arXiv:2304.14539v1 [cs.PL] 27 Apr 2023

关于编排式程序的推理

Luís Cruz-Filipe, Eva Graversen、Fabrizio Montesi 和 MarcoPeressotti

南丹麦大学数学与计算机科学系

摘要。编排式编程是一种自顶向下开发并发或分布式系统的编程范式。此类程序称为编排，详细描述进程之间期望发生的交互，并可编译为基于消息传递的分布式实现。编舞语言通常保证无死锁性，并提供编排与其编译后实现之间的操作对应，但迄今为止，关于验证其他性质的研究还很少。

本文提出一种用于推理编排行为的霍尔风格逻辑，并通过典型示例展示其用法。我们证明该逻辑是可靠且完备的，并讨论其判断的可判定性。借助编排式编程领域已有的结果，我们证明，为某个编排证明的任何函数正确性属性，在其编译后的实现中也成立。

1 引言

编写通信系统的程序很困难，因为必须确保由相互独立的程序执行的各个通信动作（例如发送或接收消息）在运行时能够正确匹配 [21]。

在编排式编程范式中 [26], ，这一挑战通过提供高级抽象来应对，使程序员能够从“全局”视角安全地表达所需的通信流程 [6,8,9,13,17,18,20,23,27]。在编排程序（即编排）中，通信通过安全协议表示法中的通信项的某种变体来表达：Alice -> Bob: M，其含义是“Alice 向 Bob 通信消息 M” [29]。这些项可以使用常见的编程语言构造组合成结构化编排。然后，编译器可以自动生成可执行分布式实现 [6, 13, 16], ，如图 1 所示。

迄今为止，关于编排式编程的研究主要聚焦于提升编排式编程语言的表达能力、语言的实现，以及对编译一般性质的形式化。编排式编程理论通常会为其配套的编译过程提供正确性证明。一项标志性成果是设计上保证无死锁性：由于编排中无法通过语法表达不匹配的通信动作，因此编译后的代码不会引发死锁 [6]。

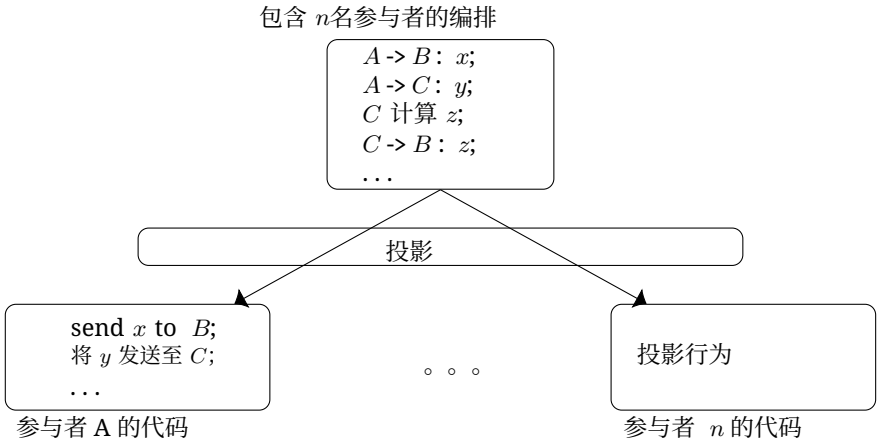


图 1. 编排式编程：系统的通信和计算行为在编排中定义，随后被投影（编译）为无死锁的分布式代码（改编自 [17]）。

相比之下，关于如何证明编排的函数正确性的一般方法，相关研究还很少。然而，编排描述了分布式协议，而推理这些协议对参与者状态的影响通常十分重要。

本研究。本文提出一种用于推理编排的霍尔逻辑。霍尔逻辑 [2,19] 是一种常见的程序推理方法。霍尔断言是一个三元组 $\{\varphi\}P\{\psi\}$，其中 φ 和 ψ 是公式（分别称为前置条件和后置条件），而 P 是程序。该三元组表明，如果 P 从满足 φ 的状态执行并终止，那么最终状态将满足 ψ。我们构建了一种霍尔逻辑，其中程序是编排，公式可以联合描述多个进程的状态。

我们的框架建立在编排式编程 [10,27], 的成熟理论之上，尤其依托于已在 Coq [11,12] 中形式化的性质。这有助于使我们的工作具有普适性和优雅性。例如，我们在元理论证明中利用合流性，并依靠先前证明的编译器正确性结果，将通过我们的逻辑证明的性质迁移到由编排编译而来的分布式实现中。

贡献。我们为以标准方式表达的编排程序定义了一种霍尔逻辑，用于进行推理。该逻辑采用模块化设计，并以状态公式的语言作为参数。我们证明，该逻辑具备霍尔逻辑应有的性质（可靠性和部分完备性），并展示如何使用它来证明以编排形式编码的特定协议的重要性质。

结构。我们回顾第2节中来自 [10] 的编舞语言。在第3节中，我们介绍我们的逻辑并证明其可靠性。第4节介绍最弱自由前置条件，并利用它们展示完备性和可判定性结果。第5节讨论其他相关工作。全文还穿插了说明性示例。

2 语言

本节回顾 [10], 中的编排语言，我们将对其进行推理。该语言用于建模独立进程系统（网络），这些系统通过同步通信进行交互。每个进程都有一个唯一的名称，网络中的所有其他进程都知道该名称；进程还可以将值存储在本地内存中，并通过变量引用这些值。假设所有进程的变量名集合相同。所有进程的集合记为 $\mathcal{P}$。

可以交换两种类型的消息：值是对表达式进行本地求值的结果；选择标签是特殊常量，用于实现对分布式行为备选方案的选择达成一致。

表达式和标签的具体集合不作规定，但我们作出一些假设。标签取自一个（较小的）有限集合。表达式由（类型化的）签名 Ξ 和进程变量集合自由生成。求值结果为布尔值的表达式也称为布尔表达式。

2.1 语法

形式上，编排的语法由以下文法定义

$$C ::= I; C \mid \mathsf{if}\ \mathsf{p}.b\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2 \mid X \mid \lceil \vec{\mathsf{q}}, X \rfloor C \mid \mathbf{0}$$
$$I ::= \mathsf{p}.x := e \mid \mathsf{p}.e \to \mathsf{q}.x \mid \mathsf{p} \to \mathsf{q}[\mathrm{L}]$$

其中， C 是编排， I 是指令，p 和 q 是进程名称， e是表达式， v 是值， x 是变量， b 是布尔表达式， L 是选择标签， X 是过程名称。

编排可以通过以下方式构建：一条指令 I 后接一个编排；两个编排 C_1 和 C_2的选择组合；过程调用；或者已终止的编排 $\mathbf{0}$。过程调用有两种形式，分别对应于：(a) 尚未被任何进程进入的过程（X），或 (b) 已经启动的过程，并附有仍需进入该过程的进程集合（$\lceil \neg \mathsf{q}\ X \rfloor C$）。

指令有三种类型：局部赋值（$\mathsf{p}.x := e$），其中 p 计算表达式 e 并将结果存储在其局部变量 x中；值通信，其中 p 计算 e 并将结果发送给 q，q 将其存储在变量x中；以及标签选择，其中 p 向 q 发送标签 L （通常用于传达局部选择的结果——见下文）。

在条件语句中，如果 $\mathsf{p}.b\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2$，进程 p 会对表达式 b 求值，以决定编排应继续为 C_1 还是 C_2。由于只有 p

知道求值结果，其余进程需要获知应如何行动——通常通过标签选择将这一信息传递给其他参与者。[1]

该语言中的重复和迭代行为通过过程调用实现。调用过程 X 只需调用与 X 对应的编排，该编排定义在单独的过程定义映射 $\mathscr{C}$ 中。由于编排执行是分布式的，进程进入过程时无需同步。因此，需要一个运行时术语 $\lceil \vec{\mathsf{q}}\ X \rfloor C$，用来表示只有部分进程已进入的过程调用。此术语同时记录仍需进入 X 的进程集合 $\vec{\mathsf{q}}$，以及编排的执行状态 C。如下文所示，编排语义允许乱序执行，因此，部分进程可能会在其他进程进入过程之前，就开始执行该过程中的相应部分。

示例 1（Diffie-Hellman）。考虑 Diffie-Hellman 密钥交换协议 [14]，它允许双方 p 和 q 建立共享密钥 s，之后可将其用于对称加密。要在我们的编舞语言中实现该协议，只需要通信、本地计算，以及支持模幂运算（$b^e \bmod m$）[16, 27] 的表达式语言。该协议假设参与者各自拥有一个私钥（a、b），并且共享一个素数 m 和一个模 m 的原根 g。

$DH = \mathsf{p}.(g^a \text{ 取模 } m) \to \mathsf{q}.a$；$\mathsf{p}$ 计算自己的公钥并发送给 q $\mathsf{q}.(g^b \text{ 取模 } m) \to \mathsf{p}.b$；$\mathsf{q}$ 计算自己的公钥并发送给 p $\mathsf{p}.s := b^a \text{ 取模 } m$；$\mathsf{p}$ 生成共享密钥 $\mathsf{q}.s := a^b \text{ 取模 } m$；$\mathsf{q}$ 生成共享密钥 0

◁

示例 2（Zeros）。寻找函数的零点是使用霍尔风格逻辑进行程序验证的常见教科书示例 [3]。在此示例中，我们考虑问题的一个变体：p 和 q 协调合作，在自然数上的函数 f 中寻找零点：p 负责选择待测试的值，q 则负责求值 f 并决定停止还是继续搜索。我们用以下递归过程来描述这一迭代协议。

$$\begin{aligned}\mathscr{C}(Z) = {} & \mathsf{p}.x \to \mathsf{q}.x; \\ & \text{if } \mathsf{q}.f(x) = 0 \text{ then } (\mathsf{q} \to \mathsf{p}[\mathrm{L}]; \mathbf{0}) \\ & \qquad \text{else } (\mathsf{q} \to \mathsf{p}[\mathrm{R}]; \mathsf{p}.x := 1 + x; Z)\end{aligned}$$

然后，要搜索 f 的定义域，我们运行编排 $\mathsf{p}.x := 0;\ Z$。 ◁

我们定义函数 pn，用于返回参与某条指令或某个编排的进程集合。该函数通过对自然数上的归纳法定义

[1] 因此，标签集合通常固定为一个含两个元素的集合，分别对应选择的两个分支。

way.

$$\mathsf{pn}(\mathsf{p}.x := e) = \{\mathsf{p}\} \qquad \mathsf{pn}(\mathsf{p}.e \to \mathsf{q}.x) = \mathsf{pn}(\mathsf{p} \to \mathsf{q}[\mathrm{L}]) = \{\mathsf{p}, \mathsf{q}\}$$
$$\mathsf{pn}(I; C) = \mathsf{pn}(I) \cup \mathsf{pn}(C) \qquad \mathsf{pn}(\mathsf{if}\ \mathsf{p}.b\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2) = \{\mathsf{p}\} \cup \mathsf{pn}(C_1) \cup \mathsf{pn}(C_2)$$
$$\mathsf{pn}(X) = \mathcal{P} \qquad \mathsf{pn}(\lceil \vec{\mathsf{q}}, X \rfloor C) = \vec{\mathsf{q}} \cup \mathsf{pn}(C)$$

为简单起见，我们假设所有进程都参与所有过程；另一种做法是在过程名称上标注其所用的进程集合，参见 [12]。这不会影响实际参与该过程的任何进程的行为；从语义上说，这只意味着某个原本会被视为已终止的进程，可能必须先进入若干个空的过程调用。

2.2 语义

编排语义使用状态这一概念，它将每个进程上的每个变量映射到该变量当前存储的值。我们可以方便地将局部状态定义为从变量到值的映射（表示某个进程上的内存状态），并将全局状态定义为函数 Σ，使得 $\Sigma(\mathsf{p})$ 是 p 处的局部状态。

为了对表达式求值，我们假设存在一个求值函数，它以局部状态为参数，根据状态将变量求值为其对应的值，并以同态方式继续求值。换句话说，求值会将 Ξ 中的每个符号映射为一个从值到值的函数。我们假设所有编排和函数都是类型正确的，也就是说，每个变量中存储的值都与其所在表达式所要求的类型相匹配。此外，我们假设求值总会终止，并用 $e \downarrow_{\Sigma(\mathsf{p})} v$ 表示：根据状态 $\Sigma(\mathsf{p})$（p 处的局部状态），e 求值为 v。

编排的形式语义通过标记转换系统定义，该系统捕捉了上述直觉，其规则见图 2。迁移由转换标签标记，这些标签抽象了可观察到的编排动作：值的通信（$\mathsf{p}.v \to \mathsf{q}$）和标签的通信（$\mathsf{p} \to \mathsf{q}[\mathrm{L}]$），或内部动作（$\tau@\mathsf{p}$）。函数 pn 自然地扩展到这些情况。

$$\mathsf{pn}(\tau@\mathsf{p}) = \{\mathsf{p}\} \qquad \mathsf{pn}(\mathsf{p}.v \to \mathsf{q}) = \mathsf{pn}(\mathsf{p} \to \mathsf{q}[\mathrm{L}]) = \{\mathsf{p}, \mathsf{q}\}$$

规则 C|Assign、C|Com、C|Sel、C|Then 和 C|Else 体现了前文所述不同编排原语背后的直觉。接下来的三条规则涉及过程调用：当一个进程决定进入过程时，该过程便启动，所有其余进程都被加入“等待列表”（规则 C|Call）；每当有新进程进入过程时，它就会从等待进程集合中移除（规则 C|Enter）；当最后一个进程进入调用时，该集合便被移除（规则 C|Finish）。

最后三条规则涉及乱序执行：无论其他进程在做什么，各进程始终可以执行对自身而言的下一步操作

$$\frac{e \downarrow_{\Sigma(\mathsf{p})} v}{\langle \mathsf{p}.x \coloneqq e; C, \Sigma\rangle \xrightarrow{\tau@\mathsf{p}}_{\mathscr{C}} \langle C, \Sigma[\langle \mathsf{p}, x\rangle \mapsto v]\rangle}\ \textsc{C|Assign}$$

$$\frac{e \downarrow_{\Sigma(\mathsf{p})} v}{\langle \mathsf{p}.e \to \mathsf{q}.x; C, \Sigma\rangle \xrightarrow{\mathsf{p}.v\to\mathsf{q}}_{\mathscr{C}} \langle C, \Sigma[\langle \mathsf{q}, x\rangle \mapsto v]\rangle}\ \textsc{C|Com}$$

$$\frac{}{\langle \mathsf{p} \to \mathsf{q}[\textsc{l}]; C, \Sigma\rangle \xrightarrow{\mathsf{p}\to\mathsf{q}[\textsc{l}]}_{\mathscr{C}} \langle C, \Sigma\rangle}\ \textsc{C|Sel} \qquad \frac{b \downarrow_{\Sigma(\mathsf{p})} \mathsf{true}}{\langle \mathsf{if}\ \mathsf{p}.b\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2, \Sigma\rangle \xrightarrow{\tau@\mathsf{p}}_{\mathscr{C}} \langle C_1, \Sigma\rangle}\ \textsc{C|Then}$$

$$\frac{b \downarrow_{\Sigma(\mathsf{p})} \mathsf{false}}{\langle \mathsf{if}\ \mathsf{p}.b\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2, \Sigma\rangle \xrightarrow{\tau@\mathsf{p}}_{\mathscr{C}} \langle C_2, \Sigma\rangle}\ \textsc{C|Else}$$

$$\frac{\mathscr{C}(X) = C}{\langle X, \Sigma\rangle \xrightarrow{\tau@\mathsf{r}}_{\mathscr{C}} \langle \lceil \mathsf{pn}(C) \setminus \mathsf{r}, X\rfloor C, \Sigma\rangle}\ \textsc{C|Call}$$

$$\frac{\mathsf{r} \in \vec{\mathsf{q}} \quad \vec{\mathsf{q}} \setminus \mathsf{r} \neq \emptyset}{\langle \lceil \vec{\mathsf{q}}, X\rfloor C, \Sigma\rangle \xrightarrow{\tau@\mathsf{r}}_{\mathscr{C}} \langle \lceil \vec{\mathsf{q}} \setminus \mathsf{r}, X\rfloor C, \Sigma\rangle}\ \textsc{C|Enter} \qquad \frac{}{\langle \lceil \mathsf{q}, X\rfloor C, \Sigma\rangle \xrightarrow{\tau@\mathsf{q}}_{\mathscr{C}} \langle C, \Sigma\rangle}\ \textsc{C|Finish}$$

$$\frac{\langle C, \Sigma\rangle \xrightarrow{\mu}_{\mathscr{C}} \langle C', \Sigma'\rangle \quad \mathsf{pn}(I) \# \mathsf{pn}(\mu)}{\langle I; C, \Sigma\rangle \xrightarrow{\mu}_{\mathscr{C}} \langle I; C', \Sigma'\rangle}\ \textsc{C|DelayI}$$

$$\frac{\langle C_1, \Sigma\rangle \xrightarrow{\mu}_{\mathscr{C}} \langle C_1', \Sigma'\rangle \quad \langle C_2, \Sigma\rangle \xrightarrow{\mu}_{\mathscr{C}} \langle C_2', \Sigma'\rangle \quad \mathsf{p} \notin \mathsf{pn}(\mu)}{\langle \mathsf{if}\ \mathsf{p}.b\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2, \Sigma\rangle \xrightarrow{\mu}_{\mathscr{C}} \langle \mathsf{if}\ \mathsf{p}.b\ \mathsf{then}\ C_1'\ \mathsf{else}\ C_2', \Sigma'\rangle}\ \textsc{C|DelayC}$$

$$\frac{\langle C, \Sigma\rangle \xrightarrow{\mu}_{\mathscr{C}} \langle C', \Sigma'\rangle \quad \vec{\mathsf{q}} \# \mathsf{pn}(\mu)}{\langle \lceil \vec{\mathsf{q}}, X\rfloor C, \Sigma\rangle \xrightarrow{\mu}_{\mathscr{C}} \langle \lceil \vec{\mathsf{q}}, X\rfloor C', \Sigma'\rangle}\ \textsc{C|DelayP}$$

图 2. 语义

。规则 C|DelayI、 C|DelayC 和 C|DelayP 对此进行建模，分别允许执行语法上并非第一条指令、条件语句或过程入口的操作。这些规则中的附加条件规定，参与所执行操作的进程不会参与被跳过的操作（对于 $X \cap Y = \emptyset$，我们记作 $X \# Y$）。此外，在 C|DelayC 中执行的操作必须是不论 p 作何选择都能执行的操作。

转换的自反传递闭包记作 $\to^*_{\mathscr{C}}$；由于转换标签的序列与当前论述无关，因此我们将其省略。

对于我们的证明，我们还需要头部转移这一概念，它是由图 2 中前 8 条规则定义的转移关系——也就是说，不允许

乱序执行。我们用 $\langle C、\ \Sigma\rangle \overset{\mu}{\Rightarrow}_{\mathscr{C}} \langle C'、\ \Sigma'\rangle$ 表示 C 向 C' 进行头部转移，并用 $\Rightarrow^*_{\mathscr{C}}$ 表示该关系的自反传递闭包。

3 编排的霍尔演算

本节介绍一种基于霍尔逻辑、用于证明编排语义性质的形式演算。我们的判断采用三元组 $\{\varphi\}C\{\psi\}$，其含义是：“如果从满足公式 φ 的状态开始执行编排 C，且执行终止，那么最终状态满足公式 ψ”。

本节将形式化定义该演算的语法和语义，首先介绍状态逻辑——公式 φ 和 ψ 所使用的语言。

3.1 状态逻辑

霍尔演算中的状态逻辑通常将性质表达为“变量 x 存储着值 v”，这类性质很容易用等式逻辑表达。我们遵循这一传统，将状态逻辑定义为等式逻辑的扩展。为了处理赋值，我们需要能够以与规则 C|Assign 中状态更新相对应的方式更新公式，但又不计算具体的值。可以通过将原公式中传递的表达式代入来实现这一点；但这意味着表达式可能会突然引用存储在不同进程中的变量，因此不再能在本地求值。

为处理这些问题，我们的状态逻辑以一组表达式为参数；这些表达式由相同的签名 Ξ 自由生成，但使用局部化变量 $\mathbf{p}.x$。我们将这些表达式记作 $\mathcal{E}$，并以自然的方式将求值扩展到它们。

状态公式定义如下：

$$\varphi, \psi ::= (\mathcal{E} = \mathcal{X}) \mid \delta \mid \varphi \wedge \varphi \mid \neg\varphi$$

其中 $\mathcal{X}$ 是一个（逻辑）变量，$\delta \in \mathfrak{D}$，其中 $\mathfrak{D}$ 是一种可判定理论，其项包含逻辑变量。以 $\mathfrak{D}$ 为参数化语言，可使公式的语法更简单，同时让用户能够灵活地定义所需的其他公式。这与我们处理局部语言的方式类似。例如，如果 $\mathfrak{D}$ 包含 $\mathcal{X} > \mathcal{X}'$，那么状态逻辑就能表达诸如 $\mathbf{p}.x > \mathbf{q}.y$ 这样的约束，假设值是整数：这可以写作 $\mathbf{p}.x = \mathcal{X} \wedge \mathbf{q}.y = \mathcal{Y} \wedge \mathcal{X} > \mathcal{Y}$。析取和蕴含按通常方式定义为缩写。

给定一个状态 Σ、一个公式 φ 以及一个从逻辑变量到值的赋值 ρ，我们根据以下规则定义 $\Sigma \Vdash_\rho \varphi$，读作“$\Sigma$ 在 ρ下满足 φ”

$$\frac{\mathcal{E} \downarrow_\Sigma \rho(\mathcal{X})}{\Sigma \Vdash_\rho \mathcal{E} = \mathcal{X}} \qquad \frac{\delta \in \mathfrak{D} \quad \varphi \text{ is true}}{\Sigma \Vdash_\rho \delta}$$

并结合逻辑联结词的常规规则。

与霍尔逻辑中的惯例一样，赋值通过替换来处理——例如，我们希望能够证明类似这样的内容。

$$\overline{\{\varphi'\}\mathsf{p}.x := e; \mathbf{0}\{\varphi\}}$$

其中，φ' 由 φ 通过将 $\mathsf{p}.x$ 替换为 e得到。然而，简单地将 $\mathsf{p}.x$ 的每次出现都替换为 e，通常会得到一个无效公式（这是由于编排和状态公式中的变量不同）。我们将 e 在 p 处的局部化记为 $L(\mathsf{p}\ e)$，它是通过将 e 中的每个（编排）变量 x 替换为 $\mathsf{p}.x$得到的（逻辑）表达式；并将局部化替换 $\mathcal{E}[\mathsf{q}.x := \mathsf{p}.e]$定义为通过将 $\mathcal{E}$ 中每次出现的 $\mathsf{q}.x$ 替换为$L(\mathsf{p}\ e)$得到的表达式。（通信规则对 p 和 q 使用不同的值。）注意，这两种操作都可以通过对表达式进行结构递归来定义。局部化替换可自然地扩展到公式。

例3. 令 φ 为公式 $\mathsf{p}.x > 3$，令 e 为表达式 $y - z$。在 φ 中用 $y - z$ 替换 $\mathsf{p}.x$ 会得到格式错误的公式 $\mathsf{p}.(y - z)\ >3$。相反，用 $L(\mathsf{p}\ y - z) = \mathsf{p}.y - \mathsf{p}.z$ 替换 $\mathsf{p}.x$ 会得到正确的公式 $\mathsf{p}.y - \mathsf{p}.z > 3$，上述判断则变为

$$\overline{\{\mathsf{p}.y - \mathsf{p}.z > 3\}\mathsf{p}.x := y - z; \mathbf{0}\{\mathsf{p}.x > 3\}}$$

其语法形式良好。 ◁

现在我们证明，已局部化到 p 的表达式在 p 中的解释与其原始求值相同。

引理 1. 设 Σ 是一个状态，v 是一个值，$\mathcal{X}$ 是一个逻辑变量，且 ρ 满足 $\rho(\mathcal{X}) = v$。对于任意进程 p 和表达式 e，$e \downarrow_{\Sigma(\mathsf{p})} v$ 当且仅当 $\Sigma \Vdash_\rho L(\mathsf{p}\ e) = \mathcal{X}$。

证明。由对 e 的结构归纳法可得。 □

然后我们证明，在公式中进行局部化替换，等价于改变环境中该变量的值。

推论 1. 设 Σ 是一个状态，p 是一个进程，e 是一个表达式，v 是一个值，且 $e \downarrow_{\Sigma(\mathsf{p})} v$。对于任意公式 φ 和赋值 ρ，$\Sigma[\langle \mathsf{p}\ x\rangle \mapsto v] \Vdash_\rho \varphi$ 当且仅当 $\Sigma \Vdash_\rho \varphi[\mathsf{q}.x :\ =\mathsf{p}.e]$。

证明。对 φ 进行结构归纳。一个基例就是引理 1，另一个基例则显然为空（因为 $\mathfrak{D}$ 中的公式不受替换影响）。两个归纳步骤可直接由归纳假设得出。□

$$\frac{}{\vdash_{\mathfrak{C}} \{\varphi\}\mathbf{0}\{\varphi\}}\ \text{H|Nil} \qquad \frac{\vdash_{\mathfrak{C}} \{\varphi\}C\{\varphi'\}}{\vdash_{\mathfrak{C}} \{\varphi[\mathsf{p}.x := \mathsf{p}.e]\}\mathsf{p}.x := e; C\{\varphi'\}}\ \text{H|Assign}$$

$$\frac{\vdash_{\mathfrak{C}} \{\varphi\}C\{\varphi'\}}{\vdash_{\mathfrak{C}} \{\varphi[\mathsf{q}.x := \mathsf{p}.e]\}\mathsf{p}.e \to \mathsf{q}.x; C\{\varphi'\}}\ \text{H|Com} \qquad \frac{\vdash_{\mathfrak{C}} \{\varphi\}C\{\varphi'\}}{\vdash_{\mathfrak{C}} \{\varphi\}\mathsf{p} \to \mathsf{q}[\mathrm{L}]; C\{\varphi'\}}\ \text{H|Sel}$$

$$\frac{\vdash_{\mathfrak{C}} \{\varphi \wedge L(\mathsf{p}, b) \overset{\mathcal{X}}{=} \mathsf{true}\}C_1\{\psi\} \quad \vdash_{\mathfrak{C}} \{\varphi \wedge L(\mathsf{p}, b) \overset{\mathcal{X}}{=} \mathsf{false}\}C_2\{\psi\} \quad \mathcal{X} \text{ fresh}}{\vdash_{\mathfrak{C}} \{\varphi\}\mathsf{if}\ \mathsf{p}.b\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2\{\psi\}}\ \text{H|Cond}$$

$$\frac{\mathfrak{C}(X) = \langle \varphi, \psi \rangle}{\vdash_{\mathfrak{C}} \{\varphi\}X\{\psi\}}\ \text{H|Call} \qquad \frac{\vdash_{\mathfrak{C}} \{\varphi\}C\{\psi\}}{\vdash_{\mathfrak{C}} \{\varphi\}\lceil \vec{\mathsf{q}}, X \rfloor C\{\psi\}}\ \text{H|Call'}$$

$$\frac{\mathfrak{D} \models \varphi \to \varphi' \quad \vdash_{\mathfrak{C}} \{\varphi'\}C\{\psi'\} \quad \mathfrak{D} \models \psi' \to \psi}{\vdash_{\mathfrak{C}} \{\varphi\}C\{\psi\}}\ \text{H|Weak}$$

图 3. 推理规则

3.2 Hoare 逻辑

现在我们可以介绍演算的规则了，这些规则如图 3 所示。为了处理过程定义，我们需要了解它们对状态的影响。为此，我们引入过程规范映射 $\mathfrak{C}$，它将每个过程名称映射到一对 $\langle \varphi、\ \psi \rangle$，其预期含义是判断 $\{\varphi\}C\{\psi\}$ 应当成立，其中 C 是 X 的定义。

赋值规则 H|Assign 前文已经给出了动机，其形式与命令式程序的标准霍尔演算中的规则类似；同样，规则 H|Nil 和 H|Cond 也都是标准规则。规则 H|Cond 中的记号 $L(\mathsf{p}\ b) \overset{\mathcal{X}}{=} \mathsf{true}$ 是合取 $L(\mathsf{p}\ b) = \mathcal{X} \wedge \mathcal{X} = \mathsf{true}$ 的缩写。

规则 H|Weak 是弱化规则，它允许我们在状态逻辑中进行推理。记号 $\mathfrak{D} \models \varphi$ 表示“φ 是一个有效公式”。

规则 H|Com 和 H|Sel 将这些规则背后的直觉调整为适用于我们的舞谱动作——通信本质上是对位于另一个进程中的变量进行赋值，而选择不会影响状态。

规则 H|Call 通过从规范映射中读取相应的判断来处理未展开的过程调用，而 H|Call' 则反映了这样一个事实：展开后的过程的当前状态是明确给出的，并且进程进入过程不会影响状态。

只有当规范映射与过程定义在以下意义上保持一致时，这些规则才有意义。

定义 1. 过程规范映射 $\mathfrak{C}$ 与过程定义集合 $\mathscr{C}$ 一致，当且仅当对于每个 X，$\vdash_{\mathfrak{C}} \{\mathsf{fst}(\mathfrak{C}(X))\}\mathscr{C}(X)\{\mathsf{snd}(\mathfrak{C}(X))\}$，其中 fst 和 snd 是有序对的标准投影算子。

这一概念的作用类似于带有 while 循环的语言的霍尔逻辑中更常见的“是一个循环不变式”这一概念，它表明每当调用 X 时，fst($\mathfrak{C}(X)$) 总是成立。

例 4（Diffie-Hellman，函数正确性）。考虑例 1，并假设 $\mathfrak{D}$ 是一个用于判定带模幂运算的算术表达式相等性的理论。Diffie-Hellman 协议的函数正确性表明，如果 p 和 q 具有相同的模数 m 和底数 g，那么协议终止后，它们将共享相同的秘密 s。以下状态公式 $\varphi = (\mathsf{p}.g \overset{\mathcal{G}}{=} \mathsf{q}.g \wedge \mathsf{p}.m \overset{\mathcal{M}}{=} \mathsf{q}.m)$ 和 $\psi = \mathsf{p}.s \overset{\mathcal{S}}{=} \mathsf{q}.s$ 捕获了这些前置条件和后置条件。因此，我们可以通过推导 $\vdash \{\varphi\}DH\{\psi\}$ 来证明 DH 的正确性：

$$\dfrac{\mathfrak{D} \models \varphi \to \varphi_1 \quad \dfrac{\vdash \{\varphi_1\}\mathsf{p}.(g^a \bmod m) \to \mathsf{q}.a; \ldots\{\psi\}}{\dfrac{\vdash \{\varphi_2\}\mathsf{q}.(g^b \bmod m) \to \mathsf{p}.b; \ldots\{\psi\}}{\dfrac{\vdash \{\varphi_3\}\mathsf{p}.s := b^a \bmod m; \ldots\{\psi\}}{\dfrac{\vdash \{\varphi_4\}\mathsf{q}.s := a^b \bmod m; \mathbf{0}\{\psi\}}{\dfrac{}{\vdash \{\psi\}\mathbf{0}\{\psi\}}\,\text{H|Nil}}\,\text{H|Assign}}\,\text{H|Assign}}\,\text{H|Com}}\,\text{H|Com}}{\vdash \{\varphi\}DH\{\psi\}}\,\text{H|Weak}$$

其中：

$\varphi_1 = \varphi_2[\mathsf{q}.a :=\ \mathsf{p}.g^a \bmod m] = (\mathsf{q}.g^{\mathsf{q}.b} \bmod \mathsf{q}.m)^{\mathsf{p}.a} \bmod \mathsf{p}.m \overset{\mathcal{S}}{=} (\mathsf{p}.g^{\mathsf{p}.a} \bmod \mathsf{p}.m)^{\mathsf{q}.b} \bmod \mathsf{q}.m \varphi_2 = \varphi_3[\mathsf{p}.b :=\ \mathsf{q}.g^b \bmod\ m] = (\mathsf{q}.b^{\mathsf{q}.b} \bmod \mathsf{q}.m)^{\mathsf{p}.a} \bmod \mathsf{p}.m \overset{\mathcal{S}}{=} \mathsf{q}.a^{\mathsf{q}.b} \bmod \mathsf{q}.m$
$\varphi_3 = \varphi_4[\mathsf{p}.s :=\ \mathsf{p}.b^a \bmod m] = \mathsf{p}.b^{\mathsf{p}.a} \bmod \mathsf{p}.m \overset{\mathcal{S}}{=} \mathsf{q}.a^{\mathsf{q}.b} \bmod \mathsf{q}.m \varphi_4 = \psi[\mathsf{q}.s :=$
$\mathsf{q}.a^b \bmod\ m] = \mathsf{p}.s \overset{\mathcal{S}}{=} \mathsf{q}.a^{\mathsf{q}.b} \bmod \mathsf{q}.m$

◁

现在我们可以证明，该演算是可靠的，也就是说，它只会推导出有效的判断。鉴于编排语义的迁移系统具有合流性 [12], ，只需证明这一性质对首步迁移成立：如果执行终止，那么任何执行路径都必须到达同一个最终状态。

引理 2。假设 $\mathfrak{C}$ 与 $\mathscr{C}$ 一致，且 $\vdash_{\mathfrak{C}} \{\varphi\}C\{\psi\}$。对于每个状态 Σ 和赋值 ρ，若 $\Sigma \Vdash_\rho \varphi$ 且 $\langle C、\ \Sigma\rangle \Rightarrow^*_{\mathscr{C}} \langle \mathbf{0}、\ \Sigma'\rangle$，则 $\Sigma' \Vdash_\rho \psi$。

证明。对从 $\langle C, \Sigma\rangle$ 到 $\langle \mathbf{0}, \Sigma'\rangle$ 的迁移次数作归纳。对于每种情况，再根据 $\vdash_{\mathfrak{C}} \{\varphi\}C\{\psi\}$ 的推导规模作归纳。下面给出几个有代表性的情况。

– 如果迁移次数为 0，则 $C = \mathbf{0}$ 和 $\Sigma = \Sigma'$。因此，$\vdash_{\mathfrak{C}} \{\varphi\}\mathbf{0}\{\psi\}$ 的推导必定以应用 H|Nil 结束——这意味着 $\psi = \varphi$，从而证明该命题——或者以应用 H|Weak 结束——此时，结合归纳假设与 $\mathfrak{D}$ 的可靠性，即可证明该命题。

– 假设 $\langle C, \Sigma\rangle \xrightarrow{\tau@\mathsf{p}}_{\mathscr{C}} \langle C', \Sigma'\rangle \to^*_{\mathscr{C}} \langle C'', \Sigma''\rangle$，且第一个迁移由规则 C|Assign 推导得到。那么 C 具有如下形式：$\mathsf{p}.x := e;\ C'$, $e \downarrow_{\Sigma(\mathsf{p})} v$，并且 $\Sigma' = \Sigma[\langle \mathsf{p}\ x\rangle \mapsto v]$。根据 $\vdash_{\mathfrak{C}} \{\varphi\}C\{\psi\}$ 的推导中最后应用的规则不同，分为两种情况。

  如果推导以应用 H|Assign 结束，那么对于某个公式 φ'，φ is $\varphi'[\mathsf{p}.x := \mathsf{p}.e]$，且 $\vdash_{\mathfrak{C}} \{\varphi'\}C'\{\psi\}$。由推论 1 可知 $\Sigma' \Vdash_\rho \varphi'$，对 C' 应用归纳假设即可得证。

  如果推导以应用 H|Weak 结束，那么如同基例一样，对该推导应用归纳假设即可得证。

– 假设 $\langle C, \Sigma\rangle \xrightarrow{\tau@\mathsf{p}}_{\mathscr{C}} \langle C', \Sigma'\rangle \to^*_{\mathscr{C}} \langle C'', \Sigma''\rangle$，且第一步转换由规则 C|Call 推导得到。那么 C 的形式为 $X, \lceil \mathsf{pn}(C) \setminus \mathsf{r}\ X\rfloor\mathscr{C}(X)$，并且 $\Sigma' = \Sigma$。同样，根据 $\vdash_{\mathfrak{C}} \{\varphi\}C\{\psi\}$ 的推导中所应用的最后一条规则，分为两种情况。如果推导以应用 H|Call 结束，那么根据 $\mathfrak{C}$ 和 $\mathscr{C}$ 的一致性，我们知道 $\vdash_{\mathfrak{C}} \{\varphi\}\mathscr{C}(X)\{\psi\}$；由此可以推断（使用 H|Call'），$\vdash_{\mathfrak{C}} \{\varphi\}\lceil \mathsf{pn}(C) \setminus \mathsf{r}\ X\rfloor\mathscr{C}(X)\{\psi\}$ 也成立。对这个编排应用归纳假设即可得证。如果推导以应用 H|Weak 结束，那么与前面的情况一样，即可得证。□

定理 1（可靠性）。假设 $\mathfrak{C}$ 与 $\mathscr{C}$ 和 $\vdash_{\mathfrak{C}} \{\varphi\}C\{\psi\}$ 一致。对于每个状态 Σ 和赋值 ρ，如果 $\Sigma \Vdash_\rho \varphi$ 和 $\langle C, \Sigma\rangle \to^*_{\mathscr{C}} \langle \mathbf{0}, \Sigma'\rangle$，那么 $\Sigma' \Vdash_\rho \psi$。

证明。根据 [12], 中的结果，如果 $\langle C, \Sigma\rangle \to^*_{\mathscr{C}} \langle \mathbf{0}, \Sigma'\rangle$，那么 $\langle C, \Sigma\rangle \Rightarrow^*_{\mathscr{C}} \langle \mathbf{0}, \Sigma'\rangle$ 也成立（结合无死锁性与合流性）。引理 2 随即证明了该论点。□

例 5（Zeros，函数正确性）。例 2 中程序的正确性要求：如果 f 有零，则程序会终止并找到它；等价地说，后置条件 $\psi = ((f(\mathsf{p}.x) = 0) \overset{\mathcal{Z}}{=} \mathsf{true})$ 成立。由于对初始状态没有任何假设，我们可以使用 ϕ 任意一个重言式作为前置条件（最好不含程序中使用的变量），例如$\varphi = (\mathsf{true} \overset{\mathcal{T}}{=} \mathsf{true})$。下面的推导表明，过程规范 $\mathsf{map}\ \ \mathfrak{C}(Z) = \langle \varphi, \psi\rangle$ 与 $\mathscr{C}$ 例 2 中的内容一致：

$$\dfrac{\mathfrak{D} \models \varphi \to \varphi_1 \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash_{\mathfrak{C}} \{\psi\}\mathbf{0}\{\psi\}}\,\text{H|Nil}}{\vdash_{\mathfrak{C}} \{\psi\}\mathsf{q} \to \mathsf{p[L]}; \mathbf{0}\{\psi\}}\,\text{H|Sel} \qquad \dfrac{\dfrac{\dfrac{\mathfrak{C}(Z) = \langle \varphi, \psi\rangle}{\vdash_{\mathfrak{C}} \{\varphi\}Z\{\psi\}}\,\text{H|Call}}{\vdash_{\mathfrak{C}} \{\varphi\}\mathsf{p}.x := x + 1; Z\{\psi\}}\,\text{H|Assign}}{\vdash_{\mathfrak{C}} \{\varphi\}\mathsf{q} \to \mathsf{p[R]}; \ldots\{\psi\}}\,\text{H|Sel}}\,\text{H|Cond}}{\vdash_{\mathfrak{C}} \{\varphi_2\}\mathsf{if}\ \mathsf{q}.f(x) = 0\ \mathsf{then} \ldots \mathsf{else} \ldots\{\psi\}}\,\text{H|Com}}{\vdash_{\mathfrak{C}} \{\varphi_1\}\mathsf{p}.x \to \mathsf{q}.x; \ldots\{\psi\}}\,\text{H|Weak}}{\vdash_{\mathfrak{C}} \{\varphi\}\mathscr{C}(Z)\{\psi\}}$$

其中：

$$\varphi_1 = ((f(\mathsf{p}.x) = 0) \overset{Z}{=} \mathsf{true} \to \psi) \wedge ((f(\mathsf{p}.x) = 0) \overset{Z}{=} \mathsf{false} \to \varphi)$$
$$\varphi_2 = ((f(\mathsf{q}.x) = 0) \overset{Z}{=} \mathsf{true} \to \psi) \wedge ((f(\mathsf{q}.x) = 0) \overset{Z}{=} \mathsf{false} \to \varphi)$$

整个程序也满足相同的前置条件和后置条件：

$$\dfrac{\dfrac{\mathfrak{C}(Z) = \langle \varphi, \psi \rangle}{\vdash_{\mathfrak{C}} \{\varphi\} Z \{\psi\}}\,\mathrm{H}|\textsc{Call}}{\vdash_{\mathfrak{C}} \{\varphi\} \mathsf{p}.x \coloneqq 0; Z \{\psi\}}\,\mathrm{H}|\textsc{Assign}$$

根据可靠性可知，任何终止的执行都会以满足 Σ s.t. $f(x) = 0 \downarrow_{\Sigma(\mathsf{p})}$ true 的状态结束。观察可知，p 从 0 开始扫描自然数，每次递增 1，因此，如果 f 中有任何一个为零，p 最终会将其中第一个零发送给 q，而 q 随即会选择终止搜索。◁

4 Hoare 演算的完备性

为了证明我们的演算具有完备性，我们采用文献中的标准方法，使用最弱自由前置条件的概念——给定 $\mathfrak{C}$、C 和 ψ 时，最弱的断言 φ，使得 $\vdash_{\mathfrak{C}} \{\varphi\} C \{\psi\}$。

4.1 最弱自由前置条件

本节定义最弱自由前置条件算子，并展示它满足预期的性质。

定义 2. 设 C 为一个编排，ψ 为一个公式，$\mathfrak{C}$ 为一个过程规约映射。在 $\mathfrak{C}$ 下，C 和 ψ 的最弱自由前置条件 $\mathsf{wlp}_{\mathfrak{C}}(C, \psi)$ 定义如下。

$$\begin{aligned}
\mathsf{wlp}_{\mathfrak{C}}((\mathsf{p}.x \coloneqq e; C), \psi) &= \mathsf{wlp}_{\mathfrak{C}}(C, \psi)[\mathsf{p}.x := \mathsf{p}.e] \\
\mathsf{wlp}_{\mathfrak{C}}((\mathsf{p}.e \to \mathsf{q}.x; C), \psi) &= \mathsf{wlp}_{\mathfrak{C}}(C, \psi)[\mathsf{q}.x := \mathsf{p}.e] \\
\mathsf{wlp}_{\mathfrak{C}}((\mathsf{p} \to \mathsf{q}[\mathrm{L}]; C), \psi) &= \mathsf{wlp}_{\mathfrak{C}}(C, \psi) \\
\mathsf{wlp}_{\mathfrak{C}}(\mathsf{if}\ \mathsf{p}.b\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2, \psi) &= (L(\mathsf{p}, b) \overset{\mathcal{X}}{=} \mathsf{true} \to \mathsf{wlp}_{\mathfrak{C}}(C_1, \psi)) \\
&\quad \wedge (L(\mathsf{p}, b) \overset{\mathcal{X}}{=} \mathsf{false} \to \mathsf{wlp}_{\mathfrak{C}}(C_2, \psi)) \\
\mathsf{wlp}_{\mathfrak{C}}(X, \psi) &= \mathsf{fst}(\mathfrak{C}(X)) \\
\mathsf{wlp}_{\mathfrak{C}}(\lceil \vec{\mathsf{q}}, X \rfloor C, \psi) &= \mathsf{wlp}_{\mathfrak{C}}(C, \psi) \\
\mathsf{wlp}_{\mathfrak{C}}(\mathbf{0}, \psi) &= \psi
\end{aligned}$$

这个算子本质上是在模仿图 3 中的规则。在条件语句的子句中，$\mathcal{X}$ 是新鲜的。唯一可能令人意外的是定义

$\mathsf{wlp}_{\mathfrak{C}}(X, \psi)$，它忽略了实际的公式 ψ：这同样是因为我们的结果要求对 $\mathfrak{C}$ 附加一个条件（即给定的条件与 $\mathsf{wlp}_{\mathfrak{C}}$ 的定义兼容），这间接确保了 ψ 也会被考虑。

例 6（Diffie-Hellman，WLP）。考虑例 1 中的编排 DH 和例 4 中的后置条件 $\psi = (\mathsf{p}.s \overset{\mathcal{S}}{=} \mathsf{q}.s)$，$\mathsf{wlp}(DH, \psi)$ 是例 4 中的公式 φ_1。◁

定义 3。若过程规范映射 $\mathfrak{C}$ 对给定的过程定义集合 $\mathscr{C}$ 而言适用于 ψ，则对于任意过程名称 X，$\mathsf{fst}(\mathfrak{C}(X))$ 与 $\mathsf{wlp}_{\mathfrak{C}}(\mathscr{C}(X)\ \psi)$ 逻辑等价，且 $\mathsf{snd}(\mathfrak{C}(X)) = \psi$。

换言之，对于每个 ψ，我们关注一个映射 $\mathfrak{C}$；该映射为每个过程提供正确的前置条件，以确保该过程若终止，ψ 就会成立。

例 7（Zeros，WLP）。例 5 中的过程规范映射 $\mathfrak{C}$，对于该例中的后置条件而言，适用于例 2 中的过程定义集合 $\mathscr{C}$。事实上，$\mathsf{wlp}_{\mathfrak{C}}(\mathscr{C}(Z)\ f(\mathsf{p}.x) = 0 \overset{Z}{=} \mathsf{true})$ 是例 5 中的公式 φ_1，它与 $\mathsf{fst}(\mathfrak{C}(Z))$ 逻辑等价。◁

接下来的结果表明，$\mathsf{wlp}_{\mathfrak{C}}(C, \psi)$ 精确刻画了这样一组状态：从这些状态执行 C，即可保证 ψ。

引理 3. 假设对于给定的 $\mathscr{C}$，$\mathfrak{C}$ 对 ψ 是充分的。那么，对于每个编舞程序 C，$\vdash_{\mathfrak{C}} \{\mathsf{wlp}_{\mathfrak{C}}(C, \psi)\} C \{\psi\}$。

证明。对 C 进行结构归纳。大多数情形可直接由 $\mathsf{wlp}_{\mathfrak{C}}$ 的定义和归纳假设得出。下面详细说明唯一的非平凡情形。

– 若 C 是 $\mathsf{if}\ \mathsf{p}.b\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2$，注意到 $\vdash_{\mathfrak{C}} \{\mathsf{wlp}_{\mathfrak{C}}(C_1, \psi)\} C_1 \{\psi\}$。由于

$$(\mathsf{wlp}_{\mathfrak{C}}(\mathsf{if}\ \mathsf{p}.b\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2, \psi) \wedge L(\mathsf{p}, b) \overset{\mathcal{X}}{=} \mathsf{true}) \rightarrow \mathsf{wlp}_{\mathfrak{C}}(C_1, \psi)$$

是一个有效的命题公式，我们可以应用规则 H|Weak 推导出 $\vdash_{\mathfrak{C}} \{\mathsf{wlp}_{\mathfrak{C}}(\mathsf{if}\ \mathsf{p}.b\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2, \psi) \wedge L(\mathsf{p}\ b) \overset{\mathcal{X}}{=} \mathsf{true}\} C_1 \{\psi\}$。对 C_2 作类似推理，可推导出规则 H|Cond 的另一个假设；将二者结合即可证明结论。– 若 C 是 X，则由假设 $\mathsf{snd}(\mathfrak{C}(X)) = \psi$.□ 可得结论。

推论 2. 如果在给定 $\mathscr{C}$ 的情况下，$\mathfrak{C}$ 适用于 ψ，那么 $\mathfrak{C}$ 与 $\mathscr{C}$一致。

推论3。假设在给定 $\mathscr{C}$ 的情况下，$\mathfrak{C}$ 对 ψ 是充分的。对于任意编排 C、状态 Σ 和赋值 ρ，如果 $\Sigma \Vdash_{\rho} \mathsf{wlp}_{\mathfrak{C}}(C, \psi)$ 且 $\langle C, \Sigma\rangle \rightarrow^*_{\mathscr{C}} \langle \mathbf{0}, \Sigma'\rangle$ 对某个状态 Σ' 成立，那么 $\Sigma' \Vdash_{\rho} \psi$。

证明。根据引理 3，$\vdash_{\mathfrak{C}} \{\mathrm{wlp}_{\mathfrak{C}}(C,\psi)\}C\{\psi\}$。根据推论 2，$\mathfrak{C}$ 与 $\mathscr{C}$ 一致。于是由定理 1 可得结论。□

引理 4. 假设 $\mathfrak{C}$ 对于给定的 $\mathscr{C}$，是 ψ 的充分条件。设 C 为一个编排，Σ 和 Σ' 为状态，ρ 为一个赋值。若 $\langle C、\Sigma\rangle \Rightarrow^*_{\mathscr{C}} \langle \mathbf{0}、\Sigma'\rangle$ 和 $\Sigma' \Vdash_\rho \psi$，则 $\Sigma \Vdash_\rho \mathrm{wlp}_{\mathfrak{C}}(C,\psi)$。

证明。对从 C 到 0 的迁移次数进行归纳。若迁移次数为 0，则 C 为 0，结论显然成立。否则，我们详细说明一些具有代表性的情况。对 C 进行分类讨论，以确定第一个迁移。

– 若 C 为 $\mathsf{p}.x := e;\ C''$，则由归纳假设可得 $\langle C,\Sigma\rangle \ {}^{\tau}@\mathsf{p}\Longrightarrow_{\mathscr{C}} \langle C'',\Sigma''\rangle \Rightarrow^*_{\mathscr{C}} \langle \mathbf{0},\Sigma'\rangle$，且 $\Sigma'' \Vdash_\rho \mathrm{wlp}_{\mathfrak{C}}(C'',\psi)$。但 $\Sigma'' = \Sigma[\langle \mathsf{p}\ x\rangle \mapsto v]$，其中 $e\downarrow_{\Sigma(\mathsf{p})} v$，因此根据推论 1，$\Sigma \Vdash_\rho \mathrm{wlp}_{\mathfrak{C}}(C'',\psi)[\mathsf{p}.x := \mathsf{p}.e]$，从而证明结论。– 若 C 为 $\mathsf{if}\ \mathsf{p}.b\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2$，则分两种情况。无妨设 $b\downarrow_{\Sigma(\mathsf{p})}$ 为 true。则 $\langle \mathsf{if}\ \mathsf{p}.b\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2,\Sigma\rangle \ {}^{\tau}@\mathsf{p}\Longrightarrow_{\mathscr{C}} \langle C_1,\Sigma\rangle \Rightarrow^*_{\mathscr{C}} \langle \mathbf{0},\Sigma'\rangle$，且由归纳假设可得 $\Sigma \Vdash_\rho \mathrm{wlp}_{\mathfrak{C}}(C_1,\psi)$。唯一非平凡的情况是$\rho(\mathcal{X}) = \mathsf{true}$——否则，$\mathrm{wlp}_{\mathfrak{C}}(C,\psi)$ 中两个蕴含的前件都为假，结论显然成立。若 $\rho(\mathcal{X}) = \mathsf{true}$，则根据引理 1，$\Sigma \vdash_\rho L(\mathsf{p}\ b) = \mathcal{X}$，且 $\mathrm{wlp}_{\mathfrak{C}}(C,\psi)$中的两个蕴含也都为真（第一个蕴含的前提和结论都为真，而第二个蕴含的前提为假）。$b\downarrow_{\Sigma(\mathsf{p})}$ 为 false 的情况类似。– 若 C 为 X，则通过应用规则 C|Call、C|Enter 和 C|Finish，直到所有进程都进入 X，可得 $\langle X,\Sigma\rangle \Rightarrow^*_{\mathscr{C}} \langle \mathscr{C}(X)\ \Sigma\rangle \Rightarrow^*_{\mathscr{C}} \langle \mathbf{0},\Sigma'\rangle$。由充分性可得，$\mathrm{fst}(\mathfrak{C}(X)) = \mathrm{wlp}_{\mathfrak{C}}(\mathscr{C}(X)\ \psi)$，而归纳假设证明了结论。□

推论 4. 假设 $\mathfrak{C}$ 对于给定的 $\mathscr{C}$ 足以满足 ψ。令 C 为一个编排，Σ 和 Σ' 为状态，ρ 为一个赋值。若 $\langle C、\Sigma\rangle \rightarrow^*_{\mathscr{C}} \langle \mathbf{0}、\Sigma'\rangle$ 和 $\Sigma' \Vdash_\rho \psi$，则 $\Sigma \Vdash_\rho \mathrm{wlp}_{\mathfrak{C}}(C,\psi)$。

证明. 结合引理 4、语义的无死锁性与合流性，并按照定理 1 的证明方式可得。□

4.2 完备性

结合上一节的结果，我们为该演算得到了一个完备性结果。

定理 2（部分完备性）。令 C 为一个编排，φ 和 ψ 为公式，并假设对于给定的 $\mathscr{C}$，$\mathfrak{C}$ 足以满足 ψ。假设对于所有状态 Σ 和 Σ' 以及赋值 ρ，若 $\Sigma \Vdash_\rho \varphi$ 且 $\langle C,$ 则 $\Sigma\rangle \rightarrow^*_{\mathscr{C}} \langle \mathbf{0}、\Sigma'\rangle$，进而 $\Sigma' \Vdash_\rho \psi$。则 $\vdash_{\mathfrak{C}} \{\varphi\}C\{\psi\}$。

证明。令 Σ 为一个状态，使得 $\langle C,\Sigma\rangle \rightarrow^*_{\mathscr{C}} \langle \mathbf{0},\Sigma'\rangle$, 蕴含 $\Sigma' \Vdash_\rho \psi$。那么$\Sigma \Vdash_\rho \mathrm{wlp}_{\mathfrak{C}}(C,\psi)$, 由推论 4 得出。由于对于所有满足$\Sigma \Vdash_\rho \varphi$的状态 Σ，情况皆如此，因此可得 $\mathfrak{D} \Vdash \varphi \rightarrow \mathrm{wlp}_{\mathfrak{C}}(C,\psi)$。但是 $\vdash_{\mathfrak{C}} \{\mathrm{wlp}_{\mathfrak{C}}(C,\psi)\}C\{\psi\}$ 由引理 3 得出，因此根据 H|Weak，结论成立。□

定理1和定理2可以与 [12], 中的 EPP定理结合起来，该定理将编排的行为与其投影的行为联系起来，从而得出关于编排生成的分布式实现执行的结果。这意味着可以在编排层面分析这些实现的性质；编排层面相对简单，也就无需为进程语言专门设计霍尔演算。

4.3 可判定性

最后，我们为霍尔演算确立一些可判定性结果。首先指出，我们假设 $\mathfrak{D}$ 可判定；由于命题逻辑可判定且求值收敛，规则 H|Weak 的前提中出现的 $\mathfrak{D} \models \varphi$ 形式的判断也都是可判定的。

引理5。判断 $\vdash_{\mathfrak{C}} \{\varphi\}C\{\psi\}$ 是可判定的。

证明。假设 $\vdash_{\mathfrak{C}} \{\varphi\}C\{\psi\}$。根据定理 1，对于每个状态 Σ 和赋值 ρ，只要 $\Sigma \Vdash_\rho \varphi$，就有：如果 $\langle C,\ \Sigma\rangle \to^*_{\mathscr{C}} \langle \mathbf{0},\ \Sigma'\rangle$，那么 $\Sigma' \Vdash_\rho \psi$。根据推论 4，这意味着 $\Sigma \Vdash_\rho \mathbf{wlp}_{\mathfrak{C}}(C,\psi)$，因此 $\mathfrak{D} \models \varphi \to \mathbf{wlp}_{\mathfrak{C}}(C,\psi)$】【。

反过来，如果 $\mathfrak{D} \models \varphi \to \mathbf{wlp}_{\mathfrak{C}}(C,\psi)$，那么对于每个状态 Σ 和赋值 ρ，只要 $\Sigma \Vdash_\rho \varphi$，就有 $\Sigma \Vdash_\rho \mathbf{wlp}_{\mathfrak{C}}(C,\psi)$。因此，若 $\langle C,\Sigma\rangle \to^*_{\mathscr{C}} \langle \mathbf{0},\Sigma'\rangle$，则根据推论3，必有 $\Sigma' \Vdash_\rho$。根据定理 2，这意味着 $\vdash_{\mathfrak{C}} \{\varphi\}C\{\psi\}$。

这表明 $\vdash_{\mathfrak{C}} \{\varphi\}C\{\psi\}$ 当且仅当 $\mathfrak{D} \models \varphi \to \mathbf{wlp}_{\mathfrak{C}}(C,\psi)$。由于 $\mathbf{wlp}_{\mathfrak{C}}$ 是可计算的，且有效性是可判定的，因此 $\vdash_{\mathfrak{C}} \{\varphi\}C\{\psi\}$ 是可判定的。□

尽管过程名称集合原则上可能是无限的，但大多数实际应用只会用到其中的有限子集。[2] 在这种情况下，一致性和充分性也变得可判定。

推论5. 如果过程名称集合是有限的，那么过程规范映射 $\mathfrak{C}$ 与过程定义集合 $\mathscr{C}$ 之间的一致性是可判定的。

引理6. 如果过程名称集合是有限的，那么过程规范映射对于某个公式和过程定义集合的充分性是可判定的。

证明. 由定义可直接得出。 □

我们以一个否定性结果结束本节：无法计算出一个充分的过程规范映射。

引理7. 不存在这样的算法：给定过程定义集合 $\mathscr{C}$和公式 ψ，总能返回一个对于给定的 $\mathscr{C}$ 而对 ψ 充分的过程规范映射 $\mathfrak{C}$。

[2] 这排除了这样的编排：例如，每个过程 X_i 都调用过程 X_{i+1}，而这在实践中并不会发生。

证明。考虑公式 $\psi = \bot$，它永远不成立。对于任意编排 C 和可满足公式 φ，判断 $\{\varphi\}C\{\bot\}$ 成立，当且仅当 C 从满足 φ 的状态出发永不终止。

这意味着，如果在给定 $\mathscr{C}$ 的情况下，$\mathfrak{C}$ 对于 $\bot$ 是充分的，那么 $\mathsf{wlp}_{\mathfrak{C}}(C, \bot)$ 刻画了从哪些状态执行 C 会发散。特别是，如果 $\mathsf{wlp}_{\mathfrak{C}}(C, \bot)$ 与 $\top$ 逻辑等价，那么 C 永不终止；而这在我们的状态逻辑中是可判定的。但是，赖斯定理意味着，始终发散的编排类是不可判定的，因此 $\mathfrak{C}$ 不可计算。□

尽管这一结果表明，充分的过程规范映射通常不可计算，但仍有可能证明它们始终存在。这样的结果将意味着我们的演算是强完备的。我们计划在未来的工作中研究这一问题。

5 相关工作

与我们最接近的工作是 [20], ，作者在其中提出了一个用于编排功能正确性的系统，旨在对分布式选择进行推理。尽管他们也为编排提出了霍尔演算，但与我们的工作相比，仍存在一些关键差异。

首先，他们引入了一种新的编舞语言，它与编排式编程中的常见做法有显著差异。例如，他们要求每个选择都涉及所有进程，无论这些进程是否参与条件分支。相比之下，我们使用的是一种已有语言，其构造符合标准做法。

其次， [20] 中用于布尔表达式的逻辑是固定的，并且与编舞语言绑定在一起。这种耦合损害了该工作的通用性，因为逻辑与编排的语法并非彼此独立。相反，我们遵循霍尔逻辑 [2,19], 的标准双层方法，并定义了一种状态逻辑，它以编排中的表达式语言和用于推理的理论为参数。

因此，我们的工作更容易应用于其他现有编舞语言，也更容易适应这些语言。

唯一另一项结合编排与逻辑的工作是线性组合式编排（LCC） [7], ，这是一种基于线性逻辑的证明论，用于推理以模块化方式将组合式编排 [28]与进程结合起来的程序。该工作受到早期关于线性命题与会话类型之间对应关系的研究启发 [5]。然而，LCC并不以函数正确性为目标：命题表示通信行为，而非关于状态的断言。

契约式设计 [25] 是一种框架，为每个协议或函数指定一个契约，说明其允许的输入和产生的输出，类似于霍尔逻辑中的前置条件和后置条件。霍尔逻辑曾被用于从全局层面推理分布式程序。该研究方向的首项工作 [4] 定义了一个用于为多方会话指定契约的框架。它基于

会话类型，因此这项工作比我们的工作更侧重于指定通信值的属性，使其能够指定比我们更多的属性，但也需要在被推理的语言中添加注释。对这一思想的扩展 [24] 描述了用于高阶二元会话的 Chaperone 合约，使合约能够在运行时动态更新。契约式设计也以 Whip [31]的形式应用于微服务。与我们的工作类似，Whip 对局部语言不作特定要求；不过，它使用全局契约直接针对局部语言进行推理。与我们的逻辑不同，Whip 旨在运行时监控通信。

另一种推理会话类型的方法是将其与依赖类型 [30] 结合起来。与 [4]，的研究类似，依赖类型可用于推理所传递的值；但与我们的研究不同，它们并非旨在用于推理前置条件和后置条件。

霍尔逻辑也被用于直接推理通信进程系统 [1,22]。这比推理编排程序复杂得多，因为这需要分别考虑每个参与者协议的性质，以及它们如何组合成全局系统。

6 结论

我们提出了一种用于推理编排式程序的新型霍尔演算。我们的逻辑具有很大的灵活性，因为它既以编舞语言的局部语言为参数，也以用户定义的可判定理论为参数。

我们已证明，霍尔逻辑的标准性质在我们的语言中成立。借助编排及其投影的操作对应定理，我们还证明了：逻辑能够为某个编排证明的任何性质，也会自动适用于由该编排生成的分布式实现。

关于可判定性的章节留下了一个尚未解答的问题：对于任意目标公式，是否总存在一个合适的过程规范映射？我们计划在未来工作中对此进行研究。我们还希望进一步探讨如何利用可判定性结果来实现有趣的算法，例如用于证明自动化。

我们的形式体系只能保证终止执行路径的性质，这意味着我们无法推断非终止编排的任何性质。然而，检查可靠性和完备性的证明（尤其是引理 2 和引理 3）可以发现，这些结果实际上保证了更强的结论，即每当编排到达过程调用时，$\mathfrak{C}$ 中描述的不变量必定成立。我们计划以这一观察为起点，研究如何利用我们的演算来断言编排非终止执行的性质。

致谢。本研究部分由 Villum Fonden 通过编号为 29518 的资助项目支持。

参考文献

1. Apt, K.R., Francez, N., de Roever, W.P.：通信顺序进程的证明系统。ACM Trans. Program. Lang. Syst. 2(3), 359—385（1980 年 7 月）。https://doi.org/10 357110 2 1145 357103. / . . Apt, K.R., Olderog, E.：霍尔逻辑五十年。CoRR abs/1904.03917（2019）3. Apt, K.R., Olderog, E.R., Apt, K.：顺序程序与并发程序的验证，第 2 卷。Springer（2009）4. Bocchi, L., Honda, K., Tuosto, E., Yoshida, N.：分布式多方交互的契约式设计理论。载于：Gastin 和 Laroussinie [15], pp. 162–176。https://doi.org/ 10 1007 978 3 642 15375 4_12 . / - - - - . Caires, L., Pfenning, F.：会话
5. 类型作为直觉主义线性命题。载于：Gastin 和 Laroussinie [15], pp. 222–236。https://doi.org/ 10 1007 978 3 642 15375 4_16 . / - - - - . Carbone, M., Montesi, F.：设计时
6. 无死锁：多方异步全局编程。载于：Giacobazzi, R., Cousot, R.（编）POPL 论文集。pp. 263–274 2013 10 1145 2429069 2429101 。ACM（）。https://doi.org/ .
7. / . . Carbone, M., Montesi, F., Schurmann, C.：从逻辑角度看编排。Distributed Comput. 31 1 8 1 51 67 2018 10 1007 00446 017 0295（），–（）。https://doi.org/ . /s - - - . Cruz-Filipe, L., Graversen, E., Lugovic, L., Montesi, F., Peressotti, M.：函数式编排编程。载于：Seidl, H., Liu, Z., Pasareanu, C.S.（编）ICTAC 论文集。《计算机科学讲义》，第 13572 卷，pp. 212–237。Springer（ 2022 10 1007 978 3 031 17715 6_15 ）。https://doi.org/ . / - - - - . Cruz-Filipe, L., Montesi, F.：过程式编排编程。载于：Bouajjani, A., Silva, A.（编） FORTE
9. 论文集。《计算机科学讲义》，第 10321 卷，pp. 92–107。Springer（2017）。https://doi.org/ 10 1007 978 3 319 60225 7_7 . / - - - - . Cruz-Filipe, L., Montesi, F.：编排
10. 式编程的核心模型。Theor. Comput. Sci. 802, 38 005 11 66 2020 10 1016 2019 07 –（）。https://doi.org/ . /j.tcs. . . . Cruz-Filipe, L., Montesi, F., Peressotti, M.：认证式编排编译。载于：Cerone, A., ¨Olveczky, P.C.（编）ICTAC 论文集。LNCS，第 12819 卷，pp. 115–133。Springer（ 2021 10 1007 978 3 030 85315 0_8 ）。https://doi.org/ . / - -
12. - - . Cruz-Filipe, L., Montesi, F., Peressotti, M.：在 Coq 中形式化一种图灵完备的编排语言。载于：Cohen, L., Kaliszyk, C.（编）ITP 论文集。LIPIcs，第 193 卷，pp. 15:1–15:18。Schloss Dagstuhl – Leibniz-Zentrum f¨ur Informatik（2021）。https://doi.org/10 15 13 4230 2021 . / LIPIcs.ITP. . . Dalla Preda, M., Gabbrielli, M., Giallorenzo, S., Lanese, I., Mauro, J.：动态编排：理论与实现。Log. Methods Comput. Sci. 13(2)（2017 2017 14 10 23638 13 2 1）。https://doi.org/ . /LMCS- (:) . Diffie, W., Hellman, M.E.：密码学的新方向。IEEE Trans. Inf. Theory 22 1055638 15 6 644 654 1976 10 1109 1976 2010（），–（）。https://doi.org/ . /TIT. . . Gastin, P., Laroussinie, F.（编）：CONCUR——并发理论，第二十一届国际会议，CONCUR 2010, 2010 年巴黎，2010 年 8 月至 9 月。会议论文集，《计算机科学讲义》，第 6269 卷。Springer（2010）16. Giallorenzo, S., Montesi, F., Peressotti, M.：作为对象的编排。CoRR abs/2005 09520 17 09520 2020 2005 .（），https://arxiv.org/abs/ . . Giallorenzo, S., Montesi, F., Peressotti, M., Richter, D., Salvaneschi, G., Weisenburger, P.：多方语言：编排式与多层案例（Pearl）。载于：Møller, A., Sridharan, M.（编）ECOOP 论文集。LIPIcs，第 194 卷，pp. 22:1–22:27。Schloss Dagstuhl - Leibniz-Zentrum f¨ur Informatik（2021）。https://doi.org/10.4230/LIPIcs.ECOOP.2021.22

18. Hirsch, A.K., Garg, D.：Pirouette：高阶类型化函数式编舞。《ACM 程序语言学报》6(POPL)，1–27（2022）。https://doi.org/10 3498684 19 1145. / . Hoare, C.：计算机编程的公理基础。《ACM 通讯》12(10)，576 363259 20 580 1969 10 1145 363235 – ( )。https://doi.org/ . / . . Jongmans, S., van den Bos, P.：编舞的谓词变换器——在编排式编程中计算前置条件。载于：Sergey, I.（编）《ESOP 论文集》。《计算机科学讲义》第 13240 卷，第 520–547 页。Springer（ 2022 10 1007 978 3 030 99336 8_19 ）。https://doi.org/ . / - - - - . 21 Leesatapornwongsa, T., Lukman, J.F., Lu, S., Gunawi, H.S.：Taxdc：数据中心分布式系统中非确定性并发错误的分类。在：Conte, T., Zhou, Y.（编）《ASPLOS 论文集》。第 517–530 页。ACM（2016）。https://doi.org/10 2872374 22 1145 2872362 . / . . Levin, G., Gries, D.：通信顺序进程的一种证明技术。《Acta Informatica》**15**, 281 302 1981 10 1007 00289266 – （）。https://doi.org/ . /BF . 23 Lopez, H.A., Nielson, F., Nielson, H.R.：在故障感知通信系统中强制实现可用性。载于：Albert, E., Lanese, I.（编）《FORTE 论文集》。《计算机科学讲义》第 9688 卷，第 195–211 页。Springer（2016）。https://doi.org/ 10 1007 978 3 319 39570 8_13 . / - - - - . Melgratti, H.C., Padovani, L.：面向高阶会话的 Chaperone 合约。《ACM 程序语言学报》1(ICFP)，35:1–35:29（2017）。https://doi.org/10 3110279 24 25 1145. / . Meyer, B.：应用“契约式设计”。《Computer》25(10)，40–51（1992）。https://doi.org/10 161279 26 1109 2. / . . Montesi, F.：编舞式编程。博士论文，哥本哈根信息技术大学（2013）27. Montesi, F.：编舞导论。剑桥大学出版社（2023）28. Montesi, F., Yoshida, N.：组合式编排。载于：D’Argenio, P.R., Melgratti, H.C.（编）《CONCUR 论文集》。《计算机科学讲义》第 8052 卷，第 425 439 2013 10 1007 978 3 642 40184 8_30 – 页。Springer（）。https://doi.org/ . / - - - - . 29 Needham, R.M., Schroeder, M.D.：在大型计算机网络中使用加密进行身份验证。《ACM 通讯》21(12)，993–999（1978）。https://doi.org/10 359659 30 1145 359657. / . . Toninho, B., Caires, L., Pfenning, F.：通过直觉线性类型理论实现依赖会话类型。载于：Schneider-Kamp, P., Hanus, M.（编）《PPDP 论文集》。第 161 2003499 31 172 2011 10 1145 2003476 – 页。ACM（）。https://doi.org/ . / . . Waye, L., Chong, S., Dimoulas, C.：Whip：面向现代服务的高阶合约。《ACM 程序语言学报》1(ICFP)，36:1–36:28（2017）。https://doi.org/10.1145/3110280