二阶抽象语法的 E-合一

Nikolai Kudasov
因诺波利斯大学，大学街1号，因诺波利斯，鞑靼斯坦共和国，俄罗斯

摘要

高阶合一（HOU）研究（扩展的） λ-演算的合一问题，可视为关于 λ-项在 $\beta\eta$-等价意义下的方程统一（E-统一）实例。我们研究具有任意变量绑定构造的语言中的项在任意二阶方程理论下的方程统一。具有一般变量绑定和参数化元变量的抽象语法，使我们能够处理任意绑定子，而无需依赖 λ-演算，也无需使用不便且容易出错的项编码，从而构建出更灵活的框架。本文针对二阶抽象语法引入 E-统一，并描述了此类问题的合一过程，该过程融合了完全高阶统一与一般 E-统一的思想。我们证明该过程是可靠且完备的。

2012年 美国计算机协会主题分类 计算理论 → 方程逻辑与重写

关键词与短语 E-合一、高阶合一、二阶抽象语法

致谢 感谢 Nikolay Shilov 对本文内容给予鼓励并进行富有成效的讨论。也感谢 Alexander Chichigin、Violetta Sim、Timur Fayzrahmanov、Alexey Stepanov 和 Dale Miller 对本工作的早期版本进行讨论。感谢 Alexander Chichigin、Timur Fayzrahmanov 和 Ikechi Ndukwe 为本文校对。最后，感谢 FSCD 2023 的匿名评审提供宝贵而详尽的意见。

arXiv:2302.05815v2 [cs.LO] 7 Jun 2023

1 引言

高阶合一（HOU）是利用函数求解符号方程的过程。考虑以下无类型 λ-演算中的方程，我们要对 M求解：

$$\mathrm{M}\ g\ (\lambda z.z\ a) \stackrel{?}{=} g\ a \tag{1}$$

该问题的一个解（称为合一器）是替换 $\theta = [\mathrm{M} \mapsto \lambda x.\lambda y.y\ x]$。事实上，将 θ 应用于该方程后，我们得到等式两边的 β-等价项：

$$\theta(\mathrm{M}\ g\ (\lambda z.z\ a)) = (\lambda x.\lambda y.y\ x)\ g\ (\lambda z.z\ a) \equiv_\beta (\lambda y.y\ g)\ (\lambda z.z\ a) \equiv_\beta (\lambda z.z\ a)\ g \equiv_\beta g\ a = \theta(g\ a)$$

高阶合一有许多应用，包括高阶逻辑中的类型检查 [22] 和自动定理证明 [24]。一般而言，高阶合一是不可判定的 [12]，若没有非平凡的优化，寻找合一器可能代价相当高。对于某些问题，只需求解一个可判定片段即可。例如，Miller高阶模式统一 [23] 及其变体 [13, 38] 常用于依赖类型推断。

传统上，高阶合一算法只考虑一种绑定子—— λ-抽象。对此的一种常见解释是诉诸高阶抽象语法（HOAS） [28]:

> 众所周知， λ-抽象足以表示公式中的量化、函数式程序中的抽象，以及许多其他变量绑定结构 [28]。（Nipkow 和 Prehofer [25, 第 1]节）

然而，高阶抽象语法（HOAS）受到了一些批评，批评者包括编程语言实现者和形式化研究人员。他们认为，HOAS 及其变体 [2, 37] 存在一些实际问题 [19, 5], ，例如难以在绑定子下进行操作、存在一般递归问题 [18]„ 以及缺乏形式化基础 [9]。

Fiore 和 Szamozvancev [9] 认为，现有的形式化、推理和实现带有变量绑定的语言的方法“虽能在一定程度上缓解问题，但代价是采用不便且容易出错的项编码，并且缺乏形式化基础”。他们建议转而考虑二阶抽象语法 [10], ，也就是带有变量绑定和参数化元变量的抽象语法。事实上，Fiore 和 Szamozvancev [9]使用二阶抽象语法，在 Agda 中为带有变量绑定的语言生成元理论。

本文提出了一种针对二阶抽象语法进行方程推理的机械化方法。我们的工作受到现有 HOU 和 E-合一研究的启发。尽管我们无法直接复用所有高度依赖 λ-演算语法的 HOU 思路，但仍能对其中许多思路加以改造，因为二阶抽象语法提供了参数化元变量，它们类似于 HOU 中的flex 项。

1.1 相关工作

据我们所知，目前尚无针对二阶抽象语法进行方程推理的机械化方法。因此，我们将本文的方法与涵盖方程推理的现有 HOU 算法进行比较。Snyder 的高阶 E-合一 [29] 将一阶等式理论扩展到 HOU。Nipkow 和 Prehofer [25]研究了高阶重写和（高阶）方程推理。如前所述，这些方法依赖 λ-抽象和类似 HOAS 的编码，以处理其他绑定构造。相比之下，我们研究的是模二阶等式理论下的任意绑定构造。

Dowek、Hardin 和 Kirchner [8] 将高阶合一表示为 $\lambda\sigma$-演算中的一阶 E-合一（ λ-演算的一种变体，带有显式替换），并将其视为模 $\beta\eta$-归约。他们的思路是使用显式替换和 de Bruijn 指标，使元变量替换不会导致名称捕获，并可归约为嫁接 （一阶替换）。这样，就可以应用一阶 E-合一算法（如窄化）。Kirch-ner 和 Ringeissen [17]将这一方法扩展到带一阶公理的高阶方程合一。在我们的工作中，参数化元变量的作用方式类似于 $\lambda\sigma$-演算中带显式替换的元变量。虽然应该可以将二阶方程编码为 σ-演算中的一阶方程（带显式替换，但不带 λ-抽象和应用），但这种方法似乎还要求我们对合一过程的规则进行编码。

由于一些等式理论可以表述为项重写系统，因此存在一条结合重写系统与类型系统的研究路线，其源头可追溯至 Tannen 的工作 [34], ，该工作将简单类型的 λ 演算扩展为支持高阶重写规则。构造演算 [1, 36, 30, 31] 和 $\lambda\Pi$演算 [7]也有类似的扩展。Cockx、Tabareau 和 Winterhalter [6]提出了重写类型理论（RTT），它是在 Martin-Löf 类型理论中加入（第一阶）重写规则的扩展。Chrząszcz 和 Walukiewicz-Chrząszcz [3] 讨论了如何为 Coq 添加重写规则。Cockx [4]介绍了 Agda 的一项实用扩展，该扩展基于与 RTT 相同的理念，支持高阶非线性重写规则 [6]。在通过求值归一化 （NbE）而非高阶合一来比较类型（及项）的证明助手中，重写尤其有用。与扩展了重写规则的类型理论不同，我们的方法依赖于简单类型语法，但允许采用任意的二阶等式理论，即使不存在合流重写系统，也能进行合一。

Kudasov [20] 在 Haskell 中，为内禀作用域语法实现了高阶（预）合一和依赖类型推断，并使用所谓的 自由作用域单子 ，根据描述节点类型的数据类型生成对象语言的语法。这种定义——

本质上是对二阶抽象语法的简化呈现。Kudasov 的预统一过程包含若干启发式方法，但这篇预印本并未给出可靠性或完备性结果。

1.2 贡献

本文的主要贡献是提出了用于二阶抽象语法的 E-合一，以及一种可靠且完备的合一过程。本文其余部分的结构如下：

- 第 2 节简要回顾 Fiore 和 Hur [10] 提出的二阶抽象语法、方程逻辑和项重写。第 3 节将传统的 E-合一概念推广为针对一组二阶方程 E 的 E-合一问题和 E-合一器。第 4 节定义了一个合一过程，该过程可枚举任意给定的 E-合一问题的解，并证明其可靠性。第 5 节借鉴现有关于 E-合一和高阶合一的研究，证明我们的合一过程具有完备性。最后，我们将在第 6 节讨论实际实现中一些可能的实用性改进，以及我们方法的局限性。

2 二阶抽象语法

本节回顾 Fiore 和 Hur [10] 提出的二阶抽象语法、二阶等式逻辑和二阶项重写。

2.1 二阶项

我们先回顾二阶签名这一概念，它本质上包含对象语言语法构造（可能带有变量绑定）的信息。

▶ 备注 1. 对于含有约束变量的语法，至少有两种细微不同的看法。具体来说，项 $\lambda x.t$ 可以看作
1.引入变量的绑定子 λx ，以及项 x，它处于扩展上下文中；在统一语法中，我们于是用一族函数符号表示 λ-抽象$\mathsf{abs}_x(t)$；

2. 将符号 λ 应用于一个 带作用域的项 $x.t$ ，其中 $(x.)$ 引入一个新变量 x ，其中于是可以在项 t 中使用；在统一语法中，我们于是用 λ-抽象表示单个函数符号$\mathsf{abs}(x.t)$ 它以一个带作用域的项作为参数。
我们遵循 Fiore 和 Hur 的做法，采用第二种解释。

二阶签名 [10, Section 2] $\Sigma = (T, O, |-|)$ 由一组类型T、一组算子[1] O以及一个元数[2] 函数 $|-|\colon O \to (T^* \times T)^* \times T$指定。对于算子 $\mathrm{F} \in O$，我们写作 $\mathrm{F} : (\sigma_1.\tau_1, \ldots, \sigma_n.\tau_n) \to \tau$，其中 $|\mathrm{F}| = ((\sigma_1, \tau_1) \ldots, (\sigma_n, \tau_n)\ \tau)$。直观而言，这意味着算子 F 接受 n 个参数，每个参数都在类型为 σ_i,1, . . . ,σ_i 的变量 $n_i = |\sigma_i|$ 的作用域内，且该项的类型为 τ_i。

除非另有说明，本文余下部分均假设存在一个环境签名 Σ。

[1] 在关于 E-统一的文献中，作者使用术语函数符号。[2] 我们遵循 Fiore 和 Hur 的术语用法。

$$\frac{x : \tau \in \Gamma}{\Theta \mid \Gamma \vdash x : \tau}\ \text{variables} \qquad \frac{\mathrm{M} : [\sigma_1, \ldots, \sigma_n]\tau \in \Theta \quad \text{for all } i = 1, \ldots, n \quad \Theta \mid \Gamma \vdash t_i : \sigma_i}{\Theta \mid \Gamma \vdash \mathrm{M}[t_1, \ldots, t_n] : \tau}\ \text{metavariables}$$

$$\frac{\mathsf{F} : (\overline{\sigma_1}.\tau_1, \ldots, \overline{\sigma_n}.\tau_n) \to \tau \quad \text{for all } i = 1, \ldots, n \quad \Theta \mid \Gamma, \overline{x_i} : \overline{\sigma_i} \vdash t_i : \tau_i}{\Theta \mid \Gamma \vdash \mathsf{F}(\overline{x_1}.t_1, \ldots, \overline{x_n}.t_n) : \tau}\ \text{operators}$$

图1 上下文中的二阶项。

一个类型上下文 [10, 节 2] $\Theta \mid \Gamma$ 由元变量类型声明 Θ 和变量类型声明 Γ组成。元变量类型声明是参数化类型：类型为 $[\sigma_1, \ldots, \sigma_n]\tau$的元变量，若以类型为 σ_1、……、 σ_n的项作为参数，则会产生一个类型为 τ的项。对于空（元）变量上下文，我们用居中点（·）表示。

例如，此上下文包含一个元变量 M ，它有两个参数和变量 x， y: $\Theta \mid \Gamma = (\mathrm{M} : [\sigma, \sigma \Rightarrow \tau]\tau \mid x : \sigma,\ y : \sigma \Rightarrow \tau)$。

▶ **定义2** ([10, 节 2])。 上下文 $\Theta \mid \Gamma \vdash - : \tau$ 中类型化项的判断式由图*1*中的规则定义。项上的变量替换按通常方式定义，详见 *[10,*节 *2]* 。

设 $\Theta = (\mathrm{M}_i : \overline{\sigma_i}]\tau_i)^{i \in \{1,\ldots,n\}}$，并考虑一个项 $\Theta \mid \Gamma \vdash t : \tau$，且对所有 $i \in \{1, \ldots, n\}$扩展[3]上下文中的一个项 $\Xi \mid \Gamma,\ \Delta,\ z_i : \overline{\sigma_i} \vdash t_i : \tau_i$。那么，**元变量替换** $t[\mathrm{M}_i[z_i] \overline{\mapsto} t_i]^{i \in \{1,\ldots,n\}}$ 按 t 的结构递归定义：

$$\begin{aligned}
x[\mathrm{M}_i[\overline{z_i}] \mapsto t_i]^{i \in \{1,\ldots,n\}} &= x \\
\mathrm{M}_k[\overline{s}][\mathrm{M}_i[\overline{z_i}] \mapsto t_i]^{i \in \{1,\ldots,n\}} &= t_k[\overline{z_k} \mapsto \overline{s[\mathrm{M}_i[\overline{z_i}] \mapsto t_i]^{i \in \{1,\ldots,n\}}}] \\
&\qquad \textit{when } k \in \{1, \ldots, n\} \textit{ and } |\overline{s}| = |\overline{z_k}| \\
\mathrm{N}[\overline{s}][\mathrm{M}_i[\overline{z_i}] \mapsto t_i]^{i \in \{1,\ldots,n\}} &= \mathrm{N}[\overline{s[\mathrm{M}_i[\overline{z_i}] \mapsto t_i]^{i \in \{1,\ldots,n\}}}] \qquad \textit{when } \mathrm{N} \notin \{\mathrm{M}_1, \ldots, \mathrm{M}_n\} \\
\mathsf{F}(\overline{x.s})[\mathrm{M}_i[\overline{z_i}] \mapsto t_i]^{i \in \{1,\ldots,n\}} &= \mathsf{F}(\overline{x.s[\mathrm{M}_i[\overline{z_i}] \mapsto t_i]^{i \in \{1,\ldots,n\}}})
\end{aligned}$$

我们将 $\theta : \Theta \mid \Gamma \to \Xi \mid \Gamma, \Delta$ 表示替换 $\theta = [\mathrm{M}_i[z_i] \mapsto t_i]^{i \in \{1,\ldots,n\}}$。
当 Γ 和 Δ 都为空时，我们将 $\theta : \Theta \to \Xi$ 作为 $\theta : \Theta \mid \cdot \to \Xi \mid \cdot$的简写。
在较大的上下文中进行单个元变量替换时，我们将省略映射到自身的元变量。也就是说，我们写作 $[\mathrm{M}_k[z] \mapsto t_k] : \Theta \mid \Gamma \to \Xi \mid \Gamma, \overline{\Delta}$ 表示$t_i = \mathrm{M}_i[z_i]$ 对所有 $i \neq k$都成立。

▶ **例3**。 以下是二阶语法中 λ-项的一些示例：

- 恒等函数 ($\lambda x.x$): abs(x.x) 丘奇数 $c_3 = \lambda s.\lambda z.s\ (s\ (s\ z))$: abs($s$.abs($z$.app($s$, app($s$,
- app(s, z))))) 交换函数 ($\lambda p.\pi_2\ p, \pi_1\ p$): abs($p$.pair(snd($p$), fst($p$))) 交换函数的应用 ((
- $\lambda p.\pi_2\ p, \pi_1\ p)\ a, b$): app(abs($p$.pair(snd($p$), fst($p$))), pair($a$, b))

[3] 这里我们对 Fiore 和 Hur 的定义稍作推广，允许在所得项中将上下文任意扩展为 Γ, Δ 。特别是在 Γ 为空时，这样做很有用。参见定义 29。

$$\frac{(\Theta \mid \Gamma \vdash s \equiv t : \tau) \in E}{\Theta \mid \Gamma \vdash s \equiv_E t : \tau}\ \text{axiom} \qquad \frac{\Theta \mid \Gamma \vdash t : \tau}{\Theta \mid \Gamma \vdash t \equiv_E t : \tau}\ \text{refl}$$

$$\frac{\Theta \mid \Gamma \vdash s \equiv_E t : \tau}{\Theta \mid \Gamma \vdash t \equiv_E s : \tau}\ \text{sym} \qquad \frac{\Theta \mid \Gamma \vdash s \equiv_E t : \tau \qquad \Theta \mid \Gamma \vdash t \equiv_E u : \tau}{\Theta \mid \Gamma \vdash s \equiv_E u : \tau}\ \text{trans}$$

$$\frac{\begin{array}{c}\mathrm{M}_1 : [\overline{\sigma_1}]\tau_1, \ldots, \mathrm{M}_n : [\overline{\sigma_n}]\tau_n \mid \Gamma \vdash s \equiv_E t : \tau \\ \text{for all } i \in \{1, \ldots, n\} \quad \Theta \mid \Delta, \overline{x_i} : \overline{\sigma_i} \vdash s_i \equiv_E t_i : \tau_i\end{array}}{\Theta \mid \Gamma, \Delta \vdash s[\mathrm{M}_1[\overline{x_1}] \mapsto s_1, \ldots, \mathrm{M}_n[\overline{x_n}] \mapsto s_n] \equiv_E t[\mathrm{M}_1[\overline{x_1}] \mapsto t_1, \ldots, \mathrm{M}_n[\overline{x_n}] \mapsto t_n] : \tau}\ \text{subst}$$

图2 二阶等式逻辑的规则。

2.2 二阶等式逻辑

现在我们定义二阶等式演算表示和二阶逻辑的规则，遵循 Fiore 和 Hur [10, Section 5]。这为我们提供了工具，用于在二阶方程理论下进行推理，例如 λ-演算的 $\beta\eta$-等价。

一个等式表示 [10, Section 5]是由若干公理构成的集合，其中每条公理都是上下文中的一对项。

▶**示例 4**。简单类型 λ演算的项由一族算子生成（对所有 σ, τ）——$\mathsf{abs}^{\sigma,\tau}\colon \sigma.\tau \to (\sigma \Rightarrow \tau)$ 和 $\mathsf{app}^{\sigma,\tau}\colon (\sigma \Rightarrow \tau, \sigma) \to \tau$。简单类型 λ-演算的等式表示则由一族公理给出：

$$\mathrm{M} : [\sigma]\tau, \mathrm{N} : []\sigma \mid \cdot \vdash \mathsf{app}(\mathsf{abs}(x.\mathrm{M}[x]), \mathrm{N}[]) \equiv \mathrm{M}[\mathrm{N}[]] : \tau \qquad (\beta)$$

$$\mathrm{M} : []\sigma \Rightarrow \tau \mid \cdot \vdash \mathsf{abs}(x.\mathsf{app}(\mathrm{M}[], x)) \equiv \mathrm{M}[] : \sigma \Rightarrow \tau \qquad (\eta)$$

请注意，这里的类型不依赖于上下文，因此只允许对相同类型的项进行等式比较是合理的。这与依赖类型系统形成对比，在依赖类型系统中，项可以具有不同（但等价）的类型。

一个等式表示 E 生成一种二阶等式逻辑 [10, 图 2]。二阶等式逻辑的规则见图2。

Fiore 和 Hur 在他们的论文中指出，零参数的元变量等价于普通变量。事实上，我们可以对每个项 $\Theta \mid \Gamma \vdash t : \tau$ 进行参数化，Θ得到一个项 $\widehat{\Gamma} \mid , \vdash \widehat{t} \cdot \tau$：其中，对于 $\Gamma = (x_1 : \sigma_1, \ldots, x_n : \sigma_n)$，我们有

$$\widehat{\Gamma} = (\mathrm{X}_1 : []\sigma_1, \ldots, \mathrm{X}_n : []\sigma_n) \qquad \widehat{t} = t[x_1 \mapsto \mathrm{X}_1[], \ldots, x_n \mapsto \mathrm{X}_n[]]$$

将参数化应用于一个等式表示 E 会得到一组参数化方程 $\widehat{E}$。注意，以下说法等价：

$$\Theta \mid \Gamma \vdash s \equiv_E t : \tau \qquad \text{iff} \qquad \Theta, \widehat{\Gamma} \mid \cdot \vdash \widehat{s} \equiv_{\widehat{E}} \widehat{t} : \tau$$

因此，从现在起，我们假设公理具有空变量上下文。

2.3 二阶项重写

最后，对于第 5 节中的完备性证明，使用项重写链而非模 E 等式的推导树会很有帮助。Fiore 和 Hur 在 二阶项重写 关系 [10, 第 8] 节中引入了这一概念。

$$\frac{(\mathrm{M}_1 : [\overline{\sigma_1}]\tau_1, \ldots, \mathrm{M}_k : [\overline{\sigma_k}]\tau_k \mid \cdot \vdash l \equiv r : \tau) \in E \qquad \Theta \mid \Gamma, \overline{x_i} : \overline{\sigma_i} \vdash t_i : \tau_i : \quad \text{for } i \in \{1, \ldots, k\}}{\Theta \mid \Gamma \vdash l[\mathrm{M}_i[\overline{z_i}] \mapsto t_i]^{i \in \{1,\ldots,k\}} \longrightarrow r[\mathrm{M}_i[\overline{z_i}] \mapsto t_i]^{i \in \{1,\ldots,k\}} : \tau}$$

$$\frac{\mathrm{M} : [\overline{\sigma}]\tau \qquad \Theta \mid \Gamma \vdash s_i \longrightarrow t_i : \sigma_i}{\Theta \mid \Gamma \vdash \mathrm{M}[\ldots, s_i, \ldots] \longrightarrow \mathrm{M}[\ldots, t_i, \ldots] : \tau}$$

$$\frac{\mathsf{F} : (\overline{\sigma_1}.\tau_1, \ldots, \overline{\sigma_n}.\tau_n) \to \tau \qquad \Theta \mid \Gamma, \overline{x_i} : \overline{\sigma_i} \vdash s_i \longrightarrow t_i : \tau_i}{\Theta \mid \Gamma \vdash \mathsf{F}(\ldots, \overline{x_i}.s_i, \ldots) \longrightarrow \mathsf{F}(\ldots, \overline{x_i}.t_i, \ldots) : \tau}$$

图 3 二阶项重写规则（简化版）。在第二条和第三条规则中，(...) 下的子项保持不变，因此每条规则仅重写一个子项。

一个等式表示 E 生成一个二阶项重写关系 $\longrightarrow_E$ [10, 图 4]。我们记作 $s \xrightarrow{*}_E t$，如果存在项序列 $u_1, \ldots, u_n$ 使得 $s = u_1 \longrightarrow_E \ldots \longrightarrow_E u_n = t$。我们记作 $s \longleftrightarrow_E t$，如果 $s \longrightarrow_E t$ 或 $t \longrightarrow_E s$。我们记作 $s \xleftrightarrow{*}_E t$，如果存在项序列 $u_1, \ldots, u_n$ 使得 $s = u_1 \longleftrightarrow_E \ldots \longleftrightarrow_E u_n = t$。

由于我们只关心公理中对元变量的替换（变量上下文为空），因此图 3 给出了这些规则的简化版本。

Fiore 和 Hur 的一项重要结果是二阶项重写的可靠性与完备性 [10, 节 8]：

$\Theta \mid \Gamma \vdash s \equiv_E t : \tau$ 当且仅当 $\Theta \mid \Gamma \vdash s \xleftrightarrow{*}_E t : \tau$。

2.4 示例：$\lambda\mu$-演算

Parigot 的 $\lambda\mu$-演算 [27] 具有 μ-抽象及以下归约规则：

$$(\mu\beta.u)\ v \longrightarrow \mu\beta.u[[\beta]w \mapsto [\beta](w\ v)] \qquad \text{（结构化约简）}$$

这条特定规则很有意思，因为它包含一个非平凡的绑定子，而且右侧的替换要求将 u 中每个形如 $[\beta]w$、且包含子项 $[\beta](w\ v)$ 的子项替换掉。这种替换可以内化为显式替换算子，也可以使用 λ-抽象和传统替换来建模（见示例 5）。

下面我们用二阶抽象语法来表示无类型的 $\lambda\mu$-演算：

▶ **示例 5**。对于无类型的 $\lambda\mu$-演算，并采用名称的显式替换算子，其签名由以下部分组成：

1. 一组类型 $T = \{\star, \text{名称}\}$2.
四个算子组成的集合：

$\mathsf{abs} : \star.\star \to \star$	（λ-抽象：$\lambda x.t$）
$\mathsf{app} : (\star, \star) \to \star$	（application：$t_1\ t_2$）
$\mathsf{mu} : \mathsf{Name}.\star \to \star$	（μ-抽象：$\mu\alpha.t$）
$\mathsf{named} : (\mathsf{Name}, \star) \to \star$	（命名项：$[\alpha]t$）
$\mathsf{subst} : (\mathsf{Name}.\star, \star) \to \star$	（名称替换：$[[\alpha]w \mapsto [\alpha](w\ v)]t$）

以下等式分别对应于结构化约简和.....的定义

名称替换：

$$
\begin{aligned}
\mathrm{M}_1 : [\mathsf{Name}]\star, \mathrm{M}_2 : \star \mid \cdot \vdash \mathsf{app}(\mathsf{mu}(\alpha.\mathrm{M}_1[\alpha]), \mathrm{M}_2[]) &\equiv \mathsf{mu}(\alpha.\mathsf{subst}(\alpha.\mathrm{M}_1[\alpha], \mathrm{M}_2[])) \\
\mathrm{M} : \star \mid x : \star \vdash \mathsf{subst}(\alpha.x, \mathrm{M}[]) &\equiv x \\
\mathrm{M}_1 : [\star]\star, \mathrm{M}_2 : \star \mid \cdot \vdash \mathsf{subst}(\alpha.\mathsf{abs}(x.\mathrm{M}_1[x]), \mathrm{M}_2[]) &\equiv \mathsf{abs}(x.\mathsf{subst}(\alpha.\mathrm{M}_1[x], \mathrm{M}_2[])) \\
\mathrm{M}_1, \mathrm{M}_2, \mathrm{M}_3 : \star \mid \cdot \vdash \mathsf{subst}(\alpha.\mathsf{app}(\mathrm{M}_1[], \mathrm{M}_2[]), \mathrm{M}_3[]) &\equiv \mathsf{app}(\mathsf{subst}(\alpha.\mathrm{M}_1[], \mathrm{M}_3[]), \mathsf{subst}(\alpha.\mathrm{M}_2[], \mathrm{M}_3[])) \\
\mathrm{M}_1, \mathrm{M}_2 : \star \mid \cdot \vdash \mathsf{subst}(\alpha.\mathsf{named}(\alpha, \mathrm{M}_1[]), \mathrm{M}_2[]) &\equiv \mathsf{named}(\alpha, \mathsf{app}(\mathsf{subst}(\alpha.\mathrm{M}_1[], \mathrm{M}_2[]), \mathrm{M}_2[])) \\
\mathrm{M}_1, \mathrm{M}_2 : \star \mid \beta : \mathsf{Name} \vdash \mathsf{subst}(\alpha.\mathsf{named}(\beta, \mathrm{M}_1[]), \mathrm{M}_2[]) &\equiv \mathsf{named}(\beta, \mathsf{subst}(\alpha.\mathrm{M}_1[], \mathrm{M}_2[])) \\
\mathrm{M}_1, \mathrm{M}_2 : \star \mid \cdot \vdash \mathsf{subst}(\alpha.\mathsf{mu}(\beta.\mathrm{M}_1[\beta]), \mathrm{M}_2[]) &\equiv \mathsf{mu}(\beta.\mathsf{subst}(\alpha.\mathrm{M}_1[\beta], \mathrm{M}_2[]))
\end{aligned}
$$

3 带二阶方程的 E-合一

本节将为二阶抽象语法表述方程统一问题，说明这类问题的解是什么，以及其是否完备。我们还将指出一类处于已求解形式的问题，即能够立即求解的问题。在很大程度上，这只是对标准 E-合一概念的直接推广 [11]。

▶ **定义 6**。一个**二阶约束** $\Theta \mid \Gamma_\exists,\ \Gamma_\forall \vdash s ?= t : \tau$ 是在一个上下文中的一对项，其中变量上下文分为两个部分：$\Gamma = (\Gamma_\exists,\ \Gamma_\forall)$。

其思想是，$\Gamma_\exists$ 包含我们需要求解的变量（自由变量），而 $\Gamma_\forall$ 包含我们不能替换的变量（约束变量）。元变量总是按存在量词处理，因此我们不拆分元变量上下文。与方程表示法类似，我们可以对约束（集合）进行参数化，得到 $\Theta,\ \widehat{\Gamma}_\exists \mid \Gamma_\forall \vdash s ?= t : \tau$。因此，从现在起，我们假设所有约束均满足 $\Gamma = \Gamma_\forall$（即 $\Gamma_\exists$ 为空）。

▶ **例 7.** 假设 $\alpha = \sigma \Rightarrow \tau,\ \beta = (\sigma \Rightarrow \tau) \Rightarrow \tau$。以下各项等价：
1. 对所有 $g : \alpha, a : \sigma$，求 $m : \alpha \Rightarrow \beta \Rightarrow \tau$，使得 $m\ g\ (\lambda z.z\ a) = g\ a$。2. $\cdot \mid m : \alpha \Rightarrow \beta \Rightarrow \tau, g : \alpha, a : \sigma \vdash \mathsf{app}(\mathsf{app}(m, g), \mathsf{abs}(z.\mathsf{app}(z, a))) ? = \mathsf{app}(g, a) : \tau$3. $\mathrm{m} : []\alpha \Rightarrow \beta \Rightarrow \tau \mid g : \alpha, a : \sigma \vdash \mathsf{app}(\mathsf{app}(\mathrm{m}[], g), \mathsf{abs}(z.\mathsf{app}(z, a))) ? = \mathsf{app}(g, a) : \tau$4. $\mathrm{m} : [\alpha, \beta]\tau \mid g : \alpha, a : \sigma \vdash \mathrm{m}[g, \mathsf{abs}(z.\mathsf{app}(z, a))] ?= \mathsf{app}(g, a) : \tau$

这里，条目 2 是将条目 1 直接编码为二阶约束。条目 3 是条目 2 的参数化版本。条目 4 在 β-等式意义下与条目 3 等价，这可由元替换 $[\mathrm{M}[] \mapsto \mathsf{abs}(x.\mathsf{abs}(y.\mathsf{app}(x, y)))]$ 和 $[\mathrm{M}[x, y] \mapsto \mathsf{app}(\mathsf{app}(\mathrm{M}[], x)\ y)]$见证。

▶ **定义 8.** 给定一个等式表示 E，**E-合一问题** $\langle \Theta, S \rangle$ 是共享元变量上下文 Θ 中二阶约束的有限集 S。我们将一个E-合一问题表示为如下形式的公式：

$$\exists(\mathrm{M}_1 : [\overline{\sigma_1}]\tau_1, \ldots, \mathrm{M}_n : [\overline{\sigma_n}]\tau_n).(\forall(\overline{z_1} : \overline{\rho_1}).s_1 \stackrel{?}{=} t_1 : \tau_1) \wedge \ldots \wedge (\forall(\overline{z_k} : \overline{\rho_k}).s_k \stackrel{?}{=} t_k : \tau_k)$$

下面是一个针对简单类型 λ-项的 E-合一问题示例：

$$\exists(\mathrm{m} : [\sigma \Rightarrow \tau, (\sigma \Rightarrow \tau) \Rightarrow \tau]\tau).\ \forall(g : \sigma \Rightarrow \tau, y : \sigma).\ \mathrm{m}[g, \mathsf{abs}(x.\mathsf{app}(x, y))] ?= \mathsf{app}(g, y) : \tau$$

▶**定义 9.** 元变量替换 $\xi :\ \Theta \to \Xi$ 称为 **E-合一器**，如果对于E-合一问题 $\langle \Theta, S \rangle$ 中的所有约束 $(\Theta \mid \Gamma_\forall \vdash s ?= t : \tau) \in S$，都有

$$\Xi \mid \Gamma_\forall \vdash \xi s \equiv_E \xi t : \tau$$

我们用 $U_E(S)$ 表示 $\langle \Theta, S \rangle$ 的所有 E-合一器的集合。

▶ **例10**。 考虑简单类型 λ-演算中的合一问题 $\langle \Theta, S\rangle$：

$$\Theta = \mathtt{M} : [\sigma \Rightarrow \tau, (\sigma \Rightarrow \tau) \Rightarrow \tau]\tau$$

$$S = \{\Theta \mid g : \sigma \Rightarrow \tau, y : \sigma \vdash \mathtt{M}[g, \mathsf{abs}(x.\mathsf{app}(x, y))] \stackrel{?}{=} \mathsf{app}(g, y) : \tau\}$$

替换 $[\mathtt{M}[z_1, z_2] \mapsto \mathsf{app}(z_2,\ z_1)]$： $\Theta \to \cdot$ 是 E-合一器，适用于 $\langle \Theta,\ S\rangle$。

3.1 已求解形式中的合一问题

在此，我们识别出一类平凡的合一问题。其思路是，看起来像元变量替换的约束可以被唯一地合一。只要约束的替换彼此足够不相交，合一问题就可以合一。更准确地说：

▶ **定义11.** 一个 E-合一问题 $\langle \Theta,\ S\rangle$ 处于 **已求解形式** 当 S 仅由以下形式的约束组成 $\Theta, \mathtt{M} : [\sigma]\tau \mid \Gamma_\forall \vdash \mathtt{M}[z] \stackrel{?}{=} t : \tau$ 且满足
1. $\vec{z} : \overline{\sigma} \subseteq \Gamma_\forall$ ($\mathtt{M}$ 的参数是 彼此不同的 变量，且属于 $\Gamma_\forall$)**2.** $\Theta \mid z : \sigma \vdash t : \tau$ ($\mathtt{M}$ 以及未出现在 z 中的变量均不出现在 t)**3.** 所有约束左侧的元变量都互不相同

▶ **例12**。 设 $\Theta = (\mathtt{M} : [\sigma, \sigma]\sigma)$。那么
1. $\{\Theta \mid x : \sigma, y : \sigma \vdash \mathtt{m}[y, x] \stackrel{?}{=} \mathsf{app}(\mathsf{abs}(z.x)\ y) : \sigma\}$ 处于已求解形式；2. $\{\Theta \mid x : \sigma, y : \sigma \vdash \mathtt{m}[x, x] \stackrel{?}{=} \mathsf{app}(\mathsf{abs}(z.x)\ y) : \sigma\}$ 不处于已求解形式，因为 m 的参数不是互异的变量，而且变量 y 出现在右侧，却没有出现在 m 的参数中；3. $\{\Theta \mid f : \sigma \Rightarrow \sigma, y : \sigma \vdash \mathtt{m}[y, \mathsf{app}(f, y)] \stackrel{?}{=} \mathsf{app}(f, y) : \sigma\}$ 不处于已求解形式，因为 m 的第二个参数不是变量；

▶ **命题 13**。 一个处于已求解形式的 E-合一问题 $\langle \Theta, S\rangle$ 有一个 E-合一器。

证明。假设 $\Theta = \{\Theta \mid \Gamma_i \vdash \mathtt{m}_i[z_i] \stackrel{?}{=} t_i : \tau_i\}^{i\in\{1,\ldots,n\}}$。令 $\xi_S = [\mathtt{m}_i[z_i] \mapsto t_i]^{i\in\{1,\ldots,n\}}$。注意，由于根据假设，每个 z_i 都是由互不相同的变量组成的序列，t_i 不引用其他变量或 $\mathtt{m}_i$，且每个元变量 $\mathtt{m}_i$ 在 ξ_S 中只被映射一次，因此 ξ_S 是一个形式良好的元替换。将 ξ_S 应用于每个约束后，我们得到平凡约束，它们由自反性满足：$\Theta \mid \Gamma_i \vdash t_i \equiv_E t_i : \tau_i$。因此，$\xi_S$ 是 Θ, S 的一个 E-合一子。◀

之后，我们将把命题 13 的证明中构造的 E-合一子称为 ξ_S。

3.2 比较 E-合一器

一般而言，一个合一问题可能有多个合一器。在这里，我们使用包含 序，将比较 E-合一子 [11] 的通常概念推广到二阶抽象语法，从而直接推广最一般合一器和完备合一器集的思想。我们不考虑推广本质合一子 [14, 33] 或同态嵌入 [32]，，尽管这些可能构成未来工作的方向。

▶ **定义 14.** 两个元变量替换 $\theta,\ \xi$： $\Theta \to \Xi$ 称为**模.....相等**E （记作 $\theta \equiv_E \xi$），如果对于所有元变量$\mathtt{M}$： $[\sigma]\tau \in \Theta$、任意上下文 Γ以及任意项$\Theta \vdash \Gamma \vdash t_i : \sigma_i$ （对于所有 $i \in \{1, \ldots, n\}$），均有

$$\Xi \mid \Gamma \vdash \theta\mathtt{M}[t_1, \ldots, t_n] \equiv_E \xi\mathtt{M}[t_1, \ldots, t_n] : \tau$$

我们称 θ 模 E **比** ξ 更一般（记作 $\theta \preccurlyeq_E \xi$），当且仅当存在一个替换 η： $\Xi \to \Xi$ 使得 $\eta \circ \theta \equiv_E \xi$。

空替换比任何替换都更一般。一个更有趣的例子可以在 λ-演算中找到。令

$$\theta_1 = [\mathrm{M}[x, y] \mapsto \mathsf{app}(\mathrm{N}[x], y)]$$
$$\theta_2 = [\mathrm{M}[x, y] \mapsto \mathsf{app}(\mathsf{abs}(z.x), y), \mathrm{N}[x] \mapsto \mathsf{abs}(z.x)]$$
$$\theta_3 = [\mathrm{M}[x, y] \mapsto x, \mathrm{N}[x] \mapsto \mathsf{abs}(z.x)]$$

那么，$\theta_1 \preccurlyeq_E \theta_2$，$\theta_2 \equiv_E \theta_3$，以及 $\theta_1 \preccurlyeq_E \theta_3$（由 $[\mathrm{N}[x] \mapsto \mathsf{abs}(z.x)] \circ \theta_1 \equiv_E \theta_3$见证）。

▶ **命题15**。若 $\theta \equiv_E \xi$ 则对于任意 E-合一问题$\langle \Theta,\ S\rangle$ 有 $\theta \in U_E(S)$当且仅当 $\xi \in U_E(S)$。

证明。对于每个约束 $\Theta \mid \Gamma_\forall \vdash s \stackrel{?}{=} t : \tau$，通过对 s和 t 的结构进行归纳，显然可得 $\Xi \mid \Gamma \vdash \theta s \equiv_E \theta t : \tau$ 当且仅当 $\Xi \mid \Gamma \vdash \xi s \equiv_E \xi t : \tau$。◀

▶ **推论16**。若 $\theta \preccurlyeq_E \xi$ 且 $\theta \in U_E(S)$ 则 $\xi \in U_E(S)$。

并非所有替换都可以比较。考虑无类型λ演算，其中 $\star$ 是任意项的类型。令 $\Theta = (\mathrm{M} : [\star, \star]\star)$ 和

$$\theta = [\mathrm{M}[z_1, z_2] \mapsto \mathsf{app}(z_2, z_1)]$$
$$\xi = [\mathrm{M}[z_1, z_2] \mapsto \mathsf{app}(z_1, \mathsf{app}(z_2, \mathsf{abs}(z.z)))]$$

这些替换都不比其他替换在无类型 λ- 演算的等式理论 E 下更一般。同时，二者都是该问题的 E-合一子

$$\exists(\mathrm{M} : [\star, \star]\star).\ \forall(g : \star, y : \star).\ \mathrm{M}[g, \mathsf{abs}(x.\mathsf{app}(x, y))] \stackrel{?}{=} \mathsf{app}(g, y) : \star$$

3.3 E-合一子的完备集

虽然一个 E-合一问题有时会有多个解，但我们通常可以期望将若干足够一般的合一子汇集到一个集合中：

▶ **定义17**。给定一个 E-合一问题 $\langle \Theta, S\rangle$，$\langle \Theta, S\rangle$ 的一个（极小）**完备E-合一子集**（记作 $\mathsf{CSU}_E(S)$）是 $U_E(S)$ 的一个子集，满足 **1.** *(完备性)* 对任意 $\eta \in U_E(S)$ 都存在 $\theta \in \mathsf{CSU}_E(S)$ 使得 $\theta \preccurlyeq_E \eta$；**2.** *(极小性)* 对任意 $\theta,\ \xi \in \mathsf{CSU}_E(S)$ 若 $\theta \preccurlyeq_E \xi$ 则 $\theta = \xi$。

我们用记号 $\mathsf{CSU}_E(S)$ 表示 E-合一子的极小完备集（即同时满足两个条件的集合）。

▶ **例18**。E-合一问题 $\langle \Theta,\ S\rangle$ 在无类型 λ-演算中具有无限的 $\mathsf{CSU}_E(S)$：

$$\langle \Theta, S\rangle = \exists(\mathrm{M} : [\star, \star]\star).\ \forall(g : \star, y : \star).\ \mathrm{M}[g, \mathsf{abs}(x.\mathsf{app}(x, y))] \stackrel{?}{=} \mathsf{app}(g, y) : \star$$
$$\mathsf{CSU}_E(S) = \{[\mathrm{M}[z_1, z_2] \mapsto \mathsf{app}(z_2, z_1)],$$
$$[\mathrm{M}[z_1, z_2] \mapsto \mathsf{app}(z_1, \mathsf{app}(z_2, \mathsf{abs}(x.x)))],$$
$$[\mathrm{M}[z_1, z_2] \mapsto \mathsf{app}(\mathsf{app}(z_2, \mathsf{abs}(x.\mathsf{abs}(f.\mathsf{app}(f, x)))), z_1)], \ldots\}$$

▶**命题19**。对于任意两个最小完备的 E-合一子集合 $\mathsf{CSU}^1_E(S)$ 和 $\mathsf{CSU}^2_E(S)$，存在一个双射 $f : \mathsf{CSU}^1_E(S) \longleftrightarrow \mathsf{CSU}^2_E(S)$ 使得

$$\forall \theta \in \mathsf{CSU}^1_E(S).\quad \theta \equiv_E f(\theta)$$

因此，$\mathsf{CSU}_E(S)$ 在模 E 意义下只在双射意义下唯一，所以从现在起我们将其称为该 E-合一子的完备集。

▶**定义20**。 当 E-合一子的完全集$\mathsf{CSU}_E(S)$ 是单元素集时，我们称其元素为 S 的**最一般 E-合一器** （记作$\mathsf{mgu}_E(S)$）。

▶ **例21**。 考虑简单类型 λ-演算中的这个 E-合一问题$\langle\Theta, S\rangle$：

$$\exists(\mathtt{M} : [\sigma \Rightarrow \tau, (\sigma \Rightarrow \tau) \Rightarrow \tau]\tau).\ \forall(g : \sigma \Rightarrow \tau, y : \sigma).\ \mathtt{M}[g, \mathsf{abs}(x.\mathsf{app}(x, y))] \stackrel{?}{=} \mathsf{app}(g, y) : \tau$$

对于这个问题，存在最一般 E-合一器： $\mathsf{mgu}_E(S) = [\mathsf{app}(z_2, z_1)]$。这个例子与例18不同，因为这里我们讨论的是简单类型 λ 演算。

▶ **命题22**。 如果 $\langle\Theta, S\rangle$ 是一个处于已求解形式的 E-合一问题，那么 $\mathsf{mgu}_E(S) \equiv_E \xi_S$。

证明。只需验证，对于任意 E-合一器 $\theta \in U_E(S)$，都有 $\xi_S \preccurlyeq_E \theta$。注意， $\theta \equiv_E \theta \circ \xi_S$，因为对于任意约束 $(\Theta \mid \Gamma_\forall \vdash M[z] ?\!= t : \tau) \in S$，使得 $\mathfrak{m} : [\sigma]\tau \in \Theta$，任意上下文 Γ，以及任意项 $\Theta \mid \Gamma \vdash t_i : \sigma_i$（对所有 $i \in \{1, \ldots, n\}$），都有

$$\Xi \mid \Gamma \vdash \theta\mathtt{M}[\overline{t}] \equiv_E \theta t[\overline{z} \mapsto \overline{t}] \equiv_E \theta(\xi_S\mathtt{M}[\overline{z}])[\overline{z} \mapsto \overline{t}] \equiv_E \theta(\xi_S\mathtt{M}[\overline{t}]) : \tau$$

◀

4 合一过程

本节将介绍一种合一过程，用于求解二阶抽象语法上的任意 E-合一问题。本节末尾将证明该过程是可靠的，而完备性结果则留待第 5 节讨论。

我们的合一过程借鉴了经典的 E-合一算法和高阶合一算法的特点。对于等式部分，我们借鉴了 Gallier 和 Snyder 关于一般（一阶） E-合一的完备变换集 [11]。对于元变量的合一，我们借鉴了 Huet 的高阶预合一 [15]，以及 Jensen 和 Pietrzykowski 的过程 [16]。Vukmirovic、Bentkamp 和 Nummelin 最近的工作 [35] 带来了一些关键见解，使我们得以进一步改进算法；不过，本文并不打算实现二阶抽象语法的高效 E-合一。

请注意，我们无法在过程中直接沿用高阶合一的思路，因为我们无法使用完整的λ-演算。相反，我们只有参数化元变量$\mathtt{M}[t_1, \ldots, t_n]$，它们类似于高阶合一中的变量应用（$m\ t_1\ \ldots\ t_n$）。不过，只要某些思路不依赖规范化或 λ-演算的特定语法，我们就可以对其加以改编。对于其他思路，我们则提出更简单但更通用的版本。这样或许能在保留完备性的同时牺牲一些效率，使搜索空间变得更大。虽然我们认为，可以优化这一过程，使其在处理 λ-演算中的合一问题时，运行时间几乎与上述高阶合一算法相同，但我们将此类优化留待未来工作。

为构建合一过程，我们遵循并推广了高阶合一和一阶 E-合一相关文献中的一些常见步骤：

1. 对某些类别的约束进行分类，找出可构成部分解的替换。其思路是，通过复合部分解得到整体解。

2. 定义转换规则，逐步逼近解。3. 确定何时停止（成功或失败）。4. 如果可能，按适当顺序组织规则，从而得到一个合一过程。

4.1 绑定

现在我们定义不同的基本替换，它们将作为合一过程中某些约束的部分解。这里，我们将 Vukmirovic、Bentkamp 和 Nummelin 收集的一组绑定 [35]加以推广。在该列表中，不使用 Huet 式投影（在高阶合一文献中也称为部分绑定 ）。相反，针对公理的模仿和 JP 式投影绑定涵盖了 Huet 式投影绑定[4]所能生成的所有替换。此外，我们在此还使用了简化版的迭代绑定，因为它与广义模仿绑定结合使用时，能够生成所有必要的绑定。

▶**定义23**。 我们定义以下几种绑定类型 ζ:

对 m 进行 *JP* 式投影。若 $m: [\sigma_1, \ldots, \sigma_k]\tau$且 $\sigma_i = \tau$，则$\zeta = [m[z] \mapsto z_i]$ 是一个 *JP* 式投影绑定。对 $\bar{m}$ 进行模仿。若 $m: [\sigma_1, \ldots, \sigma_k]\tau$，$F: (\alpha_1.\beta_1, \ldots, \alpha_n.\beta_n) \to \tau$且对于所有 i，$m_i: [\sigma_1, \ldots, \sigma_k, \alpha_i]\beta_i$，则$\zeta = [m[z] \mapsto F(x_1.m_1[z, x_1], \ldots, x_n.m_n[z, x_n])]$ 是一个模仿绑定。对 m 进行消去。若 $m: [\sigma_1, \ldots, \sigma_k]\tau$且 $1 \leq j_1 < j_2 < \ldots < j_{n-1} < j_n \leq k$，并且 $e: [\sigma_{j_1}, \ldots, \sigma_{j_n}]\tau$，则$\zeta = [m[z] \mapsto e[z_{j_1}, \ldots, z_{j_n}]]$ 是一个（参数）消去绑定。识别 m 和 n。若 $m: [\sigma_1, \ldots, \sigma_k]\tau$，$n: [\nu_1, \ldots, \nu_l]\tau$，$i: [\sigma_1, \ldots, \sigma_k, \nu_1, \ldots, \nu_l]\tau$，对于所有 $i \in \{1, \ldots, l\}$，$m_i: [\sigma_1, \ldots, \sigma_k]\nu_i$，并且对于所有 $j \in \{1, \ldots, k\}$，$n_j: [\nu_1, \ldots, \nu_l]\sigma_j$，则$\zeta = [m[z] \mapsto i[z, m_1[z], \ldots, m_l[z]], n[y] \mapsto i[n_1[y], \ldots, n_k[y], y]]$ 是一个识别绑定。对 m 进行迭代。若 $m: [\sigma_1, \ldots, \sigma_k]\tau$，$F: (\alpha_1.\beta_1, \ldots, \alpha_n.\beta_n) \to \gamma$，$h: [\sigma_1, \ldots, \sigma_k, \gamma]\tau$，且对于所有 i，$m_i: [\sigma_1, \ldots, \sigma_k, \alpha_i]\beta_i$，则$\zeta = [m[z] \mapsto h[z, F(x_1.m_1[z, x_1], \ldots, x_n.m_n[z, x_n])]]$ 是一个迭代绑定

迭代绑定允许以任意方式组合元变量的参数。这也尤其受到以下事实的影响：绑定中使用的类型 γ 可以是任意的。这类绑定会在下文的过程中引入任意分支，因此在实用实现中应谨慎使用。直观而言，我们强调迭代绑定的两种不同用例：

1. 通过应用公理，从一个或多个参数中提取新项。在这种情况下，我们使用迭代，其中公理一侧的根节点被用作算子 F。

2. 在作用域中引入新变量。在这种情况下，任何至少向作用域中引入一个变量的算子，都会在一次迭代中使用。这种用法对于该过程的完备性很重要。参见示例 35。

[4] 请注意，Huet 式投影无法用纯二阶抽象语法表述，因为它明确依赖于 abs 和 app。因此，在 E-合一中，我们只能通过某种形式的公理来恢复这类投影。Kudasov [20] 实现了一种启发式方法，其形式类似于 Huet 式投影的推广。更完善的推广留待未来工作。

4.2 转换规则

我们将合一过程的每条转换规则写成如下形式 $(\Theta \mid \Gamma_\forall \vdash s \stackrel{?}{=} t : \tau) \xrightarrow{\theta} \langle \Xi, S \rangle$，其中 θ：$\Theta \to \Xi$ 是一个元变量替换，而 S 是一个新约束集，旨在替换 $s \stackrel{?}{=} t$。当 Ξ 可从上下文中得知时，我们通常用 S 代替 $\langle \Xi, S \rangle$。

现在我们将依次介绍构成 E-合一过程的规则。前两条规则很简单。

▶**定义 24** (删除)。 如果一个约束的两边是同一个项，我们可以**删除** 它 t:

$$(\Theta \mid \Gamma_\forall \vdash t \stackrel{?}{=} t : \tau) \xrightarrow{\mathsf{id}} \varnothing$$

▶**定义 25** (分解)。 我们定义此规则的两个变体：

1. 令 $\mathsf{F} : (\sigma_1.\overline{\tau_1}, \ldots, \sigma_n.\overline{\tau_n}) \to \tau$，那么我们可以**分解**两边都有 F 的约束两侧分别转换为针对每一对（带作用域的）子项的一组约束：

$$(\Theta \mid \Gamma_\forall \vdash \mathsf{F}(\overline{x.t}) \stackrel{?}{=} \mathsf{F}(\overline{x.s}) : \tau) \xrightarrow{\mathsf{id}} \{\Theta \mid \Gamma_\forall, \overline{x_i : \sigma_i} \vdash t_i \stackrel{?}{=} s_i : \tau_i\}^{i \in \{1,\ldots,n\}}$$

2. 设 $\mathrm{M} : [\sigma_1, \ldots, \sigma_n]\tau$，那么我们可以将 **分解** 两侧均有 M 的约束，得到针对每一对参数的一组约束：

$$(\Theta \mid \Gamma_\forall \vdash \mathrm{M}[\overline{t}] \stackrel{?}{=} \mathrm{M}[\overline{s}] : \tau) \xrightarrow{\mathsf{id}} \{\Theta \mid \Gamma_\forall \vdash t_i \stackrel{?}{=} s_i : \sigma_i\}^{i \in \{1,\ldots,n\}}$$

▶**例 26.**

$$\begin{aligned}
&\Theta \mid \Gamma = (\mathrm{M} : [\sigma]\sigma \Rightarrow \sigma \mid f : \sigma \Rightarrow \sigma) \\
&\{\Theta \mid \Gamma \vdash \mathsf{abs}(x.\mathsf{app}(\mathrm{M}[x], x)) \stackrel{?}{=} \mathsf{abs}(x.\mathsf{app}(f, x))\} \\
\xrightarrow{\mathsf{id}} &\{\Theta \mid \Gamma, x : \sigma \vdash \mathsf{app}(\mathrm{M}[x], x)) \stackrel{?}{=} \mathsf{app}(f, x)\}\} && \text{(decompose)} \\
\xrightarrow{\mathsf{id}} &\{\Theta \mid \Gamma, x : \sigma \vdash \mathrm{M}[x] \stackrel{?}{=} f, \quad \Theta \mid \Gamma, x : \sigma \vdash x \stackrel{?}{=} x\} && \text{(decompose)} \\
\xrightarrow{\mathsf{id}} &\{\Theta \mid \Gamma, x : \sigma \vdash \mathrm{M}[x] \stackrel{?}{=} f\} && \text{(delete)}
\end{aligned}$$

接下来的两条规则是许多高阶合一算法中所用的模仿 和投影 规则的二阶版本。其思路是，元变量可以模仿约束的另一侧，也可以只投影其某个参数：

▶ **定义 27** (模仿)。 对于包含元变量 $\mathrm{M} : [\sigma_s]\tau$ 和算子 $\mathsf{F} : (\sigma_1.\tau_1, \ldots, \sigma_n.\tau_n) \to \tau$，我们可以使用模仿绑定来**模仿** 算子一侧（元变量 T 是新生成的）：

$$(\Theta \mid \Gamma_\forall \vdash \mathrm{M}[\overline{s}] \stackrel{?}{=} \mathsf{F}(\overline{x.t}) : \tau) \xrightarrow{[\mathrm{M}[\overline{z_s}] \mapsto \mathsf{F}(\overline{x.\mathrm{T}[\overline{z_s}, \overline{x}]})]} \{\Theta \mid \Gamma_\forall \vdash \mathsf{F}(\overline{x.\mathrm{T}[\overline{s}, \overline{x}]}) \stackrel{?}{=} \mathsf{F}(\overline{x.t}) : \tau\}$$

注意，**(模仿)** 之后可以应用**(分解)** 规则。

▶ **定义 28** (投影)。 对于含有元变量的约束 $\mathrm{M} : [\sigma_s]\tau$ 以及项 $u : \tau$，若 $\sigma_i = \tau$ 则我们可以为位置 i: 的参数生成一个 *JP* 风格的投影绑定

$$(\Theta \mid \Gamma_\forall \vdash \mathrm{M}[\overline{s}] \stackrel{?}{=} u : \tau) \xrightarrow{[\mathrm{M}[\overline{z}] \mapsto z_i]} \{\Theta \mid \Gamma_\forall \vdash s_i \stackrel{?}{=} u : \tau\}$$

下一条规则处理的是将约束的一侧与公理的一侧进行匹配。与公理匹配时，我们需要将其实例化 为具体的使用实例（毕竟，公理本身就是一种模式！）。然而，仅仅将公理中的元变量映射到对应类型的新元变量是不够的。由于我们要针对一个可能具有非空 $\Gamma_\forall$ 的特定约束来实例化公理，因此，将所有这些变量添加到每个新元变量中非常重要[5]:

▶ **定义 29.** 设 $\Gamma_\forall = (x : \alpha)$ 和 $\xi : \Xi \mid \cdot \to \Theta \mid \Gamma_\forall$.
我们称 ξ 将公理 $\Xi \mid \cdot \vdash l \equiv r : \tau$ 在上下文 $\Theta \mid \Gamma_\forall$ 中实例化，当且仅当：
1. 对任意 $(m_i : [\sigma]\overline{\tau}) \in \Xi$，$\xi$ 将 $m_i[t]$ 映射到 $n_i[\overline{t}, \overline{x}]$; *2.* $n_i = n_j$ 当且仅当 $i = j$ 对所有 i，j。

▶ **例 30.** 设 $\xi = [\mathrm{M}[z] \mapsto \mathrm{M}_1[z, g, y],\ \mathrm{N}[] \mapsto \mathrm{N}_1[g, y]]$。那么，$\xi$ 将公理实例化

$$\mathrm{M} : [\sigma]\tau, \mathrm{N} : []\sigma \mid \cdot \vdash \mathsf{app}(\mathsf{abs}(x.\mathrm{M}[x]), \mathrm{N}[]) \equiv \mathrm{M}[\mathrm{N}[]] : \tau$$

在上下文 $\mathrm{M}_1 : [\sigma, \sigma \Rightarrow \tau, \sigma]\tau$，$\mathrm{N}_1 : [\sigma \Rightarrow \tau, \sigma]\sigma \mid g : \sigma \Rightarrow \tau$，$y : \sigma$。

▶**定义 31 (变异).** 对于其中一侧与 E 中的公理相匹配的约束：

$$\Xi \mid \cdot \vdash l \equiv r : \tau$$

我们重写对应的一侧（此处，ξ 在上下文 $\Theta \mid \Gamma_\forall$中实例化公理）。

$$(\Theta \mid \Gamma_\forall \vdash t \stackrel{?}{=} s : \tau) \xrightarrow{\mathsf{id}} \{\Theta \mid \Gamma_\forall \vdash t \stackrel{?}{=} \xi l : \tau\} \uplus \{\Theta \mid \Gamma_\forall \vdash \xi r \stackrel{?}{=} s : \tau\}$$

通常，我们可以双向重写。不过，实际操作中，对某些公理（例如 $\beta\eta$-归约）选择单一方向可能更为妥当，而对其他公理（例如交换律和结合律公理）则保留双向重写。请注意，与先前的规则不同，**(变异)** 规则可能导致无限的转换序列。

其余规则处理两侧均含有元变量的约束。其中一条规则尝试统一不同的元变量：

▶**定义 32 (识别)。** 当一个约束由一对不同的元变量$\mathrm{M} : [\sigma_1, \ldots, \sigma_k]\tau$和 $\mathrm{N} : [\gamma_1, \ldots, \gamma_l]\tau$组成时，我们可以使用识别绑定（元变量$\mathrm{I}, \mathrm{M}', \mathrm{N}'$均为新鲜变量）：

$$(\Theta \mid \Gamma_\forall \vdash \mathrm{M}[\overline{s}] \stackrel{?}{=} \mathrm{N}[\overline{t}]) \xrightarrow{[\mathrm{M}[\overline{z}] \mapsto \mathrm{I}[\overline{z}, \overline{\mathrm{M}'[\overline{z}]}], \mathrm{N}[\overline{y}] \mapsto \mathrm{I}[\overline{\mathrm{N}'[\overline{y}]}, \overline{y}]]} \{\Theta \mid \Gamma_\forall \vdash \mathrm{I}[\overline{s}, \overline{\mathrm{M}'[\overline{s}]}] \stackrel{?}{=} \mathrm{I}[\overline{\mathrm{N}'[\overline{u}]}, \overline{u}]\}$$

另一条规则尝试统一具有不同参数列表的相同元变量：

▶ **定义 33** （消去）。 当一个约束的两侧具有相同的元变量 $\mathrm{M} : [\sigma_1, \ldots, \sigma_n]\tau$，并且存在一个序列 $(j_k)_{k=1}^n$ 使得 $s_{j_k} = t_{j_k}$ 对于所有 $k \in \{1, \ldots, n\}$，那么我们可以 **消去** 所有其他参数，并使剩余项保持一致（元变量 E是新鲜的）：

$$(\Theta \mid \Gamma_\forall \vdash \mathrm{M}[\overline{s}] \stackrel{?}{=} \mathrm{M}[\overline{t}]) \xrightarrow{[\mathrm{M}[\overline{z}] \mapsto \mathrm{E}[z_{j_1}, \ldots, z_{j_n}]]} \varnothing$$

[5] 这与基于一阶公理的 E-合一不同，在后者中，元变量不携带自己的上下文，因此之后可以与任意变量合一。[6] 我们会检查项的根是否匹配。从技术上说，我们不必进行此项检查，也不必对任何公理应用**(变异)** 规则（非确定性地），因为合一过程会执行完整匹配。

14 二阶抽象语法的 E-合一

最后一条规则的思想是，用已有参数的某种组合扩展参数列表。例如，考虑约束 $\forall x, y, z.\mathrm{M}[\text{有序对}(x, y)\ z] \stackrel{?}{=} \mathrm{N}[x, z]$。显然，如果我们可以处理 x 和 y 的有序对，那么也可以分别处理它们，因为我们可以使用第一投影提取 x，并使用第二投影提取 y。因此，替换 $[\mathrm{M}[p, z] \mapsto \mathrm{M}_1[p, z, \text{第一投影}(p)]$ 会产生一个新的约束 $\forall x, y, z.\mathrm{M}_1[\text{有序对}(x, y)\ z,\ \text{第一投影}(\text{有序对}(x, y))] \stackrel{?}{=} \mathrm{N}[x, z]$。现在可以通过应用（识别）、（消去）和（分解）规则来解决该约束，最终得到 $\forall x, y, z.\text{第一投影}(\text{有序对}(x, y)) \stackrel{?}{=} x : \sigma$，该约束将使用（变异）规则进行处理。

▶**定义 34**（迭代）。当一个约束由一对（可能相同的）元变量 $\mathrm{M} : [\sigma_1, \ldots, \sigma_k]\tau$ 和 $\mathrm{N} : [\gamma_1, \ldots, \gamma_l]\tau$ 组成时，可以使用迭代绑定（元变量 H、K 均为新变量）：

$$(\Theta \mid \Gamma_\forall \vdash \mathrm{M}[\overline{s}] \stackrel{?}{=} \mathrm{N}[\overline{t}]) \xrightarrow{[\mathrm{M}[\overline{z}] \mapsto \mathrm{H}[\overline{z}, \mathsf{F}(\overline{x}.\mathrm{K}[\overline{z}, \overline{x}])]]} \{\Theta \mid \Gamma_\forall \vdash \mathrm{H}[\overline{s}, \mathsf{F}(\overline{x}.\mathrm{K}[\overline{z}, \overline{x}])] \stackrel{?}{=} \mathrm{N}[\overline{t}]\}$$

以下示例展示了通过任意算子进行迭代、将变量引入作用域的重要性：

▶ **示例 35**。考虑一个简单类型 λ 演算的合一问题：

$$\begin{aligned}&\exists \mathrm{M} : [\sigma \Rightarrow \sigma \Rightarrow \tau](\sigma \Rightarrow \tau)\\ &\quad \forall f : \sigma \Rightarrow \sigma \Rightarrow \sigma \Rightarrow \sigma \Rightarrow \tau.\\ &\qquad \mathrm{M}[\lambda x.\lambda y.f\ x\ y\ x\ x] \stackrel{?}{=} \mathrm{M}[\lambda x.\lambda y.f\ y\ y\ x\ y] : \sigma \Rightarrow \tau\end{aligned}$$

它具有如下 E-合一器：$\zeta = [\mathrm{M}[g] \mapsto \lambda z.g\ z\ z]$。要从绑定构造这个合一器，我们从迭代绑定 $[\mathrm{M}[g] \mapsto \mathrm{I}[g, \lambda z.\mathrm{M}_1[g, z]]$ 开始，引入 λ 抽象，随后进行投影 $[\mathrm{I}[g, r] \mapsto r]$，接着再次迭代（以引入应用），如此继续。

最后，我们将所有转换规则编译为合一过程：

▶**定义 36**。*E*-**合一过程** 在等式表示 E 上的定义是：重复（非确定性地）应用以下转换，直到停止：
1. 如果不再有约束，则停止（成功）。*2.* 如果可能，应用（删除）规则。*3.* 如果可能，应用（变异）或（分解）规则（非确定性）。*4.* 如果存在一个由两个非元变量组成的约束，且上述转换均不适用，则停止（失败）。*5.* 如果存在形如 $m[\ldots] \stackrel{?}{=} F(\ldots)$ 的约束，则应用（模仿）或（投影）规则（非确定性）。*6.* 如果存在约束 $m[\ldots] \stackrel{?}{=} x$，则应用（投影）规则（非确定性）。*7.* 如果可能，应用（识别）、（消去）或（迭代）规则（非确定性）。*8.* 如果上述规则均不适用，则停止（失败）。

许多高阶合一算法 [23, 21] 都会实现一条规则（通常称为消去），用于在相应约束处于已求解形式时消去元变量。此处不需要这样的规则，因为**(模仿)**、**(分解)**、**(删除)**、**(识别)** 和**(消去)** 规则的组合已经涵盖了这种情况。不过，它可以简化示例的展示，也是一种实用的优化，因此我们将其作为一条可选规则纳入：

▶ **定义 37** (消去*)。 当约束 $C = (\Theta \mid \Gamma_\forall \vdash \mathrm{M}[z] \stackrel{?}{=} u)$ 处于已求解形式时，我们可以用相应的合一器 $\xi_{\{C\}} = [\mathrm{M}[z] \mapsto u]$将其消去：

$$(\Theta \mid \Gamma_\forall \vdash \mathrm{M}[\overline{s}] \stackrel{?}{=} u) \xrightarrow{[\mathrm{M}[\overline{z}] \mapsto u]} \varnothing$$

在该过程中，**消去*** 规则应与**删除**具有相同的优先级。

▶**引理 38.** 在定义 *36* 中定义的过程中，每一步都是可靠的。也就是说，如果$S \xrightarrow{\theta} S'$ 是该过程执行的单步转换，且 $\xi \in U_E(S')$ 那么 $\xi \circ \theta \in U_E(S)$。

证明。只需证明每一步相对于其所作用的约束都是可靠的。也就是说，我们考虑步骤 $\{C\} \xrightarrow{\theta} S''$，使得 $C \in S$ 且 $S'' \subseteq S'$。根据假设，$\xi \in U_E(S')$，因此 $\xi \in U_E(S'')$ 也成立。注意，对于任意约束 $D \in (S - \{C\})$，都存在相应的约束 $D' \in (S' - S'')$，使得 $D' = \theta D$。由于 ξ 合一 D'，可知 $\xi \circ \theta$ 合一 D。因此，我们只需证明 $\xi \circ \theta$ 合一 $U_E(\{C\})$。

现在我们来逐一检查可能的步骤列表：（删除）：显然，任何替换都能合一 C；（分解）：由于 ξ 能合一 S''中的所有子项对，因此它也能合一 C；（模仿）、（投影）、（识别）、（消去）、（迭代）：这些规则都只是决定如何替换某些元变量（选择 θ），并立即应用该替换。因此，$S'' = \{\theta C\}$，并且由于 ξ 能合一 S'，那么 $\xi \circ \theta$ 就能合一 S。（变异）：令 $C = (\Theta \mid \Gamma_\forall \vdash s \stackrel{?}{=} t : \tau)$，我们根据公理 $(\Xi \mid \cdot \vdash l \equiv r : \tau) \in E$ 进行变异，并由替换 ζ 实例化该公理。根据假设，ξ 能同时合一$s \stackrel{?}{=} \zeta l$ 和 $\zeta r \stackrel{?}{=} t$。此外，$\Theta \mid \Gamma_\forall \vdash \zeta l \equiv_E \zeta r : \tau$。在这条规则中，$\theta =$是恒等替换，因此我们可以证明 $\xi \circ \theta = \xi$ 能合一 $s \stackrel{?}{=} t$：$\xi s \equiv_E \xi(\zeta l) \equiv_E \xi(\zeta r) \equiv_E \xi t$

◀

▶ **定理 39.** 定义 *36* 中给出的过程是可靠的。也就是说，如果 $S \xrightarrow{\theta_1} S_1 \xrightarrow{\theta_2} \ldots \xrightarrow{\theta_n} \varnothing$ 是该过程生成的一条路径，那么 $\theta_1 \circ \theta_2 \circ \ldots \circ \theta_n \in U_E(S)$。

证明。 这是引理 38 的直接推论。 ◀

4.3 示例：带有序对的简单类型 λ-演算

这里我们给出二阶语法中简单类型 λ-演算的一个简单扩展：

▶ **例 40.** 带有序对的简单类型 λ-演算在一组基本类型 A 上的一个签名由以下部分组成：

1. 一组类型 $A^{\Rightarrow, \times}$ 定义如下

$$\frac{\tau \in A}{\tau \in A^{\Rightarrow,\times}} \qquad \frac{\sigma, \tau \in A^{\Rightarrow,\times}}{\sigma \Rightarrow \tau \in A^{\Rightarrow,\times}} \qquad \frac{\sigma, \tau \in A^{\Rightarrow,\times}}{\sigma \times \tau \in A^{\Rightarrow,\times}}$$

2. 适用于所有 σ，$\tau \in A^{\Rightarrow, \times}$

$$\begin{aligned}
\mathsf{abs}^{\sigma,\tau} &: \sigma.\tau \to (\sigma \Rightarrow \tau) && \text{(抽象)}\\
\mathsf{app}^{\sigma,\tau} &: (\sigma \Rightarrow \tau, \sigma) \to \tau && \text{(应用)}\\
\mathsf{pair}^{\sigma,\tau} &: (\sigma, \tau) \to \sigma \times \tau && \text{(序对构造子)}\\
\mathsf{fst}^{\sigma,\tau} &: \sigma \times \tau \to \sigma && \text{(第一投影)}\\
\mathsf{snd}^{\sigma,\tau} &: \sigma \times \tau \to \tau && \text{(第二投影)}
\end{aligned}$$

▶**例 41.** 考虑 λ-项 $(\lambda x.\mathrm{M}\langle f, x\rangle) \colon \sigma \Rightarrow \tau$。假设 $f \colon \sigma \Rightarrow \tau$。将此项表示为二阶语法时，我们有一个选择：

1. 如果我们将M 视为一个无参数元变量，那么

$$\mathrm{M} : []((\sigma \Rightarrow \tau) \times \sigma) \Rightarrow \tau \mid f : \sigma \Rightarrow \tau \vdash \mathsf{abs}(x.\mathsf{app}(\mathrm{M}[], \mathsf{pair}(f, x))) : \sigma \Rightarrow \tau$$

2. 如果我们将 M 视为一个带一个参数的元变量，那么

$$\mathrm{M} : [(\sigma \Rightarrow \tau) \times \sigma]\tau \mid f : \sigma \Rightarrow \tau \vdash \mathsf{abs}(x.\mathrm{M}[\mathsf{pair}(f, x)]) : \sigma \Rightarrow \tau$$

这些表示是等价的，接下来的各节将给出这种等价性的形式概念。

▶ **示例 42。** 带有序对的简单类型化 λ-演算的等式表示：

$$\mathrm{M} : [\sigma]\tau, \mathrm{N} : []\sigma \mid \cdot \vdash \mathsf{app}(\mathsf{abs}(x.\mathrm{M}[x]), \mathrm{N}[]) \equiv \mathrm{M}[\mathrm{N}[]] : \tau \qquad (\beta_\lambda)$$

$$\mathrm{M} : []\sigma,\ \mathrm{N} : []\tau \mid \cdot \vdash \text{第一投影（有序对（}\mathrm{M}[], \mathrm{N}[]\text{））} \equiv \mathrm{M}[] : \sigma \qquad (\beta_{\pi_1})$$

$$\mathrm{M} : []\sigma,\ \mathrm{N} : []\tau \mid \cdot \vdash \text{第二投影（有序对（}\mathrm{M}[], \mathrm{N}[]\text{））} \equiv \mathrm{N}[] : \tau \qquad (\beta_{\pi_2})$$

$$\mathrm{M} : []\sigma \Rightarrow \tau \mid \cdot \vdash \mathsf{abs}(x.\mathsf{app}(\mathrm{M}[], x)) \equiv \mathrm{M}[] : \sigma \Rightarrow \tau \qquad (\eta_\lambda)$$

$$\mathrm{M} : []\sigma \times \tau \mid \cdot \vdash \mathsf{pair}(\mathsf{fst}(\mathrm{M}[]), \mathsf{snd}(\mathrm{M}[])) \equiv \mathrm{M}[] : \sigma \times \tau \qquad (\eta_\times)$$

有关该问题的一个搜索路径示例，请参见图 4；该搜索路径由针对以下简单类型 λ 演算的问题所生成的 E-合一过程产生：

$$\exists \mathrm{M}\colon\ [] \left((\sigma \Rightarrow \tau) \times \left((\sigma \Rightarrow \tau) \Rightarrow \tau \right) \right) \Rightarrow \tau) 。\ (\forall g\colon\ \sigma \Rightarrow \tau,\ y\colon\ \sigma.\mathrm{app}\left(\mathrm{m}[], \text{有序对}\left(g,\ \mathrm{abs}\left(x.\mathrm{app}\left(x,\ y\right)\right)\right)\right) ? = \mathrm{app}\left(g,\ y\right) :\ \tau\ )$$

5 完备性证明

本节将证明我们的主定理，说明我们的合一过程是完备的。

我们先定义混合算子：

▶**定义 43.** 称算子 $\mathsf{F} : (\alpha_1.\beta_1, \ldots, \alpha_n.\beta_n) \to \gamma$ 为 **混合算子** 当且仅当 α_i 为空且 α_j 对某个 i 和 j不为空。

处理混合算子可能非常棘手。在下面的定理中，我们假设所有算子要么在所有子项中引入作用域，要么在任何子项中都不引入作用域。也就是说，对于每个算子 $\mathsf{F} : (\alpha_1.\beta_1, \ldots, \alpha_n.\beta_n) \to \gamma$，要么 $|\alpha_i| = 0$对所有 i成立，要么 $|\alpha_i| > 0$对所有 i成立。这个假设是合理的，因为我们总能将混合算子编码为若干非混合算子的组合。例如，令$(t_1, x.t_2)$ 可以编码为 令$(t_1$,块$(x.t_2))$。

▶**定理 44.** 假设未使用混合算子，则定义 *36* 中描述的过程是完备的，也就是说，由该过程构造的搜索树中，从根节点到所有 ***(成功)*** 叶节点的路径构成一个完备（但不一定是最小）的 E-合一器集合。更具体地说，设 E 是一个等式表示，且 $\langle \Theta, S\rangle$ 是一个 E-合一问题。那么对于任意 E-合一器 $\theta \in U_E(S)$，存在一条路径 $S \xrightarrow{\xi} \varnothing$ 使得 $\xi \preccurlyeq_E \theta$。

▶ 注释 45。合一过程可能会产生冗余合一子。例如，考虑简单类型 λ-演算中的以下合一问题：

$$\exists \mathrm{M} : [\sigma, \sigma]\tau, \mathrm{N} : [\sigma](\sigma \Rightarrow \tau).\ \forall x : \sigma.\ \mathrm{M}[x, x] \stackrel{?}{=} \mathsf{app}(\mathrm{N}[x], x)$$

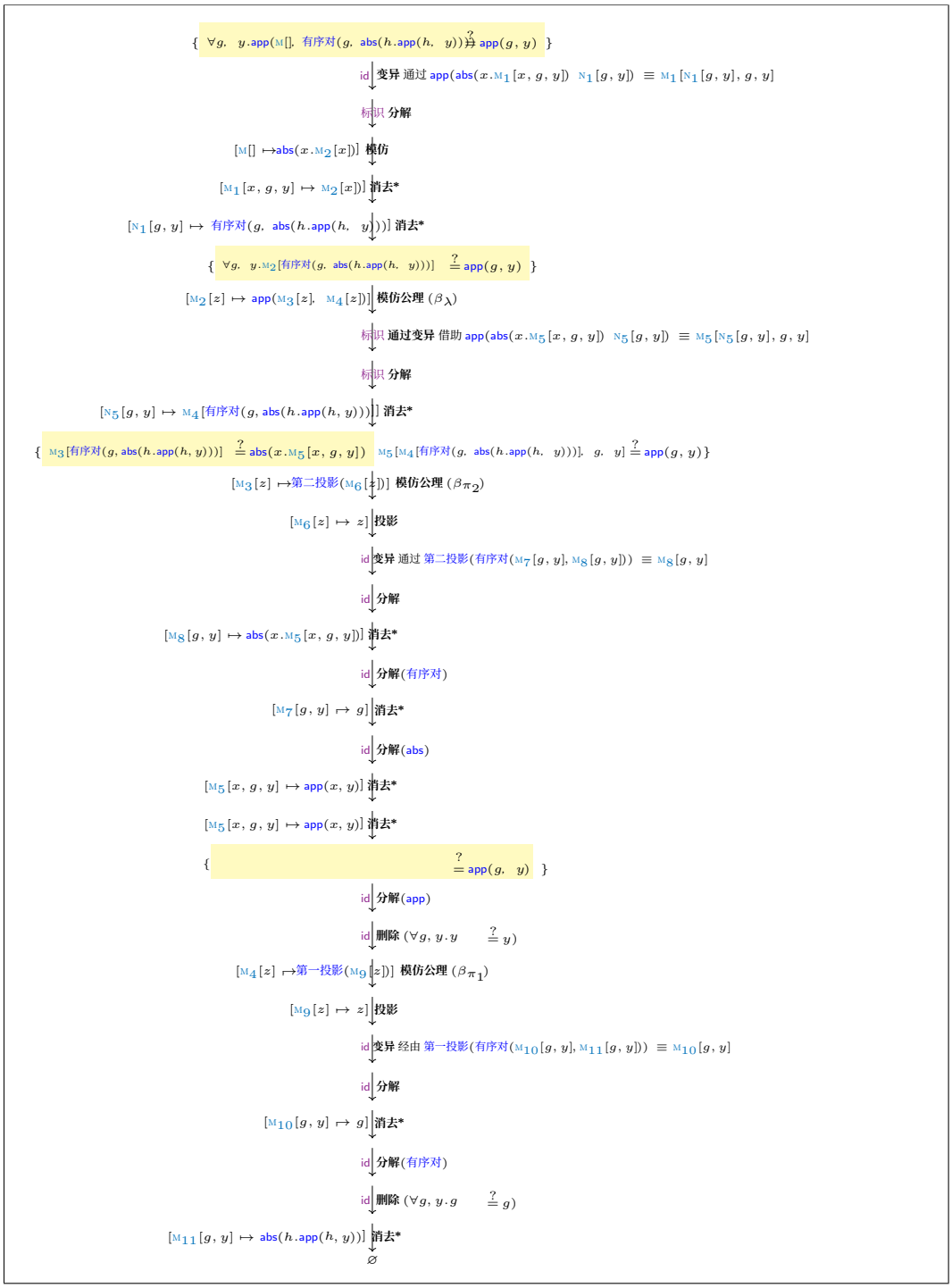

图 4 在 λ-演算中的 E-合一问题示例搜索路径，展示了许多规则：$\exists \mathrm{m} : ((\sigma \Rightarrow \tau) \times \sigma) \Rightarrow \tau.\ \forall g\colon \sigma \Rightarrow \tau, y\colon \sigma.\mathrm{m}\ g, \lambda h.h\ y \stackrel{?}{=} g\ y\colon \tau$。为节省篇幅，省略了类型。代换复合得到 $[\mathrm{m}[] \mapsto \mathrm{abs}(x.\mathrm{app}(\text{第二投影}(x), \text{第一投影}(x)))\ \ldots]$。

18 二阶抽象语法的 E-合一

根据我们先应用（**模仿**）规则还是（**变异**）规则，我们可以得到以下合一子：

$$\theta_1 = [\mathrm{M}[z_1, z_2] \mapsto \mathsf{app}(\mathrm{N}[z_1], z_2)]$$
$$\theta_2 = [\mathrm{M}[z_1, z_2] \mapsto \mathrm{T}[z_2, z_1],\ \mathrm{N}[z_1] \mapsto \mathsf{abs}(z.\mathrm{T}[z_1, z])]$$

显然， $\theta_1 \neq \theta_2$，但 $\theta_1 \preccurlyeq_E \theta_2$ （由 $[\mathrm{N}[z_1] \mapsto \mathsf{abs}\ (z.\mathrm{T}[z_1, z])$]见证）。因此，对于这个合一问题得到的 E-合一子集并非最小（根据定义17）。

我们的证明本质上结合了两种方法：一种是 Gallier 和 Snyder 在证明一般（一阶） E-合一 [11], 完备性时采用的方法，另一种是 Jensen 和 Pietrzykowski (JP) [16], 提出、后由 Vukmirovic、Bentkamp 和 Nummelin (VBN) [35]改进的完全高阶合一方法。特别是，处理参数化元变量时，我们需要重用后一种方法中的一些思路。然而，我们无法重用 JP 的 ω-简单性或 VBN 的基底简单性，因为它们关键依赖于 η长项、 λ抽象和函数应用，而这些在一般的二阶抽象语法中并不可用。相反，我们重用 Gallier 和 Snyder 的思路，以判断何时可以分解项。为了弄清何时应用（**迭代**）规则，我们也考察重写序列，而不是项的 ω-简单性。

证明的主要思路是，取合一问题 S 及其 E-合一器θ ，然后根据 θ选择过程中的一条规则。应用规则会更新约束，剩余的替换也会随之更新。为证明这一过程会终止，我们引入一个度量，它会随着每次规则应用而严格递减。

▶ **定义 46**。 设 $\theta \in U_E(S)$。然后，将 **度量** 定义在有序对 $\langle S, \theta \rangle$ 上，作为以下各项的字典序比较：
1. 所有 $\theta s \xleftrightarrow{*}_E \theta t$ 的重写序列长度之和，其中每个 $s \stackrel{?}{=} t$ 都属于 S；
2. θ中所用算子的总数；**3.** θ中所用元变量的总数；**4.** S中各项大小之和。

我们将上述四元组记作 $\mathsf{ord}(S, \theta)$。

下面的定义有助于我们理解何时应应用**(投影)** 规则：

▶ **定义 47**。 元变量 $\mathrm{M} : [\sigma]\tau$ 在 **相对于** j **而言是投影的** θ ， 当且仅当 $\theta\mathrm{M}[z] = z_j$。

证明中的一个关键点是，能否对两侧都含有两个元变量的约束应用（**识别**）或（**消去**）规则。下面的引理给出了精确条件：当元变量不同时可以（**识别**），当它们相同时则可以（**消去**）。

▶ 引理 *48*。设 $s = \boldsymbol{m}[u]$ 和 $t = \boldsymbol{n}[v]$ 满足 $\zeta s \xleftrightarrow{*}_E \zeta t$。设 $s_1, \ldots, s_n$ 是 ζs的子项，$t_1, \ldots, t_n$ 是 ζt 的子项，且重写序列 $\zeta s \xleftrightarrow{*}_E \zeta t$对应于对所有 $i \in \{1, \ldots, n\}$的独立重写 $s_i \xleftrightarrow{*}_E t_i$ 的并集。若对所有 i ，我们都有 s_i 是 ζu_{j_i} 某个出现位置的子项，或者 t_i 是 ζv_{j_i}某个出现位置的子项，则存在项 w、 u'、v'，使得

1. $\overline{u'[y \mapsto \zeta v]} \equiv_E \overline{\zeta u}$ 和 $\overline{v'[z \mapsto \zeta u]} \equiv_E \overline{\zeta v}$；
2. $\zeta\mathrm{M}[z] = w[y \mapsto v']$ 和 $\zeta\mathrm{N}[y] = w[z \mapsto u']$。

证明。我们定义一个项的辅助族 $\Xi \mid x : \alpha \vdash w'(l, r) : \tau$，用于项的有序对$\Xi \mid x : \alpha, z : \gamma \vdash l : \tau$ 以及$\Xi \mid x : \alpha, y : \beta \vdash r : \tau$，使得$l$是 $\zeta\mathrm{m}[z]$的子项，且 r是 $\zeta\mathrm{n}[y]$ 的子项，并满足 $l[z \mapsto \zeta u] \equiv_E r[y \mapsto \zeta v]$。我们根据 l 和 r的结构归纳定义$w'(l, r)$，同时保持 $l \equiv_E w'(l, r)[y \mapsto v']$ 和 $r \equiv_E w'(l, r)[z \mapsto u']$:

1. 若 $l = x_i$ 或 $r = x_i$，则 $l = r$ 且 $w'(l, r) = x_i$；2. 若 $l = z_k$，则 $w'(l, r) = z_k$ 且 $u'_k = r$；3. 若 $r = y_k$，则 $w'(l, r) = y_k$ 且 $v'_k = l$；4. 若 $l = \mathrm{F}(a.p)$ 且 $r = \mathrm{F}(a.q)$，则 $w'(l, r) = \mathrm{F}(a.w'(p, q))$；注意，对于所有 i，$w'(p_i, q_i)$ 都有定义良好，因为 $l[z \mapsto \zeta u] \equiv_E r[y \mapsto \zeta v]$ 蕴含逐分量相等 $p_i[z \mapsto \zeta u] \equiv_E q_i[y \mapsto \zeta v]$，对所有 i 均成立。事实确实如此，因为否则我们就在根部同时重写 l 和 r，但 $l[z \mapsto \zeta u]$ 和 $r[y \mapsto \zeta v]$ 都不对应于项 ζs 或 ζt 中的参数出现 ζu_j 或 ζv_j。5. 若 $l = \mathrm{m}[p]$ 且 $r = \mathrm{m}[q]$，则 $w'(l, r) = \mathrm{m}[w'(p, q)]$；这里，与前一种情况类似，对于所有 i，$w'(p_i, q_i)$ 都有定义良好。

如果 u_k（或 v_k）未针对某个 k 定义，则意味着相应的参数并非本质参数，可以将其消去。我们令[7] 这样的 u_k 为一个新的元变量 $\mathrm{U}_k[]$。我们令 $w = w'(\zeta_{\mathrm{M}}[z], \zeta_{\mathrm{N}}[y])$。根据构造，$\zeta_{\mathrm{M}}[z] = w[y \mapsto v']$ 和 $\zeta_{\mathrm{N}}[y] = w[z \mapsto u']$。◀

▶ 推论 49。设 $s = m[u]$ 和 $t = m[v]$ 满足 $\zeta s \xleftrightarrow{*}_E \zeta t$。设 $s_1, \ldots, s_n$ 是 ζs 的子项，$t_1, \ldots, t_n$ 是 ζt 的子项，且重写序列 $\zeta s \xleftrightarrow{*}_E \zeta t$ 对应于所有 $i \in \{1, \ldots, n\}$ 的独立重写 $s_i \xleftrightarrow{*}_E t_i$ 的并集。若对所有 i，我们都有：要么 s_i 是 ζu_{j_i} 某个出现位置的子项，要么 t_i 是 ζv_{j_i} 某个出现位置的子项，那么存在一个序列 $1 \le j_1 < \ldots < j_k \le$，使得且 $FV(\zeta m[z]) = \{z_{j_1}, \ldots, z_{j_k}\}$，并且对所有 i 都有 $u_{j_i} \equiv_E v_{j_i}$。

下面的引理将帮助我们概括下述证明中条目 2(e)iii 的解。

▶ **引理 50**。设 $\Xi \mid x : \alpha \vdash w : \sigma$ 是 $\Xi \mid x : \alpha \vdash t : \tau$ 的一个子项。如果 t 不包含混合算子，则存在一个替换 $\zeta_{w,\,t} = [\mathrm{H}[z, y] \mapsto h]$ 以及一个项集 $\Xi \mid x : \alpha \vdash s : \beta$，使得每个 s_i 都是 t 的子项，并且 $\zeta_{w,\,t}\mathrm{H}[w, s] = t$。

证明。注意，w 和 s 是 t 的子项，并且不在绑定子之下（因为它们具有相同的变量上下文）。于是，对 t 的结构进行归纳：
1. 如果 $t = w$，则 $\zeta_{w,t} = [\mathrm{h}[z] \mapsto z]$；2. 如果 $t = \mathrm{F}(x_1.t_1, \ldots, x_n.t_n)$，那么由于 t 不包含混合算子，对于所有 i，x_i 都是空的。现在，如果 w 是 t_i 的子项，并且 $\zeta_{w,t_i} = [\mathrm{h}_{t_i}[z, y'_1, \ldots, y'_k] \mapsto h_{t_i}]$，那么 $\zeta = [\mathrm{h}[z, y'_1, \ldots, y'_k, y_1, \ldots, y_{n-1}] \mapsto \mathrm{F}(y_1, \ldots, y_{i-1}, h_{t_i}, y_i, \ldots, y_{i-1})]$。
3. 如果 $t = \mathrm{n}[t_1, \ldots, t_n]$，使得 w 是 t_i 的子项，并且 $\zeta_{w,t_i} = [\mathrm{h}_{t_i}[z, y'_1, \ldots, y'_k] \mapsto h_{t_i}]$，那么 $\zeta = [\mathrm{h}[z, y'_1, \ldots, y'_k, y_1, \ldots, y_{n-1}] \mapsto \mathrm{n}[y_1, \ldots, y_{i-1}, h_{t_i}, y_i, \ldots, y_{i-1}]]$。

请注意，除非 $t = w$，否则 $t = x$ 的情况是不可能的（情形 1）。◀

现在我们可以证明定理 44 了。

定理 44 的证明。令 $S_0 = S$ 和 $\theta_0 = \rho \circ \theta$，其中 ρ 是某个重命名代换，使得在 $\theta_0 S_0$ 中出现的每个元变量都不在 S_0 中出现。请注意，θ_0 是 E-合一器 S，因为根据假设，θ 成立。

我们现在递归地定义 S_i、ξ_i 和 θ_i，直到得到某个 i 使得 $S_i = \varnothing$。我们确保 ord(S_i、θ_i) 每一步都递减，因此这样的步骤序列总会终止。我们在每一步维持以下不变量：

1. $\langle S_i, \theta_i \rangle \xrightarrow{\xi_i} \langle S_{i+1}, \theta_{i+1} \rangle$ 其中 $S_i \xrightarrow{\xi_i} S_{i+1}$ 根据合一过程的某条规则；2. 序数$(S_{i+1}, \theta_{i+1}) <$ 序数(S_i, θ_i)；

[7] 或者，我们也可以调整表述，只提及至少使用一次的变量子集 z 和 y

3. $\theta_i \in U_E(S_i)$;
4. $\theta_0 \equiv_E \theta_i \circ \xi_{i-1} \circ \ldots \circ \xi_0$;
5. 中出现的每个自由变量都不出现在 S_i;

如果 $S_i \neq \varnothing$，那么令 $\forall \overline{x}.\ \sigma.s \stackrel{?}{=} t :\ \tau$ 成为 S_i 中的一个约束。关于重写序列 $\Theta_i \mid x : \sigma \vdash \theta_i s \overset{*}{\longleftrightarrow}_E \theta_i t :\ \tau$，我们考虑三种主要情况：

1. **重写序列包含一次根重写**。更准确地说，存在一个序列 $\theta_i s = u_0 \longleftrightarrow_E \ldots \longleftrightarrow_E u_n = \theta_i t$ 以及某个项 u_j，使得 $u_j \longleftrightarrow_E u_{j+1}$ 是使用公理进行重写的直接应用。这意味着，s 和 t 可以通过直接使用公理进行合一。更具体地说，存在公理 ζ 的一个实例化 $\Xi \mid \cdots \vdash l \equiv r :\ \tau$ 来自 E，使得 $\zeta l = u_j$、$u_{j+1} = \zeta r$ 和 θ_i 同时将 $s \stackrel{?}{=} u_j$ 与 $u_{j+1} \stackrel{?}{=} t$ 合一。因此，我们可以应用 **（变异）**规则。由于重写序列 $s \overset{*}{\longleftrightarrow}_E t$ 现在被分成两个序列 $s \overset{*}{\longleftrightarrow} u_j$ 和 $u_{j+1} \overset{*}{\longleftrightarrow}_E t$，且新序列长度之和恰好比原序列长度少一，度量得以减少。

2. **重写序列为空，或者不包含根重写**。这意味着重写只能发生在子项中。

   a. 若 $s = x$ 且 $t = y$ 其中 x 和 y 为变量，则 $x = y$，并且我们可以应用 **(删除)**规则，其中 $\xi_i = \mathsf{id}$ 和 $\theta_{i+1} = \theta_i$。由于约束的总大小减小，而重写序列和剩余替换保持不变，因此度量减小。

   b. 若 $s = \mathsf{F}(\overline{z.u})$ 且 $t = \mathsf{F}(\overline{z.v})$，则 $\theta_i s = \mathsf{F}(\overline{z.\theta_i u})$ 且 $\theta_i t = \mathsf{F}(\overline{z.\theta_i v})$。由于不存在根重写，$\theta_i$ 会在相应的扩展上下文中合一每一对 $u_j \stackrel{?}{=} v_j$，因此我们可以应用 **(分解)** 规则，其中 $\xi_i = \mathsf{id}$ 和 $\theta_{i+1} = \theta_i$。请注意，重写链可能被分割为若干条链，但长度总和保持不变。度量的第二个分量也保持不变。由于合一问题中项的总大小减小，而重写链之和不变，因此我们减小了度量的第三个分量。

   c. 若 $s = \mathsf{M}[u]$ 且 M 相对于 j 在 θ_i 处是投影的，那么我们可以应用 **（投影）**规则，使用 $\xi_i = [\mathsf{M}[z] \mapsto z_j]$。注意，重写链保持不变，且 ξ_i 不会从 θ_{i+1} 中移除任何算子（这是将 θ_i 限制为除 M 以外的元变量）。通过减小合一问题中项的总大小来降低度量。

   d. 若 $s = \mathsf{m}[u]$，其中 m 相对于 θ_i 不是投影的，且 $\theta_i s = \mathsf{F}(\overline{z.u})$ 和 $t = \mathsf{F}(\overline{z.v})$，那么 θ_i 在相应的扩展上下文中，每个有序对 $u_i \stackrel{?}{=} \zeta v_i$ 都可合一，我们可以应用（模仿）规则，使用 $\xi_i = [\mathsf{m}[z] \mapsto \mathsf{F}(\overline{x.\mathsf{t}[z, x]})]$。设 $\overline{\theta_i \mathsf{m}[z]} = \mathsf{F}(\overline{x.w})$，那么 $\overline{\theta_{i+1}}$ 是通过从 θ_i 中移除 m 的映射并添加映射 $[\mathsf{t}_j[z, x_j] \mapsto w_j]$（对所有 j）构造的。重写链保持不变，且度量降低，因为我们减少了 θ_{i+1} 中算子的数量。

   e. 若 $s = \mathsf{M}[u]$ 其中，M 相对于 θ_i 和 $\theta_i s \overset{*}{\longleftrightarrow}_E \theta_i t$ 包含对一个子项的重写 w，该子项位于 $\theta_i s$ 中，且不是某个 $\theta_i u_i$ 的出现所包含的子项 i，则

      i. 如果 w 在 $\theta_i s$ 中处于绑定子之下，我们取绑定了被 w 捕获的变量的最外层算子 F（也就是说，$\theta_i s = \ldots \mathsf{F}(y_1.s_1, y_j.\ldots.w \ldots, y_n.s_n)$），并应用（迭代）规则，其中 $\xi_i = [\mathsf{m}[z] \mapsto \mathsf{m}'[z, \mathsf{F}(y_1.\mathsf{m}_1[z, y_1], \ldots, y_n.\mathsf{m}_n[z, y_n])]$。令 $\overline{\theta_i \mathsf{m}[z]} = \mathsf{F}(\overline{y.s'})$，则将 θ_{i+1} 定义为移除了 m 的映射并添加映射 $[\mathsf{m}_i[z, y_i] \mapsto s_i]^{\{i \in \{1,\ldots,n\}\}}$ 后的 θ_i。重写链保持不变，而 θ_{i+1} 中的算子数量减少一个，因此度量下降。

ii. 如果 $w = \mathsf{F}(\ldots)$ 且不处于绑定子之下，则我们应用 **（迭代）**规则，其中 $\xi_i = [\mathrm{M}[z] \mapsto \mathrm{M}'[z, \mathsf{F}(y_1.\mathrm{M}_1[z, y_1], \ldots, y_n.\mathrm{M}_n[z, y_n])]$。我们定义 θ_{i+1} 并证明度量以与前一种情况类似的方式下降。

iii. 如果 $w = \mathsf{w}[v]$ and 不在绑定子下，那么 $\theta_i \mathrm{m}[z]$ 包含 $w' = \mathsf{w}[v']$ 作为子项，并且 $v'_i[z \mapsto u] = v_i$ for all i （这是因为 w 不是任何一个 $\theta_i u_j$ 的子项）。由于 w' 也不在绑定子下，根据引理 50 以及不存在混合算子的假设，我们有：存在项 h、s，以及一个替换 $\zeta = [\mathsf{h}[z, y] \mapsto h]$，使得 $\zeta h[v', s] = \theta_i \mathrm{m}[z]$。令 $\theta'_i = [\mathrm{m}[z] \mapsto h[y \mapsto s]]$。我们有 $\theta_i = \zeta \circ \theta'_i$，也就是说，模 E 意义下，θ'_i 比 θ_i 更一般。重写序列保持不变。如果 $\theta_i s$ 的根部有一个算子，那么 θ'_i 包含更少的算子，从而度量减小。如果 $\theta_i s$ 的根部有一个元变量，那么 θ'_i 包含更少的元变量，从而度量减小。

f. 如果 $s = \mathsf{m}[u]$ 且 $t = \mathsf{n}[v]$，其中 $\mathsf{m} \neq \mathsf{n}$，那么 m 和 n 都不是相对于 θ_i 的投影的，并且 $\theta_i s \xleftrightarrow{*}_E \theta_i t$ 对应于对所有 $i \in \{1, \ldots, n\}$ 的独立重写的并集 $s_i \xleftrightarrow{*}_E t_i$，使得对于所有 i，我们有：s_i 是 ζu_{j_i} 某个出现位置的子项，或者 t_i 是 ζv_{j_i} 某个出现位置的子项。那么根据引理 48，存在项 w、u'、v'，使得 $\theta_i \mathrm{m}[z] = w[y \mapsto u']$ 且 $\theta_i \mathrm{n}[y] = w[z \mapsto u']$。现在我们可以应用（识别）规则，其中 $\xi_i = [\mathrm{m}[z] \mapsto \mathsf{w}[y \mapsto \mathsf{y}[z]]\mathrm{n}[y] \mapsto \mathsf{w}[z \mapsto \mathsf{z}[y]]]$。我们将 θ_{i+1} 定义为 θ_i，移除 m 和 n 的映射，并添加映射 $[\mathsf{w}[z, y] \mapsto w, \mathsf{y}[z] \mapsto v', \mathsf{z}[y] \mapsto u']$。重写序列保持不变，但根据引理 48，项 w 不是变量，因此在 θ_i 中出现了两次（分别对应 m 和 n）的某个算子或元变量，现在在 θ_{i+1} 中只出现一次（对应 w），所以 θ_{i+1} 中的算子或元变量数量至少减少了 1。因此，度量减小了。

g. 如果 $s = \mathsf{m}[u]$ 且 $t = \mathsf{m}[v]$，其中 m 相对于 θ_i 不是投影的，并且 $\theta_i s \xleftrightarrow{*}_E \theta_i t$ 对应于对所有满足以下条件的 $i \in \{1, \ldots, n\}$ 进行的独立重写 $s_i \xleftrightarrow{*}_E t_i$ 的并集：对于所有 i，我们有 s_i 是 ζu_{j_i} 某次出现的子项，或者 t_i 是 ζv_{j_i} 某次出现的子项；那么根据推论 49，我们有 $z' = FV(\theta_i \mathrm{m}[z]) \subseteq z$，使得对于每个 $z_k \in z'$，都有 $\theta_i u_k =_E \theta_i v_k$。考虑以下两个子情况：

i. 如果 $z' = z$ 我们应用 **（分解）**规则，其参数为 $\xi_i = \mathsf{id}$ 和 $\theta_{i+1} = \theta_i$。重写链保持不变，剩余的替换也不变，但约束的总规模减小，因此度量也随之减小。

ii. 如果 $z' \subset z$ 我们应用 **（消去）**规则和 $\xi_i = [\mathrm{M}[z] \mapsto \mathrm{E}[z']]$，并且 θ_{i+1} 定义为以下版本：θ_i 移除了针对 M 的映射，并添加映射 $[\mathrm{E}[z'] \mapsto \theta_i s]$。请注意，重写链保持不变，并且 ξ_i 不会从 θ_{i+1} 中移除任何算子。我们通过减少合一问题中项的总大小来降低度量（因为至少从一个元变量 M 中移除了至少一个参数）。

现在我们得到一个序列 $S_0, \theta_0 \xrightarrow{\xi_0} S_1, \theta_1 \xrightarrow{\xi_1} \ldots$。该序列是有限的，因为度量 $\mathrm{ord}(S_i, \theta_i)$ 在每一步都严格递减。因此，$S, \theta_0 \xrightarrow{\xi_0} \ldots \xrightarrow{\xi_n} \varnothing, \mathsf{id}$ 且 $\theta \equiv_E \rho^{-1} \circ \theta_i \circ \xi_{i-1} \circ \ldots \xi_0 \equiv_E \rho^{-1} \circ \xi_n \circ \ldots \circ \xi_0 \preccurlyeq_E \xi_n \circ \ldots \circ \xi_0$，证明完毕。◀

6 讨论

对我们过程进行务实的实现，可以采用以下改动。我们发现，这些改动有助于在完备性和性能之间取得合理的折衷：

1. 删除 **(迭代)** 规则；该规则牺牲了完备性，但有助于显著减少非确定性；被舍弃的解通常也非常复杂，在某些应用（例如类型推断）中可能并不需要；2. 实现 **(消去*)**规则；3. 拆分公理 $E = B \uplus R$，使得 R 构成一个合流且终止的项重写系统，并引入 **(归一化)** 规则，在应用除 **(删除)** 和 **(消去*)**之外的任何其他规则之前（惰性地）对项进行归一化；4. 限制 **(变异)** 规则的应用次数；5. 限制不会减小问题规模的绑定数量；6. 限制绑定的总数。

将经典的 E-合一和高阶合一中的思想加以改造时，由于绑定子通常不具备 λ-演算所具有的优良语法性质，因此会产生一些技术上的困难。这些困难既影响我们合一过程的设计，促使我们作出一些简化，也影响完备性证明，要求我们采用不同的方法来定义度量，并考虑一些没有对应情形的情况。

在该过程中，每当那些思想依赖规范化、 η-展开或 λ-项的特定语法时，我们都必须进行简化。许多高阶合一算法会依据项的语法性质来确定应用哪些规则。特别是，高阶合一算法常区分灵活 项和刚性 项 [15, 23]。Jensen 和 Pietrzykowski 引入了 ω-简单项的概念 [16]。Vukmirovic 、 Bentkamp 和 Nummelin [35] 引入了基简单项和实心 项的概念。这些性质关键取决于 λ-演算的特定规范化性质，而这些性质在任意二阶等式理论中可能无法获得。因此，我们的过程包含的非确定性可能超出了必要程度。

此类简化的一个显著例子体现在模仿绑定和投影绑定中。在高阶合一算法中，通常会有如下形式的替换

$$[\mathrm{M} \mapsto \lambda x_1, \ldots, x_n.f\ (\mathrm{H}_1\ x_1\ \ldots\ x_n)\ \ldots\ (\mathrm{H}_k\ x_1\ \ldots\ x_n)]$$

其中 f可以是一个 Bound 变量（即 x_1、.....、 x_n中的一个），也可以是类型为 $\sigma_1 \Rightarrow \ldots \Rightarrow \sigma_k \Rightarrow \tau$的常量。这些被称为 Huet 式投影绑定或模仿绑定 [16, 35]，或部分绑定 [15,23]。Huet 式投影（以及促使其使用的条件）很难很好地推广到任意二阶抽象语法，因此本文略去了这些内容，转而采用更简单的规则，但代价是更大的搜索空间。

在高阶合一算法的完备性证明中， λ-演算的句法性质得到了大量利用。在一般的二阶等式理论中，这些性质无法访问，这给定理 44 中完备性证明的构建带来了一些困难。其中最具挑战性的或许是处理证明中的条目 2(e)iii；该条目要求假设不存在混合算子，并使用引理 50。这些内容在高阶合一或一阶 E-合一的完备性证明中似乎都没有对应物。

7 结论与未来工作

我们为二阶抽象语法表述了方程统一问题，使我们能够自然地推理带绑定结构的语言中项的合一。这类语言包括但不限于高阶系统，例如 λ-演算，从而将潜在应用扩展到更多语言。我们还提出了求解此类问题的过程，主要结果表明该过程是完备的。

未来工作中，我们将专注于优化，并识别二阶方程上 E-统一的可判定片段。

一种值得注意的优化是将 E 拆分为两组 $R \uplus B$，其中 R 是一组有向方程，可构成合流二阶项重写系统；而 B 是一组无向方程（例如结合律和交换律公理）。

另一种潜在的优化源于 Huet 式绑定（也称为部分绑定）的推广。这可以帮助我们更有依据地决定在过程中应用哪条规则，引入**（投影）**的 Huet 式版本，并改进**（迭代）**规则，从而显著缩小搜索空间。Kudasov [20] 已将此类优化的一种实现形式设计为启发式方法，用来结合**（模仿）**和**（投影）**规则。

对于 E-合一和高阶合一，都有若干经过深入研究的片段。例如，幺半群理论中的合一，本质上就是求解半环上的线性方程 [26]。在高阶合一中，也有几个众所周知的可判定片段，例如模式合一 [23]。Vukmirovic、Bentkamp 和 Nummelin 在近期关于高效完全高阶合一的研究中，指出了一些在实践中重要的可判定片段，并发现了一个新的片段 [35]。值得探究的是，能否将这些片段推广到二阶抽象语法并将其用作判定预言机，从而有望得到一种高效的二阶抽象语法 E-合一过程，作为其过程的严格推广。

参考文献

1 Franco Barbanera、Maribel Fernández 和 Herman Geuvers。代数-λ-立方体中强规范化的模块化。《函数式编程杂志》，7(6):613–660，1997年11月。doi:10.1017/S095679689700289X。 2 Adam Chlipala。用于机械化语义学的参数化高阶抽象语法。载于第 13 届 ACM SIGPLAN 函数式编程国际会议论文集，ICFP '08，第143–156页，美国纽约州纽约市，2008年。计算机协会。doi:10.1145/1411204.1411226。 3 Jacek Chrząszcz 和 Daria Walukiewicz-Chrząszcz。迈向 Coq 中的重写，第113–131页。施普林格出版社，柏林、海德堡，2007年。 4 Jesper Cockx。《解缚的类型论：以用户定义的重写规则扩展 Agda》。载于 Marc Bezem 和 Assia Mahboubi 编，第25届证明与程序类型国际会议（TYPES 2019），《莱布尼茨国际信息学论文集》（LIPIcs）第175卷，第2:1–2:27页，德国达格施图尔，2020年。达格施图尔城堡—莱布尼茨信息学中心。网址：https://drops.dagstuhl.de/opus/volltexte/2020/13066，doi:10.4230/LIPIcs.TYPES.2019.2。 5 Jesper Cockx。1001种带绑定语法的表示方式。https://jesper.sikanda.be/posts/1001-syntax-representations.html，2021年11月。访问日期：2023年1月21日。 6 Jesper Cockx、Nicolas Tabareau 和 Théo Winterhalter。《驯服 Rew：一种带计算假设的类型论》。《ACM 程序设计语言论文集》，5(POPL)，2021年1月。doi:10.1145/3434341。 7 Denis Cousineau 和 Gilles Dowek。《在 Lambda-Pi-模演算中嵌入纯类型系统》。载于 Simona Ronchi Della Rocca 编，《带类型的 λ 演算及其应用》，第8届国际会议 TLCA 2007，法国巴黎，2007 年 6 月 26 日至 28 日，论文集，《计算机科学讲义》第 4583 卷，第102–117页。施普林格，2007年。doi:10.1007/978-3-540-73228-0_9。 8 Gilles Dowek、Thérèse Hardin 和 Claude Kirchner。《通过显式替换实现高阶合一》。《信息与计算》，157(1-2):183–235，2000年。 doi:10.1006/inco.1999.2837。 9 Marcelo Fiore 和 Dmitrij Szamozvancev。《二阶抽象语法的形式元理论》。《ACM 程序设计语言论文集》，6(POPL)，2022年1月。 doi:10.1145/3498715。 10 Marcelo P. Fiore 和 Chung-Kil Hur。《二阶等式逻辑》（扩展摘要）。载于 Anuj Dawar 和 Helmut Veith 编，《计算机科学逻辑》，第24届国际研讨会，

24 二阶抽象语法的 E-合一

CSL 2010，EACSL第19届年会，捷克共和国布尔诺，2010年8月23日至27日。论文集，《计算机科学讲义》第6247卷，第320–335页。施普林格，2010年。 doi:10.1007/978-3-642-15205-4_26。 11 Jean H. Gallier 和 Wayne Snyder。一般 E-合一的完备变换集。《理论计算机科学》，67(2&3):203–260，1989年。 doi:10.1016/0304-3975(89)90004-2。 12 Warren D. Goldfarb。二阶合一问题的不可判定性。《理论计算机科学》，13:225–230，1981年。 doi:10.1016/0304-3975(81)90040-2。 13 Adam Michael Gundry。类型推断、Haskell 与依赖类型。博士论文，斯特拉斯克莱德大学，英国格拉斯哥，2013年。网址： http://oleg.lib.strath.ac.uk/R/?func=dbin-jump-full&object_id=22728。 14 Michael Hoche 和 Peter Szabó。本质合一子。《应用逻辑杂志》，4(1):1–25，2006年。 doi:10.1016/j.jal.2004.12.001。 15 Gérard P. Huet。带类型 λ 演算的合一算法。《理论计算机科学》，1(1):27–57，1975年。 doi:10.1016/0304-3975(75)90011-0。 16 D. C. Jensen 和 Tomasz Pietrzykowski。通过合一实现 ω阶类型论的机械化。《理论计算机科学》，3(2):123–171，1976年。 doi:10.1016/0304-3975(76)90021-9。 17 Claude Kirchner 和 Christophe Ringeissen。《通过显式替换实现高阶方程合一》。载于 Michael Hanus、Jan Heering 和 Karl Meinke 编，《代数与逻辑编程》，第6届国际联合会议 ALP '97 - HOA '97，英国南安普敦，1997年9月3日至5日，论文集，《计算机科学讲义》第1298卷，第61–75页。施普林格，1997年。 doi:10.1007/BFb0027003。 18 Edward Kmett。腐烂的香蕉。 http://comonad.com/reader/2008/rotten-bananas/，2008年3月。访问日期：2023年1月21日。 19 Edward Kmett。Bound。https://www.schoolofhaskell.com/user/edwardk/bound，2015年12月。访问日期：2023年1月21日。 20 Nikolai Kudasov。函数式珍珠：通过自由高阶合一实现依赖类型推断，2022年。网址： https://arxiv.org/abs/2204.05653， arXiv:2204.05653。 21 Tomer Libal 和 Dale Miller。函数即构造子的高阶合一。载于 Delia Kesner 和 Brigitte Pientka 编，第1届计算与演绎形式结构国际会议（FSCD 2016），葡萄牙波尔图，2016年6月22日至26日，《LIPIcs》第52卷，第26:1–26:17页。达格施图尔城堡—莱布尼茨信息学中心，2016年。 doi:10.4230/LIPIcs.FSCD.2016.26。 22 Francesco Mazzoli 和 Andreas Abel。通过合一进行类型检查。CoRR，abs/1609.09709，2016年。网址： http://arxiv.org/abs/1609.09709， arXiv:1609.09709。 23 Dale Miller。一种带有 λ 抽象、函数变量和简单合一的逻辑编程语言。《逻辑与计算杂志》，1(4):497–536，1991年。 doi:10.1093/logcom/1.4.497。 24 Dale Miller 和 Gopalan Nadathur。《高阶逻辑编程》。剑桥大学出版社，2012年。 doi:10.1017/CBO9781139021326。 25 Tobias Nipkow 和 Christian Prehofer。高阶重写与方程推理。载于 W. Bibel 和 P. Schmitt 编，《自动演绎——应用基础》。第一卷：基础，第8卷《应用逻辑丛书》，第399–430页。克卢维尔，1998年。 26 Werner Nutt。幺半群理论中的合一即是在半环上求解线性方程。载于 SciDok——萨尔大学科学服务器，《研究报告／德国人工智能研究中心》，第92-01卷。德国人工智能研究中心（DFKI），1992年。 doi:10.22028/D291-24991。 27 Michel Parigot。 $\lambda\mu$演算：经典自然演绎的算法解释。载于 Andrei Voronkov 编，《逻辑编程与自动推理》，第190–201页，柏林、海德堡，1992年。施普林格柏林海德堡出版社。 28 Frank Pfenning 和 Conal Elliott。高阶抽象语法。载于 Richard L. Wexelblat 编，ACM SIGPLAN'88 编程语言设计与实现会议（PLDI）论文集，美国佐治亚州亚特兰大，1988年6月22日至24日，第199–208页。美国计算机协会，1988年。 doi:10.1145/53990.54010。

29 Wayne Snyder。高阶 E-合一。载于 Mark E. Stickel（编），第十届自动演绎国际会议论文集，凯撒斯劳滕，德意志联邦共和国，1990年7月24日至27日，《计算机科学讲义》第449卷，第573–587页。施普林格，1990年。doi:10.1007/3-540-52885-7_115。 30 Mark-Oliver Stehr。《开放构造演算（第一部分）：一种用于程序设计、规约和交互式定理证明的依赖类型等式类型论》。Fundam. Informaticae，68(1-2):131–174，2005年。URL: http://content.iospress.com/articles/fundamenta-informaticae/fi68-1-2-05。 31 Mark-Oliver Stehr。《开放构造演算（第二部分）：一种用于程序设计、规约和交互式定理证明的依赖类型等式类型论》。Fundam. Informaticae，68(3):249–288，2005年。URL: http://content.iospress.com/articles/fundamenta-informaticae/fi68-3-04。 32 Peter Szabó 和 Jörg H. Siekmann。《基于广义嵌入的 E-合一》。Math. Struct. Comput. Sci.，31(8):898–917，2021年。 doi:10.1017/S0960129522000019。 33 Peter Szabó、Jörg H. Siekmann 和 Michael Hoche。《什么是本质合一？》载于 Eugenio G. Omodeo 和 Alberto Policriti（编），《Martin Davis 论可计算性、计算逻辑与数学基础》，《逻辑学杰出贡献》第10卷，第285–314页。施普林格，2016年。 doi:10.1007/978-3-319-41842-1_11。 34 Val Tannen。《结合代数与高阶类型》。载于第三届计算机科学逻辑年度研讨会（LICS '88）论文集，爱丁堡，苏格兰，英国，1988年7月5日至8日，第82–90页。IEEE计算机学会，1988年。 doi:10.1109/LICS.1988.5103。 35 Petar Vukmirovic、Alexander Bentkamp 和 Visa Nummelin。《高效的完全高阶合一》。Log. Methods Comput. Sci.，17(4)，2021年。 doi:10.46298/lmcs-17(4:18)2021。 36 Daria Walukiewicz-Chrząszcz。《构造演算中的重写终止性》。《函数式编程杂志》，13(2):339–414，2003年3月。 doi:10.1017/S0956796802004641。 37 Geoffrey Washburn 和 Stephanie Weirich。《方框妙趣横生：用参数多态编码高阶抽象语法》。《函数式编程杂志》，18(1):87–140，2008年。doi:10.1017/S0956796807006557。 38 Beta Ziliani 和 Matthieu Sozeau。《Coq 的一种合一算法：支持宇宙多态和重载》。载于 Kathleen Fisher 和 John H. Reppy（编），第二十届ACM SIGPLAN函数式编程国际会议论文集，ICFP 2015，加拿大不列颠哥伦比亚省温哥华，2015年9月1日至3日，第179–191页。美国计算机协会，2015年。doi:10.1145/2784731.2784751。