arXiv:2304.11892v1 [cs.LO] 24 Apr 2023

论方案的表达能力

Gilles Dowek*和 Ying Jiang †

摘要

我们介绍一种称为 *the scheme-calculus* 的演算系统，它可以表示各种理论中的自然演绎证明。与 λ-演算系统不同，这种演算系统的语法与证明的语法十分接近，尤其是不为假设引入名称。我们证明，尽管存在非确定性，某些带类型的方案演算系统与相应的带类型 λ-演算系统具有相同的表达能力。

1 引言

我们介绍一种称为方案演算的演算系统，它可以表示自然演绎证明，而无需为假设引入名称。

1.1 方案演算

在由布劳威尔、海廷和柯尔莫哥洛夫提出的证明的算法解释中，证明通过带类型的 λ-演算系统的项来表达。在这种演算系统中，通常使用两类变量：逻辑自身的变量，以及为命名假设而引入的变量。例如，在系统 F 中，类型变量和证明变量通常是有区别的。

当引入变量来命名假设时，序列 $A, A \vdash A$ 的上下文中命题 A 的两次出现必须加以区分，因此上下文必须是命题的多重集。相比之下，在自动定理证明中，为了缩小搜索空间（例如 缩小到有限的空间），序列的上下文通常被视为命题的集合 [1]。因此，在证明论和自动定理证明中使用了略有不同的序列概念。此外，即使在最小命题逻辑中 [2, 3, 4]，这些假设名称也会使给定命题的证明构成一种非上下文无关语言。

本文介绍一种称为方案演算的演算系统，它允许在不为假设引入名称的情况下表达证明，并且上下文就是假设的集合。换言之，我们保留谓词逻辑中的变量

*法国国家信息与自动化研究院，法国巴黎 75214，意大利大道 23 号，CS 81321，
gilles.dowek@inria.fr
†计算机科学国家重点实验室，中国科学院软件研究所，中国北京 100190，邮政信箱 8718，
jy@ios.ac.cn

，但不为假设另行引入一类变量。

在方案演算中，最小命题逻辑乃至最小谓词逻辑正片段中给定命题的证明构成一种上下文无关语言。事实上，这种方案演算源自先前关于 λ-项集合的文法性质的研究 [5, 6, 7, 8, 9, 10, 11, 4, 12]。

根据生成给定类型的方案的文法，我们可以构造一个生成该类型所有 λ-项的算法，因为每个方案都对应于有限数量的项，而这些项可以由该方案计算得到 [4]。一个包含n 抽象和 p 变量的变量汇集，至多对应 p^n λ-项。在这个意义上，与 λ-演算相比，方案演算中有更多证明被视为相同；但是，与基于证明无关性的形式系统不同，并非所有证明都会被视为相同。例如，项 $\lambda x_P \lambda y_P\ x$ 和 $\lambda x_P \lambda y_P\ y$ 被视为相同，但项$\lambda x_P \lambda f_{P \Rightarrow P}\ x$ 和 $\lambda x_P \lambda f_{P \Rightarrow P}\ (f\ x)$ 并不相同。

尽管这一方案演算系统十分简单，我们证明它的表达能力与依赖类型 λ演算系统相当：对于某些类型系统，所有能在非谓词（即 二阶）算术中被证明为全定义的函数，都可以用方案演算系统表示。在这一表达能力结果中，确定性并非源自某种局部性质（例如合流性，这也是 λ演算系统所具备的），而是源自主体归约性质，以及依赖类型足以描述项的值这一事实。

1.2 变量的概念

要理解方案演算的基本思想，回到变量概念的起源会很有帮助。表示函数的项通常由一个函数构造子（常写作 λ）以及一个用于表示函数尚未知晓的自变量的占位符（有时写作 □）构成。例如，将一个数映射到其两倍的函数可以用以下项表示。

$$\lambda(2 \times \square)$$

将此项应用于 4，可得到一个化简为 2×4 的项。

但是，当应用该项时

$$\lambda\lambda\ (2 \times \square \times \square \times \square)$$

例如，对于包含符号 λ 多次出现的项，将其应用于参数 4 和 5，我们可能通过将符号 □ 的每次出现分别替换为 4 或 5，得到八种不同的句法结果。因此，需要使用一个指针，将每个占位符出现 □ 与一个函数构造子出现 λ 关联起来。

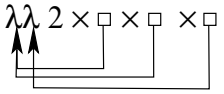
在 λ演算系统中，通过为每个 λ 的出现以及每个 □的出现指定名称来表示这一指针。随后，与占位符$\square_x$ 关联的 λ 就是将项视为树时位于其上方的第一个 λ_x。因此，将两个数映射为第一个数与第二个数的平方的乘积的两倍的函数可写作

$$\lambda_x \lambda_y\ (2 \times \square_x \times \square_x \times \square_y)$$

或者，更简单地写作

$$\lambda x \lambda y\ (2 \times x \times x \times y)$$

人们还研究了其他解决方案。一种与布尔巴基方案相关的解决方案，是用一条有向边将每个 □ 连接到对应的 λ



而在德布鲁因提出的方案中，每个 □ 都被赋予其上方对应 λ 的高度。因此我们得到

$$\lambda\lambda\ (2 \times \square_2 \times \square_2 \times \square_1)$$

将这三个项应用于项 4 和 5，会得到在每种形式系统中都只归约为 $2 \times 4 \times 4 \times 5$ 的项。

在许多情况下， λs 和 □s 都带有类型，指针必须连接类型相同的对象。类型相同保证了主体归约性质：良类型项经过归约后仍是相同类型的项。知道每个 λ 和 □ 的类型，往往能缩小原始项中符号 □ 和 λ 的出现位置之间建立连接的可能性。例如，在原始项

$$\lambda_{scal}\lambda_{vect}(2.\square_{scal}.\square_{scal}.\square_{vect})$$

只有一种方式可以为每个 □关联一个 λ，但在原始项中

$$\lambda_{nat}\lambda_{nat}\ (2 \times \square_{nat} \times \square_{nat} \times \square_{nat})$$

仍然有八种方式可以将一个 λ 与每个 □关联起来。

在方案演算中，我们不再区分八个项 $\lambda x \lambda y\,(2 \times x \times x \times x)$、 $\lambda x \lambda y\,(2 \times x \times x \times y)$……，而是考虑单个方案 $\lambda_{nat}\lambda_{nat}\,(2 \times \langle nat\rangle \times \langle nat\rangle \times \langle nat\rangle)$，其中 $\langle nat\rangle$ 是类型为 nat的规范（即唯一）变量。在该方案中，只要占位符与其上方的某个函数构造子具有相同的类型，每个占位符都可以与该函数构造子关联。方案（$\lambda_{nat}\lambda_{nat}\,(2 \times \langle nat\rangle \times \langle nat\rangle \times \langle nat\rangle)$ 4 5）聚合了八个项，并以非确定性的方式归约为（$2 \times 4 \times 4 \times 4$）、（$2 \times 4 \times 4 \times 5$）……因此，方案的归约是非确定性的，但确实具有主体归约性质。

一般而言，方案 $((\lambda_A \ldots \lambda_A \langle A\rangle)\ t_1\ t_2\ \ldots\ t_n)$ 以非确定性的 c
方式归约为每个 t_i。这正是 λ-演算系统的非确定性扩展的典型特征，例如 G. Boudol 的带有重数 [13]，的 λ-演算系统，其中项 $(\lambda x\ x)\,(t_1 \mid \ldots \mid t_n)$ 也以非确定性的方式归约为每个 t_i。

在 λ演算系统中，当我们将替换（$f\ y$）$/x$ 应用于项 $\lambda y{:}B\,(g\ x\ y)$ 时，必须重命名约束变量 y，以避免变量捕获。由于方案演算中只有一个类型为 B 的变量，我们无法再以这种方式重命名变量，因此无法避免变量捕获。

1.3 证明的算法解释

在证明的算法解释中，主体归约性质比结果唯一性更为重要。例如，考虑自然演绎证明

$$\dfrac{\dfrac{\dfrac{\dfrac{}{A, A \vdash A}\,\mathrm{ax}}{A \vdash A \Rightarrow A}\Rightarrow\mathrm{i}}{\vdash A \Rightarrow A \Rightarrow A}\Rightarrow\mathrm{i} \qquad \dfrac{t}{\vdash A}}{\dfrac{\vdash A \Rightarrow A}{\vdash A}\Rightarrow\mathrm{e}} \qquad \dfrac{u}{\vdash A}\Rightarrow\mathrm{e}$$

其中， t 和 u 是序列 $\vdash A$的两个无切证明。这个证明可以通过非确定性的方式归约为 t 或 u，但无论哪种情况，我们得到的都是 $\vdash A$的无切证明。

将该证明关联到 λ演算系统中的一个项时，我们必须为序列 $A, A \vdash A$中的每个假设指定一个变量名，并且必须选择公理规则中使用的变量。不同的选择会得到不同的证明项：$((\lambda\alpha\lambda\beta\ \alpha)\ t\ u)$和$((\lambda\alpha\lambda\beta\ \beta)\ t\ u)$，而这两个项各自都会归约为唯一的范式。

这个例子表明，在某些采用未命名假设的自然演绎表述中，证明归约是非确定性的，而 λ演算系统则以一种多少有些人为的方式引入了确定性。

正如Statman和Leivant所示（见 [14, 15]），直接定义在采用未命名假设的自然演绎证明上的证明归约过程并非强规范化，而 λ演算系统则是强规范化的。这种非终止性可视为允许变量捕获所导致的结果。由于一般而言，方案演算会失去终止性，因此必须选择一种策略。

1.4 名称和规范

在证明语言理论与编程语言理论的交叉融合中，将自然演绎证明表达为 λ-演算系统，可以看作是将变量名这一编程语言理论中所熟悉的概念引入证明论。相反，方案演算系统可以看作是将匿名假设这一证明论中所熟悉的概念引入编程语言理论。

不过，匿名资源这一理念在计算机科学中并非全新。例如，将计算机接入局域网时，我们只需使用任意一根未命名的以太网电缆。其类型“以太网电缆”足以保证计算机能够连接到网络。同样，当类型系统足够强大，能够指定程序的返回值时，就不需要为具有相同类型的不同程序命名：需要这样的程序时，任何类型正确的程序都可以使用。根据程序的规范而非名称来识别程序，或许可以避免程序和其他形式对象中的变量名泛滥。

本文将介绍的主要演算系统是一种带有依赖类型的方案演算（第3节），它允许在最小谓词逻辑中表达各种理论的证明。我们将证明这种依赖类型方案演算的三个性质：主体归约（第4节）、正规化（第5节）以及表达能力结果（第6节）。作为引入示例，我们先从简单类型方案演算开始。

2 简单类型方案演算

2.1 演算系统

定义 2.1（简单类型）令 $\mathcal{P}$ 为一个原子类型集合。简单类型归纳定义如下：

$$A = P \mid A \Rightarrow A'$$

其中 $P \in \mathcal{P}$。

定义 2.2（上下文）上下文是一个有限的简单类型集合。

定义 2.3（简单类型方案）方案归纳定义如下：

$$t = \langle A \rangle \mid \lambda_A t \mid (t\ t')$$

方案 $\langle A \rangle$ 是类型为 A的典范变量，$\lambda_A t$ 是在 t中对类型为 A 的典范变量 $\langle A \rangle$ 进行抽象后得到的方案，而 $(t\ t')$ 是将方案 t 应用于方案 t'所得的结果。

类型规则见图 1。请注意，由于上下文是集合，如果 A是 Γ的一个元素，那么 $\Gamma \cup \{A\}$ 就只是 Γ。例如，使用这些规则，方案 $\lambda_A \lambda_A \lambda_A \langle A \rangle$ 可以通过以下推导赋予类型 $A \Rightarrow A \Rightarrow A \Rightarrow A$。

$$\dfrac{\dfrac{\dfrac{\dfrac{}{A \vdash \langle A \rangle : A}}{A \vdash \lambda_A\ \langle A \rangle : A \Rightarrow A}}{A \vdash \lambda_A \lambda_A\ \langle A \rangle : A \Rightarrow A \Rightarrow A}}{\vdash \lambda_A \lambda_A \lambda_A\ \langle A \rangle : A \Rightarrow A \Rightarrow A \Rightarrow A}$$

定义 2.4（上下文中的方案）A 上下文中的方案是一个对 t_Γ，其中 t 是一个方案，Γ 是一个上下文，并且 t 在 Γ中是类型正确的。

在没有歧义时，我们有时会省略上下文 Γ。

2.2 归约

当归约方案中带下划线的可约表达式时

$$\lambda_A \ldots \lambda_A (\underline{\lambda_A (\lambda_A \ldots \lambda_A \langle A \rangle)\ u})$$

$$\frac{}{\Gamma \vdash \langle A \rangle : A}\, A \in \Gamma$$

$$\frac{\Gamma \cup \{A\} \vdash t : B}{\Gamma \vdash \lambda_A t : A \Rightarrow B}$$

$$\frac{\Gamma \vdash t : A \Rightarrow B \quad \Gamma \vdash u : A}{\Gamma \vdash (t\ u) : B}$$

图 1：简单类型方案

变量 $\langle A \rangle$ 可能受可约表达式的 λ_A 绑定，但也可能受方案中其他 λ_A绑定，该 λ_A可能位于更高处或更低处。因此，一般而言，方案u 可能会被替换为变量 $\langle A \rangle$，也可能不会被替换，这就是替换的非确定性。

然而，如果变量 $\langle A \rangle$ 在项中既没有被更高层绑定，也没有被更低层绑定，那么 $\langle A \rangle$ 唯一可能的绑定子就是可约表达式的绑定子。在这种情况下，必须替换变量$\langle A \rangle$。因此，方案 u 是否可以或必须替换变量 $\langle A \rangle$，不仅取决于被归约的可约表达式，还取决于该可约表达式在方案中的位置。因此，归约关系不能定义在方案上，而必须定义在上下文中的方案上。

为了定义归约关系，我们必须首先建立替换的概念。替换是一个有限定义域的函数，写作 $[t_1/A_1, ..., t_n/A_n]$，分别将类型为 A_1、.....、A_n 的方案 t_1、.....、t_n 与变量$\langle A_1 \rangle$、.....、$\langle A_n \rangle$关联起来。将替换应用于方案可能产生多个结果，因此这种应用会产生一个结果集合。此外，这种应用总是相对于某个上下文 Γ 执行的，该上下文规定了可以或必须进行替换的变量。更准确地说，当 A 属于 Γ 时，我们可以选择是否替换类型为 A 的典范变量；而当 A 不属于 Γ 时，这种替换就是强制的。如果 σ 是一个替换，t 是一个方案，Γ 是一个上下文，则将σ 应用于 t 并相对于上下文 Γ 所得到的结果写作 $\sigma_\Gamma t$。

为简化记号，若 S 和 S' 是方案集合，我们将 $\lambda_A S$ 记作由如下形式的方案组成的集合：$\lambda_A t$，其中 t 属于 S；将 $(S\ S')$ 记作由如下形式的方案组成的集合：$(t\ t')$，其中 t 属于 S 且 t' 属于 S'。

定义 2.5（替换）

- $\sigma_\Gamma\langle A \rangle = \{\langle A \rangle, \sigma\ (A)\}$ 若 $A \in dom\ (\sigma)$ 且 $A \in \Gamma$,
- $\sigma_\Gamma\langle A \rangle = \{\sigma\ (A)\}$ 若 $A \in dom\ (\sigma)$ 且 $A \notin \Gamma$,
- $\sigma_\Gamma\langle A \rangle = \{\langle A \rangle\}$ 若 $A \notin dom\ (\sigma)$，
- $\sigma_\Gamma\ (\lambda_A t)\ = \lambda_A \sigma_{\ (\Gamma \cup \{A\})}\ t$,
- $\sigma_\Gamma(t\ u) = (\sigma_\Gamma t\ \sigma_\Gamma u)$。

定义 2.6（归约）单步顶层 β-归约，记作$\longrightarrow$，其定义规则如下

$$((\lambda_A t)\ u)_\Gamma \longrightarrow v_\Gamma$$

其中， A、 t、 u和 Γ 是任意项，而 v 是 $[u/A]_\Gamma t$ 的任意元素。

关系 一步 β归约 $\rhd$ 是关系 $\longrightarrow$ 的上下文闭包。它归纳定义如下：

- 若 $t_\Gamma \longrightarrow t'_\Gamma$，则 $t_\Gamma \rhd t'_\Gamma$，
- 若 $t_\Gamma \rhd t'_\Gamma$，则 $(t\ u)_\Gamma \rhd (t'\ u)_\Gamma$，
- 若 $u_\Gamma \rhd u'_\Gamma$，则 $(t\ u)_\Gamma \rhd (t\ u')_\Gamma$，
- 若 $t_{\Gamma\cup\{A\}} \rhd t'_{\Gamma\cup\{A\}}$，则 $(\lambda_A\ t)_\Gamma \rhd (\lambda_A\ t')_\Gamma$。

这个 β-归约 关系 $\rhd^*$ 是 $\rhd$的自反传递闭包。

归约关系不是合流的。事实上，如果 $A \notin \Gamma$，方案 $((\lambda_A\lambda_A\ \langle A\rangle)\ t\ u)_\Gamma$ 以非确定性的方式同时归约为 t_Γ 和 u_Γ。这形式化了如下直觉：在方案 $(\lambda_A\lambda_A\langle A\rangle)_\Gamma$中，变量 $\langle A\rangle$ 可能由符号 λ_A的任一出现所绑定。

更令人惊讶的是，这个归约关系不是强规范化的。

2.3 强正规化的反例

命题 2.1（Statman [14], Leivant [15]）简单类型方案演算不是强规范化的。

证明。 令 $t = ((\lambda_A\langle A\rangle)\ \langle A\rangle)_A$。方案 $((\lambda_A t)\ t)_A$ 归约为 $[t/A]_A t$中的每个元素，例如 归约为 $((\lambda_A t)\ t)_A$。

这个反例表明，当我们在不命名假设的情况下用序列表达自然演绎时，证明归约既不具备强规范化性。例如，该证明

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A\vdash A}\,\mathrm{ax}}{A\vdash A\Rightarrow A}\Rightarrow\mathrm{i}\quad\dfrac{}{A\vdash A}\,\mathrm{ax}}{A\vdash A}\Rightarrow\mathrm{e}}{A\vdash A\Rightarrow A}\Rightarrow\mathrm{i}\qquad\dfrac{\dfrac{\dfrac{}{A\vdash A}\,\mathrm{ax}}{A\vdash A\Rightarrow A}\Rightarrow\mathrm{i}\quad\dfrac{}{A\vdash A}\,\mathrm{ax}}{A\vdash A}\Rightarrow\mathrm{e}}{A\vdash A}\Rightarrow\mathrm{e}$$

包含一个切。若消除这个切，我们必须在证明中将其替换为 t

$$\dfrac{\dfrac{\dfrac{}{A\vdash A}\,\mathrm{ax}}{A\vdash A\Rightarrow A}\Rightarrow\mathrm{i}\quad\dfrac{}{A\vdash A}\,\mathrm{ax}}{A\vdash A}\Rightarrow\mathrm{e}$$

命题 A 上的公理规则替换为证明 t 本身。由于 A 在通过切的第 $\Rightarrow$i 条规则引入之前就已经处于上下文中，因此我们可以选择是否替换每一条公理规则。如果两条都替换，就会得到我们最初的证明。

这个反例基于以下事实：方案演算允许替换约束变量。然而，即使我们禁止替换约束变量，方案演算中的变量捕获也足以危及强正规化。下面给出另一个反例。

例子。考虑上下文 $\Gamma = \{A \Rightarrow B, B \Rightarrow A, A \Rightarrow B \Rightarrow A, A, B\}$，以及 Γ、$f = \langle A \Rightarrow B \rangle$、$g = \langle B \Rightarrow A \rangle$、$h = \langle A \Rightarrow B \Rightarrow A \rangle$ 中的方案

$$a = ((\lambda_B \langle A \rangle)(f\ \langle A \rangle))$$

$$b = ((\lambda_A \langle B \rangle)(g\ \langle B \rangle))$$

$$u_0 = (h\ a\ b)$$

$$u_{n+1} = (g\ (f\ u_n))$$

$$v_n = ((\lambda_B\ u_n)\ (f\ u_n))$$

请注意，对于每个 i，方案 a 和 b 都是方案 u_i 的子方案，并且它们不出现在任何绑定子的作用域中。

方案 v_n 归约为 $[(f\ u_n)/B]_\Gamma u_n$，其中包含一个子方案 $[(f\ u_n)/B]_\Gamma b$，即 $((\lambda_A\ (f\ u_n))(g\ (f\ u_n)))$，而它又归约为 $[(g\ (f\ u_n))/A]_\Gamma (f\ u_n)$，其中包含一个子方案 $[(g\ (f\ u_n))/A]_\Gamma a$，即 $((\lambda_B (g\ (f\ u_n)))(f\ (g\ (f\ u_n))))$，它是 v_{n+1}。因此，v_n 归约为一个包含 v_{n+1} 作为子方案的方案。

2.4 策略

与任何非确定性系统一样，我们可以通过定义策略来限制方案演算系统中的归约。在方案演算系统中，非确定性源自两个不同方面。首先，与 λ-演算系统一样，当一个方案包含多个可归约式出现时，我们可以选择先归约其中一个。其次，即使选定了一个可归约式出现，仍有多种归约方式，因为替换本身具有非确定性。

最简单的策略是通过限制替换的非确定性来实现的。

最小策略的替换定义与替换的一般概念（定义 2.5）相同，但第一条除外：这里我们只取方案 $\langle A \rangle$，即

- $\underline{\sigma}_\Gamma \langle A \rangle = \{\langle A \rangle\}$, if $A \in dom(\sigma)$ and $A \in \Gamma$.

请注意，在这种情况下，$\underline{\sigma}_\Gamma t$ 总是一个单元素集合。它唯一的元素也记作 $\sigma_\Gamma t$。

例如，如果 $A \notin \Gamma$，方案 $((\lambda_A \lambda_A \langle A \rangle)\ t)_\Gamma$ 归约为 $(\lambda_A \langle A \rangle)_\Gamma$，方案 $(\lambda_A((\lambda_A \langle A \rangle)\ t))_\Gamma$ 也是如此。但方案 $((\lambda_A \langle A \rangle)\ t)_\Gamma$ 归约为t_Γ。

对偶策略是最大策略。该策略的替换定义与替换的一般概念相同，只是第一条不同：这里我们只取方案 $\sigma(A)$，即

- $\bar{\sigma}_\Gamma\langle A\rangle = \{\sigma(A)\}$, if $A \in dom(\sigma)$ and $A \in \Gamma$.

现在，方案 $((\lambda_A\lambda_A\langle A\rangle)\ t)_\Gamma$ 会归约为 $(\lambda_A t)_\Gamma$，方案 $(\lambda_A((\lambda_A\langle A\rangle)\ t))_\Gamma$ 也会如此。

直观地说，在最小策略中，必要时我们才替换变量；而在最大策略中，只要能够替换变量，我们就会替换。

一种更复杂的策略是参照最近绑定子的策略，也称为完全卸载策略 [7, 14]。在这种策略中，变量 $\langle A\rangle$ 总是指向其上方最近的绑定子。替换方式与最小策略相同，但现在对归约的定义作了修改，使得 $((\lambda_A t)\ u)_\Gamma$ 归约为 $[u/A]_\varnothing t$，而不是 $[u/A]_\Gamma t$。这样，方案 $((\lambda_A\lambda_A\langle A\rangle)\ t)_\Gamma$ 会归约为 $(\lambda_A\langle A\rangle)_\Gamma$，而方案 $(\lambda_A((\lambda_A\langle A\rangle)\ t))_\Gamma$ 会归约为 $(\lambda_A t)_\Gamma$。

对偶策略是参照最远绑定子的策略。替换与最大策略中的相同，但对归约的定义作了修改，使得当 $A \notin \Gamma$ 时，$((\lambda_A t)\ u)_\Gamma$ 归约为 $([u/A]_\Gamma t)_\Gamma$；当 $A \in \Gamma$ 时，则归约为 t_Γ。这样，方案 $((\lambda_A\lambda_A\langle A\rangle)\ t)_\Gamma$ 归约为 $(\lambda_A t)_\Gamma$，但方案 $(\lambda_A((\lambda_A\langle A\rangle)\ t))_\Gamma$ 归约为 $(\lambda_A\langle A\rangle)_\Gamma$。

第2.3节的反例表明，即使限制为弱归约，最大策略和参照最近绑定子的策略也不能正规化，也就是说，禁止在抽象下进行归约。至于最小策略和参照最远绑定子的策略能否正规化，我们暂不讨论。不过，我们将在第5节证明最小策略下弱归约的正规化性质。

最后，λ-演算系统也是方案演算的一种策略。在其中，为了化简方案 $((\lambda_A\lambda_A\langle A\rangle)\ t)_\Gamma$，我们需要知道化简的历史，这样才能确定变量 $\langle A\rangle$ 指的是哪个绑定子。在上下文 B 中的这两个类型为 $A \Rightarrow A \Rightarrow A$ 的方案中：

$$((\lambda_{A\Rightarrow A}\lambda_A\langle A \Rightarrow A\rangle)\ (\lambda_A\langle A\rangle))$$

and

$$\lambda_A((\lambda_{B\Rightarrow A}\lambda_A(\langle B \Rightarrow A\rangle\langle B\rangle))\ (\lambda_B\langle A\rangle))$$

变量 $\langle A\rangle$ 的指代没有歧义，因为它出现在单个类型为 A 的绑定子的作用域内。

归约这些方案时，两种情况下我们都会得到范式方案 $\lambda_A\lambda_A\langle A\rangle$。但要确定这个范式中变量 $\langle A\rangle$ 所指的对象，我们必须知道这个方案是从何而来的。这就是变量名的作用。将变量 x 称为变量 $\langle A\rangle$，将变量 y 称为变量 $\langle B\rangle$，将变量 f 称为变量 $\langle A \Rightarrow A\rangle$，并将变量 g称为变量 $\langle B \Rightarrow A\rangle$，第一个项 $((\lambda f\lambda x\ f)\ (\lambda x\ x))$ 归约为 $\lambda x\lambda x\ x$，第二个项 $\lambda x\ ((\lambda g\lambda x\ (g\ y))\ (\lambda y\ x))$ 归约为 $\lambda x\lambda x'\ x$；在替换过程中引入了一个新名称 x'，以避免变量捕获，并保留从变量 x 的出现位置指向其绑定子的指针。

从这个意义上说，方案演算同时推广了 λ 演算和完全消去演算。

3 依赖类型方案演算

简单类型方案演算的表达能力远不如简单类型λ-演算：采用一般的 β-归约时，范式的唯一性不复存在；若将演算限制为任何一种确定性策略，由于类型为 $A \Rightarrow A \Rightarrow A$的正规方案只有一个，因此不可能同时表达两个投影。

本节将介绍一种带有依赖类型的方案演算，并证明它与相应的带类型 λ-演算具有同等的表达能力。具体而言，我们将构造一个依赖类型系统，使其能够表达所有在非谓词算术中可证明为全函数的函数。选择非谓词算术仅仅是一个示例；我们也可以为各种理论构造类似的类型系统。

3.1 项和类型

我们首先按照多类型谓词逻辑中的通常做法，定义项和类型（或命题）。

我们考虑一种语言即一个类型集合、一组函数符号（每个函数符号都配有形如 $\langle s_1, \ldots, s_n, s\rangle$ 的元数，其中 s_1、.....、 s_n， s 是类型），以及一组谓词符号（每个谓词符号都配有元数 $\langle s_1, \ldots, s_n\rangle$，其中 s_1、.....、 s_n 是类型）。我们还为每种类型考虑一个无限变量集。类型为 s 的项归纳定义如下：

$$a = x \mid f(a_1, \ldots, a_n)$$

其中， x 是类型为 s的变量， f是元数为 $\langle s_1, \ldots, s_n, s\rangle$ 的函数符号，a_1、...、 a_n 分别是类型为 s_1、...、 s_n的项。类型通过以下方式归纳定义：

$$A = P(a_1, \ldots, a_n) \mid A \Rightarrow A' \mid \forall x\ A$$

其中， P是元数为 $\langle s_1, \ldots, s_n\rangle$ 的谓词符号， a_1、...、 a_n 分别是类型为 s_1、...、 s_n的项。

我们可以加入其他联结词和量词，所有内容都可以顺利推广。不过，我们更倾向于在第3.2节介绍的理论 HA_2中定义它们。

自由变量和约束变量、字母等价，以及替换，均按照项和类型上的通常方式定义。

一个上下文是有限的类型集合。

为了定义一种理论（例如算术），我们不考虑公理。相反，我们用一条转换规则扩展自然演绎规则

$$\frac{\Gamma \vdash A}{\Gamma \vdash B}\, A \equiv B \text{ conv}$$

$$x_1,\ \ldots\ldots,\ x_p\ \epsilon_p\ f_{\langle x_1,\ldots,x_p\rangle,\ \langle y_1,\ldots,y_n\rangle,\ A}\ (y_1,\ \ldots\ldots,\ y_n) \longrightarrow A$$

$$x = y \longrightarrow \forall c\ \ (x\ \epsilon_1\ c \Rightarrow y\ \epsilon_1\ c)$$

$$N\ (x) \longrightarrow \forall c\ (0\ \epsilon_1\ c \Rightarrow \forall y\ (N\ (y) \Rightarrow y\ \epsilon_1\ c \Rightarrow S\ (y)\ \epsilon_1\ c) \Rightarrow x\ \epsilon_1\ c)$$

$$\text{空}\ (0) \longrightarrow \forall c\ (\epsilon_0\ (c) \Rightarrow \epsilon_0\ (c)\ )$$

$$\text{空}\ (S\ (x)\ ) \longrightarrow \forall c\ \epsilon_0\ (c)$$

$$\text{前驱}(0) \longrightarrow 0$$

$$Pred(S(x)) \longrightarrow x$$

图2：重写系统 HA_2

允许我们在证明中的任意时刻，根据给定的同余，将一个命题替换为与之等价的命题，就像在模演绎 [16]中一样。该同余必须是无混淆的，也就是说，如果 $A \equiv B$，那么命题A、 B 中至少有一个是原子的，或者二者都是蕴含，或者二者都是全称量化；如果 $(A \Rightarrow A') \equiv (B \Rightarrow B')$，那么 $A \equiv B$ 且 $A' \equiv B'$；如果 $(\forall x\ A) \equiv (\forall x\ B)$，那么$A \equiv B$。

3.2 非谓词算术

依照 [17], ，我们可以在模演绎中表达谓词算术（即 一阶算术）和非谓词算术（即 二阶算术），因此这些理论的证明可以在方案演算中表达。

我们引入一个用于自然数的类型 ι，以及一个类型 κ_n（$n = 0$、1、2、.....），用于表示自然数的 n元类。函数符号为0（类型为 ι）、 S和前驱（元数为 $\langle\iota,\iota\rangle$）。谓词符号为 $=$（元数为 $\langle\iota,\iota\rangle$）、 N和 Null（元数为 $\langle\iota\rangle$），以及 ϵ_n（元数为 $\langle\iota,...,\iota,\kappa_n\rangle$）。我们用 $p\ \epsilon_1\ c$表示数 p属于（一元）类 c，并用 p_1、.....、 $p_n\ \epsilon_n\ c$ 表示序列 p_1、.....、 p_n属于 n元类 c。因此， $\epsilon_0\ (c)$是与零元类 c对应的命题。此外，对于每个命题 A以及变量序列 $\langle x_1,\ldots,x_p\rangle$、 $\langle y_1,\ldots,y_n\rangle$，只要 A的自由变量都属于 x_1、.....、 x_p、 y_1、.....、 y_n，我们就引入一个函数符号$f_{\langle x_1,\ldots,x_p\rangle}$、$_{\langle y_1,\ldots,y_n\rangle}$、$_A$。通俗地说，该符号是通过对与 A对应的概括模式实例进行斯科伦化得到的，其中 x_1、.....、 x_p作为 p元类的参数， y_1、.....、 y_n作为参数。对于所有本身不含 Skolem 符号的命题，都存在这样的符号；特别是，对于含有类量词的命题也存在这样的符号（因此具有非谓词性）。

这些符号的含义并非由公理表达，而是由图2中的重写规则表达。这些规则在项和命题上定义了同余。

众所周知，联结词和量词 $\top$、 $\bot$、 $\neg$、 $\wedge$、 $\vee$、 $\Leftrightarrow$和 $\exists$

可以在 HA_2中定义。

$$\begin{array}{rcl}
\top & = & \forall c\ (\epsilon_0(c) \Rightarrow \epsilon_0(c)) \\
\bot & = & \forall c\ \epsilon_0(c) \\
\neg A & = & A \Rightarrow \bot \\
A \wedge B & = & \forall c\ ((A \Rightarrow B \Rightarrow \epsilon_0(c)) \Rightarrow \epsilon_0(c)) \\
A \vee B & = & \forall c\ ((A \Rightarrow \epsilon_0(c)) \Rightarrow (B \Rightarrow \epsilon_0(c)) \Rightarrow \epsilon_0(c)) \\
A \Leftrightarrow B & = & (A \Rightarrow B) \wedge (B \Rightarrow A) \\
\exists x\ A & = & \forall c\ ((\forall x\ (A \Rightarrow \epsilon_0(c))) \Rightarrow \epsilon_0(c))
\end{array}$$

利用图2中的规则和转换规则所定义的同余，可以轻松证明非谓词算术的通常公理。

$$\forall x\ (x = x)$$

$$\forall x \forall y \forall c\ (x = y \Rightarrow x\ \epsilon_1\ c \Rightarrow y\ \epsilon_1\ c)$$

$$N(0)$$

$$\forall x\ (N(x) \Rightarrow N(S(x)))$$

$$\forall x \forall y\ (S(x) = S(y) \Rightarrow x = y)$$

$$\forall x\ \neg(0 = S(x))$$

$$\forall c\ (0\ \epsilon_1\ c \Rightarrow \forall y\ (N(y) \Rightarrow y\ \epsilon_1\ c \Rightarrow S(y)\ \epsilon_1\ c) \Rightarrow \forall x\ (N(x) \Rightarrow x\ \epsilon_1\ c)$$

$$\forall y_1 ... \forall y_n \exists c \forall x_1 ... \forall x_p((x_1 ... x_p\ \epsilon_p\ c) \Leftrightarrow A)$$

其中， A 是任意不含 Skolem 符号的命题，其自由变量均属于 y_1、..... y_n、 x_1、..... x_p。

3.3 方案

定义 3.1（方案）方案归纳定义如下。

$$t = \langle A \rangle \mid \lambda_A t \mid (t\ t') \mid \Lambda x\ t \mid (t\ a)$$

每种构造都对应一条自然演绎规则。类型规则见图3。它们就是自然演绎规则。

定义 3.2（上下文中的方案）上下文中的方案是一个对 t_Γ ，其中t 是一个方案， Γ 是一个上下文，并且 t 在 Γ中是类型正确的。

有时在不存在歧义时，我们会省略上下文 Γ。

现在我们定义方案上的归约关系。在此之前，我们先定义项变量替换和方案变量替换对方案的应用。

定义 3.3 设 θ 是项变量的替换， t 是一个方案。方案 θt 的归纳定义如下

$$\frac{}{\Gamma \vdash \langle A \rangle : A}\, A \in \Gamma \text{ ax}$$

$$\frac{\Gamma \cup \{A\} \vdash t : B}{\Gamma \vdash \lambda_A t : A \Rightarrow B} \Rightarrow\text{i}$$

$$\frac{\Gamma \vdash t : A \Rightarrow B \quad \Gamma \vdash u : A}{\Gamma \vdash (t\ u) : B} \Rightarrow\text{e}$$

$$\frac{\Gamma \vdash t : A}{\Gamma \vdash \Lambda x\ t : \forall x\ A}\, x \notin FV(\Gamma)\ \forall\text{i}$$

$$\frac{\Gamma \vdash t : \forall x\ A}{\Gamma \vdash (t\ a) : [a/x]A}\, \forall\text{e}$$

$$\frac{\Gamma \vdash t : A}{\Gamma \vdash t : B}\, A \equiv B \text{ conv}$$

图3：依赖类型方案

- $\theta\langle A\rangle = \langle \theta A\rangle$,
- $\theta(\lambda_A t) = \lambda_{\theta A}\theta t$,
- $\theta(u\ v) = (\theta u\ \theta v)$,
- $\theta(\Lambda x\ t) = \Lambda x'\ (\theta[x'/x]t)$，其中 x' 是一个既不出现在 $\Lambda x\ t$ 中也不出现在 θ 中的变量,
- $\theta(t\ a) = (\theta t\ \theta a)$.

请注意，与通常情况一样，这个替换通过重命名绑定项变量来避免变量捕获。

方案变量替换是一个定义域有限的函数，它将类型关联到方案。相对于上下文，将替换应用于方案的定义如下。

定义 3.4（替换）

- $\sigma_\Gamma\langle A\rangle = \{\langle A\rangle, \sigma(A)\}$，如果 $A \in dom(\sigma)$ 且 $A \in \Gamma$,
- $\sigma_\Gamma\langle A\rangle = \{\sigma(A)\}$，如果 $A \in dom(\sigma)$ 且 $A \notin \Gamma$,
- $\sigma_\Gamma\langle A\rangle = \{\langle A\rangle\}$，如果 $A \notin dom(\sigma)$,
- $\sigma_\Gamma\lambda_A t = \lambda_A \sigma_{(\Gamma \cup \{A\})} t$,
- $\sigma_\Gamma(u\ v) = (\sigma_\Gamma u\ \sigma_\Gamma v)$,
- $\sigma_\Gamma\Lambda x\ t = \Lambda x'\ \sigma_\Gamma[x'/x]t$，其中 x' 是一个既不出现在 $\Lambda x\ t$ 中，也不出现在 σ 中的变量,

- $\sigma_\Gamma\ (t\ a)\ = (\sigma_\Gamma t\ a)$ 。

定义 3.5（归约）一条一步顶层 β-归约由以下规则定义

- $((\lambda_A t)\ \ u)_\Gamma \longrightarrow v_\Gamma$，对所有 $v \in [u/A]_\Gamma t$,
- $((\Lambda x\ t)\ \ a)_\Gamma \longrightarrow ([a/x]t)_\Gamma$.

一条一步 β-归约关系 $\triangleright$ 是该关系的上下文闭包，而 β-归约关系 $\triangleright^*$ 是关系$\triangleright$的自反传递闭包。

4 主体归约

命题 4.1（替换）

- 若 $\Gamma \vdash t :\ B$，则 $[a/x]\Gamma \vdash [a/x]t :\ [a/x]B$。
- 若 $\Gamma \cup \{A\} \vdash t :\ B$ 且 $\Gamma \vdash u :\ A$，则 $\Gamma \vdash v :\ B$，对所有 v 属于 $[u/A]_\Gamma t$均成立。

证明。 对 t的结构作归纳。

请注意，尽管可以替换约束变量，也允许发生变量捕获，但该替换引理仍然成立。这是因为被捕获的变量与捕获它们的绑定子具有相同的类型。

命题 4.2（逆向）令 $\Gamma \vdash t :\ A$。

1. 若 t 是某个变量 $\langle B\rangle$，则 Γ 包含命题 B 和 $A \equiv B$。*2.* 若 $t = \lambda_B u$, 则存在一个类型 C，使得 $\Gamma \cup \{B\} \vdash u :\ C$ 且$A \equiv (B \Rightarrow C)$。*3.* 若 $t = (u\ v)$，其中 u 和 v 是方案，则存在类型 B 和 C，使得 $\Gamma \vdash u:\ B \Rightarrow C$ 且 $\Gamma \vdash v :\ B$，并且 $A \equiv C$ 。*4.* 若 $t = \Lambda x\ u$，则存在一个变量 x 和一个类型 B，使得 $\Gamma \vdash u :\ B$且 $A \equiv (\forall x\ B)$ 和 $x \notin FV(\Gamma)$。*5.* 若 $t =(u\ a)$，其中 u 是方案且 a 是项，则存在一个类型 B，使得 $\Gamma \vdash u:\ \forall x\ B$ 且 $A \equiv [a/x]B$。

证明。 对类型推导进行归纳。如果最后一条规则是转换，我们应用归纳假设以及 $\equiv$的传递性。否则，该规则的前提即可推出结果。

现在我们可以证明主体归约性质了。在此之前，我们需要证明下面的命题。

命题 4.3 若 $\Gamma \vdash t : A$ 且 $t_\Gamma \longrightarrow u_\Gamma$，则 $\Gamma \vdash u :\ A$。

证明。若 $t = ((\lambda_B t_1)\ t_2)$ 且 $u \in [t_2/B]_\Gamma t_1$，则由命题 4.2(3)，存在类型 B' 和 C'，使得 $\Gamma \vdash \lambda_B t_1 : B' \Rightarrow C'$、$\Gamma \vdash t_2 : B'$ 和 $A \equiv C'$；由命题 4.2(2)，存在类型 C，使得 $\Gamma \cup \{B\} \vdash t_1 : C$和$B' \Rightarrow C' \equiv B \Rightarrow C$。由于同余不会造成混淆，我们有 $B \equiv B'$ 和$C \equiv C'$。利用转换规则，我们有 $\Gamma \vdash t_2 : B$。由命题 4.1 可得 $\Gamma \vdash u : C$，再利用转换规则，得到 $\Gamma \vdash u : A$。

若 $t = ((\Lambda x\ t_1)\ a)$ 且 $u = [a/x]t_1$，我们选择在 Γ 中不出现的 x。利用命题 4.2、非混淆性和转换，我们得到一个类型 B'，使得 $\Gamma \vdash t_1 : B'$ 且 $A \equiv [a/x]B'$。最后利用命题 4.1 和转换即可得出结论。

定理 4.1（主体归约）若 $\Gamma \vdash t :\ A$ 且 $t_\Gamma \rhd^* u_\Gamma$，则 $\Gamma \vdash u :\ A$。

证明。我们对 $t_\Gamma \rhd u_\Gamma$ 的推导进行归纳，证明若 $\Gamma \vdash t : A$ 且$t_\Gamma \rhd u_\Gamma$，则 $\Gamma \vdash u : A$，最后再根据归约序列的长度进行归纳即可得出结论。

5 弱归约的弱正规化

现在我们证明，每个方案都可以归约为范式。鉴于第2节给出的反例，我们不能期望证明方案演算的归约具有强正规化性质。当然，可以通过模仿 λ-演算系统的归约来证明弱正规化。但规范化定理的证明所给出的方案归约策略，与定理本身同样重要；而这个简单证明所提供的策略需要引入变量名，这恰恰是我们想要避免的。因此，我们将给出另一种规范化证明，它提供的策略无需引入变量名即可定义。

获得正规化结果的第一步，是将替换限制为最小替换，即 修改定义 3.4 的第一条：不采用以下条款，

- $\sigma_\Gamma\langle A\rangle = \{\langle A\rangle, \sigma(A)\}$, if $A \in dom(\sigma)$ and $A \in \Gamma$

而采用以下条款

- $\underline{\sigma}_\Gamma\langle A\rangle = \{\langle A\rangle\}$, if $A \in dom(\sigma)$ and $A \in \Gamma$.

将替换限制为最小替换，可以排除第2节中的反例。此外，最小替换具有 λ-演算系统中替换的若干性质。特别是，约束变量从不被替换。因此，我们猜想这种最小归约是强规范化的。不过，我们将暂不解决这个问题，而是证明一个稍弱的结果：弱归约的正规化，即不在抽象下执行归约的归约。事实上，最小归约缺少 λ-演算系统的归约的一个性质：归约与替换的交换性，即每当 $t_{\Gamma\cup\{A\}}$ 归约为 $u_{\Gamma\cup\{A\}}$ 且 v 是一个方案

在上下文 Γ中具有 A 类型时，则 $(\![v/A]\!]_\Gamma t)_\Gamma$ 归约为 $(\![v/A]\!]_\Gamma u)_\Gamma$。例如，如果 $\Gamma = \{A \Rightarrow A, B \Rightarrow A, B\}$、$t = ((\lambda_A \overline{\langle A \rangle})\ (\langle A \Rightarrow A\rangle\ \langle A\rangle))$、$\overline{u = \langle A}$和 $v = (\langle B \Rightarrow A\rangle\ \langle B\rangle)$，那么 $t_{\Gamma\cup\{A\}}$ 归约为$u_{\Gamma\cup\{A\}}$。但是 $(\![v/A]\!]_\Gamma t)_\Gamma = ((\lambda_A \langle A\rangle)\ (\langle A \Rightarrow A\rangle\ (\langle B \Rightarrow A\rangle\ \langle B\rangle)))_\Gamma$ 归约为 $(\langle A \Rightarrow A\rangle\ (\langle B \Rightarrow A\rangle\ \langle B\rangle))$，而不是归约为 $(\![v/A]\!]_\Gamma u)_\Gamma = \overline{(\langle B \Rightarrow A\rangle\ \langle B\rangle)}$。

遗憾的是，基于可约性候选集的强归约规范化证明需要这一性质。但如果我们仅考虑弱归约，则不需要这一性质。

另一方面，弱归约的正规化足以证明弱头范式的存在性，而这本身足以从存在性证明中提取见证。

本节所给出的证明基于与[18]类似的思想。主要区别在于，我们考虑到了归约与替换并不交换。

5.1 归约

该一步最小顶层归约 $\longrightarrow$最小 定义如定义 3.5 中所述，但替换为最小替换。我们不考虑该关系的上下文闭包，而是如下定义一步弱最小归约。

定义 5.1（弱最小归约）一步弱最小归约 $\twoheadrightarrow$ 的定义如下：对于任意抽象以及左端为范式的任意应用，均将其视为范式；否则，归约最左归约项。它归纳定义如下。

令 t 和 u 为方案，令 a 为一个项，

- 若 $t_\Gamma \longrightarrow$最小归约 u_Γ，则 $t_\Gamma \twoheadrightarrow u_\Gamma$,
- 若 $t_\Gamma \twoheadrightarrow t'_\Gamma$,，则 $(t\ u)_\Gamma \twoheadrightarrow (t'\ u)_\Gamma$,
- 若 $t_\Gamma \twoheadrightarrow t'_\Gamma$,，则 $(t\ a)_\Gamma \twoheadrightarrow (t'\ a)_\Gamma$。

弱最小归约 关系 $\twoheadrightarrow^*$ 是关系 $\twoheadrightarrow$ 的自反传递闭包。

注意，关系 $\twoheadrightarrow$ 是函数式的（即 确定性的），也就是说，对于每个方案 t，至多存在一个方案 t' 使得 $t \twoheadrightarrow t'$。

从 t_Γ 发出的归约序列是序列$t_{0,\Gamma}$，$t_{1,\Gamma}$，$t_{2,\Gamma}$，...（有限或无限），满足 $t_{0,\Gamma} = t_\Gamma$；对于所有 i，如果存在某个 t'使得$t_{i,\Gamma} \twoheadrightarrow t'_\Gamma$,，则序列在 $i+1$ 处有定义，且 $t_{i+1,\Gamma} = t'_\Gamma$；否则$t_{i,\Gamma}$ 是该序列的最后一个元素。上下文中的方案 t_Γ 被称为可正规化，如果其归约序列是有限的。下文中，我们将 $\mathcal{N}$ 记作上下文中可正规化方案的集合。

命题 5.1（最小替换的性质）

1. 如果 t 在 Γ 中类型正确，则 $[w/A]_\Gamma t = t$。*2.* 如果 $A \in \Gamma$，则 $[w/A]_\Gamma t = t$。*3.* 如果 $B \neq A$，则 $[w/A]_{\Gamma \cup \{B\}} t = [w/A]_\Gamma t$。

证明。

1. 对 t 的结构进行归纳。唯一的非平凡情形是 $t = \langle B \rangle$。在此情形下，$B \in \Gamma$，且两个方案都等于 $\langle B \rangle$。2. 对 t 的结构进行归纳。唯一的非平凡情形是 $t = \langle A \rangle$。在此情形下，$A \in \Gamma$，因此两个方案都等于 $\langle A \rangle$。3. 对 t 的结构进行归纳。唯一的非平凡情形是 $t = \langle A \rangle$。在此情形下，要么 $A \in \Gamma$，此时两个方案都等于 $\langle A \rangle$；要么 $A \notin \Gamma$，此时 $A \notin (\Gamma \cup \{B\})$，且两个方案都等于 w。

5.2 Girard 的可约性候选集

定义 5.2（方案集上的运算）如果 E 和 F 是上下文中的方案集，我们定义集合

$$E \Rrightarrow F = \{t_\Gamma \in \mathcal{N} \mid \forall t' \forall u \ ((t_\Gamma \twoheadrightarrow^* (\lambda_A t')_\Gamma \ and \ u_\Gamma \in E) \Rightarrow ([u/A]_\Gamma t')_\Gamma \in F)\}$$

如果 S 是上下文中的方案集的集合，我们定义集合

$$\tilde{\forall}\ S = \{t_\Gamma \in \mathcal{N} \mid \forall t' \forall a \forall E \ ((t_\Gamma \twoheadrightarrow^* (\Lambda x\ t')_\Gamma \ and \ E \in S) \Rightarrow ([a/x]t')_\Gamma \in E)\}$$

定义 5.3（可约性候选集 [19]）如果一个方案对应于公理规则或消去规则，但不对应于引入规则，则称其为中性的。如果上下文中的方案集 R 满足以下条件，则称其为可约性候选集：

- 如果 $t_\Gamma \in R$，那么 t_Γ 是可正规化的，
- 如果 $t_\Gamma \in R$ 且 $t_\Gamma \twoheadrightarrow^* t'_\Gamma,$，那么 $t'_\Gamma \in R,$
- 如果 t_Γ 是中性的，并且对于每个满足 $t_\Gamma \twoheadrightarrow t'_\Gamma,$ 的 t'_Γ，我们都有 $t'_\Gamma \in R,$ 那么 $t_\Gamma \in R$。

我们用 $\mathcal{C}$ 表示可归约性候选集的集合。

注意，由于归约关系是确定性的，第三个条件可以改述为：(1) 如果 t_Γ 是中性的且处于范式，则 $t_\Gamma \in R$，以及 (2) 如果 t_Γ 是中性的，具有一个一步归约式 t'_Γ，且该归约项属于 R，则 t_Γ 属于 R。

命题 5.2 如果 E 和 F 是上下文中的方案集，那么 $E \Rrightarrow F$ 是~可约性候选集。如果 S 是上下文中的方案集的集合，那么 $\forall\ S$ 是可约性候选集。

˜证明。 根据定义，集合 $E \Rightarrow F$ 和 $\forall S$ 中的所有方案都是可正规化的。

对于归约闭包，只需注意：如果 $t_\Gamma \twoheadrightarrow^* t'_\Gamma$ 和 t_Γ 是可正规化的，那么 t'_Γ 也是可正规化的；如果 $t_\Gamma \twoheadrightarrow^* t'_\Gamma$ 和 $t'_\Gamma \twoheadrightarrow^* u_\Gamma$, ，那么 $t \twoheadrightarrow^* u_\Gamma$。

对于第三个性质，只需注意：如果 t_Γ 是上下文中的方案，并且对于所有 t'_Γ ，只要 $t_\Gamma \twoheadrightarrow t'_\Gamma$, t'_Γ 是可正规化的，那么 t_Γ 就是可正规化的；如果 t_Γ是上下文中的中性方案，并且 $t_\Gamma \twoheadrightarrow^* u_\Gamma$ ，其中 u 是一个引入，那么归约序列不为空，因此存在一个方案 t'_Γ ，使得$t_\Gamma \twoheadrightarrow t'_\Gamma \twoheadrightarrow^* u_\Gamma$。

5.3 $\mathcal{C}$-模型

取值于可约候选项代数的模型，或称 $\mathcal{C}$-模型，定义方式与经典模型相同，只是命题在可约候选项代数 $\mathcal{C}$ 中解释。因此，它由一个集合 M_s 组成：对每种类型 s，有一个函数 $\hat{}f$，从 $M_{s_1} \times \cdots \times M_{s_n}$ 到 M_s；对每个元数为 $\langle s_1, \ldots, s_n, s\rangle$ 的函数符号 f，有 $\hat{}$；对每个元数为 $\langle s_1, \ldots, s_n\rangle$ 的谓词符号 P，有一个函数 P，从 $M_{s_1} \times \cdots \times M_{s_n}$ 到 $\mathcal{C}$。赋值下项的指称按通常方式定义。命题的指称定义如下：

- $[\![P\ (a_1、\ .....、\ a_n)\ ]\!]_\phi = \hat{}P\ ([\![a_1]\!]_\phi、\ .....、\ [\![a_n]\!]_\phi)$ ，
- $[\![A \Rightarrow B]\!]_\phi = [\![A]\!]_\phi \tilde{\Rightarrow} [\![B]\!]_\phi$，˜
- $[\![\forall x\ A]\!]_\phi = \forall\ \{[\![A]\!]_{\phi + x=e} \mid e \in M_s\}$，其中 s 是变量 x的类型，而 $\phi + x = e$ 是一个与 ϕ 处处相同的赋值，但在 x处取值为 e。

定义 5.4 如果对于所有类型 A 和 B以及每个赋值 ϕ， $A \equiv B$ 蕴含 $[\![A]\!]_\phi = [\![B]\!]_\phi$, 则称同余 $\equiv$ 在 $\mathcal{C}$-模型 $\mathcal{M}$ 中有效。

5.4 弱归约的弱正规化

由于方案演算允许变量捕获，替换不能像通常那样进行复合。例如，如果 $\Gamma = \{B \Rightarrow A, C \Rightarrow B, C\}$， f 是变量$\langle B \Rightarrow A\rangle$ ，而 u 是项（$\langle C \Rightarrow B\rangle\ \langle C\rangle$），则有

$$[u/B]_\Gamma[(f\ \langle B\rangle)/A]_{\Gamma\cup\{B\}}(\lambda_B\ \langle A\rangle) = (\lambda_B\ (f\ \langle B\rangle))$$

and

$$[u/B]_\Gamma[(f\ \langle B\rangle)/A]_{\Gamma\cup\{B\}}\langle A\rangle = (f\ u)$$

但不存在这样的替换 σ 和上下文 Δ ，使得

$$\sigma_\Delta(\lambda_B\ \langle A\rangle) = (\lambda_B\ (f\ \langle B\rangle))$$

and

$$\sigma_\Delta\langle A\rangle = (f\ u)$$

因为我们不能同时拥有 $\sigma_{\Delta\cup\{B\}}\langle A\rangle =$(f $\langle B\rangle$) 和 $\sigma_\Delta\langle A\rangle =$(f u)。因此，需要引入自由替换序列这一概念。

定义 5.5（自由替换序列）设 Γ 为一个上下文，ϕ 为一个赋值，则 Γ、ϕ 中的自由替换序列归纳定义如下。

- 空序列是自由替换序列。
- 如果 ρ 是自由替换序列，C 是类型，w 是在上下文 Γ中的方案，且满足 $w_\Gamma \in [\![C]\!]_\phi$，那么 $([w/C]_\Gamma, \rho)$ 是自由替换序列。
- 如果 ρ 是自由替换序列，x 是不出现在 ρ中的项变量，并且 a 是项，那么 $([a/x], \rho)$ 是自由替换序列。

定义 5.6 设 ρ 是 Γ中的自由替换序列，ϕ 和 a 是项。项 ρa 定义如下。

- 如果 ρ 是空序列，那么 $\rho a = a$,
- 如果 $\rho = ([w/C]_\Gamma, \rho')$，那么 $\rho a = \rho' a$,
- 如果 $\rho = ([b/x], \rho')$，那么 $\rho a = [b/x](\rho' a)$。

设 A 为一种类型，类型 ρA 定义如下。

- 如果 ρ 是空序列，那么 $\rho A = A$,
- 如果 $\rho = ([w/C]_\Gamma, \rho')$，那么 $\rho A = \rho' A$,
- 如果 $\rho = ([b/x], \rho')$，那么 $\rho A = [b/x](\rho' A)$。

设 t 为一种方案，方案 ρt 定义如下。

- 若 ρ 是空序列，则 $\rho t = t$,
- 若 $\rho = ([w/C]_\Gamma, \rho')$，则 $\rho t = [w/\rho' C]_\Gamma (\rho' t)$，
- 若 $\rho = ([b/x], \rho')$，则 $\rho t = [b/x] (\rho' t)$。

在下面的命题中，我们照例证明：如果一个方案具有类型 A，那么它就是 A 的解释中的一个元素（因此我们将能够推出它是可正规化的）。

命题 5.3 设 $\equiv$ 是一个同余，$\mathcal{M}$ 是 $\mathcal{C}$-模型，$\equiv$、Γ 和 Δ 是上下文，ϕ 是一个赋值，t 是 A 类型的方案（模 $\equiv$，位于 Δ中），ρ 是 Γ中的自由替换序列，ϕ 满足 ρt 是在 Γ中良类型化的方案。则 $(\rho t)_\Gamma \in [\![A]\!]_\phi$。

证明。对 t 的类型推导进行归纳。

• 公理。方案 t 等于 $\langle A\rangle$。如果 A 不在 ρ 或 $A\in\Gamma$ 中任何替换的定义域内，那么 $(\rho t)_\Gamma=\langle\rho A\rangle_\Gamma$。因此，由于候选 $[\![A]\!]_\phi$ 包含所有正规中性项式，$(\rho t)_\Gamma\in[\![A]\!]_{\phi\circ}$ 否则，令 $[w/A]_\Gamma$ 为 ρ 中绑定 A 的最右替换。我们有 $\rho=\rho_2,[w/A]_\Gamma,\rho_1$，并且 $(([w/A]_\Gamma,\rho_1)\langle A\rangle)_\Gamma=([w/\rho_1A]_\Gamma,\langle\rho_1A\rangle)_\Gamma=w_{\Gamma\circ}$ 序列 ρ 是一个自由替换序列，方案 w 在 Γ 中类型正确，且不包含任何在 ρ_2 中绑定的项变量，因此，利用命题 5.1(*1*)，$(\rho t)_\Gamma=w_\Gamma\in[\![A]\!]_{\phi\circ}$

• ⇒i. 方案 t 的形式为 $\lambda_B u$，$A=(B\Rightarrow B')$ 和 $(\rho t)_\Gamma=(\rho(\lambda_B u))_{\Gamma\circ}$ 遍历抽象时，ρ 的替换将其上下文扩展为 $\Gamma\cup\{C\}$，其中 C为某种类型。利用命题 5.1(2)，我们从 ρ 中删除绑定类型 C 的替换；再利用命题 5.1(3)，从其余替换的上下文中擦除 C。这样，我们在 Γ、ϕ 和 $(\rho t)_\Gamma=(\lambda_{\rho B}(\rho' u))_\Gamma=(\lambda_{\rho' B}(\rho' u))_\Gamma$中得到另一个自由替换序列 ρ'。该方案是范式，因此是可归约为范式的，并且它只能归约到自身。为证明它属于 $[\![A]\!]_\phi=[\![B\Rightarrow B']\!]_\phi$，我们需要证明：对于 Γ 中满足 $v_\Gamma\in[\![B]\!]_\phi$的所有方案 v，方案 $([v/\rho' B]_\Gamma(\rho' u))_\Gamma=(([v/B]_\Gamma,\rho')u)_\Gamma$ 属于 $[\![B']\!]_{\phi\circ}$ 这由归纳假设以及 $([v/B]_\Gamma,\rho')$ 是自由替换序列这一事实可得。

• $\forall i.$ 方案 t 具有形式 $\Lambda x\ u$，不妨设 x 不出现在 ρ中。我们有 $A=(\forall x\ B)$ 和 $\rho t=\Lambda x\ \rho u_\circ$ 此方案是正规的，因此它是可正规化的，并且只能约化为自身。为了证明它属于 $[\![A]\!]_\phi=[\![\forall x\ B]\!]_\phi$，我们需要证明：对于所有项 a，以及 M_s中的 e，其中 s 是变量 x的类型，方案 $([a/x](\rho u))_\Gamma=(([a/x],\rho)u)_\Gamma$ 属于 $[\![B]\!]_{\phi+x=e\circ}$ 由于 x 不出现在 ρ中，序列 $([a/x],\rho)$ 是针对 $\Gamma,(\phi+x=e)$ 的自由替换序列。因此，根据归纳假设，此方案属于 $[\![B]\!]_{\phi+x=e}$。

• ⇒e. 方案 t 的形式为 $(u\ v)$。因此，$\rho t=(u'\ \rho v)$，其中 $u'=\rho u_\circ$ 根据归纳假设，$u'_\Gamma\in[\![B\Rightarrow A]\!]_\phi$ 且 $(\rho v)_\Gamma\in[\![B]\!]_{\phi\circ}$ 因此，方案 u'_Γ 是可正规化的。设 n 为从 u'_Γ开始的归约序列长度。我们对 n 进行归纳，证明：若 $u'_\Gamma\in[\![B\Rightarrow A]\!]_\phi$，且从 u 开始的归约序列长度为 n，并且 $v'_\Gamma\in[\![B]\!]_\phi$，则 $(u'\ v')_\Gamma\in[\![A]\!]_{\phi\circ}$ 由于 $(u'\ v')_\Gamma$ 是中性的，我们只需证明它可能的一步归约式属于 $[\![A]\!]_{\phi\circ}$ 若归约发生在 u'中，我们直接应用归纳假设。否则，归约发生在顶层。我们有 $u'_\Gamma=(\lambda_{\rho B}u'')_\Gamma$，其归约项为 $([v'/\rho B]_\Gamma u'')_\Gamma$，根据 $[\![B\Rightarrow A]\!]_\phi$ 的定义，它属于 $[\![A]\!]_\phi$。

• $\forall e.$ 方案 t 具有形式 $(u\ a)$，其中 u 具有类型 $\forall x\ B$，$A=[a/x]B$，$\rho t=(u'\ \rho a)$，其中 $u'=\rho u_\circ$ 根据归纳假设，$u'_\Gamma\in[\![\forall x\ B]\!]_{\phi\circ}$ 因此，方案 u'_Γ 是可正规化的。令 n 为从该方案开始的归约序列长度。我们通过对 n 进行归纳来证明：如果 $u'_\Gamma\in[\![\forall x\ B]\!]_\phi$，则从 u 开始的归约序列长度为

n，且 a' 是一个项，则 $(u'\ a')_\Gamma \in [\![[a/x]B]\!]_\phi = [\![B]\!]_{\phi+x=[\![a]\!]_\phi}$。由于该方案是中性的，我们只需证明其可能的一步归约式属于 $[\![B]\!]_{\phi+x=[\![a]\!]_\phi}$。如果归约发生在 u'中，我们直接应用归纳假设。否则，归约发生在顶层。我们有 $u'_\Gamma = (\Lambda x\ u'')_\Gamma$，而归约项为 $([a'/x]u'')_\Gamma$，根据 $[\![\forall x\ B]\!]_\phi$的定义，它属于 $[\![B]\!]_{\phi+x=[\![a]\!]_\phi}$。

- 转换规则。如果最后一条规则是转换规则，根据归纳假设，我们有 $(\rho t)_\Gamma \in [\![B]\!]_\phi$，其中某个 $B \equiv A$，并且有 $[\![B]\!]_\phi = [\![A]\!]_\phi$。因此 $(\rho t)_\Gamma \in [\![A]\!]_\phi$。

定理 5.1（正规化）设 $\equiv$ 是一个具有 $\mathcal{C}$-模型$\mathcal{M}$的同余。设 Γ 是一个上下文，t 是在 Γ中模 $\equiv$、类型为 A 的方案。那么 t_Γ 是可正规化的。

证明。由命题 5.3，对所有 ϕ, $t_\Gamma \in [\![A]\!]_\phi$，因此它是可正规化的。

5.5 HA_2中的正规化

命题 5.4 在 HA_2 中良类型的所有方案都是可正规化的。

证明。我们按如下方式构造一个 $\mathcal{C}$-模型。设 $M_\iota = \mathbb{N}$ 和 $M_{\kappa_n} = \mathbb{N}^n \to \mathcal{C}$。符号 0、$S$ 和 $Pred$ 按标准方式解释。函数 $\hat{\epsilon}_n$ 将 $k_1, \ldots, k_n$ 和 f 映射到 $f(k_1, \ldots, k_n)$，$\hat{}=$ 将 n 和 m 映射到 $[\![\forall c\,(x\ \epsilon_1\ c \Rightarrow \hat{} y\ \epsilon_1\ c)]\!]_{n/x,m/y}$，而 $Null$ 将 0 映射到 $[\![\forall c\,(\epsilon_0(c) \Rightarrow \epsilon_0(c))]\!]$，并将其他数映射到 $[\![\forall c\ \epsilon_0(c)]\!]$。

$\hat{}$为了定义 N，我们首先定义函数 Φ，该函数将任意函数 α 从$\mathbb{N} \to \mathcal{C}$ 映射为这样一个函数：对于赋值 n/x，在定义域为 M_ι 和 M_{κ_n}的模型中，将 n 映射为命题$\forall c\,(0\ \epsilon_1\ c \Rightarrow \forall y\,(N(y) \Rightarrow y\ \epsilon_1\ c \Rightarrow S(y)\ \epsilon_1\ c) \Rightarrow x\ \epsilon_1\ c)$ 的解释；其中 0 和 S 按标准方式解释，ϵ_n 由 $\hat{\epsilon}_n$ 解释，而 N 由 α 解释。按逐点包含排序的集合 $\mathbb{N} \to \mathcal{C}$ 是完备的，函数 Φ 是单调的，$\hat{}$ 因此它具有一个不动点 β。我们令 $N = \beta$。

这样，我们就可以解释每个不含 Skolem 符号的命题 A。最后，我们将符号 $f_{x_1,\ldots,x_p,y_1,\ldots,y_n,A}$ 解释为这样的函数：将 $a_1, \ldots, a_p$ 映射为一个函数，该函数再将 $b_1, \ldots, b_n$ 映射为 $[\![A]\!]_{a_1/x_1,\ldots,a_p/x_p,b_1/y_1,\ldots,b_n/y_n}$。

6 表达能力

现在我们将看到，尽管定义 3.5 中给出的归约具有非确定性，对于某些方案，仍可保证结果唯一性，并且每个可证明为全函数的函数都可以由这样的方案表达。

如果 n 是一个自然数，我们将 n _写作项 $S^n(0)$。

命题 6.1（Parigot 数字 [20]）设 n 为一个自然数，则存在一个类型为 $N(n)$ 的方案 ρ_n。

证明。设 $A = (0\ \epsilon_1\ c)$ 和 $B = (\forall y\ (N\ (y)\ \Rightarrow y\ \epsilon_1\ c \Rightarrow S\ (y)\ \ \epsilon_1\ c))$。取

$$\rho_0 = \Lambda c \lambda_A \lambda_B\ \langle A \rangle$$

and

$$\rho_{n+1} = \Lambda c \lambda_A \lambda_B\ (\langle B \rangle\ \underline{n}\ \rho_n\ (\rho_n\ c\ \langle A \rangle\ \langle B \rangle))$$

命题 6.2（见证性质）设 $\exists x\ A$ 是一个闭命题。从空上下文中类型为 $\exists x\ A$ 即 $\forall c\ ((\forall x\ (A \Rightarrow \epsilon_0(c))) \Rightarrow \epsilon_0(c))$ 的方案 t，我们可以提取出一个项 b 和一个在空上下文中类型为 $[b/x]A$ 的方案。

证明。考虑一个类型为 κ_0 的项变量 c，以及 $g = \langle \forall x\ (A \Rightarrow \epsilon_0(c))\rangle$。方案（$t\ c\ g$）在上下文 $\{\forall x\ (A \Rightarrow \epsilon_0(c))\}$中的类型为 $\epsilon_0(c)$，因此其弱范式的形式为（$g\ a\ u$），其中 a 是一个类型为 ι 的项，而 u 是一个在上下文 $\{\forall x\ (A \Rightarrow \epsilon_0(c))\}$中类型为 $[a/x]A$ 的方案。设 $e = f_{\exists x\ A}$ 和 w 是 $\forall x\ (A \Rightarrow \exists x\ A)$ 的一个闭证明。设 $b = [e/c]a$ 和 $v = [w/\forall x\ (A \Rightarrow \exists x\ A)]_{\varnothing}[e/c]u$。方案 v 在空上下文中的~~类型为 $[b/x]A$~~。

根据见证性质，我们得到在 HA_2 中可证明的所有函数的可表达性。首先需要用到初等逻辑中的以下结果。

命题 6.3 对于每个从 $\mathbb{N}^n$ 到 $\mathbb{N}$ 的可计算函数 f，都存在一个命题 A，使得 $[p_1/x_1, \ldots, p_n/x_n, q/y]A$ 在 HA_2 中可证明，当且仅当 $q = f(p_1, \ldots, p_n)$。

定义 6.1（可证明全函数）若函数 f 在 HA_2 中可证明全，则称其为

$$\forall x_1\ (N(x_1) \Rightarrow \ldots \Rightarrow \forall x_n\ (N(x_n) \Rightarrow \exists y\ (N(y) \wedge A)) \ldots)$$

在 HA_2 中可证明。

定理 6.1 对于每个在 HA_2 中可证明全的可计算函数 f，都存在一个方案 t，使得对于所有 $p_1, \ldots, p_n$，从方案 $(t\ p_1\ \rho_{p_1}\ p_2\ \rho_{p_2}\ \cdots\ p_n\ \rho_{p_n})$ 中提取的见证的范式为 $f(p_1, \ldots, p_n)$。

证明。任取一个类型为 $\forall x_1 (N(x_1)\ \Rightarrow \ldots \Rightarrow \forall x_n (N(x_n)\ \Rightarrow \exists y (N(y)\ \wedge A)) \ldots)$ 的方案。

在 $\mathbf{HA}_2$ 中可证明为全函数的函数集合，是否等于或严格包含于可用方案演算表达的函数集合，仍是一个开放问题。

7 未来工作

除了 HA_2 之外，定理 5.1 还适用于许多理论，例如 简单类型论和集合论的一些变体。当它们无法在理论中定义时，所有

联结词和量词都必须被视为原始的，就像在 [18] 中一样。尽管过程繁琐，但规范化证明可以顺利推广。

一个更具挑战性的问题是，证明其他归约策略也具有正规化性质，而不仅仅是弱最小归约。这可能需要将可归约性证明推广到归约与替换不可交换的情形。

8 致谢

作者谨向匿名审稿人致谢，他们在许多方面帮助作者改进了本文。本研究部分得到 NSFC 60673045、NSFC 60833001 和 NSFC 60721061 的资助。

参考文献

[1] S. C. Kleene，《元数学导论》，North-Holland，1952。

[2] M. Zaionc，关于四阶类型的 λ 可定义性的概率方法，《理论计算机科学电子笔记》140（2005）41–54。[3] T. Joly，关于 λ 可定义性 I：固定模型问题与匹配问题的推广，《计算机基础》65（1-2）（2005）135–151。[4] G. Dowek，Y. Jiang，正公式证明的枚举，《计算机杂志》52（7）（2009）799–807。[5] C. B. Ben-Yelles，λ 演算中的类型赋值；语法与语义，博士学位论文，斯旺西大学学院（1979）。[6] M. Zaionc，通过类型化λ演算中的闭项进行证明构造的机械过程，《自动推理杂志》4（1988）173–190。[7] M. Takahashi，Y. Akama，S. Hirokawa，正规方案及其文法，《信息与计算》152（2）（1996）144–153。[8] H. Comon，Y. Jurski，高阶匹配与树自动机，载于：《计算机科学逻辑》，1997，第157–176页。[9] S. Broda，L. Damas，TA-λ 中类型正规居民的上下文无关文法表示，载于：Wollic，2001，计算机科学讲义第2258卷，施普林格出版社，第145–159页。[10] S. Broda，L. Damas，关于类型的长正规居民，《逻辑与计算杂志》15（2005）353–390。[11] G. Dowek，Y. Jiang，特征变量、括号化与正的最小谓词逻辑的可判定性，《理论计算机科学》360（2006）193–208。

[12] S. Salvati，简单类型 λ 演算中的可识别性，载于：Wollic，2009。[13] G. Boudol，多重性 λ 演算，技术报告2025，法国国家信息与自动化研究所（1993）。[14] A. Troelstra，H. Schwichtenberg，基础证明论，剑桥大学出版社，1996，2000。[15] D. Leivant，自然演绎中的假设类，《数学逻辑与数学基础杂志》25（1979）1–4。¨G. Dowek，T. Hardin，C. Kirchner，模理论证明，《自动推理杂志》31（2003）33–72。[16] G. Dowek，B. Werner，作为模理论的算术，载于：J. Giesel（编），项重写及其应用，计算机科学讲义第3467卷，施普林格出版社，2005，第423–437页。[17] G. Dowek，B. Werner，模证明规范化，《符号逻辑杂志》68（2003）1289–1316。[18] J.-Y. Girard，将 G¨odel 的解释扩展到分析，并将其应用于分析中的切消去和类型论，载于：J. Fenstad（编），第二届斯堪的纳维亚逻辑研讨会，逻辑与数学基础研究第63卷，North-Holland，1971，第62–92页。[20] M. Parigot，基于证明的编程：一种二阶类型论，载于：H. Ganzinger（编），欧洲编程研讨会，计算机科学讲义第300卷，施普林格出版社，1988，第145–159页。