arXiv:2302.00744v1 [math.CT] 1 Feb 2023

DSL 组合的形式代数框架*

Zachary Flores、Angelo Taranto 和 Eric Bond

Two Six Technologies，美国弗吉尼亚州阿灵顿 22203，
zachary.flores@twosixtech.com
angelo.taranto@twosixtech.com
eric.bond@twosixtech.com

摘要。 我们探讨如何使用代数结构构建形式框架，以对一种元语言建模。该元语言能够编写和组合领域特定语言的抽象，并使这些抽象具备互操作性。此形式框架旨在验证元语言的组合性质。本文将探讨这一形式框架的构建，以及它与我们团队在 DARPAV-SPELLS 项目中的工作之间的联系；我们通过已开发的流程完成 V-SPELLS 上的验证任务。本文旨在对这一验证流程进行全面概述。该流程可分为四个主要组成部分：首先，在 Coq 中建立元语言的形式模型；其次，在 Coq 中给出所选代数结构的规约；第三，在 Coq 中实现这些代数结构的具体实例，并证明该实现按照第二步的规约确实构成一种代数结构；最后，在 Coq 中证明，第一步中元语言的形式模型是第三步所实现结构的一个实例。

关键词： 形式化验证 · 应用范畴论 · 领域特定语言 · Coq

免责声明： 文中所表达的观点、意见和/或研究结果均为作者所有，不应被解读为代表国防部或美国政府的官方观点或政策。

1 引言

DARPA V-SPELLS（大型遗留软件的已验证安全性与性能增强）计划旨在提供开发者可用能力，使其能够逐步增强大型遗留代码库中的软件组件，并使用能够与系统其余部分安全组合的经验证代码。

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 提出的问题时，我们正在开发的工具 LUMOS，

* 由 DARPA V-SPELLS HR001120S0058 资助。**分发声明 A**：已批准公开发布，无限制分发

首先运用静态分析、自然语言处理和动态分析工具处理遗留源代码，以便从构成源代码的领域特定语言中生成领域特定语义模型（DSSMs）。这些 DSSM 将使用一种我们称为元 *DSL*的语言生成。为了提供 V-SPELLS 所要求的遗留代码补丁，这些领域特定语言的模型必须以非常具体的方式组合。为了确保组合的正确性，我们借鉴范畴论中的若干思想，采用代数框架进行验证，其中建模的关键结构称为厄 *operad*。运算子在应用数学中的作用日益重要（参见 [1], [2], [3], [4], 和 [5]），我们发现它们为 V-SPELLS 的验证需求提供了出色的数学模型。

本文重点简要概述我们对领域特定语言进行形式化建模的方法，以及将在形式化过程中使用的数学框架。为此，我们在第二节首先讨论如何对领域特定语言进行形式化建模，以及在元 DSL 的编程语言语法与语义中，如何在 Coq 中表示领域特定语言的组合；第三节则讨论我们验证所需的代数框架所涉及的全部数学基础。

2 领域特定语言的形式模型与组合

2.1 领域特定语言的形式化

LUMOS 主流程中的元 DSL 语法和语义将使用 OCaml 编写，而用于验证的模型将使用 Coq 编写。鉴于 Coq 可以转换为 OCaml，我们预计形式化建模与 LUMOS 主流程之间会相互影响。在验证工作中，我们的首要目标是在元 DSL 的语境下，为领域特定语言的语法和语义提供一个形式模型，因为领域特定语言的概念对于修补遗留代码至关重要。在 Coq 中对元 DSL 建立的模型里，我们将领域特定语言视为一组类型，以及定义在这些类型上的一组有限元函数（即有限元数的函数），并可据此构造其他有限元函数。下一个例子将对此进行说明。

例*1.* 令 DSLU := {{自然数, str}, {**fprint**, **finput**}} 表示一种领域特定语言的抽象，该语言用于编写补丁，以修改作用于用户名的函数。函数 **fprint** n str 接收一个数字和一个用户名，并打印用户名的前 n 个字母；另一个函数 **finput** str 接收一个用户名，并将其作为字符串存储在内存中。

在 DSLU 中，我们可以创建一个名为 **firstn** 的函数，它通过组合**fprint** n (**finput** str)，打印写入内存的函数的前 n 个字母。

现在，为了给出该模型的 Coq 形式化定义，我们首先要明确元 DSL 所编写的每种 DSL D 所属的类型集合 D_T 是什么。为此，我们在 Coq 中将 类型标记 构造为一种归纳类型，

然后提供一个充当嵌入的函数，用来指定我们为每个类型标记分配的 Coq 类型系统中的元素。例如，我们的类型标记集合可以写作：Inductive BNU : Type := b | n | u 。接着，我们可以在 Coq 中编写一个函数，分别将 b、n 和 u 映射到 Coq 类型 bool、nat 和 unit。这为我们在求值表达式时推理其类型提供了一种方法。这两部分构成了我们所称的 类型宇宙 （这就是所谓的塔斯基式类型宇宙，详情见 [6] ）。在 Coq 中，我们使用记录类型来表示任意类型宇宙，并将其记作 Univ。这是 DSL 定义的第一个组成部分。

为了表示我们的函数集合 D_F，在我们的领域特定语言 D 中，我们通过函数名和其 类型签名 来指定函数，同时还包括该函数对 *Coq* 中的项 的作用。要指定函数名，其过程类似于创建类型宇宙：将名称作为归纳类型记录在 Coq 中（作为<注释性标记>），例如 funNames : Type，然后在 Coq 中提供一个适当的函数，将 funNames 中的标记匹配到类型签名。

指定类型签名集合的过程也类似于指定类型宇宙：我们将函数的定义域记录为来自类型宇宙的类型标记列表，并将值域记录为来自类型宇宙的类型标记。我们在 Coq 中的类型签名集合被指定为一种记录类型，记作 Sig。举一个简单例子，类型宇宙 BNU 中的自然数相等具有类型签名 n -> n-> b，而其类型签名 eqNatSig : Sig BNU 则记录为两个组成部分：定义域列表 [n, n]，以及返回类型 b。

最后，我们还需要定义函数对 Coq 中的项的作用，具体做法是将函数名称映射到实际的函数类型，并以其签名作为输入，从而通过构造确保实际函数类型的签名与之匹配。在 Coq 中，这可以表示为一个函数：将 funNames 中的名称映射到根据该名称的签名创建的 Coq 函数类型。

这些就是在 Coq 中将 DSL 规约定义为记录所需的组成部分：类型宇宙，由类型为 Univ 的对象指定，在我们的示例中是 BNU；一组函数名称，在 Coq 中由归纳类型定义，在我们的示例中是 funNames；一组类型签名，以及一个将函数名称映射到给定类型签名的映射；还有函数名称、类型签名和项定义之间恰当的配对。所有这些组成部分都会编译为 Coq 中的记录类型。在我们的示例中，我们将这组内容称为 DSLBNU，并在 Coq 中记作 DSLBNU:DSL。

2.2 DSL 组合模型

元 DSL 的主要用途之一，是组合在 LUMOS 流程中经过前序模块处理后，以元 DSL 编写的 DSL 抽象。在验证过程的这一部分，接下来我们将使用 DSL 的形式化模型，对组合两个 DSL 的概念进行形式化。我们将这种组合 DSL 的概念称为*DSL* 粘合。

分发声明 A：批准公开发布，不限分发

设 D'、D'' 是两个 DSL。映射 Φ： $D' \to D''$ 由函数 $\varphi : D'_T \to D''_T$ 构成，并且对于给定的 $f \in D'_F$，存在一个诱导函数 $f_\varphi \in D''_F$，该函数保持 f 对 D'_T 的作用。如果要组合两个 DSL 抽象 D' 和 D''，而组合过程中有若干类型表示等价的类型（例如，它们各自都包含一份自然数类型），我们希望在组合结果中将这些副本合并为一个类型。为此，我们令 Z 为一个 DSL，其类型取自一个通用的固定集合 $\mathcal{T}$（在 LUMOS 流程中，这将是 OCaml 中所有类型的集合），并首先构造 DSL 的映射： $D' \leftarrow Z \to D''$。然后， D' 与 D'' 的组合相对于Z，记作 D，定义如下列交换方块（亦称推出图）所示：

$$\begin{array}{ccc} Z & \xrightarrow{\Phi''} & D'' \\ {\scriptstyle \Phi'}\downarrow & & \downarrow{\scriptstyle h''} \\ D' & \xrightarrow{h'} & D \end{array}$$

其中， D、h、h''对于此性质是泛的，并且还使D 在唯一同构意义下唯一。

此构造的结果是 $D_T := D'_T \cup D''_T / \sim_T$，其中等价关系 $\sim_T$ 由映射 $D'_T \leftarrow Z_T \to D''_T$诱导。对于函数集合 D'_F和 D''_F，我们同样通过根据映射 $D'_F \leftarrow Z_F \to D''_F$识别函数集合，构造 $D_F := D'_F \cup D''_F / \sim_F$。

虽然这是一个一般的范畴构造，但如果我们想在编程语言的语境中对领域特定语言进行有意义的组合，就要注意， Z 不能被视为任意领域特定语言，下一例将对此进行说明。

例*2*。 考虑例1中的 DSLU，以及由 {{int,struct},{**fprint**,**ffields**}}给出的领域特定语言，记为 DSLP。这里，DSLP$_T$ 中的类型包括整数类型 int，也包括泛型结构体类型；就我们的目的而言，它的作用类似于 Coq 中的记录。DSLP 中的函数**fprint** 与 DSLU 中的**fprint** 行为相同，但函数**ffields** 会返回结构体的字段。如果在构造 DSLU 与 DSLP 的粘合时，尝试将 DSLP 中的类型 struct 与 DSLU 中的 str 粘合起来，那么组合中的 **ffields** 功能（或其缺失）可能不再符合预期。例如，假设我们将新粘合得到的类型命名为 structstr，那么如果存在输入为 str 的 structstr 类型实例，并且我们对这些实例使用**ffields** ，就会出现问题。

这个示例说明了为什么需要对领域特定语言 Z 的内容加以限制，以避免在给定上下文中将不应拼接在一起的类型或函数拼接在一起。我们概述一下解决这一问题的一般方法：首先，我们要求提供一组基本类型，记作 $\mathcal{T}$（如前所述，我们可以将其视为类型集合

分发声明 A：批准公开发布，不限分发

在 OCaml 中），并且，如果要通过领域特定语言 Z 将领域特定语言 D' 和 D'' 中的两个类型（例如 t' 和 t''）拼接起来，我们就要求以 Coq 中证明义务的形式证明：存在一个 $t \in \mathcal{T}$，并且 t 与 t'、t 与 t'' 之间分别存在等价关系。这些等价关系还必须与领域特定语言 D' 和 D'' 中的操作相容。

值得注意的是，这为我们通过另一个领域特定语言 Z 将两个领域特定语言粘合在一起的情况提供了数学描述。然而，如果有多个领域特定语言，且它们之间的关系更为复杂，我们该如何进行粘合？下一节将讨论这种情形的数学描述。

3 我们形式模型的代数框架

3.1 领域特定语言的代数模型

按照第 2 节中的数学定义，我们将一个领域特定语言视为由一组类型以及作用于这些类型的有限元函数构成的集合，并加入以不受限制的方式将已有函数复合成新函数的能力。我们发现，能够非常精确地对这种行为建模的代数结构是一个对称有色算子，我们将其简称为厄 *operad*。厄 operad 的形式化定义很长，因此我们仅列出本文必需的所有细节，其余内容请参阅 [7] 。

定义 1. 一个厄 *operad* $\mathcal{O}$ 由一组类型构成，记作 T；对于每个 $n \geq 1$, $d, \underline{c} := c_0, \ldots, c_{n-1}$，还有一组项 $\mathcal{O}\binom{d}{\underline{c}}$ 以及，

- 对于每个 $c \in T$，都有一个称为 c着色单位的元素 $\mathbb{I}^c \in \mathcal{O}\binom{c}{c}$；
- 对于每个 $i \in \{0, \ldots, n-1\}$，都有一个算子

$$\circ_i : \mathcal{O}\binom{d}{\underline{c}} \times \mathcal{O}\binom{c_i}{\underline{b}} \to \mathcal{O}\binom{d}{c_0, \ldots, c_{i-1}, \underline{b}, c_{i+1}, \ldots, c_{n-1}}$$

以及关于结合性、单位性和对称性的公理约束。

厄 *operad* 的算子态射 $F : \mathcal{O} \to \mathcal{O}'$ 由一个类型映射和一个项映射组成，且这两个映射与着色单位、算子 $\circ_i$以及所有公理约束相容。这使厄 *operad* 构成一个范畴，我们将其记为 $\mathbb{O}$。

虽然这看起来是一个极其抽象的定义，但下面的例子有助于阐明定义 1 这一抽象概念的根源。

例*3*。 若令 T 为一个集合族，我们可以通过如下设定定义一个厄 operad **集合**$_T$

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

其中，右侧的同态集是从集合$c_0 \times \cdots \times c_{n-1}$到集合 d的所有函数组成的集合。在这种设定下，我们可以明确地定义算子
$\circ_i$来自定义 1，该定义在给定 $f \in$ Hom($c_0 \times \cdots \times c_{n-1}, d$) 和 $g \in$时返回

分发声明 A：批准公开发布，不限分发

Hom($b_0 \times \cdots \times b_{m-1}, c_i$)，函数 $f \circ_i g$ 作用于 $(n+m-1)$ 元组 $(x_0, \ldots, x_{i-1}, \underline{y}, x_{i+1}, \ldots, x_{n-1})$，形式如下：

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

现在，在元 DSL 的形式模型中，每种领域特定语言都应在与例3类似的框架下建模，这一点已十分清楚。因此，我们说这种领域特定语言被建模为一个集合的厄 *operad*。为说明我们的意思，我们给出一个更具体的例子。

示例 *4.* 设 DSLU $:= \{\{$自然数, str$\}, \{$**fprint**, **finput**$\}\}$ 如例1所示。我们可以将 DSLU 规整为一个集合的厄 operad，并使用例3中的记法表示为：**fprint** $\in$ DSLU$\binom{\text{str}}{\text{str nat,str}}$，以及**finput** $\in$ DSLU$\binom{(\text{str})}{\text{str}}$。特别地，我们可以写作**firstn** $=$ **fprint** $\circ_1$ **finput** $\in$ DSLU$\binom{\text{str}}{\text{str nat, str}}$。

3.2 元 DSL 的代数模型

此时，我们希望将每种领域特定语言表示为元 DSL 的形式代数结构中的一个集合的厄 operad。这引出了下一个问题：元 DSL 的恰当代数表示是什么？正如我们所指出的，元 DSL 的主要用途之一是组合领域特定语言的抽象，因此在这一框架下，我们要寻找一种数学对象，它能够提供组合两个厄 operad 并生成另一个厄 operad 的方法。

我们采用的方法是使用图表范畴的概念。首先，我们给出范畴中的图表 $\mathcal{C}$ 的数学定义。具体来说，若 $\mathcal{J}$ 是一个范畴，则形状为 $\mathcal{J}$ 且位于范畴 $\mathcal{C}$ 中的图表是一个协变函子 $D\colon \mathcal{J} \to \mathcal{C}$。此外，固定 $\mathcal{J}$后，这些对象构成一个范畴，我们称之为形状为 $\mathcal{J}$ 且位于 $\mathcal{C}$中的图表范畴。我们也用 $\mathcal{C}^{\mathcal{J}}$表示这个范畴，这是从 $\mathcal{J}$ 到 $\mathcal{C}$的协变函子范畴的标准记法。接下来的示例对我们的讨论很重要。

示例 *5*。令 $\mathcal{J}$为一个范畴，其对象为 $-\mathbf{1}$、$\mathbf{0}$、$\mathbf{1}$，非恒等态射由图表 $-\mathbf{1} \leftarrow \mathbf{0} \rightarrow \mathbf{1}$给出。若 $\mathcal{C}$ 是一个范畴，则图表 $D\colon \mathcal{J} \to \mathcal{C}$ 的像由 $C' \xleftarrow{f} Z \xrightarrow{g} C''$ 在 $\mathcal{C}$中给出。这样的图表称为跨度。

数学与形式化正是在这里交汇：我们可以取示例 5 中一个跨度关于 Z的范畴论推出。它具有与我们在第 2 节中描述的 DSL 组合的推出相同的性质，特别是它在唯一同构意义下唯一（参见 [8] 第 III 章第 3 节）。

为了进一步阐述将多个领域特定语言结合起来并处理更复杂关系的思路，我们令 Diag($\mathbb{O}$) 为一个范畴，其中对象可以是形状为 $\mathcal{J}$、位于 $\mathbb{O}$ 中的图表，前提是 $\mathcal{J}$ 具有有限多个对象。我们希望元 DSL 生成的概念是 Diag($\mathbb{O}$) 中某个对象的余极限 （参见 [8] 第 III 章第 3 节）。也就是说，我们可以得出结论：元 DSL 可以形式化为范畴对象 Colim : Diag($\mathbb{O}$) $\to \mathbb{O}$。

分发声明 A：批准公开发布，不限分发

参考文献

1. Libkind S., Baas A., Patterson E., Fairbanks J., 动力系统的运算子建模：数学与计算，*2021* 年应用范畴论会议论文集（2021）2. Foley J.D, Breiner S., Subrahmanian E., 和 Dusel J. M., 复杂系统设计规范、分析与综合的运算子，英国皇家学会会刊（2021）3. Baez J.C., Foley J., 用于设计系统之系统的运算子，美国数学学会通告，第 68 卷，第 6 期，第 1005–1007 页（2021）4. Baez J. C., Otter N., 运算子与系统发育树，范畴论及其应用，第 32 卷第 40 期，1397–1453（2017）5. Giraudo S., Luque J., Mignot L., Nicart F., 运算子、拟序与正则语言，应用数学进展，75，56–93（2016）6. Martin-Löf, P., 类型的直觉主义理论：谓词部分，逻辑与数学基础研究，73–118，（1975）。7. Yau D., 接线图的运算子，施普林格数学讲义，13–37（2017）。8. Mac Lane S., 工作数学家的范畴论，第 2 版。施普林格，纽约（1997）。

分发声明 A：批准公开发布，不限分发