arXiv:2603.03968v1 [cs.PL] 4 Mar 2026

作为 Agda 库的名义技术*

Murdoch J. Gabbay[1] 和 Orestis Melkonian[2]

[1] 苏格兰赫瑞-瓦特大学[2] 苏格兰爱丁堡大学

引言 名义技术 [8] 为处理编程语言中的名称和变量绑定提供了一种具有数学原理依据的方法。然而，要将这些理念整合到实用且广泛使用的工具链中，进展一直缓慢；我们认为这存在一个先有鸡还是先有蛋的问题：没有人使用名义技术，因为没有人实现它们；而没有人实现它们，又是因为没有人使用它们。这令人遗憾，但也带来了一个积极的机会，让我们能够建立一个良性循环，促进对这项优美技术的更广泛理解、采纳和应用。

本文探讨了如何在 Agda 证明助手和编程语言中，将名义技术作为库提供给用户使用 [9], 。Agda 可以被视为第一作者的 Haskell `nom` 软件包 [6], 的移植版本，不过这样说并不公允，因为它有双重目的：

1. 提供一个便捷的库，以便在你自己的 Agda 形式化中使用名义技术 2. 严谨且构造性地研究名义技术的元理论

目标 1 的解决方案必须易于使用，这意味着仅仅在实现名义思想方面取得技术上的成功还不够；我们还要求取得实际层面的成功，使其开销在实际系统中可以接受。除了这是一份文学式 Agda 文件之外，我们的成果也已通过机器验证并公开发布：`https://omelkonian.github.io/nominal-agda/`。

名义设置 我们的开发基于某种抽象的**原子**类型进行，并满足若干约束，即相等性可判定且元素无穷可枚举。[1]。我们在 Agda 中使用模块参数对其建模，并可将其具体化为某个具体类型：

```
模块 _ (Atom : Type) {{ _ : DecEq Atom }} {{ _ : Enumerable∞ Atom }} 其中 :
(Atom →Type) → Type𝜙 = ∃ λ (xs : List Atom) → (∀ y → y ∉ xs → 𝜙 y)
```

量词 规定谓词对除有限多个**原子**以外的所有原子都成立，并且可以在任意类型上**交换**两个原子，但须满足一些定律：

```
记录 交换 (A : 类型) : 类型 其中字段 交换 : 原子 → 原子 → A → A(_↔_)_ = 交换
```

```
实例↔原子 : 交换原子↔原子。交换 x y z = 如果 z == x 那么y 否则如果 z == y 那么 x 否则 z
```

```
记录 SwapLaws : 类型，其中字段 swap-id : ⟪ a ↔ a ⟫x ≡ x swap-rev : ⟪ a ↔ b ⟫ x ≡ ⟪ b ↔ a ⟫ x swap-sym : ⟪ a ↔ b ⟫ ⟪ b ↔ a ⟫ x ≡ x 交换-交换 : ⟪ a ↔ b ⟫ ⟪ c ↔ d ⟫ x ≡ ⟪ ⟪ a ↔ b ⟫ c ↔ ⟪ a ↔ b ⟫ d ⟫ ⟪ a ↔ b ⟫ x
```

我们只需为原子 （由此得到可判定相等）这一基本情况提供实例，以及抽象 （下文将介绍）。据此，我们可以利用编译期宏/策略（参见后文的案例研究），系统地为所有用户定义类型推导出交换定义。

等变ZFA基础中一类特别有用的公理 [5] 是，交换在任何地方（构造子、函数、类型构造子）都满足分配性；其中交换自身的特殊情形是交换-交换。对交换的这一广义分配性概念进行公理化是相容的，我们通过一个策略来实现这一公理模式。大多数时候

*本研究发表于第29届证明与程序类型国际会议（TYPES 2023），会议于西班牙瓦伦西亚举行。幻灯片、演讲录制视频以及GitHub 上的 Agda 源代码也均可获取。[1].....在近期对局部无名方法进行的名义机械化中，也称为“非有限性” [10]。

我们处理的是具有**有限支撑**的类型，并用“新”量词表示：$2\lambda\ \mathbb{a}\ \mathbb{b}\ \to$ 交换$\mathbb{b}\ \mathbb{a}\ x\ \equiv x$。接着，我们可以定义**等变**元素，使其具有空支撑；还可以定义生成新鲜原子的操作freshAtom : $A\ \to$ 原子（因此模块要求原子是无穷可枚举的）。Agda 是构造性的，所以freshAtom也是构造性的，这与新鲜原子在（非构造性的）集合论中的用法不同。一个**抽象**只是一个原子与一个元素组成的有序对：

Abs $A = Atom \times A$

连接: 抽象 $A\ \to$ 原子 $\to A$连接 $(\mathbb{a}\ ,\ x)\ \mathbb{b}\ =$ 交换 $\mathbb{b}\ \mathbb{a}\ x$

实例
↔抽象 : 交换(抽象 A)
↔Abs .交换 $\mathbb{a}\ \mathbb{b}\ (\mathbb{c}\ ,\ x)\ =$ (交换 $\mathbb{a}\ \mathbb{b}\ \mathbb{c}$, 交换 $\mathbb{a}\ \mathbb{b}\ x$)

请注意，我们还可以提供一个正确性由构造保证的且全定义的具体化函数。在基于 Fraenkel-Mostowski集合论的名义技术中， [8] 这是不可能的；而在构造性框架下，全定义的具体化函数是可行的，这似乎是一个新颖的发现。

案例研究具备所有预期的名义设施后，特别是原子和原子抽象，就可以轻松地定义**无类型 λ-演算**中的项，而无需提及 de Bruijn 索引或类似概念。为了提升易用性并高效地证明定理，我们提供了一个基于*elaborator* 反射 [2] 的元编程宏，它能够根据任何类型的结构自动推导出其交换操作的实现。

数据Term： 类型 其中
: 原子 →Term_ · : Term
→ Term → Term_ _λ : Abs
Term →Term_unquoteDecl ↔
Term =
推导 交换 [引用 项， ↔项]

data _≈_ : Term → Term → Type where
$\nu\approx$: ‘ $x \approx$ ‘ x
$\xi\approx$: $L \approx L' \to M \approx M' \to L\ \cdot\ M \approx L'\ \cdot\ M'$
$\zeta\approx$: $(\lambda\ \mathbb{x} \to$ conc $f\ \mathbb{x} \approx$ conc $g\ \mathbb{x}) \to \lambda\ f \approx \lambda\ g$

我们可以自然地使用量词来表达 α-等价的 λ-项，并手动证明上述交换律，以及每个 λ-项都有有限支撑这一事实。不过，这些内容都可以通过系统化的数据类型泛型构造来实现，我们目前正在着手将其自动化。其余开发内容与 PLFA 教材中所介绍的机械化完全相同， [14]，尤其是“无类型”一章。与此同时，涉及繁琐指标操作的棘手“替换”附录，如今已由常见的替换名义表示取代，并辅以若干关于等变性和支撑的一般引理：

_[_:=_] : Term → $Atom$ → Term → Term
(‘ x) [$\mathbb{a}$:= N] = if x == $\mathbb{a}$ then N else ‘ x
$(L\ \cdot\ M)$ [$\mathbb{a}$:= N] = L [$\mathbb{a}$:= N] · M [$\mathbb{a}$:= N]
$(\lambda\ f)$ [$\mathbb{a}$:= N] = $\lambda\ \mathbb{x} \to$ conc $f\ \mathbb{x}$ [$\mathbb{a}$:= N]

关于 z = freshAtom (z ∷ supp f ++ supp N) 我们还需要证明几个引理，才能完整涵盖 PLFA 中关于无类型λ演算的章节，但我们看不出完成合流性证明会遇到任何根本性障碍。接下来一个不错的步骤是形式化一阶逻辑的切消去证明，因为这涉及对项和证明树进行名称抽象。

相关工作 此前已有针对无类型 λ-演算具体实例的 Agda 名义机械化工作，并包含合流性证明 [4, 3]。我们的工作与 [7], 中的非机械化表述十分接近，而 Haskell `nom` 软件包 [6] 正是据此实现的。另一种在 Agda 中表示名义集合的方法 [1] 仍处于初步阶段；我们希望我们的方法更便于使用，也更易于扩展。我们将 Agda 库视为其他名义实现（FreshML [12], 、Isabelle/HOL [13],和 Nuprl [11]）的补充；它兼具易用、轻量和易于上手等特点，并展现了名义技术在构造型类型系统中的实际兼容性。

参考文献

[1] Pritam Choudhury。Agda 中名义集合的构造性表示。硕士论文，剑桥大学罗宾逊学院，2015年。[2] David R. Christiansen 和 Edwin C. Brady。Elaborator 反射：在 Idris 中扩展 Idris。载于 Jacques Garrigue、Gabriele Keller 和 Eijiro Sumii 编，《第21届 ACM SIGPLAN 函数式编程国际会议论文集》，ICFP 2016，日本奈良，2016年9月18日至22日，第284–297页。ACM，2016年。[3] Ernesto Copello、Nora Szasz 和 Álvaro Tasistro。使用构造型类型论中的 Barendregt 变量约定，对 λ演算的丘奇-罗瑟定理进行机器检验的证明。载于 Sandra Alves 和 Renata Wasserman 编，《第十二届逻辑与语义框架及其应用研讨会》，LSFA 2017，巴西巴西利亚，2017年9月23日至24日，《理论计算机科学电子札记》第338卷，第79–95页。爱思唯尔，2017年。[4] Ernesto Copello、Alvaro Tasistro、Nora Szasz、Ana Bove 和 Maribel Fernández。构造型类型论中 λ演算的 α结构归纳与递归。载于 Mario R. F. Bene- vides 和 René Thiemann 编，《第十届逻辑与语义框架及其应用研讨会论文集》，LSFA 2015，巴西纳塔尔，2015年8月31日至9月1日，《理论计算机科学电子札记》第323卷，第109–124页。爱思唯尔，2015年。[5] Murdoch J. Gabbay。等变 ZFA 与名义技术基础。《逻辑与计算杂志》，30(2):525–548，2020年。[6] Murdoch J. Gabbay。 nom 2020 https://hackage.haskell.org/package/nom

haskell 软件包，。网址：。[7] Murdoch J. Gabbay 和 Aad Mathijssen。λ演算的名义公理化。《逻辑与计算杂志》，20(2):501–531，2010年。[8] Murdoch J. Gabbay 和 Andrew M. Pitts。具有变量绑定的抽象语法新方法。《计算形式方面》，13(3-5):341–363，2002年。[9] Ulf Norell。Agda 中的依赖类型编程。载于 Andrew Kennedy 和 Amal Ahmed 编，《TLDI'09 论文集：2009 ACM SIGPLAN 语言设计与实现中的类型国际研讨会》，美国佐治亚州萨凡纳，2009年1月24日，第1–2页。ACM，2009年。[10] Andrew M. Pitts。局部无名集合。《ACM 程序设计语言会议论文集》，7(POPL):488–514，2023年。[11] Vincent Rahli 和 Mark Bickford。直觉主义的名义探索。载于 Jeremy Avigad 和 Adam Chlipala 编，《第五届 ACM SIGPLAN 认证程序与证明会议论文集》，美国佛罗里达州圣彼得堡，2016年1月20日至22日，第130–141页。ACM，2016年。[12] Mark R. Shinwell、Andrew M. Pitts 和 Murdoch J. Gabbay。FreshML：让绑定子编程变得简单。载于 Colin Runciman 和 Olin Shivers 编，《第八届 ACM SIGPLAN 函数式编程国际会议论文集》，ICFP 2003，瑞典乌普萨拉，2003年8月25日至29日，第263–274页。ACM，2003年。[13] Christian Urban。Isabelle/HOL 中的名义技术。《自动推理杂志》，40(4):327–356，2008年。[14] Philip Wadler、Wen Kokke 和 Jeremy G. Siek。Agda 中的编程语言基础。2022年8月。网址： https://plfa.inf.ed.ac.uk/22.08/。