Egg-smol Python：用于等价图的 Pythonic 库

Saul Shanabrooks.shanabrook@gmail.com

arXiv:2305.04311v2 [cs.PL] 5 Apr 2024

摘要

等价图已成为一种用途广泛的数据结构，可通过等式饱和等技术应用于程序合成、优化和验证。本文介绍了实验性 egg-smol 库的 Python 绑定，旨在将等价图的优势带入 Python 生态系统。这些绑定提供了高层次、符合 Python 惯例的应用程序接口，为 Python 用户带来易于使用且熟悉的界面。通过将等价图技术与 Python 相结合，我们希望促进科学计算和机器学习社区各领域之间的协作与创新。我们探讨了 Python 绑定对 Python 用户和现有 egg-smol 用户的优势，以及未来可能的发展方向。

ACM 参考文献格式：

Saul Shanabrook。2023。Egg-smol Python：用于等价图的 Pythonic 库。载于 *PLDI'23* 论文集——*EGRAPHS* 研讨会。ACM，纽约州纽约市，4 页。

1 引言

等价图近来作为一种多用途数据结构日益受到欢迎，并通过等式饱和及相关技术应用于程序合成、优化和验证。开源等式图实现的数量不断增加，使得利用这项技术构建面向开发者的工具变得越来越可行。得益于丰富的库和工具生态系统，Python 已成为科学计算和机器学习领域的首选语言。因此，将等式图技术融入 Python 生态系统，能够为在这些领域工作的开发者和研究人员带来显著价值。

现有的等式图库，例如 egg Rust 库 [9]，主要通过 Rust 库提供接口，并支持使用宏以及将自定义对象用作表达式，前提是这些对象实现了用于遍历的特定接口。egg-smol 库在 egg 库之后问世，且受其启发，相较于前身[1]做出了一些改进。主要区别之一是，它主要通过一种自定义的、类似 Lisp 的 S 表达式语言提供接口，并提供具体类型，而不是允许将任意 Rust 对象作为表达式，只要它们实现了相应特征即可。egg-smol 还内置了对函数和表达式进行类型标注的支持。

[1]"egg-smol（暂定名称）是针对这一问题及 egg 其他一些不足之处的一种全新方案。目前尚不保证其会稳定下来或并入 egg。" Max Willsey [8]

PLDI'23 - EGRAPHS 研讨会，*2023*年*6*月*16*日至*21*日，佛罗里达州奥兰多2023.

Python 中已有一些等价图库，例如 Quiche [6] 和 snake-egg [5]，，后者是 egg 库的绑定。科学计算生态中的许多库都在寻求一种表示方式，用于表达和重写表达式。例如，用于在 Python 中执行分布式计算的库 Dask [3],一直在寻找一种表示高层表达式对象并对其执行优化（如重写 [4]）的方法。另一个库 Ibis[1], 提供类似 pandas 的 DataFrame 接口，同时又能编译为 SQL，也有类似需求。他们目前正尝试自行实现等式图库，此前他们曾尝试使用 snake-egg [7]。如果不同的库能够共享一个标准库，就能整合维护者的时间和精力；而使用等价图则能带来相较于传统重写系统的优势，例如无需严格遵循规则顺序。

如果底层等式图库能被多个相近领域的库使用，那么将等式图集成到 Python 中也能让终端用户受益，使用户或其他库能够定义这些库中所定义构造之间的转换。这样，终端用户便可以继续使用自己选择的库编写业务逻辑，同时利用重写规则将其转换为另一种表示形式，以便进行优化或执行。

2 egg-smol 的 Python 绑定

本文介绍了 egg-smol 库的 Python 绑定，主要提供高级构造，以一种符合"Python 风格"的方式映射 S 表达式库[2]。这一高级应用程序接口构建于较低层级的应用程序接口之上，并通过以下模块向 Python 暴露`egg_smol.bindings`模块，该模块支持 Rust 应用程序接口的所有基本功能。这些绑定借助 PyO3 Rust 库创建 [2]。

Python 应用程序接口与 S 表达式应用程序接口的对比见图1和2，图中实现了一个 e-graph，用于检验两个算术表达式是否等价。

Python 绑定使用函数、类和方法等原生 Python 构造，并配有类型注解。这些构造会在 egg-smol 库中转换为 sort 定义和函数定义。Python 绑定与 S 表达式语言的一个显著区别是，Python 中的变量必须指定类型，而 S 表达式语言中则不需要。

Python 中对变量进行类型标注的要求源于所有表达式都必须具有已知类型。这

[2]该库可以通过 `pip install egg-smol` 安装，源代码位于 https://github.com/metadsl/egg-smol-python。文档中还提供了入门教程。

```
@egraph.class_类 Num(BaseExpr ):定义 __init__(self ,
value: i64Like) ->  None: ... @classmethod定义 var(cls ,
name: StringLike) ->  Num: ...定义 __add__(self , other:
Num) ->  Num: ...定义 __mul__(self , other: Num) ->
Num: ... # 表达式1 = 2* (x + 3) 表达式1 =
egraph.define( "expr1", Num (2) * (Num.var("x") +
Num(3)), ) # 表达式2 = 6 + 2* x 表达式2 =
egraph.define( "expr2", Num (6) + Num (2) * Num.var("
x"), ) a, b, c = 变量_("a␣b␣c", Num) i, j = 变量_("i␣j",
i64) egraph.register( rewrite(a +b).to(b + a), rewrite(a
* (b + c)).to((a * b) + (a * c)), rewrite(Num(i) +
Num(j)).to(Num(i + j)), rewrite(Num(i) *
Num(j)).to(Num(i * j)), ) egraph.run (10)
egraph.check(eq(expr1 ).to(expr2 ))
```

图 1. 高级 Python 应用程序接口

这有助于静态分析和代码补全，因为它能根据类型帮助消除歧义，明确允许使用哪些方法。此外，在 S 表达式语言中，所有内容都以函数形式调用，因此无需使用带类型的变量。在 Python 中，我们允许类定义同名方法，但将其分别转换为两个不同的 egg-smol 函数。这就要求在将每个表达式发送给 egg-smol 之前，先确定它所属的类，以便知道应使用哪些方法。

Python用户的三大优势

egg-smol 库的 Python 绑定为 Python 用户带来了多项优势。首先，使用 Python 用户熟悉的内置结构，让该库更易于接触和使用。此外，借助 Rust 库，Python 绑定得以利用 Rust 所提供的速度和性能；同时，用户也能受益于等式图领域的最新创新，而无需重新发明已有的解决方案。

与 egg库 的绑定，具体来说是 snake-egg API 相比，egg-smol 库的 Python 绑定更加自成一体。在 snake-egg API 中，用户

```
(datatype Math (Num i64) (Var 字符串) (Add Math Math) (Mul
Math Math)) ;; 表达式1 = 2* (x + 3) (define expr1 (Mul (Num
2) (Add (Var "x") (Num 3)))) ;; 表达式2 = 6 + 2* x (define
expr2 (Add (Num 6) (Mul (Num 2) (Var "x")))) (rewrite (Add a b)
(Add b a)) (rewrite (Mul a (Add b c)) (Add (Mul a b) (Mul a c)))
(rewrite (Add (Num a) (Num b)) (Num (+ a b))) (rewrite (Mul
(Num a) (Num b)) (Num (* a b)))

(run 10) (check (= 表达式1 表
达式2 ))
```

图 2. 文本应用程序接口

可以将任何想映射到等式图的 Python 对象带入其中。相比之下，egg-smol 绑定的限制更多，只允许通过现有的函数和类封装器来定义。

这种限制带来了若干优势，尤其是在考虑如何实现不同下游库之间的重写时。尽管这些库可能定义了不同的领域、类型和函数，但它们都使用相同的元结构来表示。这种统一性使得在它们之间编写重写规则成为可能，从而为整个社区带来显著益处。通过提供一致而灵活的接口，egg-smol 绑定有助于在 Python 中构建一个更加统一的、基于等式图的优化与重写生态系统。

4 现有 egg-smol 用户的对比

egg-smol 库的 Python 接口与现有的 S 表达式编写方式有几处不同。使用 Python 编写某些语言元素可能会更加冗长，例如声明变量或添加重写规则。另一方面，Python 的运算符重载可以让自定义操作的数学表达式更加简洁。

通过复用现有的 Python 结构并利用 Python 的类和函数结构，Python 接口无需额外付出即可提供静态类型检查，并可借助 MyPy 或 Visual Studio Code 中的 PyLance 等现有工具。对于静态类型检查器而言，经过装饰的函数和类的类型与底层对象相同，因此即使运行时表示可能不同，类型检查仍能保持一致。

静态类型检查不仅支持创建表达式，也支持编写规则。重写规则左侧和右侧的类型必须相同，这可以

由 Python 类型检查器在静态阶段验证。示例请参见图 3：如果尝试将类型为`Num(i) * Num(j)`的项 `Num`替换为`i * j`，后者属于内置整数类型，就会产生静态错误。对这些重写进行类型检查，也是采用较冗长的流式语法 `rewrite(lhs).to(rhs)` 而非更简洁的语法 `ewrite(lhs, rhs)`的原因。第一种写法可以检查 `lhs` 和 `rhs` 的类型是否相同，而第二种写法则不行，这是 Python 类型注解的限制所致。

自动补全还能帮助用户找到适用于重写的相关方法，就像类型检查有助于确保代码正确性一样。此外，Python 接口支持 Jupyter 等交互式笔记本环境，让用户能够试验等价图表达式，并即时测试重写规则。

总体而言，Python 接口通过吸引 Python 语言的庞大用户群，并为分析和编辑器集成提供额外的保障措施，让更多人能够使用等式图。

5 未来工作

当前工作通过为 egg-smol 库提供 Python 绑定，对等式图领域进行了令人振奋的探索。不过，该方案尚未在生产环境的用例中经过测试，并且依赖于被标记为实验性、可能发生变化的 egg-smol。未来有几个潜在方向，可以帮助 Python 库作者采用这一方案：

5.1 支持嵌入现有 Python 类型

对于需要处理现有 Python 对象的 Python 开发者来说，探索如何将这些对象作为叶节点嵌入等式图，并为其编写规则，将会很有帮助。通过创建一种用于 Python 对象的新类型，以及一个能够接收若干 Python 对象并执行任意 Python 代码字符串的函数，这或许可以实现，而无需对 egg-smol 上游代码进行任何修改。

5.2 简化面向用户的应用程序接口

如果要向用户公开表达式和等价图，我们就需要开发一个更简单易用的应用程序接口，用于构建这样的处理流程：接收现有表达式，对其执行一些替换操作，再将结果输出为原生 Python 对象。这需要添加钩子，以便在调用某些方法时自动将输出转换为 Python 对象。

5.3 使用上游库进行原型开发

如果能找到一个已经采用内部表达式系统并支持重写的上游库，试着验证该库如何用于其中，以及事先需要做哪些改进，将会很有帮助。由于 Ibis 库已经在尝试使用等式图，这或许是一个可行的选择。

5.4 导出和导入等价图描述

目前，无法先用 Python 编写等式图描述，再从 S表达式语言中使用它。应该可以添加某种导出功能，例如从 Python 生成 S 表达式语言，或采用其他机器可读的表示形式，如 JSON。理想情况下，这可以实现双向转换，使文本格式与 Python 源代码之间能够相互转换。

5.5 利用 Python 的状态管理和模块

我们可以更多地利用 Python 的状态管理和模块来实现封装。目前，所有定义在运行时都绑定到特定的等式图实例。不过，在编写可复用的库时，作者可能希望将定义分散到不同的文件或模块中，并将其中几个一起导入等式图。这样会更便于分发和组合不同的模块。

5.6 在 Jupyter Notebooks 中进行交互式可视化

通过检查 egg-smol 等式图的内部状态，我们可以在 Jupyter 笔记本中创建其可视化交互视图。Dask 和 Ibis 等其他库会将这些可视化用作教学辅助工具，帮助用户了解库如何处理表达式。开发此类工具可以让用户逐步查看等式图的执行过程，了解规则是如何执行的，同时也能作为教学工具，帮助新用户理解等价图的概念。

6 结论

在 Python 生态系统中利用等式图的兴趣日益浓厚，而这项工作为让更多人接触等式图提供了契机。除了帮助解决 Python 数据科学库中的具体表达式优化问题外，egg-smol Python 绑定还可以为库作者提供一种通用语言，用于表达各自的领域并在不同领域之间进行转换，从而增强整个 Python 数据科学社区的韧性与联系。

即使对于目前未使用 Python 的用户，得益于 Python 语言现有的丰富工具，Python 绑定也能提供颇具吸引力的开发体验。借助这些工具以及 Python 的广泛普及，我们可以推动等式图技术得到更广泛的应用，并促进不同领域之间的协作与创新。

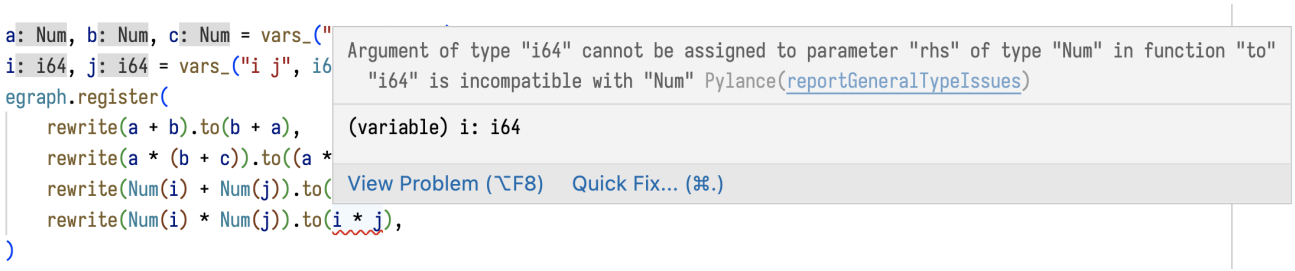


图3。 Visual Studio Code 中的类型检查错误

致谢

感谢 Recurse Center 提供社区支持与帮助，使这项工作得以开展。特别感谢 Sean Aubin 对这份提案提出的宝贵反馈，这些意见在编辑过程中发挥了极其重要的作用。如果没有 Max Willsey 及其他人对底层 Rust 库所做的全部工作，这一切都不可能实现。感谢你耐心解答我关于 egg-smol 的各种问题，并愿意与我合作开发它，即使当时这个项目还处于非常早期的阶段。

参考文献

[1] ibis维护者。2014年。 *ibis*：兼具 *Python* 的灵活性与现代 *SQL* 的规模和性能。 https://github.com/ibis-project/ibis

[2] 3 2015 *3* 3 3 PyO 维护者。 . *PyO.*

[3] 2015 https://github.com/PyO/pyoMatthew Rocklin。 . Dask：使用分块算法和任务调度进行并行计算。载于 第 *14* 届 *Python in* 科学会议，Kathryn Huff 和 James Bergstra（编）。130 – 136。[4] Matthew Rocklin。2021 年 5 月，2023 年。高级表达式。检索于 4 月，来源：https://github.com/dask/dask/issues/ 7933

[5] 2021 Snake Egg 维护者。。 *snake-egg*。https://github.com/egraphs- good/snake-egg [6] Rebecca Swords。2022 年。Quiche：E-图的 Python 实现。（2022）。 https://pldi22.sigplan.org/details/egraphs-2022-papers/ 1/Quiche-A-Python-Implementation-of-E-Graphs PLDI - 等价图。[7] Krisztián Szűcs。2023 年。feat(common)：添加纯 Python 等式图实现。检索于 2023 年 4 月 5 日，来源：https://github.com/ibis-project/ibis/pull/[8] 2022 3781

Max Willsey。。egraphs zulip - general stream，路线图讨论串。检索于 2023 年 4 月 5 日，来源：https://egraphs.zulipchat.com/#narrow/ stream/328972-general/topic/roadmap/near/297442354[9] Max Willsey、Chandrakana Nandi、Yisu Remy Wang、Oliver Flatt、Zachary Tatlock 和 Pavel Panchekha。2020 年。egg：快速且可扩展的等价饱和。*ACM* 编程语言学报5（2020），1 – 29。