arXiv:2303.08894v1 [math.CT] 15 Mar 2023

Coq中的Operad形式化

Zachary Flores ✉高保证解决方案，
Two Six Technologies
Angelo Taranto ✉High Assurance
Solutions，Two Six Technologies
EricBond ！High Assurance Solutions，
Two SixTechnologies
YakirForman ！High Assurance
Solutions，Two SixTechnologies

摘要

如何才能最大限度地确保编程语言中的执行正确性？对此问题，一种答案——也是我们采用的方案——是：如果编程语言存在指称语义，就对其进行形式化。实现这样的形式化，为确保编程语言从构造之初便正确无误树立了黄金标准。在 DARPA V-SPELLS 项目中，我们致力于为元语言的指称语义奠定基础，所用工具是一种称为幺半群的数学对象。幺半群具有组合性，这对于由较小部分构建语言至关重要。本文将讨论我们在证明助手 Coq 中对一个幺半群所做的形式化。此外，我们在 Coq 中的定义能够为其中指定的对象是幺半群提供证明。这项在 Coq 中开展的工作，为我们在 V-SPELLS 中开发元语言奠定了形式化的数学基础。据我们所知，我们的工作还首次在具有显著自动化能力的证明助手中实现了幺半群的形式化，并提供了一个无需掌握同伦类型论知识即可复现的模型。

2012 年 ACM 学科分类 计算理论 → 构造性数学

关键词和短语 Operad、形式化数学、Coq

数字对象标识符 10.4230/LIPIcs.CVIT.2016.23

补充材料 免责声明：本研究由美国国防高级研究计划局（DARPA）资助。文中表达的观点、意见和/或研究结果均属于作者，不应被解释为代表美国国防部或美国政府的官方观点或政策。

Fund支持 由 DARPA V-SPELLS HR00112 资助 0S0058。

致谢 感谢 Li-yao 在编写多重组合的 Coq 代码方面提供帮助。

1 引言

DARPA V-SPELLS（大型遗留软件的经验证安全性与性能增强）项目旨在为大型遗留代码库中的软件组件提供开发者可访问的逐步增强能力，即使用能够与系统其余部分安全组合的新型经验证代码进行增强。

在我们与约翰斯·霍普金斯大学应用物理实验室合作解决 V-SPELLS 所提出问题的方案中，我们正在开发的工具 LUMOS 首先应用

分发声明 A：已批准公开发布，不限分发

© Zachary Flores、Angelo Taranto、Eric Bond 和 Yakir Forman；依据知识共享署名 4.0 国际许可协议授权。第42届重要议题会议（CVIT 2016）。编辑：John Q. Open 和 Joan R. Access；文章编号：23；第 23:1–23:11 页 莱布尼茨国际信息学论文集 达格施图尔城堡 – 莱布尼茨信息学中心，达格施图尔出版社，德国

静态分析、自然语言处理和动态分析方法处理遗留源代码，以从构成源代码的领域专用语言（DSLs）中生成这些领域专用语言的高层抽象，我们称之为领域特定语义模型（DSSMs）。这些 DSSMs 将以一种我们称为 元领域专用语言 的语言生成。为了按照 V-SPELLS 的要求为遗留代码提供补丁，这些 DSSMs 必须以非常特定的方式复合。为了确保复合的正确性（这是 V-SPELLS 的要求），我们借助代数框架进行验证，其中运用了范畴论中的若干思想，而我们建模所依赖的关键结构称为 幺半群。幺半群在应用数学中日益发挥重要作用（参见 [6, 3, 1, 2, 4]），我们发现，它们为我们在 V-SPELLS 中的验证需求提供了出色的数学模型。

更精确地说，在我们的建模中，当用元领域专用语言编写 DSSM 时，我们将用一个幺半群来表示元领域专用语言中的 DSSM，而元领域专用语言内 DSSM 的组合则通过一种“粘合”操作来建模。从数学角度看，我们为元领域专用语言中一部分关键语言提供了指称语义。为确保元领域专用语言中 DSSM 之间组合的正确性达到最高水平，我们旨在对元领域专用语言的指称语义进行形式化。具体而言，我们需要为元领域专用语言指称语义的基础——幺半群——提供形式化定义。我们在证明助手 Coq 中完成了这一形式化，这也是本文的重点。

在第 2 节中，我们讨论幺半群的非形式化定义；第 3 节讨论了我们在 Coq 中定义幺半群时遇到的技术难题及其解决方案；第 4 节讨论我们如何在 Coq 中构造与集合幺半群等价的结构（即类型幺半群），并介绍我们在 Coq 中依据第 3 节的规范证明该结构是一个幺半群；最后，在第 5 节中，我们将我们的形式化与我们所知的唯一其他幺半群形式化工作进行比较 [5]。我们

2 非形式化定义幺半群

虽然似乎并没有对对称着色歌剧集达成一致的定义，但我们注意到，我们遵循的是 [7] 中对称着色歌剧集 的定义。不过，我们要指出， [7] 中的定义并未包含 [8] 中所谓的等变公理；我们也省略了这条公理，因为它与我们在 V-SPELLS 项目中想要实现的目标无关。尽管存在这些区别，在下文中，我们用 幺半群 指对称着色歌剧集 或着色歌剧集 。

我们的目标是在 Coq 中完整形式化幺半群的定义，因此需要精确表述。为此，我们在下面分两部分给出幺半群的完整非形式化定义。第一部分由构成幺半群的对象组成。

▶ **定义 1**（幺半群的数据）。 一个幺半群 O 由一个类型集合组成，我们将其记为 T；对于每个 $n \geq 1$ $d \in T$ c：$= c_0$，……，c_{n-1}，还有 T 中的一个类型序列，以及一个项集合 $O\binom{d}{c}$，使得

1. 对于每个 $c \in T$，我们指定一个元素 $\mathbb{1}_c \in \mathcal{O}\binom{c}{c}$，称为 c-色单位；**2.** 若 σ 是作用于 n 个字母的置换，且 $c\sigma$：$= c_{\sigma(0)}$，…，$c_{\sigma(n-1)}$，则 $\mathcal{O}\binom{d}{c}$ 与 $\mathcal{O}\binom{d}{c\sigma}$ 之间存在一个双射；

分发声明 A：已批准公开发布，不限分发

3. 对于任意b 中类型的序列 T，若我们用 $c \bullet_i b$ 表示由下式给出的序列

$$\underbrace{c_0, \ldots, c_{i-1}}_{\emptyset \ if \ i=0}, \underline{b}, \underbrace{c_{i+1}, \ldots, c_{n-1}}_{\emptyset \ if \ i=n-1},$$

然后，我们要求存在一个函数：

$$\circ_i : \mathcal{O}\binom{d}{\underline{c}} \times \mathcal{O}\binom{c_i}{\underline{b}} \to \mathcal{O}\binom{d}{\underline{c} \bullet_i \underline{b}}.$$

我们通常将函数 $\circ_i$ 称为多重复合。

▶ **示例 2.** 为了快速了解多重合成函数的类型签名是什么样的，设 $c = c_0$，c_1，c_2, $b = b_0$，b_1 和 $i = 1$，那么 $\circ_1$ 的类型签名为：

$$\mathcal{O}\binom{d}{c_0, c_1, c_2} \times \mathcal{O}\binom{c_1}{b_0, b_1} \to \mathcal{O}\binom{d}{c_0, b_0, b_1, c_2}.$$

现在，定义 1 中幺半群 $\mathcal{O}$ 的数据受到某些公理约束的限制，这构成了我们对幺半群定义的后半部分。

▶ **定义 3** （幺半群公理）。设 $c := c_0, \ldots, c_{n-1}, b := b_0, \ldots, b_{m-1}, a = a_0, \ldots, a_{\ell-1}$ 是来自类型集合 T 的序列。幺半群 O 的数据必须遵循的公理如下。

1。水平结合公理：假设 $n \geq 2$ 和 $0 \leq i < j \leq n-1$，那么 fo r $(\alpha, \beta, \gamma) \in \mathcal{O}\binom{d}{\underline{c}} \times \mathcal{O}\binom{c_i}{\underline{a}} \times \mathcal{O}\binom{c_j}{\underline{b}}$,

$$(\alpha \circ_i \beta) \circ_{\ell-1+j} \gamma = (\alpha \circ_j \gamma) \circ_i \beta$$

为了直观地描述这一公理，我们要求下图交换：

$$\begin{tikzcd}
\mathcal{O}\binom{d}{\underline{c}} \times \mathcal{O}\binom{c_i}{\underline{a}} \times \mathcal{O}\binom{c_j}{\underline{b}} \arrow[r, "(\circ_i, id)"] \arrow[d, "\cong"', "swap"] & \mathcal{O}\binom{d}{\underline{c} \bullet_i \underline{a}} \times \mathcal{O}\binom{c_j}{\underline{b}} \arrow[d, "\circ_{\ell-1+j}"] \\
\mathcal{O}\binom{d}{\underline{c}} \times \mathcal{O}\binom{c_j}{\underline{b}} \times \mathcal{O}\binom{c_i}{\underline{a}} \arrow[d, "(\circ_j, id)"'] & \mathcal{O}\binom{d}{(\underline{c} \bullet_i \underline{a}) \bullet_{\ell-1+j} \underline{b}} \arrow[d, equal] \\
\mathcal{O}\binom{d}{\underline{c} \bullet_j \underline{b}} \times \mathcal{O}\binom{c_i}{\underline{a}} \arrow[r, "\circ_i"'] & \mathcal{O}\binom{d}{(\underline{c} \bullet_j \underline{b}) \bullet_i \underline{a}}.
\end{tikzcd}$$

2. 垂直结合公理：假设 m，$n \geq 1$，$0 \leq i \leq n-1$，以及 $0 \leq j \leq m-1$。那么对于 $(\alpha, \beta, \gamma) \in \mathcal{O}\binom{d}{c} \times \mathcal{O}\binom{c_i}{b} \times \mathcal{O}\binom{b_j}{a}$,

---

分发声明 A：已批准公开发布，不限分发

$$(\alpha \circ_i \beta) \circ_{i+j} \gamma = \alpha \circ_i (\beta \circ_j \gamma)$$

也就是说，我们要求下面的图表具有交换性：

$$\begin{array}{ccc}
\mathcal{O}\binom{d}{\underline{c}} \times \mathcal{O}\binom{c_i}{\underline{b}} \times \mathcal{O}\binom{b_j}{\underline{a}} & \xrightarrow{(id,\circ_j)} & \mathcal{O}\binom{d}{\underline{c}} \times \mathcal{O}\binom{c_i}{\underline{b} \bullet_j \underline{a}} \\
\Big\downarrow{\scriptstyle (\circ_i, id)} & & \Big\downarrow{\scriptstyle \circ_i} \\
 & & \mathcal{O}\binom{d}{c \bullet_i (\underline{b} \bullet_j \underline{a})} \\
 & & \| \\
\mathcal{O}\binom{d}{\underline{c} \bullet_i \underline{b}} \times \mathcal{O}\binom{b_j}{\underline{a}} & \xrightarrow[\circ_{i+j}]{} & \mathcal{O}\binom{d}{(\underline{c} \bullet_i \underline{b}) \bullet_{i+j} \underline{a}}
\end{array}$$

3. 该左单位公理要求对于 $\alpha \in \mathcal{O}\binom{d}{c}$ 满足 $n \geq 1$，$\mathbb{1}_d \circ_1 \alpha = \alpha$.

4. 该右单位公理要求对于 $n \geq 1$，$0 \leq i \leq n-1$以及 $\alpha \in \mathcal{O}\binom{d}{c}$，$\alpha \circ_i \mathbb{1}_{c_i} = \alpha$.

在给出一个例子之前，先对定义 3 作一些说明。

►注4。

我们想对结合公理作一些合理性检查。首先注意，水平结合公理（定义 3 的第 1 条）对应图表右下角出现了以下等式：

$$\mathcal{O}\binom{d}{(\underline{c} \bullet_i \underline{a}) \bullet_{\ell-1+j} \underline{b}} = \mathcal{O}\binom{d}{(\underline{c} \bullet_j \underline{b}) \bullet_i \underline{a}} \tag{1}$$

这个等式源于以下序列之间的相等：

$$\begin{aligned}
(\underline{c} \bullet_i \underline{a}) \bullet_{\ell-1+j} \underline{b} &= (c_0, \ldots, c_{i-1}, \underline{a}, c_{i+1}, \ldots, c_{n-1}) \bullet_{\ell-1+j} \underline{b} \\
&= c_0, \ldots, c_{i-1}, \underline{a}, c_{i+1}, \ldots c_{j-1}, \underline{b}, c_{j+1}, \ldots, c_{n-1} \\
&= (\underline{c} \bullet_j \underline{b}) \bullet_i \underline{a}
\end{aligned} \tag{2}$$

具体而言，为了在 Coq 中给出幺半群的规范，我们需要证明对于 T 中的这类序列，(2) 成立。

定义垂直结合图表还需要一个类似的序列相等关系：

$$\begin{aligned}
\underline{c} \bullet_i (\underline{b} \bullet_j \underline{a}) &= c_0, \ldots, c_{i-1}, (\underline{b} \bullet_j \underline{a}), c_{i+1}, \ldots, c_{n-1} \\
&= c_0, \ldots, c_{i-1}, b_0, \ldots, b_{j-1}, \underline{a}, b_{j+1}, \ldots, b_{m-1}, c_{i+1}, \ldots, c_{n-1} \\
&= (\underline{c} \bullet_i \underline{b}) \bullet_{i+j} \underline{a}
\end{aligned} \tag{3}$$

尽管我们的定义看起来极其抽象，接下来的例子将有助于阐明定义 1 和定义 3 中抽象概念的根源。此外，接下来的例子将作为我们形式化定义 Operad 的首次应用：我们将在 Coq 中证明，该例子的实现符合我们的规范，是一个幺半群。

分发声明 A：已批准公开发布，不限分发

▶ **例 5.** 若令 T 为一个类型集合，且 T 在有限积下封闭，我们便可通过如下设定来定义一个幺半群 **集合**$_T$

$$\mathbf{Sets}_T\binom{d}{c_0,\dots,c_{n-1}} := \mathrm{Hom}(c_0 \times \cdots \times c_{n-1}, d),$$

其中，右侧的同态集是从集合的乘积$c_0 \times \cdots \times c_{n-1}$到集合 d的所有函数的集合。给定 $c \in T$， c上的恒等函数在 $\mathbf{Sets}_T\binom{c}{c} = \mathrm{Hom}(c,c)$ 中充当c-着色单位元。在这种情况下，我们可以显式定义定义 1 中的多重合成 $\circ_i$：给定 $f \in \mathrm{Hom}(c_0 \times \cdots \times c_{n-1}, d)$ 和$g \in \mathrm{Hom}(b_0 \times \cdots \times b_{m-1}, c_i)$，它返回函数 $f \circ_i g$，该函数作用于 $(n+m-1)$ 元组 $(x_0,\dots,x_{i-1},y,x_{i+1},\dots,x_{n-1})$ 时为

$$(f \circ_i g)(x_0,\dots,x_{i-1},\underline{y},x_{i+1},\dots,x_{n-1}) = f(x_0,\dots,x_{i-1},g(\underline{y}),x_{i+1},\dots,x_{n-1}).$$

定义 1 和 3 中上述未提及的其他部分，都可以利用以上定义和集合论中的基本事实对 $\mathbf{Sets}_T$ 加以证明。

3 在 Coq 中形式化建模幺半群

在定义项的集合$\mathcal{O}\binom{d}{c_0\ \dots\ ,,c_{n-1}}$ 时，在 Coq 中，定义 1 要求 d, c_i 来自集合 T。在本文的整个规范中，我们将用 Coq 内置的宇宙之一 T 替代**Type**。实际上，我们确实需要**Type**的真子集，但为简化起见，本文使用**Type**。如果需要限制为**Type**的子集，我们会在介绍 Coq 中的形式化模型后，简要讨论如何使用塔斯基宇宙来实现。

3.1 在 Coq 中编码幺半群

定义幺半群时首先要解决的问题，是为$\mathcal{O}\binom{d}{c}$给出形式化定义。

▶ 注释 6（Coq 中 $\mathrm{O}\binom{d}{c}$ 的定义）。非正式地说，幺半群 O 的一部分是一个集合族，它由一对 d : Type 和 $c := c_0,\dots,c_{n-1}$: Type 列表作为索引。由于这是一个集合族，用 Coq 中的记录类型来定义是很自然的。为此，我们在 Coq 中创建一个记录类型，并将其记为 Operad；它唯一的字段是一个类型签名为：Type $\to$ Type 列表 $\to$ Type 的函数。实例化 Operad 将得到一个函数 O : Type 列表 $\to$ Type $\to$ Type，因此 $\mathrm{O}\binom{d}{c}$ 会给出我们所需的项集合。

我们给出一个来自注释 6 的定义示例。

▶ **示例 7.** 第 4 节的目标是为 **集合**$_T$ 在 Coq 中的例 5 提供一个针对 $T =$ **Type**的版本；我们将这个幺半群记作 **Type**。在 Coq 中，若 $c = c_0,\dots,c_{n-1}$: **列表 Type**且 d : **Type**，则可以通过递归在 Coq 中定义如下内容：

$$\mathbf{Type}\binom{d}{c} := c_0 \to \cdots \to c_{n-1} \to d.$$

具体而言，*Type* 类型的项 $\binom{d}{c}$ 是 n元函数，其陪域由 c定义，返回类型为 d。

分发声明 A：已批准公开发布，不限分发

在 Coq 模型的其余部分中，我们也使用记录类型来表示构成幺半群的数据（如定义 1 所示），或表示这些数据所受的约束（如定义 3 所示）。定义 1 和定义 3 中的每一项都是必须满足的命题。我们首先详细说明如何在 Coq 中将定义 1 中的数据编码为命题。

▶ 注 8（Coq 中幺半群的数据）。1. O 中存在一个 c 色单位元（定义 1 中的 1）：对于所有 c : Type，都存在一个 $\mathbb{1}_c \in \mathsf{O}\binom{c}{c}$；2. 对于置换 σ 作用于 n 个字母，要求 $\mathsf{O}\binom{d}{c}$ 与 $\mathsf{O}\binom{d}{c\sigma}$ 之间存在双射（定义 1 中的 2）：对于所有 d : Type、c、c' : 列表 Type，且长度 c 至少为 1，并且 c 和 c' 互为置换时，$\mathsf{O}\binom{d}{c}$ 与 $\mathsf{O}\binom{d}{c'}$ 之间存在双射；3. 要求存在 $\circ_i$（定义 1 中的 3）；对于所有 i、n : $\mathbb{N}$、d、c_i : Type，b、c : 列表 Type，若 c 的长度为 n、$1 \leq n$、$i \prec n$，且 c 的第 n 个元素是 c_i，则存在一个类型为 $\mathsf{O}\binom{d}{c} \times \mathsf{O}\binom{c_i}{b} \to \mathsf{O}\binom{d}{c \bullet_i b}$ 的函数。

为了在注 8 中清楚地说明我们在 Coq 中的实现，有必要对如何在 Coq 中精确表达上述内容作几点说明：

▶注 9。1. 在此语境中，凡是用到双射一词，指的都是 Type 中的双射。也就是说，若 t, t' : Type，则存在函数 f : $t \to t'$, f' : $t' \to t$，使得 $f \circ f' = \mathrm{id}_{t'}$，且 $f' \circ f = \mathrm{id}_t$。这很容易在 Coq 中定义。2. 在 Coq 中，要构造一个命题，表明两个列表 c, c' 互为置换，可以使用 Coq 内置的类型 Permutation。它表示 Permutation $c\,c'$: Prop（其中 Prop 是 Coq 中所有命题的类型）。3. 列表上的操作 $\bullet_i$ 可以在 Coq 中这样定义：取 c 的前 i 个元素，连接列表 b，然后再将 c 的最后 $n-i-1$ 个元素连接到此前得到的列表上。4. 在注 8 的第 3 点中，我们需要使用 Coq 中的 nth 函数。该函数要求将一个默认元素作为参数之一，因此我们需要从 Type 中选取一个默认元素，并始终使用它。我们在 Coq 中选择单位类型，它用于表示单元素集合。

我们以类似的方式使用记录类型将定义 3 编码到 Coq 中，并将此记录类型记为 **OperadLaws**。然而，需要格外谨慎，部分原因是注4中的讨论。为说明这一点，下面我们将详细讨论在 Coq 中对水平结合公理的建模。

▶注释 10（Coq 中拟运算子的公理）。拟运算子中的水平结合公理（定义 3 中的 1）可以在 Coq 中通过首先列出一组参数来定义，我们将这组参数称为 P：

- n, m, ℓ, i, j : $\mathbb{N}$;
- d, c_i, c_j : **Type**;
- a, b, c : **列表Type**
- α : $\mathcal{O}\binom{d}{c}$, β : $\mathcal{O}\binom{c_i}{b}$, γ : $\mathcal{O}\binom{b_j}{a}$
- $2 \leq n$、$1 \leq m$和 $1 \leq \ell$;
- $i < j$ 和 $j < n$;
- c 的长度为 n，b 的长度为 m，而 a 的长度为 ℓ;

• c 的第 i 项是 c_i，而 c 的第 j 项是 c_j；

利用目前 P 中的内容，我们可以证明 $c \bullet_j b$ 的第 i 个元素是 c_i，并证明 $c \bullet_i a$ 的第 $(\ell - 1 + j)$ 个元素是 c_j；我们将这些证明添加到 P 中。更新 P 后，我们可以在 Coq 中陈述水平结合公理的形式化表述：对于构成 P 的所有参数，注4中的公式（2）成立，并且存在一个类型转换函数 $\mathcal{C}_{\mathbf{assoc}}$，使得

$$\mathcal{C}_{\mathbf{assoc}}\, P\, ((\alpha \circ_i \beta) \circ_{\ell-1+j} \gamma) = (\alpha \circ_j \gamma) \circ_i \beta$$

类型转换函数 $\mathcal{C}_{\mathbf{assoc}}$ 是必要的，因为我们在 Coq 中为每个 d： **Type** 和 c： **list Type** 都定义了 $\mathcal{O}\binom{d}{c}$ 是 Type 中的一个类型，而转换函数则提供了证明，表明方程 (1) 中的类型相等成立。然而，$\mathcal{C}_{\mathbf{assoc}}$ 的存在完全依赖于方程 (2) 中列表相等的证明。现在，方程 (2) 中的等式需要付出大量努力才能证明，而形式化这一公理最困难的部分就是提供其证明。

为定义 3 中所有其他公理提供形式化规范，并将其插入我们记录类型 **OperadLaws** 的字段中，遵循与上述相同的步骤：

1. 仔细整理出公理所需的正确参数集合 P；2. 补充任何可由当前 P 中的内容推导出的必要证明；3. 证明所需的任何列表相等关系成立（定义 3 中的所有公理都需要这一步）；4. 创建必要的类型转换函数。

关于我们模型中的选择，最后还有一点需要说明。

▶ 注 11。 在 [8] 幺半群的定义中指出，如果 $c = \emptyset$，来自集合 T 的符号列表为空，那么符号 $\mathcal{O}\binom{d}{\emptyset}$ 仍然有意义。请注意，在定义 1 中，我们不允许存在这样的符号，因为我们要求列表 c 是 非 空的。这样做的原因是，我们的主要应用依赖于在 Coq 中给出例 5 的一个版本。在 **集合**$_T$ 中，如果 $c = \emptyset$，那么空集合列表的乘积是一个单元素集合，$\{\bullet\}$，因此 **集合**$_T\binom{d}{\emptyset} \cong$ **集合**$_T\binom{d}{\bullet \{\}}$。在 Coq 中，我们可以通过令 c 为唯一元素是 $\mathbb{U}$ 的列表来建模这种情况： **Type**，即单位类型。

3.2 塔斯基宇宙

一种解决方案是将 T Type 的一个子集 在 Coq 中定义为 T 一个塔斯基宇宙。这会定义 T : **Type**，以及一个解释，使得 T 中的项可以被视为编码，对应实际类型。这样，类型 T 就是一个集合，并带有到 **Type** 的单射映射，这正是 **Type** 的一个子集所包含的数据。我们在 Coq 中实现此定义的方法如下：

1. 在 Coq 中定义一个带零元构造子的类型 $\mathcal{B}$，我们称之为 基本类型，其项称为 类型符号；
2. 定义类型的构造子 T，其中包括：具有签名的构造子 $\mathbf{Ty} : \mathcal{B} \to T$，它将基本类型编码为 T；

分发声明 A：已批准公开发布，不限分发

- 其他构造子可以表示积，例如 $\mathbf{p} : T \to T \to T$，或者 $\mathbf{fn} : T \to T \to T$，后者可以表示函数；

3. 为 $\mathcal{B}$ 在Type中的赋值，以及递归定义的解释函数$\mathbf{El} : T \to \mathbf{Type}$，它为每个 Type 中的 $t : T$赋予一个值。

我们明确举例说明这会是什么样子。

▶ **示例 12.** 我们在 Coq 中定义基本类型集合$\mathcal{B}$ 及其零元构造子N，U和 B。在 Coq 中，我们创建一个函数 $\mathbf{I}$ 来解释这些类型符号： N 被赋予 $\mathbb{N}$（自然数类型）； U 被赋予 $\mathbb{U}$（单位类型）； B 被赋予 $\mathbb{B}$，即布尔类型。

如果我们想在 T 中对积和函数建模，那么可以用以下构造子定义 T：

1. $\mathbf{Ty} : \mathcal{B} \to T$;2. $\mathbf{p} : T \to T \to T$;3. $\mathbf{fn} : T \to T \to T$。

现在，El 将通过以下递归给出嵌入 Coq 的方式：

1. $\mathbf{El}\,(\mathbf{Ty}\,t) \Rightarrow \mathbf{I}\,t$
2. $\mathbf{El}\,(\mathbf{p}\,t\,t') \Rightarrow \mathbf{El}\,t \times \mathbf{El}\,t'$
3. $\mathbf{El}\,(\mathbf{fn}\,t\,t') \Rightarrow \mathbf{El}\,t \to \mathbf{El}\,t'$

举一个 T 中代码的具体示例，我们有 $\mathbf{p}\,(\mathbf{Ty}\ N)(\mathbf{Ty}\ N) : T$；通过嵌入 El，它在 Coq 中是 $\mathbb{N} \times \mathbb{N}$ 的一个模型。

4 使用 OurModel 的证明

本节的目标是讨论一个形式化证明：Coq 中与例 5 等价的对象（我们记作 **Type**，并在示例 7 中定义）按照我们的模型是一个幺半群。

要形式化地证明 **Type** 是一个幺半群，我们首先需要为记录类型 **Operad**（见注释 6）中唯一字段里的函数下定义。接下来，在记录类型 **OperadLaws** 中，我们需要为 $c :$ **Type** 定义 c-着色单位元（定义 1 中的第 1 项），提供证明以表明 $\mathbf{Type}\binom{d}{c}$ 在 c 的重排下保持不变（相差一个单射）（定义 1 中的第 2 项），定义多重合成函数（定义 1 中的第 3 项），并证明定义 3 中的所有公理均成立。将这些赋值和证明整合起来，就得到一个类型为 **Operad** 和 **OperadLaws** 的项，从而得到我们所需的形式化证明。

在示例 7 中，我们给出了 **Type** 上的 **Type**所需函数的定义：给定$c = c_0$、.....、 $c_{n-1} :$ **list** Type 以及 $d :$ Type，我们写作：

$$\mathbf{Type}\binom{d}{c} := c_0 \to \cdots \to c_{n-1} \to d,$$

它是以 n 为元、陪域由 c 定义且返回类型为 d 的函数类型。我们对 **Type** 上的 **OperadLaws** 的实例化将始终使用此定义。

$\mathrm{Type}\binom{d}{c}$ 的右侧通过一个递归函数定义，我们将其记作**arr**（箭头的缩写），其类型签名为 **列表**Type $\to$ Type $\to$ Type。特别地，我们定义 $\mathbf{arr}\ \emptyset\ d = d$（其中 $\emptyset$ 是空列表）。

4.1 在 Coq 中实现 Type 的 Data

接下来，我们将在一系列笔记中讨论定义 1 在 Coq 中针对 **Type** 的实现，以及为此实现而开发的工具。

▶ 注 13（c-着色单位元于 **Type**）。如果 c 只有一个条目 c： **Type**，那么**Type** $\binom{c}{c} = c \to c$，它是所有定义域和值域均由 c给出的函数的类型。于是 $\mathbb{1}_c$ ： =单位元$_c$，即 c 上的恒等函数。

▶ 注 14（**Type**$\binom{d}{c} \cong$ **Type**$\binom{d}{c\sigma}$）。我们的目的是在 Coq 中给出例 5 的等价形式，以及幺半群 $\mathbf{Sets}_T$ 对应的同构，即

$$\mathrm{Hom}(c_0 \times \cdots \times c_{n-1}, d) \cong \mathrm{Hom}(c_{\sigma(0)} \times \cdots \times c_{\sigma(n-1)}, d),$$

给定 $\underline{c} = c_0, \ldots, c_{n-1}$ 以及一个作用于 σ 上的 n 字母置换。Coq 语境下的同构要求我们构造上述两个集合之间的双射。根据定义 8 和注 9 中的评论，对于**Type** ，我们可以将其翻译成 Coq 表述如下：对所有 d: **Type**，c，c'： **Type列表**，其长度 c至少为1，且**置换** $\tau\, c'$，存在**Type**$\binom{d}{c}$与**Type**$\binom{d}{c'}$ 之间的双射： **Type**。

我们可以在 Coq 中对 $\bar{c}$ 的长度进行归纳法证明这一点，并结合一些关于函数复合和同构在此情形下性质的前置引理。

▶ 注释 15（$\circ_i$ 用于 **Type**）。最后，我们需要在 Coq 中为 **Type** 编写定义 8 的（3）。将 **Type**$\binom{d}{c}$ 写成柯里化形式，而不是照搬例 5 中 $\mathbf{Sets}_T\binom{d}{c}$ 的逐字翻译，就能通过部分应用提供所需的灵活性，从而在 Coq 中根据定义 1 的（3）为 **Type** 实现多重合成函数 $\circ_i$。我们在 Coq 中定义的最重要部分，是定义一个递归函数 **compose** ，其类型签名为

$$\mathbf{arr}\, \underline{c}'\, (t \to t') \to \mathbf{arr}\, \underline{b}\, t \to \mathbf{arr}\, \underline{c}'\, (\mathbf{arr}\, \underline{b}\, t');$$

其中 t，t': **Type**，且 $\underline{c}'$，$\underline{b}$: **列表 Type**。若 $\underline{c} = c_0, \ldots, c_{i-1}, c_i, c_{i+1}, \ldots, c_{n-1}$，我们令$c' = c_0, \ldots, c_{i-1}$，$t = c_i$，且 $t' = c_i \to c_{i+1} \to \cdots \to c_{n-1} \to d$，只要写入正确的类型推断，这便给出了所需的多重合成算子 $\circ_i$ ，用于 **Type**。

4.2 在 Coq 中实现 Type 的公理

定义 1 为构建幺半群的公理奠定了基础，这些公理见定义 3；我们在此讨论其在 Coq 中的实现。定义 3 列出了若干公理，正如注释 10 所述，我们将讨论重点放在水平结合公理（定义 3 中的公理 1）上，因为证明 **Type** 满足定义 3 中的所有其他公理，也可以在 Coq 中通过类似的步骤完成。

我们首先遇到的难题是：我们在注释 15 中对 $\circ_i$ 的定义在数学上正是我们想要的，但 Coq 不会自动识别类型相等：

$$\underline{c}'\, (\mathbf{arr}\, \underline{b}\, t') = \mathbf{arr}(\underline{c} \bullet_i \underline{b})\, d,$$

其中 $\underline{c} = c_0, \ldots, c_{i-1}, c_i, c_{i+1}, \ldots, c_{n-1}$, $c' = c_0, \ldots, c_{i-1}$, $t = c_i$, $t' = c_i \to c_{i+1} \to \cdots \to c_{n-1} \to d$，使得（参见定义 1 的第 3 条） $c \bullet_i b = c_0, \ldots, c_{i-1}, b, c_{i+1}, \ldots, c_{n-1}$。由于我们

分发声明 A：已批准公开发布，不限分发

要求 $f\colon \mathbf{Type}\binom{d}{c}$，$g\colon \mathbf{Type}\binom{c_i}{b}$，且 $f \circ_i g\colon \mathbf{Type}\binom{d}{c \bullet_i b}$，这就带来了一个问题。与注释 10 中一样，解决方法是使用类型转换函数。 – –

我们在 Coq 中对**Type** 进行形式化的其余部分，堪称类型转换的精彩展示。下文将讨论我们在这一证明中使用的工具。首先，我们给出在 Coq 中使用的类型转换形式化定义。

▶ **定义 16.** 给定 A，$B : \mathit{Type}$，以及一个等式 $A = B$，类型转换$\mathcal{C}_{A=B}$ 是一个函数，对 $a : A$，有 $\mathcal{C}_{A=B}\, a : B$。

为了处理证明**Type** 满足定义 3 中公理的过程中出现的类型转换，我们在 Coq 中证明了一些关于类型转换的一般事实，下文将对此进行讨论。

▶ 注释 17（Coq 中定义 3 的类型转换）。1. 两个类型转换的复合仍是一个类型转换：给定类型的等式 $A = B$ 和 $B = C$，有 $\mathsf{C}_{B=C} \circ \mathsf{C}_{A=B} = \mathsf{C}_{A=C}$。2. 使用类型等式 $A = A$ 进行的类型转换（即在 Coq 看来相同的两个类型之间的类型转换）等于单位元：对于 $a : A$，$\mathsf{C}_{A=A}\, a = a$。3. 同一对类型之间的两个类型转换（即都使用类型等式 $A = B$）相等：对所有 $a : A$，$\mathsf{C}_{A=B}\, a = \mathsf{C}'_{A=B}\, a$。4. 对函数进行类型转换后再将其应用于一个参数，等同于将原函数应用于经过类型转换的参数：若 $f : B \to C$ 且 $a : A$，则 $(\mathsf{C}_{A\to B=B\to C}\, f)\ a = f\ (\mathsf{C}_{A=B}\, a)$

这些事实简化了证明**Type** 满足定义 3 中拟运算子公理的过程，因为这需要对多个类型转换进行操作。例如，注释 17 中的 2 和 3 确保我们无需追踪这些操作对类型转换所依赖的公式造成的影响，因为要证明相等，我们只需确保涉及的类型 匹配即可。

现在，我们将讨论如何运用注释 17 中证明的事实，并说明如何在证明**Type**满足水平结合公理（定义 3 中的 1）时使用这些事实。第一步是证明以下涉及**复合** 函数的关键等式引理：

$$\mathbf{复合}(\mathcal{C}(\mathbf{复合}(\mathcal{C}\ \alpha)\ \beta))\,\gamma = \mathcal{C}(\mathbf{复合}(\mathcal{C}(\mathbf{复合}(\mathcal{C}\ \alpha)\ \gamma))\,\beta)。 \tag{4}$$

这里，α，β，γ是幺半群**Type**中类型适当的项，类型转换公式已从记法中省略。我们可以通过对相应的列表使用归纳法，并多次应用注释 17 中的 4，来证明公式 4 中的等式。请注意，公式 4 本质上就是**Type**中的水平结合公理，事实确实如此：为了证明水平结合公理，我们运用注释 17 中的工作对类型转换进行操作，直到它们与公式 4 中的等式相符。

在 Coq 中为 **Type** 证明定义 3 中其余公理，遵循类似的模式：证明给定公理所对应的模式对 **compose** 成立，然后根据注释 17 适当地处理类型转换，从而得到所需的公理。

分发声明 A：已批准公开发布，不限分发

5 相关工作

在 [5], 中，作者使用 Cubical Agda 对一种更简单的幺半群进行了形式化；Cubical Agda 是 Agda 的扩展，加入了立方类型论。立方类型论是同伦类型论的一种替代方案，更适合构造性解释。因此，要充分理解 [5] 中的 Operad 实现，需要掌握同伦类型论的一个变体，并了解如何在 Agda 中使用其实现。

此外，Agda 缺乏显著的自动化能力，因此，例如，要展示我们在第 4 节中的证明，就需要多做很多工作。不过，我们认为将我们的工作转换为 Agda 并非不可能，只是需要编写更多样板代码（例如，手动编写结构归纳策略）。

我们还想将我们工作中的形式化内容与 [5] 的工作进行比较。他们在 [5] 中用来指称幺半群的是一个幺半群，它带有一个类型集合 T，满足 $|T| = 1$。特别是，$\mathcal{O}\binom{d}{c}$ 可以由自然数参数化，因此我们可以写作 $\mathcal{P}(n) := \mathcal{O}\binom{d}{c}$，若 $|c| = n$，其中 $c = c_0, \ldots, c_{n-1}$，且 $\_ c_i \neg= d$。此外，单位元唯一（$1 \in \mathcal{P}(1)$），函数 $\circ_i$ 的类型为 $\mathcal{P}(n) \to \mathcal{P}(m) \to \mathcal{P}(n + m - 1)$，并且定义 3 中的结合公理得到了显著简化。我们还注意到 [5]，他们将单色算子定义为满足 [8] 中给出的等变公理。

参考文献

1 John C. Baez 和 John Foley。用于设计系统之系统的 Operad。CoRR，abs/2009.12647，2020年。网址：https://arxiv.org/abs/2009.12647, arXiv:2009.12647。2 John C. Baez 和 Nina Otter。Operad 与系统发育树。Theory Appl. Categ.，32：第 40 号论文，1397–1453，2017年。3 John D. Foley、Spencer Breiner、Eswaran Subrahmanian 和 John M. Dusel。用于复杂系统设计规范、分析与综合的 Operad。CoRR，abs/2101.11115，2021年。网址：https://arxiv.org/abs/2101.11115, arXiv:2101.11115。4 Samuele Giraudo、Jean-Gabriel Luque、Ludovic Mignot 和 Florent Nicart。Operad、准序与正则语言。Adv. Appl. Math.，75：56–93，2016年。doi:10.1016/j.aam.2016.01.002。5 Brandon Hewer 和 Graham Hutton。同伦类型论算子。编程语言原理研讨会，2023年。网址：http://www.cs.nott.ac.uk/~pszgmh/operads.pdf。6 Sophie Libkind、Andrew Baas、Evan Patterson 和 James Fairbanks。动力系统的 Operad 建模：数学与计算。理论计算机科学电子论文集，372：192–206，2022年11月。doi:10.4204/eptcs.372.14。7 David Spivak。接线图的 Operad：为数据库、递归和即插即用电路形式化图形语言。2013年。网址：https://arxiv.org/pdf/1305.0297.pdf,arXiv:1305.0297v1。8 Donald Yau。接线图的 Operad，数学讲义第 2192 卷。Springer，Cham，2018年。doi:10.1007/978-3-319-95001-3。

分发声明 A：已批准公开发布，不限分发