arXiv:2304.04886v1 [cs.PL] 10 Apr 2023

让流分析框架重焕生机：重新审视流框架

Roland Meyer[1], Thomas Wies[2],and Sebastian Wolff[2(✉)]

[1] 布伦瑞克工业大学，布伦瑞克，德国，roland.meyer@tu-bs.de[2] 纽约大学，纽约，美国，{wies,sebastian.wolff}@cs.nyu.edu

摘要 我们提出了一种新的流框架，用于通过分离逻辑推理操作一般图的程序。该框架克服了早期发展中的问题：它基于标准的不动点理论，保证存在最小流，排除了消失流，并且提供了易于理解的足迹概念，这是框架规则具有可靠性所必需的。此外，我们提出了用于自动化框架规则的算法，并在从并发数据结构的线性化证明中提取的图更新上对其进行了评估。评估表明，我们的算法有助于自动化这些证明的关键方面，而这些方面此前一直依赖用户指导或启发式方法。

关键词： 分离逻辑 · 图算法 · 框架推断。

会议版本： Meyer, R., Wies, T., Wolff, S. (2023). 让流分析框架重焕生机：重新审视流框架。载于：Sankaranarayanan, S., Sharygina, N.（编）《系统构造与分析工具和算法》。TACAS 2023。《计算机科学讲义》，第 13993 卷。施普林格。https://doi. org/10.1007/978-3-031-30823-9_32

1 引言

流框架 [23, 24] 是一种基于分离逻辑 [6,32, 40] 的抽象机制，能够以局部方式推理一般图的全局归纳不变量。该框架已被证明有助于验证一些难以用其他技术处理的复杂算法，例如优先级继承协议、面向对象设计模式以及复杂的并发数据结构 [22, 24, 27, 34]。然而，这些工作也暴露出底层元理论的一些不足之处，它们要么限制了表达能力，要么妨碍了自动化。本文提出一种新的流框架元理论，旨在平衡这些相互冲突的要求。此外，我们还提出了有助于实现证明自动化的算法。

背景。 流框架的核心概念是流。给定一个交换幺半群（$\mathbb{M}$，$+$，0）（例如带加法的自然数），以及一个节点为X的图和一个边函数 $E\colon X \times X \to \mathbb{M} \to \mathbb{M}$，流是一个函数$\mathit{fl}\colon X \to \mathbb{M}$，它满足流方程：

$$\forall x \in X.\quad \mathit{fl}(x) = in_x + \textstyle\sum_{y \in X} E_{(y,x)}(\mathit{fl}(y)) \ .$$

也就是说，fl 是这样一个函数的不动点：该函数为每个节点 x 赋予初始值$in_x \in \mathbb{M}$作为其流入，然后根据边函数将这些值传播到整个图中。这类似于前向数据流分析，其中幺半群运算 $+$ 用作并。通过选择合适的流幺半群、流入和边函数，可以用仅引用每个节点流值 $\mathit{fl}(x)$ 的条件来表达图的归纳性质（可达性、有序性等）。

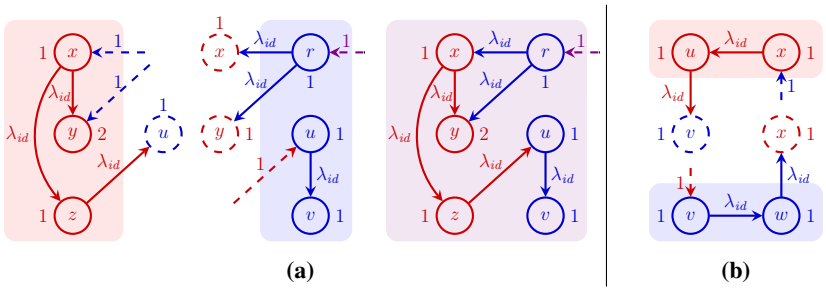


图 1。（a）两个流图h_1，分别带有节点 $h_1.X = \{x, y, z\}$（左）和 h_2 带有节点 $h_2.X = \{r, u, v\}$（中），对应于带加法的自然数流幺半群。边标签 λ_{id} 表示恒等函数。省略的边标记为常数 0 函数。虚线边表示流入。节点分别标注其流和流出。右侧显示复合 $h = h_1 * h_2$。**（b）**两个流图 h_1 带有$h_1.X = \{u, x\}$（上）以及 h_2 带有$h_2.X = \{v, w\}$（下），它们的复合由于消失流而未定义。

带有流入和相关流的图称为流图。图 1a 右侧展示了一个流图示例h。这里，节点 w 的流值$\mathit{fl}(w)$表示从 r 到 w 的路径数。流图可以是不完整的，并且可以包含指向 X 之外节点的边，例如图 1a 中h_1 对应的节点 u。如果在流的计算中包含这些节点，那么它们的流值就构成了该流图的流出。例如，h_1 对于 u 的流出为 1。

流图配备了不相交复合的概念，$h = h_1 * h_2$。图 1a 给出了一个示例。只有当h_1 和h_2 的流的并集再次成为h的流时，复合才有定义。这并不总是成立。例如，h_1 和h_2 的流入和流出可能彼此不兼容，例如h_1 向 u 发送流出 2，而h中的 u 的流入只有 1。

流图的复合构成一种分离代数。也就是说，如果我们将流图用作程序状态（例如堆）的抽象，就可以使用分离逻辑，对以诱导出的流图表示的程序属性进行局部推理。例如，假设程序通过在节点 r 和u之间插入一条标记为 λ_{id} 的新边，将图 1a 中的流图h 更新为新的流图 h'。这会使 u 和 v 的流量从 1 增至 2。我们可以按如下方式分解这次更新。首先，将h 分解为h_1 和 h_2。接着，通过插入这条边，并将 u 和 v 的流量更新为 2，得到h_2'。最后，再将h_2' 与h_1 复合，得到 h'。注意，复合$h_1 * h_2'$ 仍然有定义。这意味着，以h_1部分中的流为基础表达的任何属性，在h'中仍然成立。这就是分离逻辑中著名的框架规则在流图上的具体应用。

应用框架规则的关键在于证明复合 $h_1 * h_2'$ 确实有定义。可以通过局部证明更新 $h_2 \rightsquigarrow h_2'$ 是保持框架的来实现这一点；也就是说，对于满足 任意 h_1 使得 $h_1 * h_2$ 有定义的情况，$h_1 * h_2'$ 也有定义。

通常，保持框架更新所涉及的流子图$h_2 \rightsquigarrow h_2'$ 包含的节点多于受更新直接影响的节点。例如，考虑我们示例中h 和h' 的子图，它们仅由 $\{r, u\}$ 这些因插入该边而直接受到影响的节点组成。这些子图并不构成保持框架更新，因为在 r 和 u 之间插入边，也会使流向 v 的流出从 1 变为 2。因此，$\{r, u\}$ 的更新子图将不再能与h的其余部分复合，因为其中 v的流仍为 1，而不是 2。我们将诸如 $\{r, u, v\}$这样能够标识保持框架更新的节点集合称为该更新的足迹。

流图的元理论。除了确保流图复合构成分离代数之外，设计流图元理论时还必须考虑两个期望：

– 获得唯一的流。 使用流来编码归纳性质时，人们通常会关注某个特定的流，最常见的是给定流入对应的流方程的最小不动点。因此，需要一种方法，将推理聚焦于所关注的特定流。– 识别保帧更新。 为了能够应用框架规则，需要一种有效计算候选足迹并检查它们是否代表保帧更新的方法。

第一个子目标对于表达能力至关重要，第二个则关系到证明自动化。实现其中一个子目标，会使另一个更难实现。具体而言，考虑 [24] 中提出的元理论。该理论要求流幺半群 $(\mathbb{M}, +, 0)$ 也具有消去性（$m + n_1 = o$ 且 $m + n_2 = o$ 蕴含 $n_1 = n_2$）。要求具有消去性有一个好处：很容易检查更新 $h \rightsquigarrow h'$ 是否保帧；只需证明 h 和 h' 具有相同的流入和流出。消去性还确保对于每个流 $\boldsymbol{fl}$，都存在唯一的流入能够产生 $\boldsymbol{fl}$。因此，只需跟踪 $\boldsymbol{fl}$，因为流入是可推导出的量。然而，反过来则不成立。

事实上，获取唯一的流对于可消去的 $\mathbb{M}$ 变得更加困难。人们希望对 $\mathbb{M}$ 施加的一个自然要求是，由 $+$ 诱导的预序构成一个完全偏序（cpo），甚至是一个完全格。这样一来，只需关注最小流；如果应用标准的不动点定理，只需对边函数施加较弱的假设，就可以保证最小流存在。然而，消去性在本质上与标准的域理论前提不相容。例如，唯一的、作为有向 cpo 的有序交换幺半群是平凡幺半群：$\mathbb{M}_0 = \{0\}$。同样，$\mathbb{M}_0$ 是唯一具有最大元的此类幺半群。

对于需要唯一流的情况，[24] 对边函数（幂零）或图结构（实质上无环）施加了额外要求。前者在表达能力方面限制颇多。后者则再次增加了计算保帧更新的复杂性：现在必须确保更新后的图h_2'与其框架h_1复合时不会引入任何环。事实上，对于实质上无环的情况，[24] 仅提供了一个充分条件，用于判断给定足迹是否能产生保帧更新，却没有给出计算此类足迹的算法。

贡献。 本文提出了一种新的流元理论，基于构成 ω-cpo 的流幺半群（但不要求其可消去）。cpo 要求带来了所需的最小不动点语义。对流幺半群要求的不同，意味着需要一种新的流图复合概念。具体而言，对于流的最小不动点语义，只有当 h_1 和 h_2 的流均不消失时，$h = h_1 * h_2$ 才有定义。图 1b 展示了这种情况的一个例子：由于所形成的环没有外部流入，h_1 和 h_2 中的流会在 $h_1 * h_2$ 中消失为 0。此外，如果 h 和 h' 以相同方式将流入导向流出，那么更新 $h \rightsquigarrow h'$ 就是框架保持的。我们通过图的传递函数的上下文等价概念，将这一条件形式化。这些传递函数是流方程的最小不动点，以流入为参数，并限制在图外的节点上。随后，我们确定了实践中通常满足、且能够有效检查传递函数上下文等价性的边函数条件。这一结果意义非凡，因为流幺半群可以具有无限升链，而流图也可以包含环。基于这一等价性检查，我们提出了一种迭代算法，用于计算更新的足迹。该算法使得能够自动应用框架规则，以推理操纵流图的程序。

从而自动应用框架规则，以推理操纵流图的程序。我们在一个基准测试套件上评估了所提出的算法。该套件包含从使用工具 plankton [26, 27] 构造的并发搜索结构的线性化证明中提取的流图更新。评估结果表明，我们的算法有助于自动化这些证明中此前依赖用户指导或启发式方法的关键环节。

2 流图分离代数

我们首先介绍新的流图分离代数。

给定一个交换幺半群 $(\mathbb{M}, +, 0)$，我们通过 $n \leq m$ 定义 $\leq$ 在 $\mathbb{M}$ 上的二元关系，如果存在 $o \in \mathbb{M}$，使得 $m = n + o$。流值取自一个流幺半群，即一种交换幺半群，其中关系 $\leq$ 是一个 ω-cpo。也就是说，$\leq$ 是一个偏序，并且 $\mathbb{M}$ 中的每个升链 $K = m_0 \leq m_1 \leq \ldots$ 都有一个最小上界，记作 $\bigsqcup K$。我们要求加法与上界兼容，$n + \bigsqcup K = \bigsqcup(n + K)$。下文中，我们固定一个流幺半群 $(\mathbb{M}, +, 0)$。

令 $ContFun(\mathbb{M} \to \mathbb{M})$ 表示 $\mathbb{M} \to \mathbb{M}$ 中的连续函数。回顾一下，函数 $f\colon \mathbb{M} \to \mathbb{M}$ is 连续 [43]，当它与升链的极限相交换时，$f(\bigsqcup K) = \bigsqcup f(K)$ 对 $\mathbb{M}$ 中的每条链 K 都成立。我们以预期的方式将 $+$ 和 $\leq$ 提升为函数 $\mathbb{M} \to \mathbb{M}$。空的迭代和 $\sum_{i \in \varnothing} m_i$ 定义为 0。

引理 1. *$(ContFun(\mathbb{M} \to \mathbb{M})\ \circ, id)$ 是一个幺半群。此外，如果 $(\mathbb{M}, \leq)$ 是一个 ω-cpo，那么 $(ContFun(\mathbb{M} \to \mathbb{M})\ \leq)$ 也是如此。*

流图 是一个元组 $h = (X, E, in)$，由有限节点集合 $X \subseteq \mathbb{N}$、边集合 $E\colon X \times \mathbb{N} \to ContFun(\mathbb{M} \to \mathbb{M})$（边由连续函数标记）以及流入 $in\colon (\mathbb{N} \setminus X) \times X \to \mathbb{M}$ 组成。我们用 FG 表示所有流图的集合，并用 $h_\varnothing \triangleq (\varnothing, \varnothing, \varnothing)$ 表示空流图。

我们为流图定义两个派生函数。首先，流是满足流方程的最小函数 流 : X → M，其满足流方程：流 (x) = in$_x$ + rhs$_x$(流)，对于所有 x ∈X 。这里，in$_x \triangleq \sum_{y \in (\mathbb{N} \setminus X)}$ in (y, x) 是一个幺半群值，而 rhs$_x \triangleq \sum_{y \in X}E_{(y,x)}$是一个类型为 ContFun((X → M) → M) 的函数。最后，我们还定义流出 out : X × (N \ X) → M，其定义为 out (x , y) ≜E$_{(x,y)}$(流(x)).

示例 *1*。对于并发搜索结构的线性化证明，可以使用一种流，为每个数据结构节点 x 标注其内集，即键的集合 k'，线程搜索 k' 时可能遍历该节点 x [22, 23]。转换到我们的场景中，相关的流幺半群是键的幂集 $\mathbb{P}(\mathbb{Z} \cup \{-\infty, \infty\})$，其中集合并为加法。图 2 展示了两个键集流图，它们抽象表示了基于有序链表的并发集合实现的潜在状态。当一个键 k 从集合中移除时，存储该键的节点 x 首先会被标记，以表明 x 已被逻辑删除。第二步，x 随后会从链表中被物理摘链。该抽象的思想是：对于存储键 k 的节点 x，其出边在 x 未被标记时标记为函数 λ_k，否则标记为 $\lambda_{-\infty}$。这是因为，当且仅当 $k < k'$ 或 x 已被标记时，搜索 $k' \in \mathbb{Z}$ 才会遍历离开 x 的边。在图中，l 和 r 被假定为未标记，分别存储键 6 和 8。节点 t 被假定为已标记。流图 h_2 由 h_1 通过物理摘链已标记的节点 t 得到。借助键集流，随后可以表达基于局部推理的线性化证明所需的关键数据结构不变式（例如，节点的逻辑内容始终是其内集的子集这一不变式）。

我们注意到，抽象程序状态的全局流图的流入可以用于规范中。在该示例中，将 流入$_r = \mathbb{Z}$ 用于数据结构的根节点 r，并将所有其他节点的 流入$_x = \varnothing$ 用于表示所有搜索都从 r 开始。□

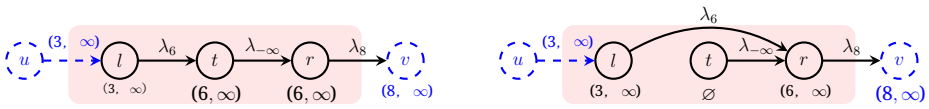


图 2. 两个流图 h_1（左）和 h_2（右），其 $h_1.X = h_2.X = \{l, t, r\}$ 对应键集流幺半群 $\mathbb{P}(\mathbb{Z} \cup \{-\infty, \infty\})$。键 k 的边标签 λ_k 表示函数 $\lambda m.(m \setminus [-\infty, k])$。

不含消失流的复合。要定义流图的复合，$h_1 * h_2$，我们分两步进行。首先，我们定义一种辅助复合，该复合可能会出现消失流，即在复合过程中消失的局部流。也就是说，对于图1b所示的流图，这种复合是有定义的。在复合后的图中，每个节点的流值在复合前为1 的地方都变为0——该流消失了。这意味着，辅助复合无法将各个组成部分中的流值下界提升到复合后的图。因此，实际的复合会将辅助复合限制为排除这类消失流。辅助复合的定义性要求h_1 与h_2中的节点互不相交。此外，一个流图的流出必须与另一个流图的流入预期相匹配：

$$h_1 \mathbin{\#\#} h_2 \quad \text{if} \quad X_1 \cap X_2 = \varnothing \ \wedge \ \forall x \in X_1,\, y \in X_2.\ out_1(x, y) = in_2(x, y) \wedge out_2(y, x) = in_1(y, x)\ .$$

辅助复合 $h_1 \uplus h_2$ 会移除另一个组件提供的流入：

$$h_1 \uplus h_2 \quad \triangleq \quad (X_1 \uplus X_2, E_1 \uplus E_2, (in_1 \uplus in_2)|_{(\mathbb{N} \setminus (X_1 \uplus X_2)) \times (X_1 \uplus X_2)})\ .$$

为了排除消失流，我们在复合的定义性中加入流上的一个适当等式：

$$h_1 \# h_2 \quad \text{if} \quad h_1 \mathbin{\#\#} h_2 \ \wedge \ h_1.flow \uplus h_2.flow = (h_1 \uplus h_2).flow\ .$$

只有当后一个等式成立时，我们才有复合 $h_1 * h_2 \triangleq h_1 \uplus h_2$。值得注意的是，$h_1.$流 $\uplus\, h_2.$流 $\geq (h_1 \uplus h_2)$。流 始终成立。定义性真正要求的是反向不等式。

回顾 [6]，分离代数 是一个偏交换幺半群 $(\Sigma, *, \mathsf{emp})$，其单位元集合为 $\mathsf{emp} \subseteq \Sigma$。

引理 2。$(FG\ *, \{h_\varnothing\})$ 是一个分离代数。

3 保帧更新

由于流图构成一个分离代数，我们可以像 [24] 中那样使用分离逻辑断言来描述流图的集合，进而用它们证明分离逻辑霍尔三元组。在这类证明中使用的一条关键证明规则是框架规则。给定分离逻辑断言 P_1 和 P_2，以及一条命令 c，框架规则指出：如果霍尔三元组$\{P_1\}\, c\, \{P_2\}$有效，那么对于任意框架 F，$\{P_1 * F\}\, c\, \{P_2 * F\}$ 也有效。本文余下部分将重点介绍用于自动化应用该证明规则的算法。

断言所描述的流图可能具有无界大小（例如，由于使用了迭代分离合取）。下文中我们只考虑有界流图；众所周知，无界情形是一个挑战，对于该挑战，正交

相关技术正在开发中（参见第 6 节）。然而，即使流图的大小有界，它们仍可能有无穷多个，因为流入和边函数是在流幺半群的逻辑理论中以符号方式编码的。为了便于教学，我们用具体流图而非符号流图来介绍算法。不过，假设底层流幺半群理论是可判定的，我们的论述很容易扩展到符号表示。事实上，我们在第 5 节讨论的实现使用的就是符号流图。

框架规则的可靠性依赖于这样一个假设：由命令 c 诱导的状态更新满足某种局部性条件。在我们的设定中，这一条件相当于检查 P_1 在 c 下的更新是否相对于流图复合是 保持框架 的。对于由 P_1 描述的流图h_1 以及h_1 在 c下的后像中的所有流图h_2，这意味着$h_1 \# h$蕴含$h_2 \# h$，对所有h</style>均成立。直观地说，如果h_1 和h_2 以相同方式将流入传递为流出，则$h_2 \# h$ 仍然成立。

形式上，对于流图 h，我们定义其 传递函数 $tf(h)$，将流入映射到流出，$tf(h)$: $((\mathbb{N} \setminus X) \times X \to \mathbb{M}) \to X \times (\mathbb{N} \setminus X) \to \mathbb{M}$, 通过

$$tf(h)(in') \triangleq h[in \mapsto in'].out\ .$$

对于给定的流入 in，我们也用 $tf(h_1) =_{in} tf(h_2)$ 表示：对于所有流入$in' \leq in$，$tf(h_1)(in') = tf(h_2)(in')$。

定义 1。流图 h_1,h_2 是上下文等价的，记作 $h_1 =_{ctx} h_2$,，如果我们有 $h_1.X = h_2.X$，h_1.属于 $= h_2$.属于，并且 $tf(h_1) =_{h_1.in} tf(h_2)$。

定理1（框架保持）。对于所有流图 $h_1 =_{ctx} h_2$ 和 h，$h_1 \# h$ 当且仅当$h_2 \# h$，并且在定义性情况下，$h_1 * h =_{ctx} h_2 * h$。

为了自动化应用命令 c 和前置条件 P的框架规则，我们需要找出一个分解 $P = P_1 * F$，从而推断出 $\{P_1\}\, c\, \{P_2\}$，然后应用框架规则，推导出后置条件 $Q = P_2 * F$的 $\{P\}\, c\, \{Q\}$。这与框架推断问题 [4]密切相关。当命令将流图 h_1 修改为h_2,时，我们的目标是找出h_1 中一组（希望尽可能小的）受此更新影响的节点 Y，即流足迹。也就是说，Y 捕获了更新前后流图之间的差异，而 Y 的补集则定义了框架。要对此进行形式化，我们需要将流图限制到节点子集上，由此便可以定义流图分解的概念。为此，考虑h 和$Y \subseteq \mathbb{N}$。我们定义

$$h|_Y \ \triangleq\ (h.X \cap Y, h.E|_{(h.X \cap Y) \times \mathbb{N}}, in)$$

使得对于所有 z $\in \mathbb{N} \setminus$h.X、y $\in$h.X $\cap$Y，流入 in 满足 in(z , y) $\triangleq$h.in(z , y); 并且对于所有 x $\in$h.X $\setminus$Y、y $\in$h.X $\cap$Y，in(x , y) $\triangleq$h.E$^{(x,y)}$(h.flow(x))。

定义 2. 考虑 h_1 和 h_2，其中 $X \triangleq h_1.X = h_2.X$，且 $h_1.in = h_2.in$。对于 h_1 与 h_2 之间的差异，流足迹 是节点的一个子集 $Y \subseteq X$，使得 $h_1|_Y =_{ctx} h_2|_Y$且 $h_1|_{X \setminus Y} = h_2|_{X \setminus Y}$。所有此类足迹组成的集合为 $FFP(h_1, h_2)$。

节点集合或流入不同的流图永远不会有流足迹。前一项要求只是为了简化表述。为此，我们假设程序执行期间将分配的所有节点都已存在于初始流图中。此假设可以取消。后一项要求的依据是，全局流入属于规范的一部分，正如前文示例 1 所述。

在继续讨论如何计算流足迹这一问题之前，我们先强调它们的一些性质。

引理 3（足迹单调性）。若 $Z \in FFP(h_1, h_2)$ 且 $Z \subseteq Y \subseteq h_1.X$，则 $Y \in FFP(h_1, h_2)$。

单调性的一个结果是规范流足迹的存在：只要存在流足迹，那么所有节点构成的集合就可以作为足迹。当然，这个规范足迹大得不尽如人意。它对应于全局推理流图更新、放弃应用框架规则的情况。遗憾的是，不存在包含极小流足迹。

命题 1（规范足迹）。我们有：$FFP(h_1, h_2) \neq \varnothing$ 当且仅当 $h_1.X \in FFP(h_1, h_2)$。不存在包含极小流足迹；特别地，集合 $FFP(h_1, h_2)$ 在交集下不封闭。

单调性的证明需要更好地理解限制算子，以下引理对此作出了说明。

引理 4（限制）。考虑 h 和 Y、$Z \subseteq \mathbb{N}$。那么 *(i)* $h|_Y.\text{流} = h.flow|_Y$，*(ii)* $h|_Y \# h|_{X \setminus Y}$ 以及 $h|_Y * h|_{X \setminus Y} = h$，并且 *(iii)* $(h|_Y)|_Z = h|_{Y \cap Z}$。

由于流足迹是通过限制定义的，该引理也表明流足迹具有良好的性质。例如，将限制应用于足迹 Y 不会改变节点 $y \in Y$ 的流，也不会改变节点 $x \in h.X \setminus Y$ 的流。更正式地说，这意味着 $h|_Y.\text{流}(y) = h.\text{流}(y)$，以及 $h|_{X \setminus Y}.\text{流}(x) = h.\text{流}(x)$，由引理 4(i)可知。

对于我们的开发而言，采用传递函数更具操作性的表述会很方便。为此，我们将流图理解为一个以流入为参数、并生成流近似值转换器的函数：

$$h : ((\mathbb{N} \setminus X) \times X \to \mathbb{M}) \to (X \to \mathbb{M}) \to X \to \mathbb{M}$$

$$\text{defined by} \quad h[in](\sigma)(x) = in_x + rhs_x(\sigma)\ .$$

回忆一下，$\mathrm{in}_x \triangleq \sum_{y \in \mathbb{N} \setminus X} \mathrm{in}(y, x)$ 和 $\mathrm{rhs}_x(\sigma) = \sum_{y \in X} \mathrm{E}(y,x)(\sigma(y))$。$\mathrm{h}[\mathrm{in}]$ 的最小不动点为 $\bigsqcup_{i \in \mathbb{N}} \mathrm{h}[\mathrm{in}]^i(\bot)$，其中 $\mathrm{h}^0 = \mathrm{id}_{X \to \mathbb{M}}$ 且 $\mathrm{h}^{i+1} = \mathrm{h}^i \circ \mathrm{h}$，这由克莱尼定理得出。定义 $\mathrm{out} : (\mathrm{X} \to \mathbb{M}) \to \mathrm{X} \times (\mathbb{N} \setminus \mathrm{X}) \to \mathbb{M}$，使得 $\mathrm{out}(\sigma)(y, z) \triangleq \mathrm{E}(y,z)(\sigma(y))$。由此得到传递函数和流的如下刻画。

引理 5（传递）. 对于所有流图 h，我们有 *(i)* $tf(h) = out \circ (lfp.h[-])$ 且 *(ii)* $lfp.h[h.in]) = h.flow$。

4 计算足迹

我们提出一种算法，用于计算给定的两个流图之间差异的足迹。我们分两步进行。首先，我们给出该算法的高层描述，暂不考虑可计算性问题。第二步，我们说明如何解决这些可计算性问题。在整个推导过程中，我们假设流图 h_1 和 h_2 具有相同的节点 $X \triangleq h_1.X = h_2.X$，且具有相同的流入 $h_1.in = h_2.in$。根据定义，如果这一假设不成立，则流足迹不存在。

4.1 算法

我们将流足迹计算为一个不动点。首先，取足迹候选集 Z，其中包含出边在 h_1 和 h_2 中不同的节点。然后，我们迭代地添加那些流出当前足迹候选集 Z 的流出在 $h_1|_Z$ 和 $h_2|_Z$ 中不同的节点。所谓流出不同，是指传递函数 $tf(h_1|_Z)$ 和

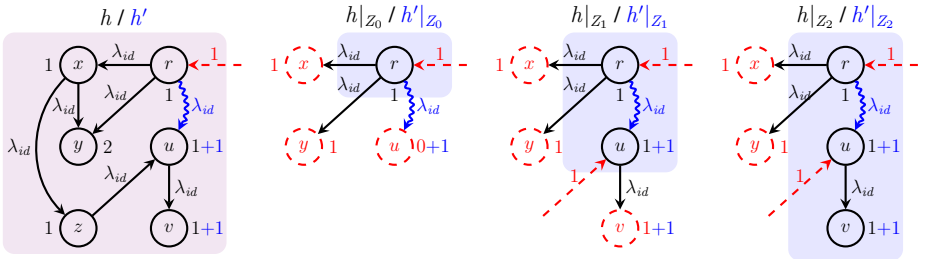


图3. 计算h与h'的差异的足迹，依次遍历集合$Z_0 \triangleq \{r\}$、$Z_1 \triangleq \{r, u\}$和$Z_2 \triangleq \{r, u, v\}$。后者是$ext_{h,h'}$的最小不动点，并得到所需的足迹，即$Z_2 \in FFP(h, h')$。

$tf(h_2|_Z)$不同，因此候选集Z不是足迹。反过来，如果所有流出都一致，那么传递函数就相同，且Z如期构成一个足迹。

严格来说，我们是在带有特定顶元的节点幂集格上计算不动点：$(\mathbb{P}(X)^\top, \sqsubseteq)$，其中 $\mathbb{P}(X)^\top \triangleq \mathbb{P}(X) \uplus \{\top\}$。元素$\top$ 表示足迹计算失败。如果足迹未被X覆盖，即超出流图h_1, h_2，就可能出现这种情况。

我们的不动点计算从$Z = odif_{h_1,\ h_2} \subseteq X$开始，如下所定义：

$$odif_{h_1,h_2} \triangleq \{\, x \in X \mid \exists z \in \mathbb{N}. h_1.E(x,z) \neq h_2.E(x,z) \,\}.$$

只要与$h_1|_Z$和$h_2|_Z$相关联的传递函数不匹配，不动点就会继续扩展Z。为定义扩展，我们将传递失败 定义为$Z \subseteq X$的后继节点中，可能从h_1和h_2接收到不同流出的节点：

$$tfail_{h_1,h_2}(Z) \triangleq \left\{ x \in \mathbb{N} \setminus Z \;\middle|\; \begin{array}{l} \exists\, in \leq h_1|_Z.in\ \ \exists z \in Z. \\ \qquad [tf(h_1|_Z)(in)](z,x) \neq [tf(h_2|_Z)(in)](z,x) \end{array} \right\}.$$

这个集合就是原因，解释了为什么当前足迹候选集Z不是足迹；也就是说，$Z \notin FFP(h_1, h_2)$。将传递失败并入Z后，会得到一个新的候选集。我们会检查新候选集是否被X覆盖（即是否包含h_1, h_2之外的节点）。如果检查未通过，新候选集将为$\{\top\}$，表示无法计算出足迹。以下定义将精确说明扩展过程。

定义3。 函数 $ext_{h_1,h_2} : \mathbb{P}(X)^\top \to \mathbb{P}(X)^\top$ 定义如下

$$ext_{h_1,h_2}(Z) \triangleq \begin{cases} \top & \text{if } tfail_{h_1,h_2}(Z) \not\subseteq X \\ Z \sqcup odif_{h_1,h_2} \sqcup tfail_{h_1,h_2}(Z) & \text{otherwise} \end{cases}.$$

通过传递失败迭代扩展候选 Z，最终会得到表示h_1与h_2, 之差的足迹，或者以$\top$失败。该方法是可靠的。

定理 2（可靠性）。 设 $F \triangleq lfp_{\emptyset} ext_{h_1,h_2}$。若 $F \neq \top$，则 $F \in FFP(h_1, h_2)$。

示例 2。 如图 3 所示，考虑对所提出方法的说明。我们在此应用不动点计算，为流图h与 h'之差寻找足迹。正如第 1 节所述，h' 是在h 中插入一条新边的结果，该边连接节点 r 和 u，标签为 λ_{id}。

不动点计算从 $Z_0 \triangleq \{r\} = odif_{H,H'}$开始，因为它是唯一一个出边发生变化的节点。接下来，我们计算 $tfail_{h,h'}(Z_0)$。这会产生

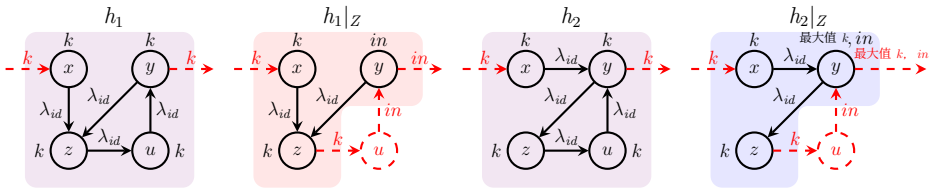


图 4. 使用幺半群 $(\mathbb{N}\cup\{\infty\}, \text{最大值}, 0)$ 说明完备性的反例。虽然集合 $\{x, y, z, u\}$ 是流图 h_1 与h_2,之差的一个足迹，但我们的不动点会生成候选项 $\{x\}$ 和$Z \triangleq \{x, y, z\}$，随后因 $\{\top\}$而失败。

$\{u\}$ 因为 u 在 h_0 中从 Z′ 接收到 0，而在 h_0 中则因新增的边接收到 1。Z′ 流向其余节点的流出在 h 和 h_0 中一致。因此，将传递失败应用于 $Z_1 \triangleq$ 后，得到 $Z_h ext_{h',0}(Z = \{u, r\})_h$。同理，我们计算 $tfail_{h',1}(Z_2 \triangleq)$，得到 $Z_h ext_{h',1}(Z = \{r, u, v\})$ v。由于 $_2$ 没有出边，Z_h 是 $ext_{h',2}$ 的最小不动点。因为 Z′ 是 h 和 $h_2 \in$ 的节点集合的子集，所以它是一个足迹，Z′FFP(h, h□)。

为了得到定理 2，我们必须证明不动点$F \triangleq lfp.ext_{h_1,\ h_2}$ 确实是一个足迹，前提是$F \neq \top$。也就是说，根据定义 2，我们必须确立以下两个性质：(i) $h_1|_F =_{ctx} h_2|_F$，以及 (ii) $h_1|_{X\setminus F} = h_2|_{X\setminus F}$。

要理解后一个性质，请注意，$h_1|_{X\setminus F}$与$h_2|_{X\setminus F}$的图结构（节点和边）相同，因为$odif_{h_1,\ h_2} \subseteq F$。流入也相同，因为直观而言，它们由流图的整体流入$h_1.in = h_2.in$和足迹的流出组成；由于$h_1|_F =_{ctx} h_2|_F$，两个流图中的流出相等。

健全性证明中有趣的部分在于证明性质 (i)，即上下文等价 $h_1|_F =_{ctx} h_2|_F$。由于 F 是 ext_{h_1,h_2} 的不动点，我们知道$tfail_{h_1,h_2}(Z) = \varnothing$，因此$h_1|_F$和$h_2|_F$的传递函数相同。因此，只需证明$h_1|_F.in = h_2|_F.in$，即可得到所需的上下文等价（定义 1）。证明中的这一关键步骤借助以下引理得以完成。

引理 6. 设 $odif_{h_1,h_2} \subseteq F \subseteq X$，且 $tfail_{h_1,h_2}(F) = \varnothing$。则 $h_1|_F.in = h_2|_F.in$。

要证明该引理，必须证明从非足迹部分 Y $\triangleq$X \F 流入 F 的流，在 h_1 和 h_2中相同。由于流中存在循环依赖，这颇具挑战：来自 Y 的流入取决于 F 的流出，而 F 的流出又取决于来自 Y 的流入。为应对这一挑战，我们将 h_i 的流方程改写为 h $_i|_F$和 $h_i|_Y$的两个独立流方程的配对，其中 $i \in \{1, 2\}$。直观来说，这些配对在给定固定流入的情况下（初始为 h_i.in），分别计算 $h_i|_F$和 $h_i|_Y$中的局部流。随后，将流入 $h_i|_F$ 更新为来自 h_i 外部的流入与来自 $h_i|_Y$的流入之和，并对流入 $h_i|_Y$作类似更新。重复这一过程，直到达到不动点。具体而言，我们依靠 Beki´c 引理 [1]来计算这些配对。接着，我们注意到 tf ($h_1|_F$) =tf ($h_2|_F$)，因为 $tfail_{h_1,h_2}$(F) = ∅；同时，tf ($h_1|_Y$) =tf ($h_2|_Y$)，因为 $odif_{h_1,h_2} \subseteq$ F。粗略地说，这意味着 h_1 和 h_2 的流配对也必须相同，因为各个部分传递的值相同。换言之，在每次迭代中，$h_1|_F$ 和 $h_2|_F$ 的更新流入，以及 $h_1|_Y$ 和 $h_2|_Y$ 的更新流入，分别相同。总之，这意味着 h $_1|_F$.in =$h_2|_F$.in。

我们以正向方式计算流足迹：从流图不同的节点出发，沿着边继续追踪。因此，如果要确定流足迹时需要考虑某个迭代过程的前驱节点 Z，这种方法可能会失败。示例见图 4。使用幺半群 $(\mathbb{N}\cup\{\infty\}, \max, 0)$，不难看出，集合 $\{x, y, z, u\}$是 h_1 与 h_2之间差异的一个足迹。然而，我们的不动点将从

从 $\{x\}$ 开始，并将其扩展至 $Z \triangleq \{x, y, z\}$。设 v 为指向其外部的流图节点。于是，下一个传递失败是 $tfail_{h_1,h_2}(Z) = \{v\}$，因为对于 $in < k$，从 y 到 v 的流出在 $h_1|_Z$ 和 $h_2|_Z$ 中有所不同。我们的方法无法计算出足迹。

事实3（不完备性）存在一些流图 h_1 和 h_2，对于这些流图，尽管存在流足迹，我们的算法仍无法确定该流足迹。

4.2 比较传递函数

在实现上述不动点计算时，难点在于证明给定传递函数之间的等价性，以便得出传递失败：[传递函数 $(h_1|_Z)$(–)](–，x) = [传递函数 $(h_2|_Z)$(–)](–，x)？众所周知，比较两个函数在算法层面很难实现。更棘手的是，传递函数被定义为最小不动点，这意味着我们没有可供比较的函数闭式表示。

我们的方法是对边函数的集合施加额外要求。这些要求在我们的所有实验中都得到满足，因此并不意味着我们的方法在适用性方面受到限制。我们证明，如果边函数不仅是连续的，而且还是可分配的，那么传递函数就可以根据穿过底层流图的路径来理解。如果边函数还具有递减性，并且底层幺半群的加法是幂等的，那么无环路径就足够了。这两个结论对于仅仅连续的边函数都不成立。

分配性。我们的第一个附加假设是，边函数 $f\colon \mathbb{M} \to \mathbb{M}$ 不仅连续，而且还是**分配的**，即对于所有 $m, n \in \mathbb{M}$，都有 $f(m + n) = f(m) + f(n)$，并且 $f(0) = 0$。我们用 $DistFun(\mathbb{M})$ 表示 $\mathbb{M}$ 上所有连续且分配的函数组成的集合。引理 1 中提出的性质仍然成立。

对于连续且可分配的传递函数，我们可以根据长度为 i、经过 $h[in]^i$ 的路径来理解 $h[in]$。例如，$i = 3$ 得出

$$\begin{aligned}[h[in]^3](\bot)(z) &= in_z + \sum_{y\in X} E_{(y,z)}\big(\, in_y + \sum_{x\in X} E_{(x,y)}(in_x + \sum_{u\in X} E_{(u,x)}(\bot(u))\,)\\ &= in_z + \sum_{y\in X} E_{(y,z)}(in_y) + \sum_{y\in X}\sum_{x\in X} E_{(y,z)}(E_{(x,y)}(in_x))\,.\end{aligned}$$

第一个等式根据定义成立，第二个等式体现了分配律。特别是，$\bot(u) = 0$ 因此 $E_{(y,z)}(\, E_{(x,y)}(\, E_{(u,x)}(\, \bot(u)\,)\,)\,) = 0$。最后一项表明，我们将节点 x 处给定的流入转发到中间节点 y，再从那里转发到我们所关注的节点 z。对于 $h[in]$, 的更高次幂，我们取更长的路径。对于 $h[in]^*$，因此我们得到所有节点 x 以及从 x 到 z、经过流图的所有路径之和。我们需要一些定义来使这一点更加精确。

流图中的一条路径 p，是一个有限的非空节点序列，其中除最后一个节点位于流图之外外，其余节点都属于该流图：

$$p \;=\; x_0 \cdot \ldots \cdot x_n \cdot z \;\in\; X^+ \cdot (\mathbb{N} \setminus X)$$

其中 $\cdot$ 表示路径拼接。我们使用 **首个**$(p) = x_0$ 分别使用 **最后一个**$(p) = x_n$ 来从流图 h 中提取首个和最后一个节点。通过 **路径集**(h, x, y, z)，我们表示所有经过流图 h 的路径的集合，这些路径从节点 **首个**$(p) = x$ 开始，并从节点 **最后一个**$(p) = y$ 离开 h，移动到 $z \in \mathbb{N} \setminus X$。给定节点的集合 $X' \subseteq X$，我们使用

路径集(h, X', y, z) 对所有 $x \in X'$的集合的并集 路径集(h, x, y, z)。该路径诱导出函数E_P： $\mathbb{M} \to \mathbb{M}$ 该函数将沿路径的边函数进行复合：

$$E_x = id \qquad\qquad E_{x.p} = E_p \circ E_{(x,first(p))} .$$

结合引理 5，上述分析得出了流图传递函数的首个闭式表示；此前，该函数的求解涉及不动点计算。

定理 4（闭式表示）。如果h 标记在$DistFun(\mathbb{M})$上，则：

$$[tf(h)(in)](y, z) \quad = \quad \sum_{x \in X} \sum_{p \in Paths(h,x,y,z)} E_p(in_x) .$$

定理 4 将传递函数的不动点计算推入集合路径集(h, x, y, z)中，而这些集合本身是归纳定义的，并且可能是无限的。下文中，我们将在不要求流图具有无环性的情况下缓解这一问题。

幂等性。我们的第二个假设是，幺半群中的加法具有幂等性，即对于所有$m \in \mathbb{M}$，$m + m = m$。幂等性确保对于可比较元素，加法退化为并：对于所有$m \leq n \in \mathbb{M}$，$m + n = m \sqcup n = n$。除非另有说明，以下我们均假设加法具有幂等性。

根据定理 4，我们只需比较路径上的和。利用幂等性，我们证明可以进一步简化问题，针对单条路径而非路径和进行推理。我们证明，h_1 中的每条路径都可以替换为h_2, 中的一个路径集，反之亦然。更进一步，我们只需考虑边发生变化的节点所对应的路径。路径替换条件的精确定义如下。

定义 4。对于节点集合相同、节点集为 X 且由 $DistDecFun(\mathbb{M})$ 标记的流图 h_1 和 h_2，路径替换条件 要求：对于每个 $x \in odif_{h_1,h_2}$，以及每个 $y \in X$ 和每个 $z \in \mathbb{N} \setminus X$，均有

$$\forall p \in Paths(h_1, x, y, z) \ \exists P \subseteq Paths(h_2, x, y, z). \ E_p \leq E_P \triangleq \sum_{q \in P} E_q .$$

例 3。对于图 4 中的流图h_1 和h_2，我们有h_1 可由h_2, 替换，反之亦然。为此，考虑h_1 中的路径 $p \triangleq x \cdot z \cdot u \cdot y \cdot v$ 和h_2, 中的$q \triangleq x \cdot y \cdot v$，其中 v 是h_1,h_2 之外指向该节点的节点。由于所有边都标记为 λ_{id}，因此我们有$E_p = \lambda_{id} = E_q$。值得注意的是，在这个例子中，我们可以忽略h_1 和h_2中的环。稍后，我们将介绍对边函数的限制，以避免一般情况下出现环。

类似地，对于图 2 中的流图，我们有路径替换。具体而言，$E_p = \lambda_8 = E_q$ 对于路径 $p \triangleq l \cdot t \cdot r \cdot v$（位于 h_1 中）和 $q \triangleq l \cdot r \cdot v$（位于 h_2中）成立。□

主要结论是，路径替换对于证明传递函数等价既是可靠的，也是完备的。

定理 5（路径替换原理）。当且仅当 h_1 到 h_2 的路径替换以及 h_2 到 h_1 的路径替换成立时，$tf(h_1) = tf(h_2)$。

该定理在几个方面都非常引人注目。首先，我们原本会认为必须替换 h_1 中所有节点的路径。相反，我们可以关注出边发生变化的节点。其次，我们原本会认为替换路径P 从 h_2 中的任意节点开始。这样的路径集将产生类型为 $(Y \to \mathbb{M}) \to \mathbb{M}$ 的传递函数。相反，我们可以使用类型为 $\mathbb{M} \to \mathbb{M}$ 的函数。更进一步，我们可以关注从与待替换路径相同的节点开始的路径。最后，我们用于替换的路径不受任何约束，从而为启发式方法留出了空间。

证明从在 X 上且由 $DistFun(\mathbb{M})$ 标记的、将 h_1 替换为 h_2，的完全路径替换条件开始。完全路径替换与定义 4 相一致，但它从完整的 X 中取x，而不是从 $x \in odif_{h_1,h_2}$ 中取值。在连续且分配的边函数情况下，完全路径替换刻画了具有幂等加法的幺半群中传递函数之间的等价性。

引理 7. 我们有 $tf(h_1) = tf(h_2)$ 当且仅当 h_1 被 h_2 完全路径替换以及 h_2 被 h_1 完全路径替换成立。

该结果是定理 4 的推论。定理 4 将 $\mathrm{tf}(\mathrm{h}_1)$ 等同于对所有 $\mathrm{x} \in \mathrm{X}$ 以及所有路径集 $p \in$ 路径集$(\mathrm{h}_1, \mathrm{x}, \mathrm{y}, \mathrm{z})$ 中的所有 E_P 求和。完全路径替换允许我们改为对 E_P 求和，其中 $\mathrm{P} \subseteq$ 路径集$(\mathrm{h}_2, \mathrm{x}, \mathrm{y}, \mathrm{z})$。用所有路径 路径集$(\mathrm{h}_2, \mathrm{x}, \mathrm{y}, \mathrm{z})$ 对 P 进行过近似，可得 $\mathrm{tf}(\mathrm{h}_1)$ 的上界。显然，所得的和可改写为定理 4 的形式，从而得到 $\mathrm{tf}(\mathrm{h}_1) \leq \mathrm{tf}(\mathrm{h}_2)$。类似地，我们得到 $\mathrm{tf}(\mathrm{h}_1) \geq \mathrm{tf}(\mathrm{h}_2)$，因此 $\mathrm{tf}(\mathrm{h}_1) = \mathrm{tf}(\mathrm{h}_2)$，如所需。引理的反向推导类似。

为完成定理 5 中路径替换原理的证明，下面的引理指出，将 h_1 替换为 h_2的完全路径替换与（普通）路径替换是一致的。为说明这一点，考虑任意路径 $p \in$ 路径集$(\mathrm{h}_1, \mathrm{x}, \mathrm{y}, \mathrm{z})$，其中 $\mathrm{x} \in \mathrm{X}$。目标是证明，对于某个 $\mathrm{P} \in$ 路径集$(\mathrm{h}_2, \mathrm{x}, \mathrm{y}, \mathrm{z})$，有 $\mathrm{E}_p \leq \mathrm{E}_P$。为此，将该路径分解为 $p = p_1 \cdot p_2$，使得 $\mathrm{x}' \triangleq \mathrm{first}(p_2)$ 是 p from odif_{h_1, h_2} 中的第一个节点。普通路径替换得到 $\mathrm{Q} \in$ 路径集$(\mathrm{h}_2, \mathrm{x}', \mathrm{y}, \mathrm{z})$，且有 $\mathrm{E}_{p_2} \leq \mathrm{E}_Q$。现在，取 $\mathrm{P} \triangleq \{ p_1 \cdot q \mid q \in \mathrm{Q} \}$。由于 p_1 存在于 h_1 和 h_2 中，并且具有完全相同的边标签，我们立即得到所需的 $\mathrm{E}_p \leq \mathrm{E}_P$。

引理 8. 当且仅当 h_1 到$by h_2$ 的路径替换成立时，h_1 到 h_2 的完全路径替换成立。

递减性。我们假设边函数 $f\colon \mathbb{M} \to \mathbb{M}$ 不仅连续且具有分配性，而且还是递减的：$f(m) \leq m$，对所有$m \in \mathbb{M}$均成立。边函数递减这一假设是合理的，因为遍历流图的程序会逐步构建关于结构状态的信息，而较小的流值意味着更多信息（如同经典数据流分析中那样）。我们用$DistDecFun(\mathbb{M})$表示 $\mathbb{M}$ 上所有连续、具有分配性且递减的传递函数所构成的集合；引理 1 同样适用于该集合。幺半群中的加法仍假定为幂等的。

如果所有边函数都是递减的，那么流图中的每个环也都是递减的。关键观察是，给定幂等加法，在对路径集合求和时，可以避免包含递减边函数的环。

引理 9. 设 h 在 $DistDecFun(\mathbb{M})$ 和 $p_1 \cdot p \cdot p_2 \in$ 路径集(h, x, y, z)上标记，其中 $last(p) = first(p)$。那么 $p_1 \cdot p_2 \in$ 路径集(h, x, y, z) 和 $E_{p_1 \cdot p \cdot p_2} \leq E_{p_1 \cdot p_2}$。

如果一条路径不重复经过任何节点，就称它为简单 路径；用$SimplePaths(h, x, y, z)$表示所有经过h、从x 到y 并朝z离开流图的简单路径构成的集合。注意，有限图只包含有限条简单路径。

定理 6（简单路径）。假设边函数连续、可分配且递减，并且加法具有幂等性，则定理 *4* 和定理 *5* 中每一处出现的 $Paths(h, x, y, z)$ 都可替换为 $SimplePaths(h, x, y, z)$。

实际上，路径计数流、键集流、可达性流、最短路径流和优先级继承流与我们的理论相关 [22–24, 27] 并且兼容。

5 评估

我们通过在从文献中提取的真实世界流图集合上进行评估，证明了新方法的实用性。我们说明了如何获得基准数据，以及如何实现和评估我们的方法。

基准测试套件。 正如第 1 节所述，流框架已用于验证复杂的并发数据结构。具体而言，它已被用于借助 plankton 工具 [26, 27] 自动构造证明。plankton 会在支持流的分离逻辑中进行穷尽式证明搜索，并提供用于证明线性化的其他高级功能；这些功能与本文的评估无关。为处理堆更新，plankton 会为当前证明状态（以分离逻辑断言表示）的流图$h_1 = h * h^{frame}$生成足迹。随后，它会框定流图h^{frame}中非足迹部分h_1，以便在足迹h范围内局部计算堆更新后的状态h'。所得结果是新的流图$h_2 = h' * h^{frame}$。我们将这一对 (h_1, h_2) 视为评估用的基准测试套件 。

我们对plankton 进行了改造，使其导出构造了足迹的流图对。由此，我们从一组包含 10 种并发集合数据结构的证明构造过程中发生的堆更新中，获得了 1272 个基准测试。此基准测试套件中的所有流图最多包含 4 个节点。

我们的基准测试套件受限于plankton的能力和限制。特别是，我们继承了其仅适用于并发搜索结构的限制。这是因为plankton 仅集成了对键集流的支持（参见示例 1）。我们的评估将针对该流计算足迹。

实现。 我们在名为krill [28]的工具中实现了第 4 节所述的固定点计算，用于求出给定两个流图h_1,h_2 时的足迹。该工具集成了三种方法，用于计算足迹候选集Z的传递失败$tfail_{h_1,h_2}(Z)$:

1. **NAIVE**： 一种朴素方法，计算足迹Z内的流。按照[24], 的要求，此方法要求流图无环，以避免计算流时求解不动点方程。2. **NEW**： 我们提出的新方法，利用简单路径的路径替换条件（参见定理 5 和定理 6）。此方法要求底层幺半群中的边函数具有分配性和递减性，且加法满足幂等性。3. **DIST**： 我们提出的新方法的一种变体，利用闭式表示（参见定理 4）。此方法要求边函数具有分配性，并要求流图无环，以避免对闭式表示中的所有路径进行无界求和。

我们的基准测试套件满足这三种方法的所有要求。NAIVE 和DIST 方法包含一个（充分的）检查，以确保更新后的流图具有无环性，从而保证所得足迹的可靠性。

这三种方法都将传递函-数之间所必需的等价性检查编码为 SMT 公式，然后使用现成的SMT 求解器z3 [31]进行求解。我们的编码使用带量词的整数理论。NAIVE 方法还使用自由函数来编码整数集合。

实验。我们在基准测试套件上运行了krill，并比较了三种不同的传递失败计算方法的运行时间。结果汇总于图 5（左）。对于我们从中提取基准测试的每种搜索结构，图中列出了：(i) 提取的流图对数量 #FG；(ii) 各方法的总运行时间

结构	#FG	NAIVE	DIST	NEW	加速
精细集合 [13]	12	75 ms	48 ms	46 ms	39%
惰性集合 [12]	14	73 ms	52 ms	51 ms	30%
ORVYY 集 [33]	20	106毫秒	76 ms	74 ms	30%
VY DCAS 集合 [46]	19	109ms	74 ms	73 ms	33%
VY CAS 集合 [46]	28	139ms	104毫秒	102毫秒	27%
Michael 集合 [29]	225	1216毫秒	887毫秒	874毫秒	28%
Michael 集合（无等待）	186	996ms	731 毫秒	721 毫秒	27%
Harris 集合 [11]	352	2242毫秒	1490毫秒	1443毫秒	36%
Harris 集合（无等待）	296	1859ms	1242ms	1205ms	35%
FEMRS 树 [10]	120	519ms	409ms	407ms	22%
总计	1272	7335毫秒	5114毫秒	4996毫秒	32%

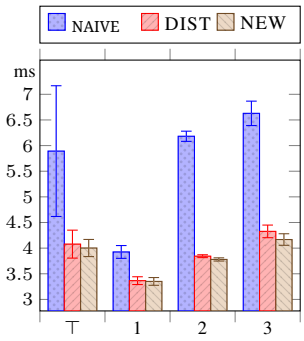


图5. 在 Apple M1 Pro 上进行 1000 次重复运行所得的实验结果平均值。**（左）**为高度并发的集合数据结构自动构造证明期间出现的流图计算足迹的总运行时间。加速比表示 NEW 相较于 NAIVE 的相对性能提升。**（右）**按足迹大小划分的单个足迹计算平均运行时间（⊤ 表示失败）。

，用于计算所有流图对的足迹；(iii) NEW 相较于 NAIVE 的加速比（百分比）。实验在 Apple M1 Pro 上进行。

图 5（左）显示，所有方法的运行时间与计算足迹的数量大致呈线性关系。此外，计算足迹的绝对时间很短，因此这些方法具有实用性。该图还显示，我们的 NEW 和 DIST 方法相比 NAIVE 方法具有性能优势。NEW 方法比 NAIVE 方法快 22% 至 39%。我们认为差异相对较小，只是因为无环性假设避免了可能无法终止的不动点计算。在存在环的情况下避免这一不动点计算，是我们的 NEW 方法优于 NAIVE 和 DIST 方法的一项主要优势。DIST 和 NEW 之间的性能差异可以忽略不计，因为无环性检查的开销可以忽略不计。

我们还根据所得足迹的大小对基准测试的运行时间进行了分组。图 5（右）给出了计算单个足迹的平均运行时间和标准差，并按足迹大小细分。如果找不到足迹，其大小记为 ⊤。这些足迹构造失败的情况与`plankton`的方法一致，不会导致验证失败。

6 相关工作

先前的工作 [23, 24]提出了两种关于流框架的替代元理论。与我们的设置类似，原始流框架 [23] 要求流域是一个 ω-cpo，以获得最小不动点语义。然而，它提出了不同的流图复合方式，由此得到的上下文等价概念依赖于流入等价类。这使证明自动化变得复杂。此外，该流域被假定为半环，且边函数被限制为与常数相乘。这限制了表达能力。

如第 1 节所述， [24] 提出的修订后的流框架要求流幺半群是可消去的，但不是 ω-cpo。这意味着流的唯一性本身并不能得到保证。相反，唯一性是通过对边函数施加额外条件来实现的。然而，这些条件比我们框架中的条件更严格。 [24] 中引入的流图的容量与我们的传递函数概念密切相关。基于路径集上的求和所得到的闭式表示用于检查容量是否等价。然而，这种推理是重.....

限制为无环图。此外， [24] 没有提供计算流足迹的算法。

从某种意义上说，我们的工作在此前的两种元理论之间取得了平衡：它既保证流的唯一性，又不牺牲表达能力，同时还能实现更好的证明自动化。尽管如此，我们认为， [24]中提出的框架本身也具有独立的研究价值，尤其是在应用不要求流唯一时（即不要求对流施加某些下界，而在存在消失流时这些下界可能会平凡地成立）。消去性使人们能够将流入和流出聚合为一元函数，从而得到更小的流足迹（即更局部的证明）。

用于评估的基准测试套件取自plankton [26,27],，这是一个利用键集流验证并发搜索结构的工具。当程序修改符号堆时，plankton 会为被修改的节点以及与这些节点距离不超过 k 的所有节点创建流图。该流图被视为足迹，并会对上下文等价进行检查。此检查与 NAIVE 的基本相同。然而，论文没有给出底层流图概念的元理论，也没有为所实现的、用于推理流图的算法提供任何正确性论证。

流图构成一个分离代数。因此，所发展的理论可以与任何对底层分离代数具有参数化性质的现有分离逻辑结合使用，例如 [6, 7, 18, 27, 41, 44]。识别更新的足迹与分离逻辑中的框架推断问题有关，该问题已得到广泛研究 [4, 5, 15, 25, 35, 36, 42]。然而，现有工作主要关注以归纳谓词表示的断言的框架推断。这些技术并不适合对操纵一般图的程序进行推理，包括叠加结构；这类结构在实践中经常使用，并且可以很容易地用流表示。在分离逻辑中对一般堆图进行推理的一种常见方法，是使用迭代分离合取 [14, 39, 44, 47]，通过一个不依赖于程序状态的纯图来抽象堆。不过，对依赖于纯图的归纳性质的规范进行验证时，又会诉诸经典的一阶推理，并且难以实现自动化。一个例外是 [45] ，它使用SMT 求解器对由迭代分离合取描述的图中的二元可达关系进行框架推断。然而，该技术仅限于此类可达性性质。

在计算递归谓词 [8] 的后像时，很早就遇到了无界足迹。这引发了人们对分离逻辑片段的兴趣，因为其中的推理可以得到高效自动化 [2,3,9,17,20,35,38]。所有这些工作背后都存在一个限制：以某种方式假设堆具有树正则性，而流正是为克服这一限制而设计的。在程序（或幽灵代码）遍历无界足迹的情况下（更新之前或之后），近期的工作 [24, 27] 找到了一种将推理归约为有界足迹块的方法。

正如第 1 节所指出的，流的定义与经典的前向数据流分析表述非常相似。尤其是，对于分配边函数，其流方程的最小不动点可表征为流图中所有路径上的并，这与数据流分析中关于最大不动点的对偶结果相呼应 [19, 21]。类似地，流图的上下文等价概念与指称语义中的上下文程序等价和完全抽象模型有关 [16, 30, 37]。事实上，我们在定理 1 和引理 6 的证明中使用的 Beki´c 引理 [1], ，最初就是受对这类模型的研究启发而提出的。流图可以作为程序的抽象（而不仅仅是程序状态的抽象）。因此，我们相信，我们的结果也可能有助于开发增量式和组合式数据流分析框架。

数据可用性声明

本文生成和/或分析的 krill 工件和数据集可在 Zenodo 存储库 [28], https://zenodo.org/record/7566204 中获取。

致谢

本研究部分由美国国家科学基金会（NSF）资助（项目编号：1815633）。第一作者获得德国研究联合会（DFG）项目 *EDS@SYN*：面向综合的有效指称语义 的支持。第三作者获得西蒙斯基金会初级研究员奖学金的资助（855328，SW）。

参考资料

1. Beki´c, H.：一般代数中的可定义运算，以及自动机理论与流程图。载于：编程语言及其定义。计算机科学讲义，第 177 卷，0048939 2 30 55 1984 10 1007 页码 – 。施普林格（ ）。https://doi.org/ . /BFb 。Berdine, J., Calcagno, C., O'Hearn, P.W.：分离逻辑的一个可判定片段。载于：FSTTCS。计算机科学讲义，第 3328 卷，第 97–109 页。施普林格（ 2004）。https://doi.org/10.1007/978-3-540-30538-5_93。Berdine, J., Calcagno, C., O' Hearn, P.W.：Smallfoot：基于分离逻辑的模块化自动断言检查。载于：FMCO。计算机科学讲义，第 4111 卷，第 115– 137 2005 10 1007 11804192_6 页。施普林格（ ）。https://doi.org/ . / 。Berdine, J., Calcagno, C., O'Hearn, P.W.：使用分离逻辑进行符号执行。载于：APLAS。计算机科学讲义，第 3780 卷，第 52–68 页。施普林格（2005）。https: //doi.org/10.1007/11575467_55。Calcagno, C., Distefano, D., O' Hearn, P.W., Yang, H.：通过双向溯因进行组合式形状分析。载于：POPL。第 289–300 页。ACM（2009）。https://doi.org/10.1145/ 1480881.1480917 6. Calcagno, C., O'Hearn, P.W., Yang, H.：局部动作与抽象分离逻辑。载于：LICS。第 366 30 7 378 页。IEEE 计算机学会（2007）。https://doi.org/ . /LICS. . . Dinsdale-Young, T., Birkedal, L., Gardner, P., Parkinson, M.J., Yang, H.：视图：并发程序的组合式推理。载于：POPL。第 287–300 页。ACM（2013）。https: //doi.org/10.1145/2429069.2429104 8. Distefano, D., O'Hearn, P.W., Yang, H.：基于分离逻辑的局部形状分析。载于：TACAS。计算机科学讲义，第 3920 卷，第 287–302 页。施普林格（2006）。https://doi.org/10.1007/11691372_199。Enea, C., Lengál, O., Sighireanu, M., Vojnar, T.：SPEN：分离逻辑求解器。载于：NFM。计算机科学讲义，第 10227 卷，第 302–309 页（2017）。https://doi.org/10.1007/978-3-319-57288-8_2210。Feldman, Y.M.Y., Enea, C., Morrison, A., Rinetzky, N., Shoham, S.：从混沌中理出秩序：使用局部视图证明线性化。载于：DISC。LIPIcs，第 121 卷，第 23:1–23:21 页。施洛斯·达格斯图尔—莱布尼茨信息学中心（2018）。https://doi.org/10.4230/LIPIcs. DISC.2018.23 11. Harris, T.L.：非阻塞链表的一种务实实现。载于：DISC。计算机科学讲义，第 2180 卷，第 300–314 页。施普林格（2001）。https://doi.org/ 10.1007/3-540-45414-4_2112。Heller, S., Herlihy, M., Luchangco, V., Moir, M., III, W.N.S., Shavit, N.：一种惰性并发链表集合算法。载于：OPODIS。计算机科学讲义，第 3974 卷，第 3 16 2005 10 1007 11795490_3 – 13 页。施普林格（ ）。https://doi.org/ . / 。Herlihy, M., Shavit, N.：多处理器编程艺术。Morgan Kaufmann（2008）14. Hobor, A., Villard, J.：数据结构中共享所带来的影响。载于：POPL。第 523–536 页。ACM（2013）。https://doi.org/10.1145/2429069.2429131 15. Holík, L., Peringer, P., Rogalewicz, A., Soková, V., Vojnar, T., Zuleger, F.：底层双向溯因。载于：ECOOP。LIPIcs，第 222 卷，第 19:1–19:30 页。施洛斯·达格斯图尔—莱布尼茨信息学中心（2022）。https://doi.org/10.4230/LIPIcs.ECOOP.2022.19

16. Hyland, J.M.E., Ong, C.L.：关于 PCF 的完全抽象：i、ii 和 III。《信息与计算》**163**(2), 285–408 (2000)。https://doi.org/10.1006/inco.2000.2917 17. Iosif, R., Rogalewicz, A., Vojnar, T.：使用树自动机判定归纳分离逻辑中的蕴含关系。载于：ATVA。计算机科学讲义，第 8837 卷，第 201–218 页。施普林格（2014）。https://doi.org/10.1007/978-3-319-11936-6_1518。Jung, R., Krebbers, R., Jourdan, J., Bizjak, A., Birkedal, L., Dreyer, D.：从基础开始构建 Iris：高阶并发分离逻辑的模块化基础。《函数式编程学报》**28**, e20 (2018)。https://doi.org/10.1017/S0956796818000151 19. Kam, J.B., Ullman, J.D.：单调数据流分析框架。《信息学学报》**7**, 305–317 (1977)。https://doi.org/10.1007/BF00290339 20. Katelaan, J., Zuleger, F.：超越符号堆：使用归纳定义判定分离逻辑。载于：LPAR。《计算领域 EPiC 系列》，第 73 卷，第 390–408 页。EasyChair（2020）。https://doi.org/10.29007/vkmj 21. Kildall, G.A.：一种全局程序优化的统一方法。载于：POPL。第 194–206 页。ACM出版社（1973）。https://doi.org/10.1145/512927.512945 22. Krishna, S., Patel, N., Shasha, D.E., Wies, T.：验证并发搜索结构模板。载于：PLDI。第 181–196 页。ACM（2020）。https://doi.org/10.1145/3385412. 3386029 23. Krishna, S., Shasha, D.E., Wies, T.：顺流而行：并发数据结构的组合式抽象。《ACM 程序设计语言汇刊》**2**(POPL), 37:1–37:31 (2018)。https: //doi.org/10.1145/3158125 24. Krishna, S., Summers, A.J., Wies, T.：针对全局图性质的局部推理。载于：ESOP。计算机科学讲义，第 12075 卷，第 308–335 页。施普林格（2020）。https:// doi.org/10.1007/978-3-030-44914-8_1225。Le, Q.L., Sun, J., Qin, S.：用于分离逻辑归纳蕴含证明的框架推断。载于：TACAS (1)。计算机科学讲义，第 10805 卷，第 41–60 页。施普林格（ 2018 ）。https://doi.org/10.1007/978-3-319-89960-2_326。Meyer, R., Wies, T., Wolff, S.：《具有未来与历史的并发程序逻辑》配套资料（2022 年 9 月）。https://doi.org/10.5281/zenodo.7080459 27. Meyer, R., Wies, T., Wolff, S.：具有未来与历史的并发程序逻辑。《ACM 程序设计语言汇刊》**6**(OOPSLA2), 1378–1407 (2022)。https://doi.org/10. 1145/3563337 28. Meyer, R., Wies, T., Wolff, S.：《让流分析框架重焕生机：重新审视流框架》配套资料（2023 年 1 月）。https://doi.org/10.5281/zenodo.7566204 29. Michael, M.M.：高性能动态无锁哈希表和基于链表的集合。载于：SPAA。第 73–82 页。ACM（2002）。https://doi.org/10.1145/564870.564881 30. Milner, R.：类型化 *lambda* 演算的完全抽象模型。《理论计算机科学》**4**(1), 1–22 (1977)。https://doi.org/10.1016/0304-3975(77)90053-6 31. de Moura, L.M., Bjørner, N.S.：Z3：高效的 SMT 求解器。载于：TACAS。计算机科学讲义，第 4963 卷，第 337–340 页。施普林格（2008）。https://doi.org/10.1007/978-3-540-78800-3_2432。O'Hearn, P.W., Reynolds, J.C., Yang, H.：对改变数据结构的程序进行局部推理。载于：CSL。计算机科学讲义，第 2142 卷，第 1–19 页。施普林格（2001）。https://doi.org/10.1007/3-540-44802-0_133。O'Hearn, P.W., Rinetzky, N., Vechev, M.T., Yahav, E., Yorsh, G.：利用事后推理验证线性化。载于：PODC。第 85–94 页。ACM（2010）。https://doi.org/10.1145/1835698.1835722 34. Patel, N., Krishna, S., Shasha, D.E., Wies, T.：验证并发多副本搜索结构。《ACM 程序设计语言汇刊》**5**(OOPSLA), 1–32 (2021)。https://doi.org/10.1145/3485490 35. Piskac, R., Wies, T., Zufferey, D.：使用 SMT 实现分离逻辑自动化。载于：CAV。计算机科学讲义，第 8044 卷，第 773–789 页。施普林格（2013）。https://doi.org/10.1007/978-3-642-39799-8_5436。Piskac, R., Wies, T., Zufferey, D.：使用树与数据实现分离逻辑自动化。载于：CAV。计算机科学讲义，第 8559 卷，第 711–728 页。施普林格（2014）。https://doi.org/10.1007/978-3-319-08867-9_47

37. Plotkin, G.D.：将 LCF 视为一种编程语言。《理论计算机科学》**5**(3)，223–255 5 38 1977 10 1016 0304 3975 77 90044 ( )。https://doi.org/ . / - ( ) - . Qiu, X., Wang, Y.：一种用于带度量的树形数据结构的可判定逻辑。载于：VMCAI。《计算机科学讲义》，第 11388 卷，第 318–341 页。施普林格（2019）。https:// doi.org/10.1007/978-3-030-11245-5_1539. Raad, A., Hobor, A., Villard, J., Gardner, P.：验证并发图算法。载于：APLAS。《计算机科学讲义》，第 10017 卷，第 314–334 页（2016）。https: // doi.org/10.1007/978-3-319-47958-3_1740. Reynolds, J.C.：分离逻辑：一种用于共享可变数据结构的逻辑。载于：LICS。第 55 1029817 41 74 2002 10 1109 2002 – 页。IEEE 计算机学会（）。https://doi.org/ . /LICS. . . da Rocha Pinto, P., Dinsdale-Young, T., Gardner, P.：Tada：一种用于时间和数据抽象的逻辑。载于：ECOOP。《计算机科学讲义》，第 8586 卷，第 207–231 页。施普林格（ 2014 10 1007 978 3 662 44202 9_9 ）。https://doi.org/ . / - - - - . Rowe, R.N.S., Brotherston, J.：分离逻辑中递归过程的自动循环终止证明。载于：CPP。第 53–65 页。ACM（2017）。https://doi.org/10.1145/ 3018610.3018623 43. Scott, D.：计算数学理论纲要。技术报告 PRG02，牛津大学计算机实验室（1970）44. Sergey, I., Nanevski, A., Banerjee, A.：细粒度并发程序的机械化验证。载于：PLDI。第 77–87 页。ACM（2015）。https://doi.org/10.1145/2737924. 2737964 45. Ter-Gabrielyan, A., Summers, A.J., Müller, P.：分离逻辑中堆可达性属性的模块化验证。《ACM 程序设计语言汇刊》**3**(OOPSLA)，121:1–121:28（2019）。https://doi.org/10.1145/3360547 46. Vechev, M.T., Yahav, E.：推导线性化的细粒度并发对象。载于：PLDI。第 125 1375598 47 135 2008 10 1145 1375581 – 页。ACM（）。https://doi.org/ . / . . Yang, H.：BI 指针逻辑中局部推理的一个示例：Schorr-Waite 图标记算法。载于：SPACE 研讨会论文集（2001）

A 评估详情

详细运行时间 我们在图 6 中再次展示图 5（右）的放大版本，其中给出了计算单个足迹的平均运行时间，并按足迹大小划分。在图 7 中，我们还按照足迹所属的数据结构，对计算单个足迹的运行时间进行划分。

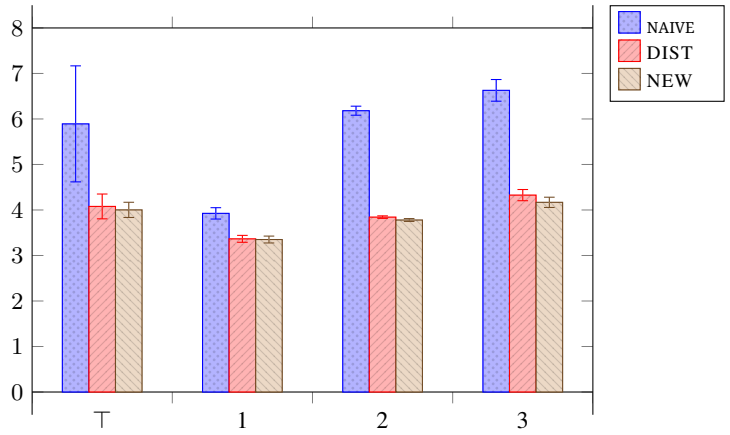


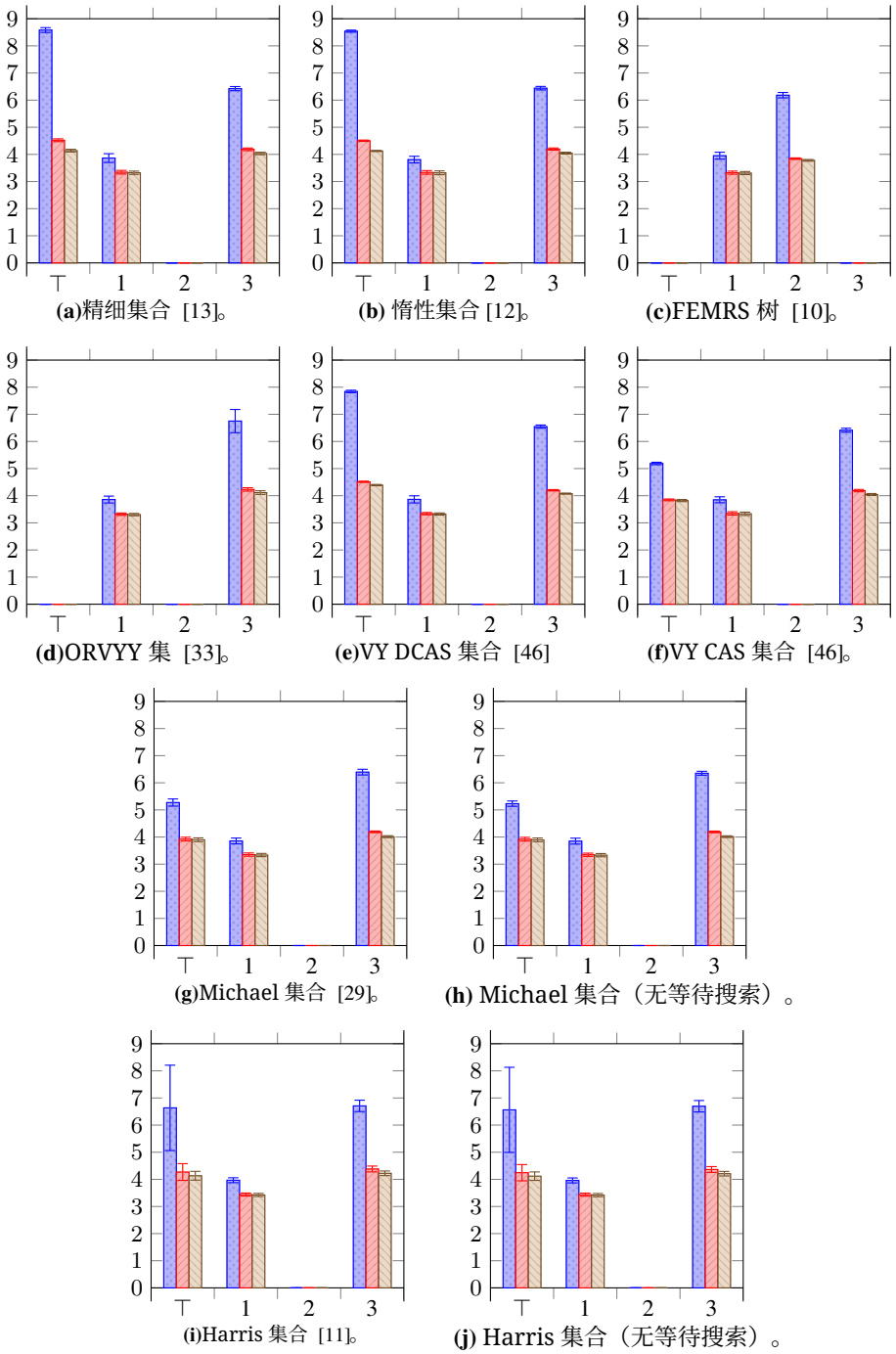
图 6。 按足迹大小划分的单个足迹计算平均运行时间（⊤ 表示失败）。重复自图 5（右）。

符号流图 回顾第 5 节，为了进行评估，我们提取了 plankton 工具会为其计算足迹的流图。在 plankton 中，流图通过分离逻辑断言以符号形式编码，A 其形式为 $A \triangleq F * \divideontimes_{x \in N} \text{Obj}(x)$，其中 F 可视为纯栈（它不包含资源，$F * F = F$），而迭代分离合取则包含谓词 Obj(x)，反映堆中每个对象 $x \in N$ 所对应的资源。（这意味着 N 表示一个可能无界的集合。）Obj(x) 谓词大致上是一个指向谓词，其中包含对象 x 的字段。例如，如果 x 的类型为 T，且其定义如下：

struct T { int 键; bool 标记; T* next; }

那么，x的资源 Obj(x) 定义为 Obj(x) $\triangleq x \mapsto M$，k，m，n其中 M、k、m 和 n 是存在量化的（在断言层面）符号变量，分别指代 x 的流、键字段、标记字段和 next 字段。注意，流 M 是幽灵状态：它并非节点物理表示的一部分。这些符号变量的值可能受栈 F 约束，也可能不受其约束。

为了将断言 A 转换为流图 h $=$(X , E , in)，我们需要定义 A 如何导出节点 X、边 E 和流入 in。对于 X，我们只需选择堆中的对象，X $=$ N。边模拟堆结构。也就是说，如果对象 x 的指针字段引用对象 y，那么我们就将该指针作为一条边加入 E。在上面的示例中，如果 $A \models \text{x} \mapsto M, k, m, n * n = \text{y} * \text{true}$，且边函数 f 选择得当，则有 (x , y f) $\in$ E。我们将讨论如何才算 f 选择得当。在 plankton 中，边函数由资源 Obj(–) 和栈 F 导出。更准确地说，用户为每个类型 T 的每个指针字段 next 指定一个生成器函数 $\text{gen}_{\text{T,next}}$，该函数为一条边生成边函数 f。



(a)精细集合 [13]。 **(b)** 惰性集合 [12]。 **(c)**FEMRS 树 [10]。

(d)ORVYY 集 [33]。 **(e)**VY DCAS 集合 [46] **(f)**VY CAS 集合 [46]。

(g) Michael 集合 [29]。 **(h)** Michael 集合（无等待搜索）。

(i) Harris 集合 [11]。 **(j)** Harris 集合（无等待搜索）。

图 7. 计算单个足迹的平均运行时间，按数据结构和足迹大小分类（⊤表示失败）。

介于 x 和y之间。为此，生成器的输入是x的所有字段的赋值（物理字段，而非幽灵状态）。继续这个例子，我们使用一个类型为

$$gen_{\mathtt{T},\mathtt{next}} : \mathbb{Z} \times \mathbb{B} \times N \to \mathit{ContFun}(\mathbb{M} \to \mathbb{M})$$

的生成器，它依次以对象的键、标记和 next 字段作为输入。然后，我们可以如下定义生成器，以获得例如示例 1/图 2 中的边标记：

$$gen_{\mathtt{T},\mathtt{next}}(k, m, n) \triangleq \begin{cases} \lambda_{-\infty} & \text{if } m = \mathit{true} \\ \lambda_k & \text{otherwise .} \end{cases}$$

值得指出的是，栈可能不会约束 m。在这种情况下， A 表示多个流图：一种情况是我们有 $gen_{\mathtt{T},\mathtt{next}}(k, m, n) = \lambda_{-\infty}$，另一种情况是我们有 $gen_{\mathtt{T},\mathtt{next}}(k, m, n) = \lambda_k$。对于流入，我们选择任意函数$in$，使流与栈 F保持一致。同样，这会产生多个、可能无穷多个 in 的选择。

基于上述对断言如何表示流图的理解，图 2（左）中的流图可由以下断言描述

$$l \mapsto M_l, 6, \mathit{false}, t \ * \ t \mapsto M_t, k_t, \mathit{true}, r \ * \ r \mapsto M_r, 8, \mathit{false}, v \ * \ M_l = (3, \infty)$$

其中，我们将一些值内联到指向谓词中，使断言更易于阅读。请注意，我们仅指定了 l、$M_l = (3, \infty)$ 中的流。对于 t 和 r 中的流，该断言与上述生成器共同导出 $M_t = (6, \infty) = M_r$。

我们从 `plankton` 中提取的基准测试套件采用半符号表示。这意味着我们提取了流图的节点 $X\,h$ 及其（符号）字段值。不过，我们并未显式存储边 E 和流入 in，而是记录栈 F 和边函数生成器。随后，我们的工具 `krill` 将第 4 节中讨论的技术应用于这种半符号表示。我们使用`Z3`来处理此过程涉及的 SMT 查询。

B 第 2 节的证明

引理 2 的证明。 交换律和单位元律可直接由复合的定义得出。对于结合律，我们证明 $h_2 \# h_3$ 和 $h_1 \#(h_2 * h_3)$ 成立，当且仅当 $h_1 \# h_2$ 和 $(h_1 * h_2) \# h_3$ 成立。我们证明从左向右的方向。反向则由定义性的对称性得出。

由于 $\mathrm{h}_2 \# \mathrm{h}_3$，我们有$(\mathrm{h}_2 \uplus \mathrm{h}_3).\mathrm{流} \geq \mathrm{h}_2.\mathrm{流} \uplus \mathrm{h}_3.\mathrm{流}$。此外，由于 $\mathrm{h}_1 \#(\mathrm{h}_2 * \mathrm{h}_3)$我们有$(\mathrm{h}_1 \uplus (\mathrm{h}_2 * \mathrm{h}_3)).\mathrm{流} \geq \mathrm{h}_1.\mathrm{流} \uplus (\mathrm{h}_2 * \mathrm{h}_3).\mathrm{流}$。注意，我们有$(\mathrm{h}_1 \uplus (\mathrm{h}_2 * \mathrm{h}_3)).\mathrm{流} = (\mathrm{h}_1 \uplus \mathrm{h}_2 \uplus \mathrm{h}_3).\mathrm{流}$。

为证明 $\mathrm{h}_1 \# \mathrm{h}_2$，，我们必须论证$(\mathrm{h}_1 \uplus \mathrm{h}_2).\mathrm{流} \geq \mathrm{h}_1.\mathrm{流} \uplus \mathrm{h}_2.\mathrm{流}$。为了解这一点，请注意，$(\mathrm{h}_1 \uplus \mathrm{h}_2).\mathrm{流} \geq (\mathrm{h}_1 \uplus \mathrm{h}_2 \uplus \mathrm{h}_3).\mathrm{流}$|$_{(h_1 \uplus h_2).X} \geq \mathrm{h}_1.\mathrm{流} \uplus \mathrm{h}_2.\mathrm{流}$。后一个不等式由假设得出。对于前一个不等式，我们注意到，$(\mathrm{h}_1 \uplus \mathrm{h}_2).\mathrm{流}$的不动点迭代从$\mathrm{h}_3$的贡献（给定为流入）开始，而$(\mathrm{h}_1 \uplus \mathrm{h}_2 \uplus \mathrm{h}_3).\mathrm{流}$的迭代只有在达到不动点时才会接收到该贡献。根据单调性，左侧的每个不动点近似值都大于右侧对应的近似值，因此不动点也是如此。

对于 $(h_1 * h_2) \# h_3$，，我们注意到

$$\begin{aligned}((h_1 * h_2) \uplus h_3).\mathit{flow} = (h_1 \uplus h_2 \uplus h_3).\mathit{flow} &\geq h_1.\mathit{flow} \uplus h_2.\mathit{flow} \uplus h_3.\mathit{flow} \\ &\geq (h_1 \uplus h_2).\mathit{flow} \uplus h_3.\mathit{flow} \ .\end{aligned}$$

第一个不等式由上述假设得出。正如上文所述，第二个不等式始终成立。□

C 第3节的证明

定理1(i)的证明。对于 $h_2.X \cap h.X = \varnothing$，我们根据 $h_1.X \cap h.X = \varnothing$，利用 $h_1 \# h$以及 $h_1.X = h_2.X$。为了说明 h_2 的流出与 h 的流入相匹配，我们考察 $x \in h_2.X$ 和 $y \in h.X$，并作如下推理：

$$h_2.out(x,y) = h_1.out(x,y) = h.in(x,y).$$

前一个等式由 $h_2.in = h_1.in$ 和 $tf(h_1) =_{h_1.in} tf(h_2)$ 得出。第二个等式由 $h_1 \# h$ 得出。根据假设，流入保持不变，因此我们得到以下等式：$h.out(y,x) = h_1.in(y,x) = h_2.in(y,x)$.

还需证明 $(h_2 \uplus h).流 = h_2.流 \uplus h.流$。我们使用 Beki´c 引理 [1]。将两个定义在相同定义域 A 上的函数 $f\colon A \to B$ 和 $g\colon A \to C$ 的目标配对定义为函数 $\langle f,g\rangle\colon A \to B \times C$，其中 $\langle f,g\rangle(a) \triangleq (f(a)\ g(a))$。我们将 $h_2 \uplus h$ 的流计算为目标配对 $\langle f,g\rangle$ 的最小不动点，其中

$$f : ((h_2.X \uplus h.X) \to \mathbb{M}) \to h_2.X \to \mathbb{M}$$
$$g : ((h_2.X \uplus h.X) \to \mathbb{M}) \to h.X \to \mathbb{M}\,.$$

函数 f 会根据h中的流/来自h的流入，更新h_2 中节点的流。函数 g 负责h中节点的流。来自$h_2 \uplus h$ 外部节点的流入是常量。该定义保证了（$h_2 \uplus h$）。流 $=$ 最小不动点.$\langle f,g\rangle$。我们将前一个函数柯里化，

$$f\colon (h.X \to \mathbb{M}) \to (h_2.X \to \mathbb{M}) \to h_2.X \to \mathbb{M}\,,$$

并对每个$cval\colon h.X \to \mathbb{M}$，得到函数

$$f(cval)\colon (h_2.X \to \mathbb{M}) \to h_2.X \to \mathbb{M}\,.$$

这个函数仍然是单调的，因此存在一个最小不动点。因此，函数

$$f^\dagger : (h.X \to \mathbb{M}) \to h_2.X \to \mathbb{M}$$

将赋值$cval\colon h.X \to \mathbb{M}$映射到最小不动点最小不动点.$f(cval)$，这是定义良好的。

Beki´c 引理告诉我们如何通过依次消去变量，计算上文这类目标配对$\langle f,g\rangle$的最小不动点。我们首先确定 $f^\dagger$，它是$h.X \to \mathbb{M}$中的一个函数。我们将这个函数代入 g，得到一个仅关于$h.X \to \mathbb{M}$的函数。确切地说，由于 g需要一个从（$h_2.X \uplus h.X$）$\to \mathbb{M}$出发的函数，我们将$f^\dagger$与$id = id_{h.X \to \mathbb{M}}$配对，得到

$$\langle f^\dagger, id\rangle : (h.X \to \mathbb{M}) \to (h_2.X \uplus h.X) \to \mathbb{M}\,.$$

我们将此函数与 g 组合，得到

$$g \circ \langle f^\dagger, id\rangle : (h.X \to \mathbb{M}) \to h.X \to \mathbb{M}\,.$$

我们计算这个复合的最小不动点，以获得 $h.X$ 上所关注的最小不动点的值。对于 $h_2.X$ 上的值，我们将 $h.X$ 的值重新代入$f^\dagger$。Beki´c 引理保证了这一逐步消元过程的正确性：

$$lfp.\langle f,g\rangle \;=\; (f^\dagger(cval), cval) \qquad \text{with} \qquad cval \;=\; lfp.g \circ \langle f^\dagger, id\rangle\,.$$

为完成证明，我们回顾 $tf(h_2) =_{h_1.in} tf(h_1)$。此外，对于所有 $y \in h.X$和 $x \in h_1.X$，我们有 $h_1.in(y,x) = h.out(y,x) = h.E(y,x)(h.流(x))$。综上

结合边函数的单调性，这意味着 $g \circ \langle f^\dagger, id\rangle =_{h.flow} g \circ \langle e^\dagger, id\rangle$。这里， e： $((h_2.X \uplus h.X) \to \mathbb{M}) \to h_2.X \to \mathbb{M}$ 是从h_1 推导出的变换器，其方式与从h_2推导出 f 的方式相同。因此，对于所有$x \in h.$X</style}：

$$\begin{aligned}
& & (h_2 \uplus h).flow(x) \\
\text{(Definition } flow, f, g) & = & lfp.\langle f, g\rangle(x) \\
\text{(Bekić's lemma, } x \in h.X) & = & lfp.g \circ \langle f^\dagger, id\rangle(x) \\
(tf(h_2) =_{h_1.in} tf(h_1) \text{, see above}) & = & lfp.g \circ \langle e^\dagger, id\rangle(x) \\
\text{(Bekić's lemma, } x \in h.X) & = & lfp.\langle e, g\rangle(x) \\
\text{(Definition flow, } f, g) & = & (h_1 \uplus h).flow(x) \\
(h_1 \# h, x \in h.X) & = & h.flow(x).
\end{aligned}$$

我们论证，对于节点 $x \in h_2.X$我们也有 $(h_2 \uplus h).$流$(x) = h_2.$流(x)：

$$\begin{aligned}
& & (h_2 \uplus h).flow(x) \\
\text{(Definition } flow, f, g) & = & lfp.\langle f, g\rangle(x) \\
\text{(Bekić's lemma, } x \in h_2.X) & = & [f^\dagger(cval)](x) \\
\text{(See above)} & = & [f^\dagger(h.flow)](x) \\
\text{(See below)} & = & h_2.flow(x).
\end{aligned}$$

要看出最后一个等式，注意，对于所有 $y \in h.X$， $h_2.in(y, x) = h.out(y, x) = h.E(y, x)$ ($h.flow(x)$) 成立。因此，我们从流入计算得到的 $h_2.flow$，是用$h.flow$ 固定计算出的最小不动点。□

定理 1(ii) 的证明。通过类似但更简单地应用 Beki´c 引理即可得出，其应用方式与定理1(i)相同。□

引理4 的证明。 我们使用 X 表示节点集合 h。X。

(i) 我们立即得到 $\sqsupseteq$。对于 $\sqsubseteq$，我们利用如下事实：如引理 5(ii) 所明确指出，流被定义为一个最小不动点：$h|_Y$.流 = 最小不动点。($h|_Y[h|_Y.in]$) and$h$。流$|_Y$ =(最小不动点。$h[h$。$in])|_Y$。不动点通过 Kleene 迭代计算：最小不动点。($h|_Y[h|_Y.in]$) 是升链的并

$$(h|_Y[h|_Y.in])^0(\bot) \sqsubseteq (h|_Y[h|_Y.in])(\bot) \sqsubseteq (h|_Y[h|_Y.in])^2(\bot) \sqsubseteq \ldots$$

其中，无迭代时的值为 $(h|_Y[h|_Y.in])^0(\bot) \triangleq \bot$ 而对于迭代 $i+1$，我们有 $(h|_Y[h|_Y.in])^{i+1}(\bot) \triangleq (h|_Y[h|_Y.in])((h|_Y[h|_Y.in])^i(\bot))$。现在我们将证明，这些所谓的 Kleene 逼近序列中的每一个都被 $h.$流$|_Y$ 支配。因此，受限流是序列中所有元素的上界，从而大于或等于并。

在基本情形中，我们有 $\bot \sqsubseteq h$。流$|_Y$。假设 $(h|_Y[h|_Y.$属于$])^i(\bot) \sqsubseteq h$。流$|_Y$。我们有

$$\begin{aligned}
(h|_Y[h|_Y.in])^{i+1}(\bot) &= (h|_Y[h|_Y.in])((h|_Y[h|_Y.in])^i(\bot)) \\
&\sqsubseteq (h|_Y[h|_Y.in])(h.flow|_Y) \\
&= h.flow|_Y\ .
\end{aligned}$$

第一个等式根据Kleene 逼近序列的定义得出。下面的不等式是边函数的单调性与归纳假设相结合的结果。对于最后一个等式，我们利用了流是最小不动点这一事实，$h[h.in]$($h.$流) $= h.$流，以及受限流图的流入定义。

(ii)我们首先证明复合的定义性。节点集合的不相交性是显然的。流的相等性由 (i) 得出。对于流入和流出的相容性，令 $y \in X \cap Y$以及 $x \in X \setminus Y$。我们证明 $h|_Y.out(y, x) = h|_{X \setminus Y}.in(y, x)$。我们有

$$\begin{aligned} h|_Y.out(y,x) &= h|_Y.E_{(y,x)}(h|_Y.flow(y)) \\ &= h.E_{(y,x)}(h|_Y.flow(y)) \\ &= h.E_{(y,x)}(h.flow(y)) \,. \end{aligned}$$

第一个等式是流出的定义，下一个等式根据限制的定义得出，最后一个等式是 (i)。同样，根据限制的定义，我们有：

$$h|_{X \setminus Y}.in(y,x) = h.E_{(y,x)}(h.flow(y)) \,.$$

这两个值相同，符合要求。反方向的论证是对称的。剩下需要证明复合满足 $h|_Y * h|_{X \setminus Y} = h$。节点集合立即相同，边集合也相同。根据限制的定义，来自 $h.X$ 外部的流入相同。来自另一个流图的流入（分别出现在$h|_Y$和$h|_{X \setminus Y}$中）会被复合移除，符合要求。

(iii) 限制后的节点集合在两种情况下都是$h.X \cap Y \cap Z$，我们将其记为X'。边也相同，因为它们是原始边在$X' \times \mathbb{N}$上的限制。从$h.X$外部的节点到X'的流入，对于 $(h|_Y)|_Z$ 和 $h|_{Y \cap Z}$ 立即相同。考虑从节点 $z \in h.X \setminus X'$ 到 $x \in X'$的流入。我们有

$$h|_{Y \cap Z}.in(z,x) = h.E_{(z,x)}(h.flow(z)) \,.$$

在 $(h|_Y)|_Z$ 中，有两种情况。如果 $z \in h.X \cap (Y \setminus Z)$，则有

$$(h|_Y)|_Z.in(z,x) = h|_Y.E_{(z,x)}(h|_Y.flow(z)) = h.E_{(z,x)}(h.flow(z)) \,.$$

第一个方程是限制的定义。在下一个方程中，边相等同样依据限制的定义。根据 (i)，流重合。如果 $z \in h.X \setminus Y$，我们有

$$\begin{aligned} (h|_Y)|_Z.in(z,x) &= h|_Y.in(z,x) \\ &= h.E_{(z,x)}(h.flow(z)) \,. \end{aligned}$$

根据限制的定义，两个方程均成立。推理表明，流入也相同。□

引理 3 的证明。 设 $Z \in FFP(h_1, h_2)$ 且 $Z \subseteq Y \subseteq X \triangleq h_1.X$。我们首先证明$h_1|_Y =_{ctx} h_2|_Y$。对于 $i = 1,2$，由$Z \subseteq Y$这一事实以及引理 4(iii)可得：

$$h_i|_Z = h_i|_{Z \cap Y} = (h_i|_Y)|_Z \,.$$

结合$h_1|_Z =_{ctx} h_2|_Z$这一事实，即Z是一个流足迹，我们得到

$$(h_1|_Y)|_Z =_{ctx} (h_2|_Y)|_Z \,. \tag{1}$$

此外，由于 Z是一个足迹，我们有 $h_1|_{X \setminus Z} = h_2|_{X \setminus Z}$。这意味着 $(h_1|_{X \setminus Z})|_Y = (h_2|_{X \setminus Z})|_Y$。我们使用引理 4(iii)和 $Y \subseteq X$，推导出 $i = 1,2$：

$$(h_i|_{X \setminus Z})|_Y = h_i|_{(X \setminus Z) \cap Y} = h_i|_{Y \setminus Z} = (h_i|_Y)|_{Y \setminus Z} \,.$$

因此，我们有了

$$(h_1|_Y)|_{Y \setminus Z} = (h_2|_Y)|_{Y \setminus Z} \,. \tag{2}$$

方程 (1) 和 (2) 结合引理 4(ii) 和定理 1 中的定义性，得出

$$(h_1|_Y)|_Z * (h_1|_Y)|_{Y\setminus Z} =_{ctx} (h_2|_Y)|_Z * (h_2|_Y)|_{Y\setminus Z} . \tag{3}$$

引理 4(ii) 表明，对于 $i = 1,2$：

$$(h_i|_Y)|_Z * (h_i|_Y)|_{Y\setminus Z} = h_i|_Y .$$

结合方程（3），可得所需结果

$$h_1|_Y =_{ctx} h_2|_Y .$$

还需证明 $h_1|_{X\setminus Y} = h_2|_{X\setminus Y}$。由于 $h_1|_{X\setminus Z} = h_2|_{X\setminus Z}$，我们有

$$(h_1|_{X\setminus Z})|_{X\setminus Y} = (h_2|_{X\setminus Z})|_{X\setminus Y} . \tag{4}$$

由引理 4(iii) 和 $Z \subseteq Y$，对于 $i = 1,2$，我们得到：

$$(h_i|_{X\setminus Z})|_{X\setminus Y} = h_i|_{(X\setminus Z)\cap(X\setminus Y)} = h_i|_{X\setminus Y} .$$

结合方程（4），可得

$$h_1|_{X\setminus Y} = h_2|_{X\setminus Y} .$$

证明至此结束。 □

D 第 4 节的证明

引理 6 的证明。来自X之外节点的流入 $\triangleq h_1$.流入 $= h_2$.流入在两个流图中相同。为了证明来自$X \setminus Z$的流入一致，我们证明$X \setminus Z$中的流一致。

我们首先将h_i中的流计算为$h_i|_Z$与$h_i|_{X\setminus Z}$之间的值交换。根据引理 4，我们有$h_i = h_i|_Z * h_i|_{X\setminus Z}$。我们可以将$h_i|_Z$和$h_i|_{X\setminus Z}$理解为函数

$$f_i : (X \to \mathbb{M}) \to Z \to \mathbb{M}$$
$$g_i : (X \to \mathbb{M}) \to (X \setminus Z) \to \mathbb{M} .$$

函数 f_i 计算节点从Z流入的流，该流取决于 Z和 $X \setminus Z$中当前的流。来自X 外部的流入恒定为in。类似地， g_i计算 $X \setminus Z$ 中的流，该流取决于 $X \setminus Z$和 Z中当前的流，其中in 同样保持恒定。根据 $odif_{h_1,h_2} \subseteq Z$这一事实，从 $X \setminus Z$中节点出发的边相重合，因此我们有 $g_1 = g_2$。我们将此函数称为 g。现在我们考虑目标配对

$$\langle f_i, g\rangle : (X \to \mathbb{M}) \to [(Z \to \mathbb{M}) \times ((X \setminus Z) \to \mathbb{M})] =_{iso} (X \to \mathbb{M}) .$$

由引理 5(ii) 及 f_i 和 g 的定义，我们有

$$h_i.flow \quad = \quad lfp.h_i[in] \quad = \quad lfp.\langle f_i, g\rangle . \tag{5}$$

Bekic 引理 [1] 说明了如何计算后一个不动点：

$$lfp.\langle f_i, g\rangle \;=\; (f_i^\dagger(cval_i), cval_i) \quad \text{with} \quad cval_i \;=\; lfp.g \circ \langle f_i^\dagger, id_{(X\setminus Z)\to\mathbb{M}}\rangle . \tag{6}$$

为了定义函数 $f_i^\dagger$，我们对函数 f_i 进行柯里化，得到

$$f_i : ((X \setminus Z) \to \mathbb{M}) \to (Z \to \mathbb{M}) \to Z \to \mathbb{M} .$$

给定一个函数 $cval : (X \setminus Z) \to \mathbb{M}$，我们有

$$f_i(cval) : (Z \to \mathbb{M}) \to Z \to \mathbb{M} .$$

该函数再次连续，因此存在一个最小不动点。我们定义

$$\begin{aligned} f_i^\dagger : ((X \setminus Z) \to \mathbb{M}) &\to Z \to \mathbb{M} \\ cval &\mapsto lfp.f_i(cval) . \end{aligned}$$

函数 $f_i^\dagger$ 具有直观的含义。它给出 Z 中的流，其中 $X \setminus Z$中的流为$cval$。换一种表述，它给出 Z 中的流，其中从 $X \setminus Z$ 流入的值为 $out(cval)|_{(X\setminus Z)\times Z}$，其中 out 的定义与传递函数中的定义相同，而从$\mathbb{N} \setminus X$ 流入的值为 in。

$$in_{cval} \triangleq in|_{(\mathbb{N}\setminus X)\times Z} \cup out(cval)|_{(X\setminus Z)\times Z}$$

我们得到

$$f_i^\dagger(cval) \quad = \quad lfp.f_i(cval) \quad = \quad lfp.h_i|_Z[in_{cval}] .$$

我们将 $f_i^\dagger$ 与 out 复合，并应用引理 5：

$$out \circ f_i^\dagger(cval) \quad = \quad out \circ lfp.h_i|_Z[in_{cval}] \quad = \quad tf(h_i|_Z[in_{cval}]) .$$

事实是 $tfail_{h_1,\ h_2}(Z) = \varnothing$意味着 $tf(h_1|_Z) =_{h_1|_Z.in} tf(h_2|_{X\setminus Z})$。此外，由于 $h_1|_Z$.属于$(x, z) = h_2|_Z$.属于$(x,\ z)$ 对所有 $x \in \mathbb{N} \setminus X$、$z \in Z$，且由于边函数是单调的，我们有

$$out \circ f_1^\dagger(cval) \quad = \quad out \circ f_2^\dagger(cval) \quad \text{for all } cval \le h_1|_{X\setminus Z}.flow . \tag{7}$$

现在我们将证明，$X \setminus Z$ 中的流对于h_1 和h_2是一致的。我们将引理 4(i)与方程 (5) 和 (6)结合起来，得到

$$h_i|_{X\setminus Z}.flow = (h_i.flow)|_{X\setminus Z} = (lfp.\langle f_i, g\rangle)|_{X\setminus Z} = lfp.g \circ \langle f_i^\dagger, id\rangle . \tag{8}$$

因此，我们旨在证明 $lfp.g \circ \langle f_1^\dagger, id\rangle = lfp.g \circ \langle f_2^\dagger, id\rangle$。展开这些函数的定义会有所帮助。给定一个属于 $X \setminus Z$ 的流 $cval$，它们为 $x \in X \setminus Z$ 产生以下值：

$$\begin{aligned} &(g \circ \langle f_i^\dagger, id\rangle)(cval)(x) \\ &\quad = \sum_{z\in Z} out(f_i^\dagger(cval))(z, y) + \sum_{y\in X\setminus Z} h_i.E_{(y,x)}(cval(y)) + in_x . \end{aligned} \tag{9}$$

此处，$in_x = \sum_{y\in\mathbb{N}\setminus X} in(y,\ x)$ 与传递函数的定义中相同。请注意，$(g \circ \langle f_1^\dagger, id\rangle)(cval)$ 和 $(g \circ \langle f_2^\dagger, id\rangle)(cval)$ 之间唯一的区别，分别在于将 $out \circ f_1^\dagger$ 和 $out \circ f_2^\dagger$ 应用于 $cval$。然而，根据方程（7），当$cval \le h_1|_{X\setminus Z}$ ***flow*** 时，这些函数相同。

现在，我们将 $(g \circ \langle f_1^\dagger, id\rangle)(cval)$ 与 $(g \circ \langle f_2^\dagger, id\rangle)(cval)$ 相等这一推测推广到不动点。根据克莱尼定理，$lfp.g \circ \langle f_i^\dagger, id\rangle$ 是以下升链的并

$$(g \circ \langle f_i^\dagger, id\rangle)^0(\bot) \sqsubseteq (g \circ \langle f_i^\dagger, id\rangle)(\bot) \sqsubseteq (g \circ \langle f_i^\dagger, id\rangle)^2(\bot) \sqsubseteq \ldots$$

其中包含 $(g\circ\langle f_i^\dagger, id\rangle)^0(\bot) \triangleq \bot$和 $(g\circ\langle f_i^\dagger, id\rangle)^{j+1}(\bot) \triangleq (g\circ\langle f_i^\dagger, id\rangle)[(g\circ\langle f_i^\dagger, id\rangle)^j(\bot)]$。我们将证明，对于所有 j，都有 $(g\circ\langle f_i^\dagger, id\rangle)^j(\bot) \leq h_1|_{X\setminus Z}.\text{流}$。因此，上述推理表明，Kleene 逼近序列彼此一致。这一结论也适用于并。

我们对 j 进行归纳。在基础情形 $j = 0$ 中无需做任何处理，因为 $(g \circ \langle f_i^\dagger, id\rangle)^0(\bot) = \bot$。为了证明 $(g \circ \langle f_i^\dagger, id\rangle)^{j+1}(\bot) \leq h_1|_{X\setminus Z}.\text{流}$，我们使用以下蕴含关系：

$$cval \leq h_1|_{X\setminus Z}.flow \quad \text{implies} \quad (g \circ \langle f_i^\dagger, id\rangle)(cval) \leq h_1|_{X\setminus Z}.flow\ .$$

对于 $g \circ \langle f_1^\dagger, id\rangle$，我们使用 $g \circ \langle f_1^\dagger, id\rangle$ 加上 $cval \leq h_1|_{X\setminus Z}.\text{流}$ 以及根据方程（8），$h_1|_{X\setminus Z}.\text{流}$ 是 $g \circ \langle f_1^\dagger, id\rangle$ 的不动点：

$$(g \circ \langle f_1^\dagger, id\rangle)(cval) \leq (g \circ \langle f_1^\dagger, id\rangle)(h_1|_{X\setminus Z}.flow) = h_1|_{X\setminus Z}.flow\ .$$

对于 $g \circ \langle f_2^\dagger, id\rangle$，我们采用类似的方法，但补充上文的观察：由方程（9）结合方程（7）可得 $(g \circ \langle f_2^\dagger, id\rangle)(h_1|_{X\setminus Z}.\text{流}) = (g \circ \langle f_1^\dagger, id\rangle)(h_1|_{X\setminus Z}.\text{流})$：

$$\begin{aligned}(g \circ \langle f_2^\dagger, id\rangle)(cval) &\leq (g \circ \langle f_2^\dagger, id\rangle)(h_1|_{X\setminus Z}.flow)\\ &= (g \circ \langle f_1^\dagger, id\rangle)(h_1|_{X\setminus Z}.flow) = h_1|_{X\setminus Z}.flow\ .\end{aligned}$$

□

定理2 的证明。令 $F \triangleq \text{lfp.ext}_{h_1,h_2} \neq \top$，即 $F \subseteq X$。我们首先证明 $h_1|_F =_{ctx} h_2|_F$。根据限制的定义，节点集合相同。根据 $\text{odif}_{h_1,h_2} \subseteq F \subseteq X$、$\text{tfail}_{h_1,h_2}(F) = \varnothing$ 以及引理 6，流入相同。我们将此流入记为 $\text{in}_F \triangleq h_1|_F.\text{in} = h_2|_F.\text{in}$。根据 $\text{tfail}_{h_1,h_2}(F) = \varnothing$，传递函数相同。这意味着，对于所有 $x \in \mathbb{N}\setminus F$、每个函数 $\text{in} \leq \text{in}_F$，以及所有 $z \in F$，我们都有 $[\text{tf}(h_1|_F)(\text{in})](z, x) = [\text{tf}(h_2|_F)(\text{in})](z, x)$。然而，这正是 $\text{tf}(h_1|_F) =_{in_F} \text{tf}(h_2|_F)$ 的定义。

剩下要证明 $h_1|_{X\setminus F} = h_2|_{X\setminus F}$。根据限制的定义，节点集合再次相同。由于 $\text{odif}_{h_1,h_2} \subseteq F$，因此边集合相同，所以 $X \setminus F \subseteq X \setminus \text{odif}_{h_1,h_2}$。反过来，后者是边相同的最大节点集合。准确地说，对于每个 $x \in X \setminus \text{odif}_{h_1,h_2}$ 以及每个 $z \in \mathbb{N}$，都有 $h_1.E(x, z) = h_2.E(x, z)$。然而，这表示边函数相等，即 $h_1.E = h_2.E$。若将边函数限制在 $F \times \mathbb{N}$上，这种相等关系仍然成立。对于来自 X 之外节点的流入，根据 $h_1.\text{in} = h_2.\text{in}$，二者相等。对于$i = 1, 2$，考虑 $h_i|_{X\setminus F}.\text{in}(z, x)$，其中 $z \in F$。根据限制的定义，我们有

$$h_i|_{X\setminus F}.in(z, x) = h_i.E_{(z,x)}(h_i.flow(z))\ . \tag{10}$$

我们有 $h_i.\text{流}(z) = h_i|_F.\text{流}(z)$，由引理 4(i) 得出。根据引理 5(ii)，我们有

$$h_i|_F.flow(z) = lfp.h_i|_F[h_i|_F.in]\ . \tag{11}$$

利用引理 5(i)，我们得到

$$h_i.E_{(z,x)}(lfp.h_i|_F[h_i|_F.in]) = [tf(h_i|_F)(h_i|_F.in)](z, x)\ . \tag{12}$$

总而言之，这些方程得出

$$\begin{aligned}h_i|_{X\setminus F}.in(z, x) &\overset{(10)}{=} h_i.E_{(z,x)}(h_i.flow(z))\\ &\overset{(11)}{=} h_i.E_{(z,x)}(lfp.h_i|_F[h_i|_F.in])\\ &\overset{(12)}{=} [tf(h_i|_F)(h_i|_F.in)](z, x)\ .\end{aligned} \tag{13}$$

我们在上文证明了$h_1|_F =_{ctx} h_2|_F$ 成立，因此还可得到$h_1|_F.in = h_2|_F.in$以及$tf(h_1|_F) =_{h_1|_F.in} tf(h_2|_F)$。结合公式（13），即可得到所需的$h_1|_{X\setminus F}.in(z,x) = h_2|_{X\setminus F}.in(z,x)$。□

引理 7 的证明。我们首先证明从左到右的蕴含关系。考虑节点$x, y \in X$和 $z \in \mathbb{N}\setminus X$以及一条路径 $p \in$路径集(h_1, x, y, z)。如果存在一个适合替换的路径集，那么完整的集合 $P =$路径集(h_2, x, y, z)即可满足要求。为证明$E_p \le E_P$，考虑一个值$m \in \mathbb{M}$。令in^m 为将 x 映射到m、并将其余节点映射到 0 的流。对于所需不等式的左侧，我们有

$$E_p(m) \le \sum_{x\in X} \sum_{p\in Paths(h_1,x,y,z)} E_p(in^m(x)) = [tf(h_1)(in^m)](y,z)\,.$$

后一个等式使用了定理 4。对于所需不等式的右端，我们有

$$\begin{aligned} E_P(m) &= \sum_{p\in Paths(h_2,x,y,z)} E_p(in^m(x)) + \sum_{x'\in X\setminus\{x\}} \sum_{p\in Paths(h_2,x',y,z)} E_p(0) \\ &= \sum_{x\in X} \sum_{p\in Paths(h_2,x,y,z)} E_p(in^m(x)) = [tf(h_2)(in^m)](y,z)\,. \end{aligned}$$

第一个等式是 P 的定义，并且 $f(0) = 0$ 对所有边函数成立。第二个等式重新排列了求和，并使用了 in^m 的定义。最后一个等式再次使用了定理 4。由于 $[tf(h_1)(in^m)](y,z) = [tf(h_2)(in^m)](y,z)$根据假设成立，因此我们可以得出 $E_P(m) \le E_P(m)$，正如所要求的。将 h_2进行路径替换为h_1 的论证是对称的。

假设 h_1 被h_2 完全路径替换。我们证明 $tf(h_1) \le tf(h_2)$。结论由此得出。考虑一个流入in、一个节点$y \in X$和一个节点$z \in \mathbb{N}\setminus X$。根据定理 4，我们有

$$[tf(h_1)(in)](y,z) = \sum_{x\in X} \sum_{p\in Paths(h_1,x,y,z)} E_p(in_x)\,.$$

考虑一个节点 x 和一个路径 $p \in$路径集(h_1, x, y, z)。我们证明以下性质成立：$E_P(in_x) \le [tf(h_2)(in)](y, z)$。于是，根据幂等性可得不等式 $[tf(h_1)(in)](y, z) \le [tf(h_1)(in)](y, z)$，证毕。

根据路径替换，存在一个路径集$P \subseteq$路径集(h_2, x, y, z)，据此可以推导出：

$$\begin{aligned} E_p(in_x) \le E_P(in_x) &= \sum_{q\in P} E_q(in_x) \\ &\le \sum_{q\in Paths(h_2,x,y,z)} E_q(in_x) + \sum_{x'\in X\setminus\{x\}} \sum_{q\in Paths(h_2,x',y,z)} E_q(in_{x'}) \\ &= [tf(h_2)(in)](y,z)\,. \end{aligned}$$

第一个不等式来自路径替换。接下来的等式是E_P对路径集的定义。下一个不等式加入了更多路径。最后一个等式是定理 4。

□

引理 8 的证明。完全路径替换意味着路径替换。反过来，我们首先考虑节点 $x \in odif_{\#}(h_1, h_2)$、$y \in X$、$z \in \mathbb{N}\setminus X$，以及

一条路径 $p \in$ 路径集($h_1, x,\ y,\ z$)。根据路径替换的假设，存在一个路径集 $P \subseteq$ 路径集($h_2, x,\ y,\ z$)，满足 $E_p \leq E_P$。

对于一个节点 x $\in$ X $\setminus$ odif #(h_1,h_2) 和一条路径 $p \in$ Paths(h_1, x , y, z)，有两种情况：要么该路径不会经过 odif #(h_1,h_2) 中的节点，要么会经过。前一种情况下，这条路径仍存在于 h_2 中，因此我们可以用它本身替换它，并选择 P $= \{ p \}$。后一种情况下，我们将路径分解为 $p = p_1.p_2$，使得 last(p_1) = first(p_2) 是该路径首次经过 odif #(h_1,h_2) 的位置。对于 p_2，路径替换会得到一个路径集合 Q $\subseteq$ Paths(h_2, first(p_2), y, z)，且 $E_{p_2} \leq E_Q$。现在 p_1.Q $= \{ p_1.q \mid q \in$ Q $\}$ 是 h_2 中的一个路径集合。

我们按如下方式展示 $E_{p_1.Q} \geq E_P$：

$$\begin{aligned} E_{p_1.Q} &= \sum_{q \in Q} E_{p_1.q} = \sum_{q \in Q} (E_q \circ E_{p_1}) = (\sum_{q \in Q} E_q) \circ E_{p_1} \\ &= E_Q \circ E_{p_1} \geq E_{p_2} \circ E_{p_1} = E_p \ . \end{aligned}$$

第一个等式是将边函数推广到路径集并结合 $p_1.Q$的定义。第二个等式是路径的边函数定义。第三个等式利用了函数之和按如下方式求值这一事实：($\sum_{q \in Q} E_q$)(x) = $\sum_{q \in Q} E_q(x)$。第四个等式再次使用了路径集的边函数定义。不等式源于 $E_{p_2} \leq E_Q$。最后一个等式再次使用了路径的边函数定义以及路径 p的分解。□