arXiv:2601.13325v1 [cs.LO] 19 Jan 2026

通过计时器和排序验证无限状态系统的一阶时序性质

Raz Lotan[1,2], Neta Elad[1], Oded Padon[3], 和 Sharon Shoham[1]

[1] 特拉维夫大学，特拉维夫，以色列[2] Certora，特拉维夫，以色列[3] 魏茨曼科学研究所，雷霍沃特，以色列 lotanraz@gmail.com

摘要。 我们提出了一个基于良基排序的统一演绎验证框架，用于验证一阶时序性质，并使用 SMT 求解器证明验证条件。为此，我们提出了一种新颖的规约方法，将任意时序性质的验证规约为终止性验证。该规约通过为系统添加预言计时器变量，预测沿轨迹经过多少步后，下一个特定时序公式（包括被否定的性质）将成立。与将问题规约为公平终止性的标准表格法规约不同，我们的规约不引入公平性假设。为了验证增强系统的终止性，我们采用传统方法，为每个状态赋予一个来自良基集合的秩，并证明该秩在每次转换中都会递减。我们利用近期提出的隐式排序形式化方法，通过 SMT 求解器表达并自动验证秩的递减，即使秩无法用一阶逻辑表达也不例外。我们将隐式排序从有限域扩展到无限域，从而能够验证更一般的系统，并使其适用于规约生成的增强系统；这使我们能够在终止性证明中利用计时器的递减。我们在一系列以往工作中的时序验证任务上评估了该技术，并在我们的框架中为这些任务给出了简单直观的证明。

1 引言

近年来，已有多项研究 [28, 29, 32, 33, 41] 致力于验证以一阶逻辑描述的迁移系统的时序性质。这类系统的时序性质通常使用一阶线性时序逻辑（FO-LTL）来描述。FO-LTL 在标准一阶逻辑的基础上扩展了时序算子，用于推理迁移系统随时间演变的行为，并可在各种公平性假设下描述安全性质和活性性质。

现有针对一阶逻辑中给定的迁移系统活性性质的验证方法，大致可分为采用活性到安全性归约的方法和使用良基排序的方法。 [32]提出的活性到安全性归约可以针对任意一阶线性时序逻辑性质，并且功能十分强大

，若结合时序预言 [22,33]，效果会更好。然而，验证所得系统的安全性颇为复杂，因为这需要对系统的一个受监控副本进行推理，该副本会跟踪动态有限抽象下的重复状态。相比之下，良基排序在原始系统层面运作，因此使用起来更直观；但该领域现有研究 [28,29,41] 仅针对特定的活性性质，且能够涵盖的排序论证类型有限。

我们的目标是设计一个一阶迁移系统验证框架，使其既便于用户直观使用，又适用于任意时序性质。为此，我们结合了两个关键思想。首先，为了获得统一的方法，我们的框架基于一种新颖的规约，将 FO-LTL 性质验证规约为终止性验证。我们的规约是可靠且完备的，并会生成一个易于推理的迁移系统，因为它通过加入用于跟踪性质违反情况的预言变量，对原迁移系统进行了扩充。随后，验证所得迁移系统的终止性，就等价于证明原系统不存在违反属性的轨迹。其次，在终止性验证方面，我们采用并扩展了 [28]，中提出的隐式排序概念，使用户能够编码一系列良基排序，其中可以包含规约所引入的预言变量。隐式排序由用户提供，并由 SMT 求解器自动验证。

我们的终止性规约以迁移系统 T 和时序性质 φ 为输入，将 $T \models \varphi$ 的验证规约为验证一个增广转移系统的终止性；该增广转移系统由 T 与 $T_{\neg\varphi}$的同步组合得到，即由 $\neg\varphi$的计时器迁移系统 构成。 $T_{\neg\varphi}$ 的无限轨迹恰好是所有满足 $\neg\varphi$ 的无限轨迹。计时器迁移系统类似于标准的表格法构造，但不包含公平性假设。因此，我们得到的是到终止性验证而非公平终止验证的规约，后者更易于处理。为构造 $T_{\neg\varphi}$ ，我们针对 $\neg\varphi$ 的每个子公式 ψ 引入一个计时器变量 t_ψ，其取值属于 $\mathbb{N} \cup \{\infty\}$，用于预测直到 ψ 下一个得到满足所需的转换次数；如果不存在这样的次数，则取值为 ∞ 。 $T_{\neg\varphi}$的转换强制实现了这种行为。随后，将 $T_{\neg\varphi}$ 的初始状态定义为满足 $t_{\neg\varphi} = 0$ 的状态，这意味着它们所有的出发无限轨迹都满足 $\neg\varphi$。由此可知，组合 $T \times T_{\neg\varphi}$是带有计时器的 T 的扩充，其中包含 T 满足 $\neg\varphi$的所有无限轨迹。因此，当且仅当通过该组合得到的增广转移系统不存在无限轨迹，即终止时， $T \models \varphi$ 成立。

为了证明规约所生成迁移系统的终止性，我们采用隐式排序方法 [28]。隐式排序是一阶公式，用于捕捉两个状态之间某个排序函数的递减，而不显式定义该函数。隐式排序有助于推理那些难以用一阶逻辑表达的排序函数，例如涉及无界集合基数或求和的函数。 [28] 中定义的隐式排序构造子，为良基排序提供了一种表达能力强的语言，可以使用 SMT 求解器在一阶逻辑中进行推理；但这些构造子仅适用于运行在有限但可能无界的

论域上的系统。具体而言，尽管计时器的论域是良基的，它们仍无法在排序中利用计时器。为解决这一问题，我们扩展了隐式排序形式体系，使其能够对具有无限论域的系统进行可靠推理。为此，我们引入了一个显式的可靠性条件，例如允许我们要求某个动态更新的集合在所有可达状态中都是有限的，或者某个关系是良基的。我们还将 [28] 的构造子推广到了这一设定。具体而言，这使我们能够在用于证明规约所生成增强系统终止性的排序中，利用计时器的递减。重要的是，终止性证明中对计时器的使用依赖于其直观含义，无需理解增强系统的构造过程。

我们在一个演绎验证工具中实现了规约和隐式排序的构造。该工具以迁移系统规约和一阶时序性质为输入，同时还需提供隐式排序（由构造器的复合给出）以及辅助不变式；辅助不变式可以引用计时器变量。该工具利用这些要素生成验证条件，以刻画在加入计时器后的系统的每个可达迁移中，隐式排序所编码的排序值递减。随后，工具使用 SMT 求解器自动证明这些验证条件，从而验证系统满足该性质。我们通过验证先前研究中的一组具有挑战性的示例来评估工具，其中包括分布式协议的活性性质和程序的终止性。我们在框架内为这些示例给出了简单直观的证明。

贡献

– 我们提出一种规约，将任意一阶线性时序逻辑性质的验证转化为终止性验证，并通过预言计时器变量扩充迁移系统。– 我们将近期工作的隐式排序及其构造器定义推广到无限域情形。– 我们将终止性规约与隐式排序相结合，构建了一个演绎验证框架：用户为增强系统构造隐式排序，随后由 SMT 求解器自动验证增强系统每一步中的排序值递减。– 我们针对文献中的一组迁移系统及其性质实现并评估了这一方法。

本文其余部分安排如下：第 2 节概述一个动机示例；第 3 节介绍一阶逻辑和一阶线性时序逻辑的背景；第 4 节介绍从一阶线性时序逻辑到终止性的规约；第 5 节给出隐式排序的推广定义、用于利用隐式排序证明终止性的证明规则，以及隐式排序的构造器；第 6 节详细介绍我们的实现和评估；第 7 节讨论相关工作并总结全文。所有证明均推迟至附录 A。

2 运行示例

为说明我们的方法，我们考虑用于确保互斥的票据协议，这是 Lamport 面包店算法 [25] 的一个变体。在票据协议中，对临界区的访问由自然数票据控制。该协议维护下一个可用票据和当前正在处理的票据，二者均初始化为零。线程从空闲状态开始，可以通过获取下一个可用票据并随即将其递增，转移到等待状态。一旦处于等待状态，线程必须等待其票据成为当前正在处理的票据，才能进入临界区。立即-

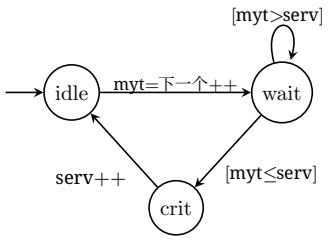

图 1。 Ticket 协议

线程进入临界区后，会立即返回空闲状态，同时将当前正在处理的票号递增。每个线程的控制流图如图 1 所示。

我们要验证的时序性质是：在公平调度下，每个等待进入临界区的线程最终都会进入临界区。我们用一阶线性时序逻辑将该性质形式化为 φ = (∀x □◊ scheduled(x)) → (∀x □(waiting(x) →◊critical(x)))，其中 □ 表示“全局”，◊ 表示“最终”。要验证该性质，我们需要证明系统的任何无限轨迹都不满足 φ 的否定。为此，我们考虑一个增广转移系统，它将原系统的无限轨迹限制为满足斯科伦化否定性质 sk(¬φ) = ∀x.□◊scheduled(x) ∧ ◊(waiting(x_0) ∧□¬ critical(x_0)) 的轨迹。验证问题由此归结为证明该增广转移系统不存在无限轨迹，也就是说，该系统会终止。

增强转移系统通过将原始系统与针对 sk(¬φ) 的计时器迁移系统组合而自动构造。一般性的构造见第5.2节。这里我们只说明构造的要点。增强转移系统包含一个计时器变量 $t_{\text{starved}} \in \mathbb{N} \cup \{\infty\}$ 和一个计时器函数 t_{sched} : 线程 → $\mathbb{N} \cup \{\infty\}$。变量 t_{starved} 充当预言变量，用于预测 waiting(x_0) ∧ □¬ critical(x_0) 下一个成立还需多少步。当 t_{starved} 达到 0 时，表示 x_0 将永远无法进入临界区。函数 t_{sched}(x) 预测线程 x 下次被调度还需多少步。该构造确保计时器根据这一含义更新：若其值为正数，则每一步都递减；若其值为无穷大，则保持无穷大；若其值为 0，则确保公式成立。由于我们想要保证 sk(¬φ)，增强转移系统会将 t_{starved} 初始化为某个自然数，随后它在系统的每一步中递减。根据自然数的性质，这意味着它会在有限步后变为 0。此外，为了保证公平调度，对于每个线程 x，该构造确保 t_{sched}(x) 无限多次等于 0。具体做法是先将其初始化为某个自然数，并在其达到 0 时将其重置为任意正数。总之，这保证了增强系统的任意无限轨迹

都满足 sk($\neg\varphi$)（无需任何公平性假设）。接下来只需证明增强系统会终止。

在本节余下部分中，我们将说明如何利用计时器证明增强系统的终止性。我们强调，在证明中引入计时器时，用户可以依赖其直观含义，无需担心增强系统的构造。我们的终止性证明基于一个排序函数，该函数将系统状态映射为某个良基集合中的排序值，并且该值在每次转换中都会减小。不过，我们不显式编码该函数，而是采用一种扩展的隐式排序概念 [28]，它能编码两个状态之间排序值的减小，并且可以利用我们提供的构造器自然地定义（参见第5.2节）。受篇幅限制，我们只展示底层的排序函数。鉴于我们扩展了隐式排序，使其能够引入论域无限的计时器，因此使用我们的构造器将其编码为隐式排序相当直接。

我们通过考察（增强）系统可能发生的转换来构造一个字典序排序函数。排序的第一个分量是 t_{starved}。这个计时器会反复递减，直到变为 0；此后它保持为 0，而 x_0 则无法进入临界区。排序的后续分量用于证明从这一时刻起，轨迹的其余部分是有限的。为此，我们注意到，由于 x_0 始终处于等待状态，其票号值 myt(x_0) 永远不会改变，因此差值 myt(x_0) –serv 永远不会增加。当线程离开临界状态时，serv 会增加，差值则会减小。因此，我们将该差值用作排序的下一个分量。接下来，当没有线程处于临界区时，我们会等待某个线程进入。因此，排序的第三个分量表示是否有线程处于临界区。最后，被调度线程可能既不在临界区，也无法进入临界区。为确保这种情况下排序值会减小，我们利用计时器，并引入排序的第四个分量；对于持有当前正在处理的票号、且下一个将进入临界区的线程 x，该分量按照 $t_{\mathrm{sched}}(x)$ 递减。将所得排序编码为隐式排序后，SMT 求解器可以自动验证增强迁移系统中每个（可达）步骤的排序值都会减小，从而确保终止性。

3 预备知识

一阶逻辑。 我们使用带等式的多排序一阶逻辑。一个签名 Σ 由一个类型集合（记作 sorts(Σ)）、带排序的关系符号、常量符号和函数符号组成。项是常量符号、变量或函数应用。我们用粗体表示变量序列 $\boldsymbol{x}$ 或项序列 $\boldsymbol{t}$。原子公式是 $t_1 = t_2$ 或关系应用 $R(\boldsymbol{t})$。非原子公式由逻辑联结词 $\neg$、$\wedge$、$\vee$、$\rightarrow$ 以及带排序量词 $\forall$、$\exists$构成。对于公式 α（或项），我们用 $\alpha(\boldsymbol{x})$ 表示其自由变量包含在 $\boldsymbol{x}$ 中；对于项序列 $\boldsymbol{t}$，我们用 $\alpha(\boldsymbol{t})$ 表示同时替换 $\alpha[\boldsymbol{x} \mapsto \boldsymbol{t}]$。

一阶公式在结构与赋值对上进行求值。一个 Σ-结构是一个二元组 $s = (D,\ I)$，其中 D 将每个类型 σ 映射到其论域 $D(\sigma)$，

该论域是一个非空集合，而 I 是Σ中所有符号的解释函数。我们用 struct(Σ, D) 表示 D 上的所有结构构成的集合，用 struct(Σ) 表示所有 Σ-结构构成的类。从有排序变量 $\boldsymbol{x}$ 到D 的有排序赋值 v 是一个函数，它将 $\boldsymbol{x}$中类型为 σ 的每个变量 x_i 映射到 $D(\sigma)$中的一个元素。我们用 assign($\boldsymbol{x}, D$) 表示从 $\boldsymbol{x}$ 到 D 的所有赋值构成的集合。对于定义在互不相交的变量序列 $\boldsymbol{x}$、$\boldsymbol{y}$上的两个赋值 u、v，我们用 $u{\cdot}v$表示定义在 $\boldsymbol{x}{\cdot}\boldsymbol{y}$ 上的赋值；对于任意 i、j，该赋值满足 $u{\cdot}v(x_i) = u(x_i)$和 $u{\cdot}v(y_j) = v(y_j)$。对于公式 α，以及结构 s 与对 α的自由变量赋值 v 构成的一对 (s, v)，我们写作 $(s, v) \models \alpha$，表示 (s, v) 满足 α。

对于签名 Σ，我们用 Σ' 表示 Σ 的一个不相交副本，其定义为 $\Sigma' = \{a' \mid a \in \Sigma\}$。对于变量序列 $\boldsymbol{x} = (x_i)_{i=1}^m$，我们用 $\boldsymbol{x}'$ 表示序列 $(x_i')_{i=1}^m$。对于论域 Σ 上的公式 α（或项 t），我们用 α' 表示将每个符号 $a \in \Sigma$ 替换为 $a' \in \Sigma'$ 后得到的、论域为 Σ' 的公式。稍微滥用一下记号，我们有时将一个 Σ-结构 $s = (D, I)$ 视为一个 Σ'-结构，反之亦然，方法是令 $I(a') = I(a)$。类似地，我们有时将对 $\boldsymbol{x}$ 的赋值视为对 $\boldsymbol{x}'$ 的赋值，反之亦然，方法是令 $v(x_i') = v(x_i)$。对于论域 $\Sigma \uplus \Sigma'$ 上自由变量为 $\boldsymbol{x}, \boldsymbol{x}'$ 的公式 τ，以及共享论域上的结构 s, s'，我们用记号 $(s, v), (s', v') \models \tau$ 表示：τ 在结构—赋值对下成立，其中论域为 s, s' 的共享论域，$\Sigma, \boldsymbol{x}$ 按照 (s, v) 解释，而 $\Sigma', \boldsymbol{x}'$ 按照 (s', v') 解释。如果 τ 是闭公式，我们省略赋值，并写作 $s, s' \models \tau$。

用一阶逻辑描述迁移系统。签名为 Σ 的迁移系统记作 $\mathcal{T}(\mathcal{S})$，由一对一阶规约 $\mathcal{T}$ 和预期语义 $\mathcal{S}$ 给出。一阶规约由 $\mathcal{T} = (\Gamma, \iota, \tau)$ 给出，其中 Γ 是 Σ 上的闭合公式，充当背景理论；ι 是 Σ 上的闭合公式，用于定义系统的初始状态；τ 是 $\Sigma \uplus \Sigma'$ 上的闭合公式，用于定义系统可能发生的转换。预期语义 $\mathcal{S}$ 是一类 Σ-结构。$\mathcal{T}(\mathcal{S})$ 的状态由 $\mathcal{S}$ 中满足 Γ 的 Σ-结构给出。$\mathcal{T}(\mathcal{S})$ 的轨迹由状态序列 $\pi = (s_i)_{i=0}^n$ 给出，其中 $n \in \mathbb{N} \cup \{\infty\}$，使得序列中的所有状态具有相同的论域，$s_0 \models \iota$，并且对于每个 $i \leq n$，都有 $s_i, s_{i+1} \models \tau$。如果存在有限轨迹 $\pi = (s_i)_{i=0}^n$，其中 $n \in \mathbb{N}$ 且 $s_n = s$，则状态 s 是可达的。如果 s 可达，且 $s, s' \models \tau$，则一对状态 s, s' 是可达迁移。若 $\mathcal{T}(\mathcal{S})$ 没有无限轨迹，我们就称其会终止。

一阶线性时序逻辑。 我们使用 FO-LTL 来描述迁移系统的时序性质。给定一个一阶签名 Σ，FO-LTL 公式的定义与一阶公式类似，只是增加了时序算子。为简便起见，我们仅考虑时序算子“全局” $\Box$ 和“最终” $\Diamond$，我们的方法也可以轻松扩展到算子“下一个” $\bigcirc$和“直到” $\mathcal{U}$。FO-LTL 公式 ψ 的语义定义在一对 (π, v) 上，其中 π 是由共享论域上的 Σ- 结构构成的无限序列 $\pi = (s_i)_{i=0}^\infty$，而 v 是对 ψ的自由变量的赋值。对于 $\pi = (s_i)_{i=0}^\infty$ 和$k \in \mathbb{N}$，我们记作 $\pi^k = (s_{i+k})_{i=0}^\infty$。对于 ψ 和上述的 (π, v)，FO-LTL 的语义如下：若 ψ是原子命题，则当且仅当 $s_0, v \models \psi$时，有 $\pi, v \models \psi$，在

一阶语义下，π，$v \models \Box\psi$ 当且仅当 π^k，$v \models \psi$ 对每个 $k \in \mathbb{N}$ 都成立；π，$v \models \Diamond\psi$ 当且仅当存在某个 $k \in \mathbb{N}$ 使得 π^k，$v \models \psi$ 成立；一阶算子则按通常方式解释。时序性质 φ 由一个闭合的 FO-LTL 公式给出。迁移系统 $\mathcal{T}(\mathcal{S})$ 满足时序性质 φ，记作 $\mathcal{T}(\mathcal{S}) \models \varphi$，当且仅当它的所有无限轨迹都满足 φ。

4 计时器

本节介绍如何将一阶线性时序逻辑性质的验证规约为终止性验证。首先，在第 4.1 节中，我们介绍时序性质 φ 的计时器迁移系统，并证明关于它的两个特征事实。随后，在第 4.2 节中，我们介绍该规约；该规约使用待验证的取反时序性质的计时器迁移系统。

4.1 计时器迁移系统

时序性质 φ 的计时器迁移系统是一种构造出来的迁移系统，使得 (i) 它的所有无限轨迹都满足 φ，且 (ii) 它包含所有满足 φ 的无限轨迹。我们通过引入一种新的类型来构造计时器迁移系统，该类型记为 σ_{time}，其预期语义为扩展自然数 $\mathbb{N}\cup\{\infty\}$；此外，对每个子公式 ψ，我们都添加一个函数符号 t_ψ，称为 ψ 的计时器。

形式化地，对于签名 Σ 上的一阶线性时序逻辑性质 φ，我们用 $A(\varphi)$ 表示包含 φ 的所有子公式，并且对每个子公式 $\Box\psi$，还包含 $\neg\psi$ 的公式集合。计时器迁移系统 $\mathcal{T}_\varphi(\mathcal{S}_{\text{time}})$ 定义在签名 Σ_φ 上，该签名具有类型 $\text{sorts}(\Sigma_\varphi) = \text{sorts}(\Sigma) \cup\{\sigma_{\text{time}}\}$，以及签名 $\Sigma_\varphi = \Sigma \cup \{t_\psi \mid \psi \in A(\varphi)\} \cup \{\leq, 0, \infty, -1\}$；其中，对于 $\psi(\boldsymbol{x}) \in A(\varphi)$，$t_\psi$ 是一个从 $\boldsymbol{x}$ 的类型映射到 σ_{time} 的函数符号；$\leq$ 是 σ_{time} 上的二元关系符号；0 和 ∞ 是类型为 σ_{time} 的常量符号；而 -1 是一个从 σ_{time} 映射到 σ_{time} 的函数符号。计时器的预期语义 $\mathcal{S}_{\text{time}}$ 由 Σ_φ 的结构类给出，其中 σ_{time} 的论域为 $\mathbb{N}\cup\{\infty\}$，并对 $\leq$、0、∞ 和 -1 采用标准解释。$\mathcal{T}_\varphi$ 的公理和迁移公式定义为：对于任意 $\psi \in A(\varphi)$，$t_\psi(\boldsymbol{x})$ 表示满足 $\psi(\boldsymbol{x})$ 所需的转换次数。最后，初始状态公式确保 $t_\varphi = 0$ 在所有初始状态中成立，这将意味着 φ 在每条轨迹中都成立。

定义 1. 设 φ 是签名为 Σ 的一阶线性时序逻辑性质，并定义上述 $A(\varphi)$、Σ_φ 和 $\mathcal{S}_{\text{time}}$。$\varphi$ 的计时器迁移系统是一个在 Σ_φ 上定义的迁移系统 $\mathcal{T}_\varphi(\mathcal{S}_{\text{time}})$，其中 $\mathcal{T}_\varphi = (\Gamma_\varphi, \iota_\varphi, \tau_\varphi)$ 定义如下：

$$\Gamma_\varphi = \bigwedge_{\psi(\boldsymbol{x})\in A(\varphi)} \forall \boldsymbol{x}\ ((t_\psi(\boldsymbol{x}) = 0) \leftrightarrow \zeta_\psi(\boldsymbol{x})) \qquad \iota_\varphi = (t_\varphi = 0)$$

$$\tau_\varphi = \bigwedge_{\psi(\boldsymbol{x})\in A(\varphi)} \forall \boldsymbol{x} \left(\begin{array}{l} (0 < t_\psi(\boldsymbol{x}) < \infty \rightarrow t'_\psi(\boldsymbol{x}) = t_\psi(\boldsymbol{x}) - 1) \wedge \\ (t_\psi(\boldsymbol{x}) = \infty \rightarrow t'_\psi(\boldsymbol{x}) = \infty) \wedge \\ (t_\psi(\boldsymbol{x}) = 0) \leftrightarrow \eta_\psi(\boldsymbol{x}) \end{array} \right)$$

表1. ζ_ψ， η_ψ 一阶线性时序逻辑公式 $\psi(\boldsymbol{x})$ 的定义

$\psi(\boldsymbol{x})$	ζ_ψ	η_ψ
$R(\boldsymbol{x})$	$R(\boldsymbol{x})$	
$\neg\rho$	$t_\rho(\boldsymbol{x}) \neq 0$	
$\rho \circ \chi$ 对于 $\circ \in \{\wedge, \vee, \rightarrow\}$	$(t_\rho(\boldsymbol{x}) = 0) \circ (t_\chi(\boldsymbol{x}) = 0)$	
$Qy\ \rho$ 对于 $Q \in \{\forall, \exists\}$	$Qy\ (t_\rho(\boldsymbol{x} \cdot y) = 0)$	
$\Diamond\rho$	$t_\rho(\boldsymbol{x}) < \infty$	$t_\rho(\boldsymbol{x}) = 0 \vee t'_{\Diamond\rho}(\boldsymbol{x}) = 0$
$\Box\rho$	$t_{\neg\rho}(\boldsymbol{x}) = \infty$	$t_\rho(\boldsymbol{x}) = 0 \wedge t'_{\Box\rho}(\boldsymbol{x}) = 0$

其中 ζ_ψ 和 η_ψ 根据表*1*定义。如果 η_ψ 未定义，则从 τ_φ中省略引用它的合取项。

例*1.* 考虑 $\varphi = \forall x.\Box\Diamond\text{sched}(x)$。 $\mathcal{T}_\varphi$ 中的计时器为 $t_{\forall x.\Box\Diamond\text{sched}(x)}$,$t_{\Box\Diamond\text{sched}(x)}(x)$ $t_{\neg\Diamond\text{sched}(x)}(x)$ $t_{\Diamond\text{sched}(x)}(x)$ $t_{\text{sched}(x)}(x)$。在 $\mathcal{T}_\varphi$ 的初始状态中，我们有 $t_\varphi = 0$，因此由 Γ_φ 可得，在所有初始状态中都有 $\forall x.t_{\Box\Diamond\text{sched}(x)}(x) = 0$ 。由 τ_φ 可知，在所有可达状态中都有 $\forall x.t_{\Diamond\text{sched}(x)}(x) = 0$ 。根据为 $\Diamond$生成的公理，对于任意线程 x，在任意可达状态中都有 $t_{\text{sched}(x)}(x) < \infty$。如果 $t_{\text{sched}(x)}(x)$，由τ_φ 中计时器的递减可知，该计时器会在有限步内减至 0。因此，对于任意线程 x ，在任意轨迹中，$t_{\text{sched}(x)}(x) = 0$ 都会无限次成立，从而保证公平调度。也就是说， $\mathcal{T}_\varphi$ 的每条无限轨迹都满足 φ。

备注*1*。 计时器迁移系统的构造与表格法构造 [33, 39]类似。我们没有像后者那样使用时序公式集合来表示状态，而是为状态添加计时器。在表格法构造中，公平性约束用于过滤掉不理想的轨迹，例如所有状态都被标记为 $\Diamond p$ 的无限轨迹，而这种标记依据的是某个未来对 p的满足，但这个满足永远不会到来。相反，在我们的构造中，我们依靠计时器的预期语义来避免这类情况。由于这类轨迹会导致计时器值无限递减，而自然数是良基的，因此这类轨迹并不存在。

备注*2*。 为了处理任意 FO-LTL 公式，我们还会考虑时序算子 $\bigcirc$和 $\mathcal{U}$。为了处理时序子公式 $\bigcirc\rho$，我们会将断言 $t_{\bigcirc\rho} = 0 \leftrightarrow t'_\rho = 0$合取到 τ_φ 中。为了处理时序子公式$\chi\mathcal{U}\rho$，我们会将断言 $t_{\chi\mathcal{U}\rho} = 0 \rightarrow t_\rho < \infty$合取到 Γ_φ 中，并将断言 $t_{\chi\mathcal{U}\rho} = 0 \leftrightarrow t_\rho = 0 \vee (t_\chi = 0 \wedge t'_{\chi\mathcal{U}\rho} = 0)$合取到$\tau_\varphi$ 中。

我们的第一个引理指出，对于任意 $\psi \in A(\varphi)$，计时器的值为 0 表示相应时序公式已满足。在 ι_φ 中，我们要求 $t_\varphi = 0$，因此 $\mathcal{T}_\varphi(\mathcal{S}_{\text{time}})$ 的每条无限轨迹都满足 φ。

引理 *1*。设 $\hat{\pi} = (\hat{s}_i)_{i=0}^{\infty}$ 是 $\mathcal{T}_\varphi(\mathcal{S}_{\text{time}})$ 的一条轨迹，其论域为 D, $\psi(\boldsymbol{x}) \in A(\varphi)$,$v \in$ 赋值$(\boldsymbol{x}, D)$ 和 $i \in \mathbb{N}$。那么 $\hat{s}_i, v \models (t_\psi(\boldsymbol{x}) = 0)$ 当且仅当 $\hat{\pi}^i, v \models \psi(\boldsymbol{x})$。

下面的引理表明，任何满足 φ 的状态序列都可以扩展为 $\mathcal{T}_\varphi(\mathcal{S}_{\text{time}})$ 的一条轨迹。这实质上意味着， $\mathcal{T}_\varphi(\mathcal{S}_{\text{time}})$ 展示了满足 φ 的所有方式。引理的证明通过说明我们总能用计时器的“自然定义”扩展轨迹来完成：将 $t_\psi(\boldsymbol{x})$ 求值为使该序列的后缀满足 ψ 所需经过的最少转换次数；如果不存在这样的后缀，则取 ∞ 。

引理 2. 给定一个共享论域上的 Σ-结构序列 $\pi = (s_i)_{i=0}^{\infty}$ ，满足 $\pi \models \varphi$，则存在一条 $\mathcal{T}_\varphi(\mathcal{S}_{\text{time}})$ 的轨迹 $\hat{}\pi = (\hat{s}_i)_{i=0}^{\infty}$ ，使得 $\hat{}\pi|_\Sigma = \pi$。

注释 *3.* 虽然引理 2 的证明表明，我们总是可以根据轨迹中时序公式的满足情况，为轨迹补充计时器的自然定义，但引理 1 仅指出， $\mathcal{T}_\varphi(\mathcal{S}_{\text{time}})$ 的轨迹会对取值为 0 的计时器强制满足自然定义。实际上，我们可以证明，对于任意计时器值 $k \in \mathbb{N} \cup \{\infty\}$，同样的结论也成立：对于轨迹 $\hat{}\pi = (\hat{s}_i)_{i=0}^{\infty}$ 和赋值 v，对于 $k \in \mathbb{N}$ ，$\pi^i, v \models (t_\psi(\boldsymbol{x}) = k)$ 当且仅当 $\pi^{i+k}, v \models \psi$；并且 $\pi^i, v \models (t_\psi(\boldsymbol{x}) = \infty)$ 当且仅当 $\pi^{i+j}, v \not\models \psi$，对所有 $j \in \mathbb{N}$ 均成立。这是因为 $\mathcal{T}_\varphi(\mathcal{S}_{\text{time}})$ 要求取值为 ∞ 的计时器在转换后仍保持为 ∞。由于我们在引理 1 的证明中并未使用这一要求，而且去掉它只会为 $\mathcal{T}_\varphi(\mathcal{S}_{\text{time}})$ 增添轨迹，因此不会影响引理 2，所以我们可以从 $\mathcal{T}_\varphi(\mathcal{S}_{\text{time}})$ 的定义中省略这一要求，而不影响下一小节所述规约的可靠性和完备性。

4.2 计时器规约

将 FO-LTL 性质的满足性规约为终止性，本质上是将 $\mathcal{T}(\mathcal{S})$ 与被否定性质$\mathcal{T}_{\neg\varphi}(\mathcal{S}_{\text{time}})$的计时器迁移系统组合起来。形式化地，对于两个定义在签名 Σ_1, Σ_2 上的迁移系统，其规约分别为 $\mathcal{T}_1 = (\Gamma_1, \iota_1, \tau_1)$、 $\mathcal{T}_2 = (\Gamma_2, \iota_2, \tau_2)$，预期语义分别为 $\mathcal{S}_1, \mathcal{S}_2$，我们将它们的不交并不一定成立的签名 $\Sigma_1 \cup \Sigma_2$ 上的乘积迁移系统定义如下。乘积迁移系统记作 $(\mathcal{T}_1 \times \mathcal{T}_2)(\mathcal{S}_1 \times \mathcal{S}_2)$，其定义满足 $\mathcal{T}_1 \times \mathcal{T}_2 = (\Gamma_1 \wedge \Gamma_2, \iota_1 \wedge \iota_2, \tau_1 \wedge \tau_2)$，且$\mathcal{S}_1 \times \mathcal{S}_2$ 包含任意满足 $s|_{\Sigma_1} \in \mathcal{S}_1$ 和 $s|_{\Sigma_2} \in \mathcal{S}_2$ 的 $(\Sigma_1 \cup \Sigma_2)$-结构 s 。在我们的情形中， $\mathcal{T}(\mathcal{S})$ 和 $\mathcal{T}_{\neg\varphi}(\mathcal{S}_{\text{time}})$ 分别定义在签名 Σ, Σ_φ 上，且满足 $\Sigma \subseteq \Sigma_\varphi$。因此，乘积迁移系统定义在 Σ_φ 上，可以理解为用 $\mathcal{T}_{\neg\varphi}(\mathcal{S}_{\text{time}})$ 的计时器变量及语义扩充 $\mathcal{T}(\mathcal{S})$。乘积系统 $\mathcal{T}(\mathcal{S}) \times \mathcal{T}_{\neg\varphi}(\mathcal{S}_{\text{time}})$ 包含 $\mathcal{T}(\mathcal{S})$ 中满足 $\neg\varphi$ 的那些轨迹，因此， $\mathcal{T}(\mathcal{S})$ 满足该性质，当且仅当乘积系统终止。以下定理概述了规约为终止性在可靠性和完备性方面的结果。

定理 1。 对于迁移系统 $\mathcal{T}(\mathcal{S})$ 和一阶线性时序逻辑性质 φ ，我们有$\mathcal{T}(\mathcal{S}) \models \varphi$ 当且仅当 $\mathcal{T}(\mathcal{S}) \times \mathcal{T}_{\neg\varphi}(\mathcal{S}_{\text{time}})$ 终止。

5 排序

本节讨论如何扩展并使用 [28] 中的隐式排序框架，以证明转换系统的终止性。第 5.1 节给出

隐式排序的定义，并将其推广为由系统可达状态上的显式可靠性条件参数化的形式，同时说明如何在终止性证明中使用隐式排序。第5.2节介绍基于 [28], 构造器的隐式排序递归构造，并将其扩展为生成可靠性条件，还展示如何利用计时器构造隐式排序。在第 5.3 节，我们将说明如何通过一阶逻辑中的有效性检查，履行基于隐式排序的终止性证明所需的证明义务。

5.1 使用隐式排序验证终止性

我们首先介绍扩展后的封闭隐式排序定义，这类排序用于证明。接着，我们给出不一定封闭的隐式排序定义，这类排序可用于递归构造。在本节余下部分，我们固定一个一阶签名 Σ。为简化记号，我们假设 Σ 是单排序的。推广到我们在实践中使用的多排序签名是直接的。封闭隐式排序是一个闭合公式 $\tau_>$ ，它编码了某个隐式排序函数相对于某个隐式序的递减。公式 $\tau_>$ 定义在双重签名 $\Sigma \uplus \Sigma'$ 上，其中排序在前状态（由 Σ表示）与后状态（由 Σ'表示）之间递减。在 [28], 中，要求隐式序是良基的。我们放宽这一要求，并扩展隐式排序的概念，将状态上的可靠性条件 $\mathcal{C}$ 纳入其中；这样，只需保证隐式序在满足 $\mathcal{C}$ 的状态上由排序函数得到的像是良基的。我们使用可靠性条件来指定保证隐式序良基所必需、但无法直接用一阶逻辑定义的要求。我们使用的可靠性条件包括：要求某个元素集合是有限的，以及要求某个状态关系是良基的。

定义 2（封闭隐式排序）。给定一个关于$\Sigma \uplus \Sigma'$ 的闭合公式 $\tau_>$ 以及一类 Σ-结构 $\mathcal{C}$, 如果对任意论域 D，都存在一个偏序集 $(A, <)$ 和一个函数 $f : \mathrm{struct}(\Sigma, D) \to A$, 使得以下条件成立，则称 $\tau_>$ 是满足可靠性条件 可靠性条件 $\mathcal{C}$ 的封闭隐式排序：

1. 对每个 s， $s' \in \mathsf{struct}(\Sigma,\ D)$ 都有 s， $s' \models \tau_> \implies f(s) > f(s')$
2. $<$ 限制在 $\{f(s) \mid s \in \mathcal{C}\}$ 上是良基的。

我们称$(A,\ <)$ 为 D 的排序值域，并称 f 为 D的排序函数。

我们注意到，在终止性证明中使用隐式排序时，只需要求条件 1（甚至只需要求 f 有定义）对 $\mathcal{C}$中的结构成立即可，但这并不能帮助我们构造有用的隐式排序。

我们使用带有可靠性条件 $\mathcal{C}$ 的封闭隐式排序 $\tau_>$ ，通过证明迁移系统 $\mathcal{T}(\mathcal{S})$ 的所有可达状态都符合可靠性条件 $\mathcal{C}$ ，且所有可达迁移都满足 $\tau_>$ ，来证明该系统 $\mathcal{T}(\mathcal{S})$具有终止性。如果系统存在一条无限

轨迹，就意味着存在一个在良基序中无限递减的序列。我们在以下定理中将其形式化：

定理2。 如果 $\mathcal{T}(\mathcal{S})$ 的每个可达状态 s 都满足 $s \in \mathcal{C}$，并且 $\mathcal{T}(\mathcal{S})$ 的每个可达转换 (s, s') 都满足 $s, s' \models \tau_{>}$，那么 $\mathcal{T}(\mathcal{S})$ 会终止。

在第 5.3 节中，我们将讨论如何在实践中验证定理2的要求。在此之前，我们先介绍隐式排序的构造。与 [28], 类似，我们注意到，可以通过识别有用的基本构件并将它们组合起来，构造隐式排序。为了支持将隐式排序用作构建模块、递归构造复杂隐式排序， [28] 从两个方面推广了封闭隐式排序的定义。首先，隐式排序可以包含自由变量，称为参数；在这种情况下，每个论域对应的底层（隐式）排序函数定义在结构与自由变量赋值构成的有序对上。参数有助于构造能够捕获论域上聚合的隐式排序。其次， [28] 中的隐式排序还包括公式 $\tau_{\geq}$，用于近似排序值域上的弱偏序 $\leq$。这样，递归构造也可以依赖于 $\leq$。我们还引入第三项推广，即公式 $\kappa_{\min}$，用于近似判断一个元素是否具有最小排序。这项推广用于定义通过无限域上的聚合构造的隐式排序的可靠性条件。

定义 3（隐式排序）。一个带有参数 $\boldsymbol{x}$和可靠性条件 $\mathcal{C}$ 的隐式排序是一个三元组$\mathrm{Rk} = (\tau_{>}, \tau_{\geq}, \kappa_{\min})$，其中 $\tau_{>}, \tau_{\geq}$ 是关于 $\Sigma \uplus \Sigma'$ 的公式，其自由变量为 $\boldsymbol{x}, \boldsymbol{x}'$，$\kappa_{\min}$ 是关于 Σ 的公式，其自由变量为 $\boldsymbol{x}$，而 $\mathcal{C}$ 是关于 Σ的结构类，使得对于任意论域D，都存在一个偏序集$(A, <)$和一个函数 $f\colon \mathrm{struct}(\Sigma, D) \times \mathrm{assign}(\boldsymbol{x}, D) \to A$，使得以下条件成立：

1. $(s, v), (s', v') \models \tau_{>}(\boldsymbol{x}, \boldsymbol{x}') \implies f(s, v) > f(s', v')$ 2. $(s, v), (s', v') \models \tau_{\geq}(\boldsymbol{x}, \boldsymbol{x}') \implies f(s, v) \geq f(s', v')$ 3. $(s, v) \models \kappa_{\min} \implies f(s, v)$ 在$(A, <)$中是极小的 4. $<$限制于 $\{f(s, v) \mid s \in \mathcal{C}, v \in \mathrm{assign}(\boldsymbol{x}, D)\}$ 是良基的。

其中，条件 *1*、*2*、*3* 的取值范围是任意 (s, v)、$(s', v') \in \mathrm{struct}(\Sigma, D) \times \mathrm{assign}(\boldsymbol{x}, D)$。我们称 $(A, <)$ 为 D 的排序值域，称 D为 f 的排序函数。

与封闭隐式排序一样，我们可以放宽要求，只要求条件 1、2、3 对 $s, s' \in \mathcal{C}$ 成立，但事实证明这并没有什么用处。

5.2 隐式排序的构造器

我们重新审视了 [28]的构造器。对于每个构造器，我们还明确规定了所得隐式排序中 $\mathcal{C}$ 和 $\kappa_{\min}$ 的定义。此外，我们利用本文定义的隐式排序可靠性条件，将 [28], 中仅对具有有限域的类型可靠的有限域构造器推广到允许较弱的有限性假设。我们仅介绍其中一部分

构造器，这些构造器用于展示计时器在排序中的用法，且具有非平凡的可靠性条件；其余构造器留待附录 B 介绍。

定理 3. 所介绍的所有构造器都是可靠的，也就是说，只要它们的参数满足相应的假设，就会输出具有适当可靠性条件的隐式排序。

我们介绍的第一个构造器依赖系统中域元素之间已有的序，来定义一个隐式排序，其底层排序函数相对于该序递减。该序由关系符号 $\ell(y_1, y_2)$ 指定，并要求满足公式 order(ℓ)；该公式规定 ℓ 是不可变的，并定义一个严格偏序。这个隐式排序的可靠性条件是由 ℓ 定义的序为良基。我们将良基性作为可靠性条件，是因为无法用一阶逻辑加以描述。虽然这看起来像是在推迟问题，而不是解决问题，但我们通常能够利用迁移系统的预期语义，例如计时器 $\mathcal{S}_{\text{time}}$ 的预期语义，来确保可靠性条件成立，而无需显式地对其进行编码（参见第 5.3 节）。

构造器 1。 位置构造器 接收一个关于 Σ 且带有自由变量 $\boldsymbol{x}$ 的项 t 以及一个二元关系符号 $\ell \in \Sigma$，返回一个隐式排序 $\text{Pos}(t, \ell) = (\tau_>, \tau_\geq, \kappa_{\min})$，其参数为 $\boldsymbol{x}$，可靠性条件 $\mathcal{C}$ 定义如下：

$$\tau_>(\boldsymbol{x}, \boldsymbol{x}') = \text{order}(\ell) \wedge \ell(t'(\boldsymbol{x}'), t(\boldsymbol{x}))$$
$$\tau_\geq(\boldsymbol{x}, \boldsymbol{x}') = \text{order}(\ell) \wedge (\ell(t'(\boldsymbol{x}'), t(\boldsymbol{x})) \vee t'(\boldsymbol{x}') = t(\boldsymbol{x}))$$
$$\kappa_{\min}(\boldsymbol{x}) = \forall y. \neg\ell(y, t(\boldsymbol{x})) \qquad \mathcal{C} = \{s \in \text{struct}(\Sigma) \mid \ell \text{ is well-founded in } s\}$$

示例 2。 考虑第 2 节中的票据协议，在为系统添加计时器之后，我们按照如下方式在隐式排序中使用计时器和位置构造器。关系 ℓ 是 σ_{time} 上的序关系 $<$。取 $t = t_{\text{waiting}(x_0) \wedge \Box\neg\text{critical}(x_0)}$，我们得到捕获 t_{starved} 递减的封闭隐式排序。取 $t(x) = t_{\Box\Diamond\text{scheduled}(x)}(x)$，我们得到一个带参数 x 的隐式排序，需要对其进行聚合才能得到封闭隐式排序。

下一个构造器接收一个隐式排序 Rk$^\circ$ 和一个公式 α 作为输入，并返回一个细化给定隐式排序的隐式排序：仅当 α 满足时，状态才基于 Rk$^\circ$ 进行排序；不满足 α 的状态的排序低于满足该条件的状态。因此，根据此排序，不满足 α 的每个状态都是极小的。当我们只能保证在 α 满足时 Rk$^\circ$ 的保持性（$\leq$）时，此构造器非常有用。

构造器 2。 条件构造器 接受一个隐式排序 $\text{Rk}^\circ = (\tau_>^\circ、\ \tau_\geq^\circ、\ \kappa_{\min}^\circ)$，其参数为 $\boldsymbol{x}$，可靠性条件为 $\mathcal{C}^\circ$，以及一个公式 $\alpha(\boldsymbol{x})$。它返回一个隐式排序 $\text{Cond}(\text{Rk}^\circ、\ \alpha) = (\tau_>、\ \tau_\geq、\ \kappa_{\min})$，其参数为 $\boldsymbol{x}$，可靠性条件为 $\mathcal{C}$，定义如下：

$$\tau_>(\boldsymbol{x}, \boldsymbol{x}') = (\alpha(\boldsymbol{x}) \wedge \neg\alpha'(\boldsymbol{x}')) \vee (\alpha(\boldsymbol{x}) \wedge \alpha'(\boldsymbol{x}') \wedge \tau_>^\circ(\boldsymbol{x}, \boldsymbol{x}'))$$
$$\tau_\geq(\boldsymbol{x}, \boldsymbol{x}') = \neg\alpha'(\boldsymbol{x}') \vee (\alpha(\boldsymbol{x}) \wedge \alpha'(\boldsymbol{x}') \wedge \tau_\geq^\circ(\boldsymbol{x}, \boldsymbol{x}')) \quad \kappa_{\min}(\boldsymbol{x}) = \neg\alpha(\boldsymbol{x}) \quad \mathcal{C} = \mathcal{C}^\circ$$

例 *3*。 在例 2 中，我们展示了如何在隐式排序中使用计时器 $t_{\Box\Diamond \mathrm{scheduled}(x)}(x)$。在票据协议中，每个线程都会被无限次调度，因此上述计时器并不总能保持不变。此外，线程被调度并不一定会改变状态，例如，它尝试进入临界区但并不持有 serv 时就是如此。为此，我们只希望对持有当前所服务票据的线程考虑上述计时器。这由隐式排序 Cond(Pos($t_{\Box\Diamond \mathrm{scheduled}(x)}(x) <$)、myt($x$) = serv) 表示。

我们接下来重新介绍 [28] 的基于论域的构造器。这些构造器聚合由域元素参数化的排序。在 [28] 中，这些构造器仅在论域有限时才被定义为可靠。这里，我们稍微放宽这一假设，而是将其纳入可靠性条件，只要求基础排序中非最小元素的集合有限。由于最小值不会减少，非最小值有限就足以确保聚合排序具有良基性。

论域逐点构造器将排序值域为 $(P, <_P)$ 的隐式排序 Rk$^\circ$ 提升为一种隐式排序，其排序值域为函数集$Y \to P$，其中 Y 可以是无限的。集合 $Y \to P$ 按逐点序排序：当且仅当存在 $y \in Y$ 使得 $a(y) <_P b(y)$，并且对于每个 $y \in Y$，都有 $a(y) \leq_P b(y)$ 时，$a <_{\mathrm{pw}} b$。如果 $<_P$ 是良基的，即使 Y 是无限的，我们也可以将生成的偏序 $<_{\mathrm{pw}}$ 限制为良基，只考虑满足以下条件的函数 $a \in Y \to P$：使 $a(y)$ 为非最小值的 $y \in Y$ 只有有限多个。构造器的可靠性条件体现了这一点。

构造器 3。 论域逐点构造器 接受一个隐式排序$\mathrm{Rk}^\circ = (\tau^\circ_>, \tau^\circ_\geq, \kappa^\circ_{\min})$，其参数为 $\boldsymbol{x} = \boldsymbol{y}{\cdot}\boldsymbol{z}$，可靠性条件为 $\mathcal{C}^\circ$。它返回一个隐式排序 $\mathrm{DomPW}(\mathrm{Rk}^\circ, \boldsymbol{y}) = (\tau_>, \tau_\geq, \kappa_{\min})$，其参数为 $\boldsymbol{z}$，可靠性条件为 $\mathcal{C}$，定义如下：

$$\tau_>(\boldsymbol{z}, \boldsymbol{z}') = \tau_\geq(\boldsymbol{z}, \boldsymbol{z}') \wedge (\exists \boldsymbol{y}.\tau^\circ_>(\boldsymbol{y}{\cdot}\boldsymbol{z}, \boldsymbol{y}{\cdot}\boldsymbol{z}'))$$
$$\tau_\geq(\boldsymbol{z}, \boldsymbol{z}') = \forall \boldsymbol{y}.\tau^\circ_\geq(\boldsymbol{y}{\cdot}\boldsymbol{z}, \boldsymbol{y}{\cdot}\boldsymbol{z}') \qquad \kappa_{\min}(\boldsymbol{z}) = \forall \boldsymbol{y}.\kappa^\circ_{\min}(\boldsymbol{y}{\cdot}\boldsymbol{z})$$
$$\mathcal{C} = \{s \in \mathrm{struct}(\Sigma) \mid s \in \mathcal{C}^\circ \text{ and } \forall v \in \mathrm{assign}(\boldsymbol{z}).\ \mathrm{finite}_{\boldsymbol{y}}(\neg\kappa^\circ_{\min}(\boldsymbol{y}{\cdot}\boldsymbol{z}), s, v)\}$$

where $\mathrm{finite}_{\boldsymbol{y}}(\alpha, s, v) := |\{u \in \mathrm{assign}(\boldsymbol{y}) \mid (s, u{\cdot}v) \models \alpha(\boldsymbol{y}{\cdot}\boldsymbol{z})\}| < \infty$.

例 *4*。 例 3 中构造的隐式排序捕捉了某个线程 x 持有当前服务的 Ticket 时其调度计时器的排序值递减的情况，因此它以 x 作为参数。若我们使用 DomPW(Cond(Pos($t_{\Box\Diamond \mathrm{scheduled}(x)}(x) <$),myt($x$) = serv) x) 对所有此类线程进行汇总，就能考察所有此类线程的调度情况。这是我们在隐式排序中使用计时器的主要模式，因此我们为此引入一种“语法糖”：

$$\mathrm{TimerRank}(\psi(\boldsymbol{x}), \alpha(\boldsymbol{x})) = \mathrm{DomPW}(\mathrm{Cond}(\mathrm{Pos}(t_\psi(\boldsymbol{x}), <), \alpha(\boldsymbol{x}), \boldsymbol{x}).$$

最后一个构造子将排序值域为 $(P, <_P)$ 的隐式排序 Rk$^\circ$ 提升为一种隐式排序，其排序值域为函数集 $Y \to P$，其中 $(Y, <_Y)$ 是一个可能无限的良基集合。该函数集按照

逆字典序排列： $a <_{\text{lex}} b$ 当且仅当对于每个 $y \in Y$，若 $a(y) \not\leq_P b(y)$，则存在 y_0，使得 $y <_Y y_0$ 且 $a(y_0) <_P b(y_0)$。$<_Y$ 的良基性以及 Rk° 中非最小元素集合的有限性，确保了逆字典序的良基性。逆字典序至关重要。例如，考虑 $\{0,1\}^{\mathbb{N}}$ 上的标准字典序，它不是良基的，因为它包含如下无限递减序列：(1，0，.....)，(0，1，0，.....)，(0，0，1，0，.....) .....

构造器 4。定义域字典序构造子接受一个隐式排序 $\text{Rk}^\circ = (\tau_>^\circ, \tau_\geq^\circ, \kappa_{\min}^\circ)$，其参数为 $\boldsymbol{x} = y{\cdot}\boldsymbol{z}$，可靠性条件为 $\mathcal{C}^\circ$，以及论域 Σ 上的关系 $\ell(y_1, y_2)$。它返回一个隐式排序 $\text{DomLex}(\text{Rk}^\circ, y, \ell) = (\tau_>, \tau_\geq, \kappa_{\min})$，其参数为 $\boldsymbol{z}$，可靠性条件 $\mathcal{C}$ 定义如下：

$$\tau_>(\boldsymbol{z}, \boldsymbol{z}') = \tau_\geq(\boldsymbol{z}, \boldsymbol{z}') \wedge (\exists y. \tau_>^\circ(y{\cdot}\boldsymbol{z}, y{\cdot}\boldsymbol{z}'))$$
$$\tau_\geq(\boldsymbol{z}, \boldsymbol{z}') = \text{order}(\ell) \wedge \forall y. (\tau_\geq^\circ(y{\cdot}\boldsymbol{z}, y{\cdot}\boldsymbol{z}') \vee \exists y_0. (\ell(y, y_0) \wedge \tau_>^\circ(y_0{\cdot}\boldsymbol{z}, y_0{\cdot}\boldsymbol{z}')))$$
$$\kappa_{\min}(\boldsymbol{z}) = \forall y. \kappa_{\min}^\circ(y{\cdot}\boldsymbol{z})$$
$$\mathcal{C} = \left\{ s \in \text{struct}(\Sigma) \;\middle|\; \begin{array}{l} s \in \mathcal{C}^\circ, \ell \text{ is well-founded in } s, \\ \forall v \in \text{assign}(\boldsymbol{z}).\ \text{finite}_y(\neg\kappa_{\min}^\circ(y{\cdot}\boldsymbol{z}), s, v) \end{array} \right\}$$

例*5*。考虑一个程序，它以自然数数组$c[0, \ldots, n-1]$作为输入，并在每次迭代中选择一个满足 $c[i] > 0$ 的索引 i，然后执行操作 $c[i] := c[i] - 1;\ c[i-1] = *$。可以证明，该程序会终止，其隐式排序为 $\text{DomLex}(\text{Pos}(c[i], <_{\mathbb{N}})\ i, <_{\mathbb{N}})$。

5.3 验证秩递减和可靠性条件

我们现在重新审视定理2，并说明如何验证其两个前提以证明终止性。固定一个签名 Σ、一个迁移系统 $\mathcal{T}(\mathcal{S})$，其中包含 $\mathcal{T} = (\Gamma, \iota, \tau)$，以及一个带有可靠性条件 $\mathcal{C}$ 的封闭隐式排序 $\tau_>$。我们需要证明，$\mathcal{T}(\mathcal{S})$ 的任意可达状态都满足可靠性条件 $\mathcal{C}$，且任意可达迁移都满足 $\tau_>$。为证明这些前提，我们使用归纳不变式 θ 来对可达状态进行过近似；为此，需要验证 θ 在初始状态中成立，且 θ 在转换下保持不变。然后，对于第二个前提，我们需要验证：从满足 θ 的状态开始的转换都满足 $\tau_>$。这些要求可通过以下一阶逻辑验证条件的有效性来刻画：(i) $\iota \wedge \Gamma \to \theta$ (ii) $\theta \wedge \Gamma \wedge \tau \wedge \Gamma' \to \theta'$ (iii) $\theta \wedge \Gamma \wedge \tau \wedge \Gamma' \to \tau_>$。

关于可靠性条件 $\mathcal{C}$，在一阶逻辑中，并非总能验证某个结构 s 满足$\mathcal{C}$ 该条件。这是因为我们利用可靠性条件来编码排序中那些无法用一阶逻辑定义的要求。具体而言，我们的构造器引入了两类可靠性条件：关系的良基性和集合的有限性。这些概念无法用一阶逻辑定义，因此我们无法将其验证归约为一阶有效性检查。相反，我们采用两种互补策略来验证可靠性条件：依赖预期的系统语义，或寻找可用一阶逻辑表达的充分条件。

策略*1*。 在某些情况下，预期的系统语义可以直接推出某些可靠性条件有效：1. 具有有限域语义的类型的元素集合是有限的。2. 具有有限域语义的类型上的偏序是良基的。3. 在系统的预期语义下良基的偏序，例如，具有扩展自然数预期语义的类型上的 $<$ 关系（如计时器中的关系），是良基的。

策略*2*。 第二种策略旨在证明，在所有可达状态下，元素集合 S 都是有限的，即使相关类型的论域并非有限。该方法基于 [29], ，并作了一些推广。其核心思想是：如果存在 $m \in \mathbb{N}$，使得初始时 B 中至多有 m 个元素，并且每次转换向 B 中至多添加 m 个元素，那么根据归纳，所有可达状态下的 B 都是有限的。这也意味着，任何满足以下条件的集合 A：在所有可达状态下均有 $A \subseteq B$，都是有限的。在我们考虑的可靠性条件中，集合 A 由某个 $\boldsymbol{z}$ 参数化；对于给定的 $\boldsymbol{z}$，它被定义为满足公式 $\alpha(\boldsymbol{y}{\cdot}\boldsymbol{z})$ 的 $\boldsymbol{y}$ 上的赋值集合，而我们的目标是验证对于任意 $\boldsymbol{z}$，A 都是有限的。因此，我们通过考虑公式 $\beta(\boldsymbol{y}{\cdot}\boldsymbol{z})$ 来编码上述思想，并将其用作 $\alpha(\boldsymbol{y}{\cdot}\boldsymbol{z})$ 的归纳有限过近似。由此得到以下验证条件，对于任意 $\boldsymbol{z}$，针对 $m = 1$ 概述如下：(i) $\alpha(\boldsymbol{y}{\cdot}\boldsymbol{z}) \wedge \theta \wedge \Gamma \to \beta(\boldsymbol{y}{\cdot}\boldsymbol{z})$；(ii) $\iota \wedge \theta \wedge \Gamma \to \forall \boldsymbol{z}\ \exists \boldsymbol{y}_0\ \forall \boldsymbol{y}\ \beta(\boldsymbol{y}{\cdot}\boldsymbol{z}) \to \boldsymbol{y} = \boldsymbol{y}_0$；以及 (iii) $\theta \wedge \Gamma \wedge \tau \wedge \Gamma' \to \forall \boldsymbol{z}\ \exists \boldsymbol{y}_0\ \forall \boldsymbol{y}\ \beta'(\boldsymbol{y}{\cdot}\boldsymbol{z}) \to \boldsymbol{y} = \boldsymbol{y}_0 \vee \beta(\boldsymbol{y}{\cdot}\boldsymbol{z})$。

例*6.* 接着例4，可靠性条件可理解为：在所有可达状态中，使 $\alpha(x) = (\mathsf{myt}(x) = \mathsf{serv})$ 成立的线程 x 数量有限。若取 $\beta(x) = \neg(\mathsf{idle}(x))$ 和一个适当的不变式 θ，我们便可以轻松验证上述策略2的验证条件 $m = 1$，从而确认该可靠性条件。

6 实现与评估

我们在一个演绎验证工具中实现了使用计时器和隐式排序验证时序性质的方法。我们的工具使用 Python 和 Z3 API [30], 实现，可在 [15] 获取。该工具以一阶逻辑中的迁移系统规约、FO-LTL 时序性质、由构造子组合给出的隐式排序、归纳不变式，以及在应用第 5.3 节的策略2时还包括归纳有限近似作为输入；这些内容均可使用计时器变量。该工具计算第 4.2 节中的计时器归约，生成第 5.3 节中的验证条件，并使用 Z3 求解器将其证明。当某个验证条件不成立时，工具会以增强系统的一次转换为形式生成反例。在 SMT 查询中，计时器类型 σ_{time} 使用解释整数编码（其中 ∞ 由 -1编码）。未解释函数、量词和整数的组合有时会给求解器带来困难。因此，我们实现了一项优化：对于不涉及计时器的不变式，允许针对不含计时器的原始系统检查其归纳性。此外，与 [28], 类似，我们允许为存在量词提供提示项，以帮助处理含有量词交替的查询。

表 2. 评估结果。时间以秒为单位。Con. 表示隐式排序中构造器的数量，Fin 表示归纳有限近似的数量，Inv. 表示不变式中合取项的数量，|证明|表示整个证明中的词元数量，而 |L2S 证明 | 表示 Ivy 文件中采用活性到安全性归约后整个证明中的词元数量。

示例	来源	时间（秒）	Con.	Fin	Inv.	|证明|	L2S 证明|
Ticket（第 2 节）	[32]	2	6	2	20	256	407
Paxos		101	11	1	20	238	521
ABP		4	31	2	34	539	860
HRB-Correct	[6]	135	7	0	6	99	446
HRB-Relay		56	8	0	17	316	593
阿克曼	[22]	0.8	5	2	9	211	748
词典数组（例 5）	-	0.08	4	1	2	30	-
TimestampedQueue	[29]	0.5	6	1	2	73	-
CascadingQueue		1.2	9	2	7	175	-
ReorderingQueue		2.3	9	2	11	220	-
MutexRing	[28]	0.5	8	0	5	93	-
LeaderRing		0.3	8	1	4	76	-
ToyStabilization		0.3	5	0	3	69	-
Dijkstra k-状态		64.8	6	0	3	213	-
二进制计数器		0.02	4	0	0	21	-
SAT 回溯		0.3	9	0	8	157	-
SAT 冲突驱动子句学习		5.3	8	0	8	148	-

我们使用一组来自以往工作的迁移系统及其时序性质示例 [6,22,28,29,32]，评估了该工具，并采用用户提供的隐式排序和不变式。我们在第 C 节详细介绍了所有示例。我们在一台搭载 Apple M1 Pro CPU 的 MacBook Pro 上运行该工具，使用的 Z3 版本为 4.13.3。

结果。 表 2 总结了我们的实验结果。为了直观展示证明的复杂度，我们针对每个示例报告验证验证条件所需的时间、隐式排序中使用的构造器数量（其中 TimerRank 计为一个）、所需的归纳有限近似数量，以及不变式中的合取项数量。此外，我们还报告证明的总规模，以词元（或项）的数量衡量。对于存在基于活性到安全方法的证明的情况，我们也报告相应证明的这一指标，数据取自 Ivy [34]。

对于来自 [28,29]，的示例，我们的证明与原始证明类似；不同之处在于，那些工作使用了特定的活性证明规则，而我们证明的是增强系统的终止性。将这些证明修改为适用于我们通用

框架相对简单，只需将上述规则的前提转换为 TimerRank 构造器的实例化即可。对于通过活性到安全性转换证明的示例，我们的证明更简单，因为不需要对有限抽象或状态的受监控副本进行推理。在一些示例中，证明规模的显著差异便体现了这一点。

7 相关工作

我们提出了一种方法，用于验证在无界域上运行、以一阶逻辑建模的系统的时序性质。许多研究致力于验证类似的系统和性质。一些研究采用适用于受限系统的全自动方法 [5, 9–11, 16, 17, 21, 23, 35],值得注意的是，这些研究不适用于以一阶逻辑描述的系统这一场景。另一些研究提供了用于设计分布式系统并以机械方式证明其活性的框架 [7,20,24,27,31,36–38,43] 还有一些研究提供了用于分析一阶线性时序逻辑证明机制的理论框架 [1, 18,42]。下面我们将讨论最相关的研究；与我们的工作类似，这些研究提供了依赖一阶逻辑求解器的半自动方法。

我们的工作基于 [28] ，并在两个方面对其进行了扩展。首先，原工作仅针对带参数化公平性假设的响应性质，而我们的工作针对任意 FO-LTL 性质。其次，原工作中对隐式排序进行的基于域的聚合假设相关域是有限的。在我们的工作中，只要聚合排序的非最小元素集合保持有限，就允许在无限域上进行基于域的聚合。

在 [29] 中，使用关系排名函数来验证时序性质。关系排名函数是隐式排序的一部分，由逐域排名函数进行字典序组合而成。与我们的工作类似，该工作也允许系统具有无限域，而我们在第 5.3 节中的策略2受到其强制有限性的做法启发。不同于我们统一的方法， [29] 针对不同类型的性质使用不同的证明规则来处理不同的时序性质，这种方法更复杂，也不够通用。

[41] 采用半自动化方法，从系统状态中的整数变量综合出多项式排名函数。这些变量可以来自规约本身、某些集合的基数，也可以是用户引入的公平性变量。这些公平性变量与我们在定义 1 中引入的计时器类似，也是计时器的灵感来源，但它们只能刻画特定类型的公平性假设，且需手动引入，因此不具备形式化保证。与我们一样，该方法允许用户将某些类型指定为有限解释，但对有限性的处理不够完备。虽然两篇论文有一些相同的示例，但他们会以非平凡的方式对公平性假设进行一些手动处理，从而大幅简化证明。

在 [22, 32, 33] 中，采用活性到安全性归约，将时序性质的验证规约为安全性验证。该归约对系统进行动态有限抽象。随后，为监控系统提供一个不变式，证明抽象后系统的无环性；在该监控系统中，当前状态会与

已保存状态在抽象意义下进行比较。活性到安全性归约是可靠的，但并不完备，而且需要借助时序预言来扩展其能力。此外，为增强系统编写不变式可能比给出隐式排序更复杂，也不够直观。

致谢 我们感谢匿名审稿人和 Eden Frenkel 提供的宝贵意见。开展本研究并取得这些成果所需的经费由欧洲研究委员会通过欧盟 Horizon 2020 研究与创新计划提供（资助协议编号 [759102-SVIS]）。本研究还部分获得了以色列科学基金会（ISF）第 2117/23 号资助、魏茨曼科学研究所新科学家中心的研究资助以及 Azrieli基金会的资助。

数据可用性声明 本文附有一个工件，其中包含复现结果所需的软件和数据。本文所用版本对应的工件快照可在 [15]获取。为便于重复使用和未来开发，源代码可在 [14]获取。

参考文献

1. Abadi, M.：时序证明的力量。《理论计算机科学》**65**(1)，35–83（1989），https://doi.org/10 2 2 1016 0304 3975 89 90138 . / - ( ) - . Ackermann, W.：关于希尔伯特实数构造。Mathematische An- nalen **99**(1)，118–133（1928）3. Bartlett, K.A., Scantlebury, R.A., Wilkinson, P.T.：关于通过半双工链路进行可靠全双工传输的注记。Commun. ACM **12**(5)，260–261（1969年5月），https://doi.org/10 362970 4 1145 362946. / . . Bayardo Jr, R.J., Schrag, R.：使用CSP回溯技术求解现实世界中的可满足实例。载于：Aaai/iaai。第203–208页。Citeseer（1997）5. Ben-Amram, A.M., Lee, C.S.：用于规模变化终止的排序函数 II。《计算机科学逻辑方法》**5**4382 6 2 2009 0903 ( ) ( )，http://arxiv.org/abs/ . . Berkovits, I., Lazic, M., Losa, G., Padon, O., Shoham, S.：通过分解为可判定逻辑验证基于阈值的分布式算法。载于：计算机辅助验证——第31届国际会议，CAV 2019（2019），https:// doi.org/ 10 1007 978 3 030 25543 5_15 . / - - - - . Brunel, J., Chemouil, D., Tawa, J.：分析Chord协议的基本活性性质。载于：计算机辅助验证中的形式化方法，FMCAD 2018 8603001 8 2018 10 23919 2018 ( )，https://doi.org/ . /FMCAD. . . Chang, E.J.H., Roberts, R.：针对进程环形配置中去中心化极值查找的改进算法。Commun. ACM **22**(5)，281–283（1979 10 1145 359104 359108 ），https://doi.org/ . / . . Chen, Y., Heizmann, M., Lengal, O., Li, Y., Tsai, M., Turrini, A., Zhang, L.：用于程序终止性检查的高级基于自动机的算法。载于：编程语言设计与实现，PLDI 2018（2018），https: //doi.org/10 3192405 10 1145 3192366 . / . . Cimatti, A., Griggio, A., Magnago, E.：无限状态系统中的LTL反例验证。《信息与计算》（2022），https://www.sciencedirect.com/ science/article/pii/S0890540122001328

11. Daniel, J., Cimatti, A., Griggio, A., Tonetta, S., Mover, S.：通过隐式抽象和良基关系将无限状态活性转化为安全性。载于：计算机辅助验证——第28届国际会议，CAV 2016，多伦多，安大略省，加拿大，7月
17 23, 2016 2016 10 1007 978 3 319 41528 4_15 - ( ),
12 https://doi.org/ . / - - - - . Dershowitz, N., Manna, Z.：利用多重集序证明终止性。Commun. ACM **22**359142 13 8 465 476 1979 10 1145 359138 ( )，–（8月），https://doi.org/ . / . . Dijkstra, E.W.：分布式控制下的自稳定系统。Commun. ACM **17**361202 14 11 643 644 1974 10 1145 361179 ( )，–（11月），https://doi.org/ . / . . Elad, N., Lotan, R.：implicit-rankings-timers，https://github.com/neta-elad/implicit-rankings-timers 15. Elad, N., Lotan, R.：通过计时器和排序验证无限状态系统的一阶时序性质（工件）（2026年1月），https://doi.org/10.5281/ zenodo.18157346 16. Fang, Y., Piterman, N., Pnueli, A., Zuck, L.D.：带隐式排序的活性。《国际软件工具与技术转换杂志》**8**(3)，261–279（2006）。https://doi.org/10. 1007/S10009-005-0193-X 17. Farzan, A., Kincaid, Z., Podelski, A.：证明参数化程序的活性。载于：ACM/IEEE逻辑与计算机科学年度研讨会，LICS ’16（2016 2935310 18 10 1145 2933575）。https://doi.org/ . / . . Geatti, L., Gianola, A., Gigante, N.：一阶自动机。载于：第三十九届 AAAI 人工智能会议会议论文集（2025），https://doi. org/10 33638 19 1609 39 14 . /aaai.v i . . Grossman, J.W., Zeitman, R.S.：阿克曼函数的一种内在迭代计算。《理论计算机科学》**57**，327–330（1988），https://doi.org/10. 1016/0304-3975(88)90046-1 20. Hawblitzel, C., Howell, J., Kapritsos, M., Lorch, J.R., Parno, B., Roberts, M.L., Setty, S.T.V., Zill, B.：Ironfleet：证明实用分布式系统的安全性与活性。Commun. ACM
60 7 83 92 2017 10 1145 3068608 ( )，–
21 （），https://doi.org/ . / . Herrmann, R., Rummer, P.：展示参数化迁移系统终止性的新方法。载于：自动机的实现与应用（2026），https://doi.org/
10 1007 978 3 032 02602 6_14 . / - - - - . Hoenicke, J., Padon,
22 O., Shoham, S.：时序预言的力量，第40–53页。施普普林格·自然瑞士，Cham（2026），https://doi.org/10.1007/ 978-3-032-13711-1_323. Hong, C., Lin, A.W.：数组系统的正则抽象。《ACM程序设计语言会议录》**8**3632864 24 638 666 2024 10 1145 (POPL)，–（），https://doi.org/ . / . Ioannidis, E., Zakowski, Y., Zdancewic, S., Angel, S.：用于机械化程序验证的结构化时序逻辑（2025 14906 25 2410），https://arxiv.org/abs/ . . Lamport, L.：Dijkstra并发编程问题的新解。《Commun. ACM》**17**361093 26 8 453 455 1974 10 1145 361082**16**2 133 ( )，–（），https://doi.org/ . / . . Lamport, L.：兼职议会。《ACM计算机系统汇刊》（），–169（1998年5月 279229 27 10 1145 279227），https://doi.org/ . / . . Leung, D., Zeldovich, N., Kaashoek, F.：Shipwright：证明包含拜占庭参与者的分布式系统的活性（2025 14080 28 2507），https://arxiv.org/abs/ . . Lotan, R., Shoham, S.：用于在一阶逻辑中验证活性性质的隐式排序。载于：系统构造与分析工具及算法，TACAS
2025 2025 10 1007 978 3 031 90643 5_20 ( )。
29 https://doi.org/ . / - - - - . McMillan, K.L.：迈向大规模活性证明。载于：计算机辅助验证——第36 27届国际会议，CAV蒙特利尔，魁北克省，加拿大，2024年7月-会议论文集，第I部分。《计算机科学讲义》，第24 14681卷，第255–276页。施普林格（2024）。https://doi.org/10.1007/978-3-031-65627-9_13

30. de Moura, L.M., Bjørner, N.S.: Z3：一种高效的 SMT 求解器。载于：系统构造与分析工具和算法，第 14 届国际会议 TACAS
2008，2008 10 1007 978 3 540 78800 3_24 ( )。
31
https://doi.org/ . / - - - - . Ouyang, L., Sun, X., Tang, R., Huang, Y., Jivrajani, M., Ma, X., Xu, T.: 面向分布式系统模型检测与验证的多粒度规范。载于：第二十届欧洲计算机系统会议论文集。Eu- roSys '25 3696069 32 2025 10 1145 3689031 ( )。https://doi.org/ . / . . Padon, O., Hoenicke, J., Losa, G., Podelski, A., Sagiv, M., Shoham, S.: 将活性规约为一阶逻辑中的安全性。Proc. ACM Program. Lang. **2**(POPL), 26:1– 26 3158114 33 33 2018 10 1145 : ( )。https://doi.org/ . / . Padon, O., Hoenicke, J., McMillan, K.L., Podelski, A., Sagiv, M., Shoham, S.: 用时序预言证明无限状态系统的时序性质。Formal Methods Syst. Des. **57**(2), 246–269 (2021)。https://doi.org/10.1007/S10703-021-00377-1 34. Padon, O., McMillan, K.L., Panda, A., Sagiv, M., Shoham, S.: Ivy：通过交互式泛化进行安全性验证。载于：PLDI '16：第 37 届 ACM SIGPLAN 编程语言设计与实现会议论文集。第 614 2908118 35 630 2016 10 1145 2908080 19 – 页（Jun ）。https://doi.org/ . / . . Podelski, A., Rybalchenko, A.: 转移不变式。载于：第 th 届 IEEE 计算机科学逻辑研讨会（LICS 2004），2004 年 7 月 14–17 日，芬兰图尔库，会议论文集。第 32–41 页。IEEE 计算机学会（2004）。https://doi.org/10.1109/LICS.2004.1319598 36. Qiu, L., Kim, Y., Shin, J., Kim, J., Honor´e, W., Shao, Z.: Lido：具有基于精化的活性证明的线性化拜占庭分布式对象。Proc. ACM Program. Lang. **8**3656423 37 1140 1164 2024 10 1145 (PLDI), – ( )，https://doi.org/ . / . Qiu, L., Xiao, J., Shin, J.Y., Shao, Z.: Lido-dag：用于验证基于 DAG 的共识协议的安全性与活性的框架。Proc. ACM Program. Lang.**9**3729306 38 2025 10 1145 (PLDI) (Jun), https://doi.org/ . / . Sun, X., Ma, W., Gu, J.T., Ma, Z., Chajed, T., Howell, J., Lattuada, A., Padon, O., Suresh, L., Szekeres, A., Xu, T.: Anvil：验证集群管理控制器的活性。载于：第 18 届 USENIX 操作系统设计与实现研讨会（2024），https://www.usenix.org/conference/osdi24/presentation/ sun-xudong 39. Vardi, M.Y.: 线性时序逻辑的自动机理论方法。载于：并发逻辑——结构与自动机（1995）。https://doi.org/10.1007/ 3-540-60915-6_640. Widder, J., Schmid, U.: 混合进程与链路故障下部分同步系统中的时钟同步启动。Distributed Comput. **20**(2), 115– 140 (2007)。https://doi.org/10.1007/S00446-007-0026-0, https://doi.org/10.1007/s00446-007-0026-0 41. Yao, J., Tao, R., Gu, R., Nieh, J.: 使用排序函数对分布式协议的活性性质进行基本自动化验证。Proc. ACM Program. Lang.
8 1028 1059 2024 10 1145 3632877 (POPL), – ( )。
42
https://doi.org/ . / . Zhang, W.: 一阶 Büchi 自动机及其在 LTL 规格验证中的应用。《逻辑与代数方法及其程序应用》**142**, 101021 (2025)。https: //doi.org/10 101021 43 1016 2024 . /J.JLAMP. . . Zhao, Q., Pˆırlea, G., Grzeszkiewicz, K., Gilbert, S., Sergey, I.: 复合拜占庭协议的组合式验证。载于：计算机与通信安全会议。CCS '24，计算机协会，纽约，NY，美国（2024）。https://doi.org/10.1145/3658644.3690355

A 证明

引理 1 的证明

证明。对 FO-LTL 公式的结构进行归纳证明。对于原子公式 $p(\boldsymbol{x})$，由公理: $\Gamma_p = t_p(\boldsymbol{x}) = 0 \leftrightarrow p$ 可得。对于 $\psi = \Diamond\rho$: 若$\hat{s}_i, v \models t_{\Diamond\rho}(\boldsymbol{x}) = 0$，则由 $\Gamma_{\Diamond\rho}$ 可知，t_ρ 的解释为 $k < \infty$; 根据转换中计时器的递减，可知$\hat{s}_{i+1}$ 将 t_ψ 解释为 $k-1$; 重复此论证 k 次，可得$\hat{s}_{i+k}, v \models t_\rho(\boldsymbol{x}) = 0$。接着，由归纳假设可得 $\hat{\pi}^{i+k}$, $v \models \rho$，因此根据 $\Diamond$ 的语义可得 $\hat{\pi}^i, v \models \Diamond\rho$ 。反过来，若 $\hat{\pi}^i, v \models \Diamond\rho$，则存在 $k \in \mathbb{N}$ 使得 $\hat{\pi}^{i+k}, v \models \rho$; 因此根据归纳假设，我们有$\hat{s}_{i+k}, v \models (t_\rho = 0)$; 由 $\Gamma_{\Diamond\rho}$ 可得$\hat{s}_{i+k}, v \models (t_{\Diamond\rho} = 0)$。现在，我们反复应用 $\tau_{\Diamond\rho}$，得到对所有 $j \leq i+k$，均有$\hat{s}_j, v \models (t_{\Diamond\rho} = 0)$; 而对于 $j = i$，则可得到所需结果。对于 $\psi = \Box\rho$: 若$\hat{s}_i$, $v \models t_{\Box\rho}(\boldsymbol{x}) = 0$，则由 $\tau_{\Box\rho}$ 可得$\hat{s}_i, v \models t_\rho(\boldsymbol{x}) = 0$ 且$\hat{s}_{i+1}, v \models t'_{\Box\rho}(\boldsymbol{x}) = 0$; 对 $i+1$ 等情形重复此论证，可得对每个 $k \geq i$，均有$\hat{s}_i, v \models t_\rho(\boldsymbol{x}) = 0$; 随后由归纳假设可得 $\hat{\pi}^k, v \models \rho$，并根据 $\Box$ 的语义可得 $\hat{\pi}^i, v \models \Box\rho$ 。反过来，若 $\hat{\pi}^i, v \models \Box\rho$，则对每个 $k \in \mathbb{N}$，均有 $\hat{\pi}^{i+k}, v \models \rho$，因此根据归纳假设，我们有$\hat{s}_{i+k}, v \models (t_\rho = 0)$。观察$\hat{s}_i, v$ 中 $t_{\neg\rho}$ 的取值，若其为$k < \infty$，则根据计时器的递减，对于 $\hat{s}_{i+k}$，我们将有 $(t_{\neg\rho} = 0)$，并由 $\Gamma_{\neg\rho}$ 可得 $t_\rho \neq 0$，这与上述结论矛盾。因此必有 $\hat{s}_i, v \models (t_{\neg\rho} = \infty)$; 于是由 $\Gamma_{\Box\rho}$ 可得$\hat{s}_i, v \models (t_{\Box\rho} = 0)$，正如所需。其他算子的归纳情形较为简单，可由一阶推理得出。□

引理 2 的证明

证明。为了构造轨迹 $\hat{\pi}$，我们需要将每个结构 s_i 扩展为在 Σ_φ 上的结构$\hat{s}$，使其满足计时器规范和预期语义。首先，对于每个结构 s_i，通过将 σ_{time}、$\leq$、0、∞、-1解释为带有通常序的扩展自然数 $\mathbb{N} \cup \{\infty\}$ 来扩展它。接下来需要定义 $A(\varphi)$中每个公式的计时器函数的解释。令 $\psi \in A(\varphi)$是一个自由变量为 $\boldsymbol{x}$ 的公式，且给定结构$\hat{s}_i$，将t_ψ 的解释定义为一个从 $\boldsymbol{x}$ 的类型到 σ_{time} 的函数，使得对于任意一列类型化元素 $\boldsymbol{d}$ 和赋值 $v = \boldsymbol{x} \mapsto \boldsymbol{d}$，$t_\psi$ 将 $\boldsymbol{d}$ 映射到满足 π^{i+j}、$v \models \psi$ 的极小 $j \in \mathbb{N}$; 若不存在这样的 j，则映射到 ∞。由于 π 和 $\hat{\pi}$ 在 Σ 中的符号上相同，因此可以互换使用它们的满足关系。

接下来需要证明，这些解释满足计时器规范。首先，$\pi \models \varphi$，由此可得，对于$\hat{s}_0$，有 $t_\varphi = 0$，满足ι_φ。其次，我们证明所有状态$\hat{s}_i$ 都满足公理 Γ_φ，这意味着每个$\psi \in A(\varphi)$ 都满足 Γ_ψ: 对于原子命题 p，有$\hat{s}_i \models t_p(\boldsymbol{x}) = 0$ 当且仅当接着 $\pi^i \models p$ 当且仅当$\hat{s}_i \models p$，由此可得，$t_p(\boldsymbol{x}) = 0 \leftrightarrow p$ 在所有状态$\hat{s}_i$中均成立。对于 $\psi = \Diamond\rho$: $\hat{s}_i \models (t_{\Diamond\rho} = 0)$ 当且仅当 $\pi^i \models \Diamond\rho$当且仅当存在 $k \in \mathbb{N}$ 使得 $\pi^{i+k} \models \rho$ 当且仅当 $\hat{s}_i \models t_\rho < \infty$。对于 $\psi = \Box\rho$，有$\hat{s}_i \models (t_{\Box\rho} = 0)$ 当且仅当 $\pi^i \models \Box\rho$ 当且仅当对于所有 $k \in \mathbb{N}$，都有 $\pi^{i+k} \models \rho$，当且仅当对于每个 $k \in \mathbb{N}\ \pi^{i+k} \not\models \neg\rho$，当且仅当 $\hat{s}_i \models t_{\neg\rho} < \infty$。满足

Γ_ψ 由其他算子构造的公式，其满足性证明很简单，可由一阶推理得出。最后，我们希望证明 $\hat{}\pi$ 的连续状态满足τ_φ。对于任意子公式 ψ，若$\hat{s}_i \models 0 < t_\psi < \infty$，令 j 为$\hat{s}_i$中对 t_ψ的解释，它是满足 $\pi^{i+j} \models \psi$的极小值，因此 t_ψ 在$\hat{s}$ 中被解释为 $j-1$，于是 $(\hat{s}_i, \hat{s}_{i+1}) \models t'_\psi(\boldsymbol{x}) = t_\psi(\boldsymbol{x}) - 1$。此外，若$\hat{s}_i$将 $t_\psi(\boldsymbol{x})$ 解释为 ∞，则π^{i+j}，对于任意 j，$v \not\models \psi$，由此可得，对于任意 j，π^{i+1+j}，$v \not\models \psi(\boldsymbol{x})$，因此$\hat{s}_{i+1}$也将 $t'_\psi(\boldsymbol{x})$ 解释为 ∞，于是 $(\hat{s}_i, \hat{s}_{i+1}) \models t'_\psi(\boldsymbol{x}) = \infty \to t_\psi(\boldsymbol{x}) = \infty$，正如所需。对于 $\psi = \Diamond\rho$，$\hat{s}_i \models (t_{\Diamond\rho} = 0)$当且仅当存在一个极小的 $k \in \mathbb{N}$使得 $\pi^{i+k} \models \rho$，$k = 0$ 当且仅当 $t_\rho(\boldsymbol{x}) = 0$；否则，$k > 0$ 当且仅当 $\pi^{i+1} \models \Diamond\rho$ 当且仅当$\hat{s}_{i+1} \models (t'_{\Diamond\rho} = 0)$，因此这种情形等价于 $(\hat{s}_i, \hat{s}_{i+1}) \models t_\rho(\boldsymbol{x}) = 0 \vee t'_{\Diamond\rho}$。对于 $\psi = \Box\rho$，$\hat{s}_i \models (t_{\Box\rho} = 0)$当且仅当对于每个$k \in \mathbb{N}$ 都有 $\pi^{i+k} \models \psi$，当且仅当对于 $\pi^i \models \rho$ 和 $\pi^{i+1} \models \Box\rho$，当且仅当 $(\hat{s}_i, \hat{s}_{i+1}) \models t_\rho(\boldsymbol{x}) = 0 \wedge t'_{\Box\rho} = 0$。□

定理 1 的证明

证明。可靠性：假设 $\mathcal{T}(\mathcal{S}) \times \mathcal{T}_{\neg\varphi}(\mathcal{S}_{\text{time}})$ 会终止。设 π 是$\mathcal{T}(\mathcal{S})$ 的一条轨迹，我们希望证明 $\pi \models \varphi$。反设 $\pi \models \neg\varphi$，令 $\hat{}\pi$ 为引理 2 中从 $\mathcal{T}_{\neg\varphi}(\mathcal{S}_{\text{time}})$ 得到的扩展轨迹，使得 $\hat{}\pi|_\Sigma = \pi$。由此可得，$\hat{}\pi$ 是 $\mathcal{T}(\mathcal{S}) \times \mathcal{T}_{\neg\varphi}(\mathcal{S}_{\text{time}})$ 的一条轨迹，这与该系统会终止的假设矛盾。完备性：假设 $\mathcal{T}(\mathcal{S})$ 满足 φ。反设 $\hat{}\pi = \hat{s}_0, \hat{s}_1, \ldots\ldots$是 $\mathcal{T}(\mathcal{S}) \times \mathcal{T}_{\neg\varphi}(\mathcal{S}_{\text{time}})$ 的一条轨迹。根据初始状态公理，我们有 $\hat{s}_0 \models (t_{\neg\varphi} = 0)$，因此根据引理 1，我们有 $\hat{}\pi \models \neg\varphi$。考虑投影 $\pi = \hat{}\pi|_\Sigma$，它必然是 $\mathcal{T}(\mathcal{S})$ 的一条轨迹，并且 $\pi \models \neg\varphi$，这与假设矛盾。□

定理2的证明

证明。反设 $\mathcal{T}(\mathcal{S})$ 不终止，则存在一条轨迹 $\pi = (s_i)_{i=1}^\infty$，其基于某个共享论域 D。由此可知，$\tau_>$ 是一个封闭隐式排序，其可靠性条件为 $\mathcal{C}$，因此对于 D，存在一个排序值域 $(A, <)$ 和一个排序函数 $f\colon \text{struct}(\Sigma, D) \to A$。考察序列 $(f(s_i))_{i=0}^\infty$，对于每个 $i \in \mathbb{N}$，s_i 都是可达的，并且 (s_i, s_{i+1}) 是一次可达迁移。因此，根据假设，对于所有 $i \in \mathbb{N}$，都有 $s_i \in \mathcal{C}$，且 $(s_i, s_{i+1}) \models \tau_>$；根据定义，有 $f(s_{i+1}) < f(s_i)$，因为 $f(s_i) \in \{f(s) \mid s \in \mathcal{C}\}$，由此可知，序列 $(f(s_i))_{i=0}^\infty$ 是良基偏序中的无限递减序列，从而导致矛盾。□

定理 3 的证明 我们证明构造器 3 和构造器 4 的可靠性；其他构造器的可靠性证明与 [28] 中的证明类似。在证明这些构造器的可靠性之前，我们先证明一个命题，以表明其底层序是良基的。

命题。给定一个良基集合（$P, <_P$）和一个集合 Y，$(\text{Fin}(Y \to P), <_{\text{pw}})$，从 Y 到 P 的函数中像非极小的次数有限的函数所构成的集合，在逐点序下是良基的。

证明。反设存在一个无限序列 $(f_i)_{i=0}^{\infty}$，使得对所有 i，$f_i \in \mathsf{Fin}(Y \to P)$ 和 $f_i >_{\mathrm{pw}} f_{i+1}$ 均成立。令 $S = \{y \in Y \mid f_0(y) \text{ 在 } P\}$ 中是非极小的，定义一个序列 $(y_i)_{i=0}^{\infty}$，使得 $f_{i+1}(y_i) <_P f_i(y_i)$。由此可知，对所有 $i \in \mathbb{N}$，都有 $y_i \in S$，因为 $f_{i+1}(y_i) <_P f_i(y_i) \leq_P f_0(y_i)$。$S$ 是有限的，因此存在 $y_\star$，它在序列 $(y_i)_{i=0}^{\infty}$ 中无限次出现，于是对无限多个 i，有 $f_{i+1}(y_\star) <_P f_i(y_\star)$。此外，对任意 i，都有 $f_{i+1}(y_\star) \leq f_i(y_\star)$，因此序列 $(f_i(y_\star))_{i=0}^{\infty}$ 在 $(P, <_P)$ 中包含一个无限递减序列，这与良基性矛盾。□

命题。 构造器 3 是可靠的。

证明。给定一个带有参数 $\boldsymbol{x} = y{\cdot}\boldsymbol{z}$ 和可靠性条件 $\mathcal{C}^\circ$ 的隐式排序 Rk°，我们需要证明，$\mathsf{DomPW}(\mathsf{Rk}^\circ, y)$ 是一个带有参数 $\boldsymbol{z}$ 和可靠性条件 $\mathcal{C}$ 的隐式排序，其定义见构造器 3。对于论域 D，令 A°、$f_\circ$ 分别为 Rk° 和 D 的排序值域与排序函数。定义 $A = \mathsf{assign}(y, D) \to A^\circ$，其序由 $<_{\mathrm{pw}}$ 给出，并且 $f\colon \mathsf{struct}(\Sigma, D) \times \mathsf{assign}(\boldsymbol{z}, D) \to A \text{ by } f((s,v)) = \lambda u \in \mathsf{assign}(y, D).f_\circ(s, u{\cdot}v)$。

假设 (s,v)、$(s',v') \models \tau_{\geq}(\boldsymbol{x}, \boldsymbol{x}') = \forall y.\tau^\circ_{\geq}(y{\cdot}\boldsymbol{z}, y{\cdot}\boldsymbol{z}')$。那么对于任意赋值 u 到 y，我们有 (s,v)、$(s',v')\, u \models \tau^\circ_{\geq}(y{\cdot}\boldsymbol{z}, y{\cdot}\boldsymbol{z}')$，根据我们的约定，这等价于 $(s, u{\cdot}v)$、$(s', u{\cdot}v') \models \tau^\circ_{\geq}(\boldsymbol{y}{\cdot}\boldsymbol{z}, \boldsymbol{y}{\cdot}\boldsymbol{z}')$。根据 Rk° 的定义，我们得到 $f_\circ(s, u{\cdot}v) \geq_\circ f_\circ(s', u{\cdot}v')$。这对任意 $u \in \mathsf{assign}(y, D)$ 都成立，因此可推出：$\lambda u.f_\circ(s, u{\cdot}v) \geq_{\mathrm{pw}} \lambda u.f_\circ(s', u{\cdot}v')$，从而 $f(s, v) \geq_{\mathrm{pw}} f(s', v')$。此外，若有 (s,v)、$(s',v') \models (\exists y.\tau^\circ_{>}(y{\cdot}\boldsymbol{z}, y{\cdot}\boldsymbol{z}'))$，则存在 $u \in \mathsf{assign}(y, D)$，使得 $f_\circ(s, u{\cdot}v) >_\circ f_\circ(s', u{\cdot}v')$，由此可得 $f(s,v) >_{\mathrm{pw}} f(s', v')$。

对于极小性，假设 $(s,v) \models \kappa_{\min}(\boldsymbol{z}) = \forall y.\kappa^\circ_{\min}(y{\cdot}\boldsymbol{z})$，再反设 $f(s,v)$ 并非极小，从而导出矛盾。于是存在某个函数 $g \in A$，使得 $g <_{\mathrm{pw}} f(s,v)$，因此存在 $u \in \mathsf{assign}(y, D)$，使得 $g(u) <_\circ f(s,v)(u) = f_\circ(s, u{\cdot}v)$，故 $f_\circ(s, u{\cdot}v)$ 在 A° 中并非极小。但是，$(s,v)\ u \models \kappa^\circ_{\min}(y{\cdot}\boldsymbol{z})$，因此根据定义可知 $f(s, u{\cdot}v)$ 在 A° 中是极小的，矛盾。

最后，考虑将 $<_{\mathrm{pw}}$ 限制到 $P = \{f(s,v) \mid s \in \mathcal{C}, v \in \mathsf{assign}(\boldsymbol{z}, D)\}$，其中 $\mathcal{C} = \{s \in \mathsf{struct}(\Sigma) \mid s \in \mathcal{C}^\circ \text{ 和 } \forall v \in \mathsf{assign}(\boldsymbol{z}).|\{u \in \mathsf{assign}(y) \mid (s, u{\cdot}v) \models \neg\kappa^\circ_{\min}(y{\cdot}\boldsymbol{z})\}| < \infty\}$。根据递归情况，序 $<_\circ$ 在集合 $P_\circ = \{f_\circ(s, u{\cdot}v) \mid s \in \mathcal{C}^\circ, u{\cdot}v \in \mathsf{assign}(y{\cdot}\boldsymbol{z}, D)\}$ 上是良基的。由上文可知，$\mathsf{Fin}(\mathsf{assign}(y, D) \to P_\circ)$ 在 $<_{\mathrm{pw}}$ 下是良基的；我们将证明 P 包含于其中。设 (s,v) 满足 $s \in \mathcal{C}$，且 $v \in \mathsf{assign}(\boldsymbol{z}, D)$，则有 $s \in \mathcal{C}^\circ$；因此，对于任意 $u \in \mathsf{assign}(y, D)$，均有 $f_\circ(s, u{\cdot}v) \in P_\circ$。接下来只需证明 $f(s,v)$ 只有有限多个非极小输出。事实上，根据假设 $s \in \mathcal{C}$，因此对于 v，有 $|\{u \in \mathsf{assign}(y) \mid (s, u{\cdot}v) \models \neg\kappa^\circ_{\min}(y{\cdot}\boldsymbol{z})\}| < \infty$。对于赋值 $u \in \mathsf{assign}(y, D)$，如果 $f_\circ(s, u{\cdot}v)$ 不是极小的，那么 $(s, u{\cdot}v) \not\models \kappa^\circ_{\min}(y{\cdot}\boldsymbol{z})$；否则，根据递归情况，它必定是极小的。因此 $(s, u{\cdot}v) \models \neg\kappa^\circ_{\min}(y{\cdot}\boldsymbol{z})$，所以 $f(s,v)$ 的输入中，至多只有有限多个满足 $f(s,v)(u)$ 非极小，于是 $f(s,v) \in \mathsf{Fin}(\mathsf{assign}(y, D) \to P_\circ)$。□

命题。 给定两个良基集合 $(Y, <_Y)$，$(P, <_P)$，$(\mathsf{Fin}(Y \to P), <_{\mathrm{lex}})$，从 Y 到 P 的函数中像仅有有限个非极小值的函数所组成的集合，在逆字典序下是良基的。

证明。我们的证明仿照 Manna-Dershowitz 定理 [12] 的证明。反设存在一个无限序列 $(f_i)_{i=0}^{\infty}$，使得对所有 i、$f_i \in$，Fin$(Y \to P)$ 且 $f_i >_{\text{lex}} f_{i+1}$。按以下过程构造一棵无限树：添加一个根节点 r；第一步，对每个满足 $f_0(y)$ 非极小的 y，添加顶点 $(y, f_0(y))$，并将它们全部连接到 r。第二步，对每个满足 $f_1(y) \neq f_0(y)$ 的 $y \in Y$，添加顶点 $(y, f_1(y))$。若 $f_1(y) <_P f_0(y)$，则连接一条边 $(y, f_0(y)) \to (y, f_1(y))$。否则，若 $f_1(y) \not<_P f_0(y)$，则存在 $x \in Y$，使得 $y <_Y x$ 且 $f_1(x) <_P f_0(x)$；由此可知 $f_0(x)$ 非极小，因此 $(x, f_0(x))$ 在图中，于是连接边 $(x, f_0(x)) \to (y, f_1(y))$。对任意 i 和 $i+1$ 重复这一构造。该构造满足以下性质：构造进行到第 i 步后，图的叶节点恰好是具有非极小 $f_i(y)$ 的节点 $(y, f_i(y))$；每一步只添加有限个顶点；每一步添加的边只连接叶节点；每个顶点的度数有限；每一步至少添加一个顶点，因此该图是无限的；每条边 $(x, p) \to (y, q)$ 都满足 $x >_Y y$ 或 $x = y \wedge p >_P q$。现在，由于该图是一棵有限度数的无限树，我们应用 K″onig 无限引理，该引理指出图中存在一条无限路径。将这条无限路径记为 $v_0, v_1, v_2, \ldots$。注意，对每个 i，都有一条边 $v_i \to v_{i+1}$，因此 $v_{i+1} <_{Y \times P} v_i$，其中 $<_{Y \times P}$ 是 $Y \times P$ 上的标准字典序。由于 $<_Y$ 和 $<_P$ 均为良基的，因此 $<_{Y \times P}$ 也是良基的，矛盾。□

命题。 构造器 4 是可靠的。

证明。给定一个隐式排序 Rk°、参数 $\boldsymbol{x} = y \cdot \boldsymbol{z}$、可靠性条件$\mathcal{C}^\circ$，以及论域 Σ 上的关系 $\ell(y_1, y_2)$，我们需要证明 DomLex(Rk°, y, ℓ) 是一个具有参数 $\boldsymbol{z}$ 和可靠性条件 $\mathcal{C}$ 的隐式排序，如构造器 4 中所定义。对于论域 D，令 $A^\circ, f_\circ$ 分别为 Rk° 和 D 的排序值域与排序函数。定义 interp(ℓ) $= \mathcal{P}$(assign($y_1 \cdot y_2, D$))，它是二元关系 ℓ 所有可能解释构成的集合。对于一个结构s，定义 $I^\ell(s) \in$ interp(ℓ) 为 ℓ在 s中的解释。我们将 Rk 的排序值域定义为 $A =$ interp(ℓ) $\times$ assign(y, D) $\to A^\circ$，并按如下方式排序：$(I_1^\ell, a_1) \leq (I_2^\ell, a_2)$ 当且仅当 $I^\ell := I_1^\ell = I_2^\ell$, I^ℓ 是一个偏序，且 $a_1 \leq_{\text{lex}} a_2$ 按照函数集上的字典序排列，并依据 I^ℓ。接着，我们将排序函数定义为: f: struct(Σ, D) $\times$ assign($\boldsymbol{z}, D$) $\to A$，其定义为$f((s, v)) = (I^\ell(s)\ \lambda u \in$ assign(y, D).$f_\circ(s, u \cdot v))$。

若 (s, v)、$(s', v') \models \tau_{\geq}^\circ(\boldsymbol{x}, \boldsymbol{x}') =$序($\ell$) $\wedge \forall y.(\tau_{\geq}^\circ(y \cdot \boldsymbol{z}, y \cdot \boldsymbol{z}') \vee \exists y_0.(\ell(y, y_0) \wedge \tau_{>}^\circ(y_0 \cdot \boldsymbol{z}$, $y_0 \cdot \boldsymbol{z}')))$。那么，特别地，我们有 $I^\ell(s) = I^\ell(s')$，且它是一个偏序。于是，对于任意赋值 u y s v , s' v' $u \models \tau_{\geq}^\circ$ $y \cdot \boldsymbol{z}$ $y \cdot \boldsymbol{z}' \vee$，都有 $\exists y_0.\ \ell\ y\ y_0 \wedge \tau_{>}^\circ\ y_0 \cdot \boldsymbol{z}\ y_0 \cdot \boldsymbol{z}'$ $f_\circ\ s\ u \cdot v\ \geq_\circ f_\circ\ s'\ u \cdot v'$ (,) (,) ((,) ((,) (,))), 因此，要么 (,) (,)，要么存在 $u_0 \in$赋值(y, D)，使得 $u \cdot u_0 \in I^\ell(s)$ 且 $f_\circ(s, u_0 \cdot v) >_\circ f_\circ(s', u_0 \cdot v')$。这表明，按照由 $I^\ell(s)$ 定义的字典序，$\lambda u \in$ 赋值(y, D).$f_\circ(s, u \cdot v) \geq_{\text{lex}} \lambda u \in$赋值($y, D$).$f_\circ(s', u \cdot v')$，因此根据上述定义的序，$f(s, v) \geq f(s', v')$。此外，若有 (s, v)、$(s', v') \models (\exists y.\tau_{>}^\circ(y \cdot \boldsymbol{z}, y \cdot \boldsymbol{z}'))$，则由相同的论证可知，存在赋值 $u \in$ 赋值(y, D)，使得$f_\circ(s, u \cdot v) >_\circ f_\circ(s', u \cdot v')$，从而可得 $f(s, v) > f(s', v')$。$\kappa_{\min}$ 的定义与构造器 3 相同，因此证明也相同。

最后，我们需要证明，$P = \{f(s, v) \mid s \in \mathcal{C}, v \in \text{赋值}(\boldsymbol{z}, D)\}$ 按照上文定义的序是良基的。排序值域 A 是 $\text{interp}(\ell) \times \text{赋值}(y, D) \to A^\circ$，其中 $\leq$ 中的元素对具有相同的解释 I^ℓ，对应于 ℓ。容易看出，只需证明在 I^ℓ 固定时，$\text{赋值}(y, D) \to A^\circ$ 上定义的序是良基的。换言之，我们需要证明，对于固定的 I^ℓ，满足 $I^\ell(s) = I^\ell$ 的 $\{\lambda u \in \text{赋值}(y, D).f_\circ(s, u{\cdot}v) \mid s \in \mathcal{C}$，其中 $v \in \text{赋值}(\boldsymbol{z}, D)\}$ 按照 $<_{\text{lex}}$ 排序，是良基的。定义 $(Y = \text{赋值}(y, D) <_Y = I^\ell)$。对于状态 $s \in \mathcal{C}$，我们有 $I^\ell(s)$ 是良基的，因此 $<_Y$ 是良基的。取 $P = \{f_\circ(s, u{\cdot}v) \mid s \in \mathcal{C}, u{\cdot}v \in \text{赋值}(\boldsymbol{x}, D)\}$，按 $<_P = <_\circ$ 排序。对于状态 $s \in \mathcal{C}$，我们有 $s \in \mathcal{C}^\circ$，因此根据递归情况，P 上的序是良基的。最后，对于任意 $s \in \mathcal{C}$ 和 $v \in \text{赋值}(\boldsymbol{z}, D)$，有限的 $\text{赋值}(\neg\kappa^\circ_{\min}(y{\cdot}\boldsymbol{z}) s, v)$，因此 $\lambda u \in \text{赋值}(y, D).f_\circ(s, u{\cdot}v) \in \text{Fin}(Y \to P)$。由上面的命题可知，该集合在 $<_{\text{lex}}$ 序下是良基的。□

B 其他构造器

我们介绍实现中使用的另外四种构造器，这些构造器未在第5.2节中介绍，均取自 [28], ，并具有预期的极小公式和可靠性条件。

构造器 5。二元构造器 接受一个定义在 Σ上的公式 $\alpha(\boldsymbol{x})$。它返回一个隐式排序 $\text{Bin}(\alpha) = (\tau_>, \tau_\geq, \kappa_{\min})$，其参数为 $\boldsymbol{x}$，可靠性条件 $\mathcal{C}$ 定义如下：

$$\tau_>(\boldsymbol{x}, \boldsymbol{x}') = \alpha(\boldsymbol{x}) \wedge \neg\alpha'(\boldsymbol{x}') \qquad \tau_\geq(\boldsymbol{x}, \boldsymbol{x}') = \neg\alpha(\boldsymbol{x}) \to \neg\alpha'(\boldsymbol{x}')$$
$$\kappa_{\min}(\boldsymbol{x}) = \neg\alpha(\boldsymbol{x}) \qquad \mathcal{C} = \text{struct}(\Sigma)$$

构造器 6。逐点构造器 接受 $i = 1, \ldots, m$对应的隐式排序 $\text{Rk}^i = (\tau^i_>, \tau^i_\geq, \kappa^i_{\min})$，每个排序都带有参数 $\boldsymbol{x}$和可靠性条件 $\mathcal{C}^i$。它返回一个隐式排序 $\text{PW}(\text{Rk}^1, \ldots, \text{Rk}^m) = (\tau_>, \tau_\geq, \kappa_{\min})$，其参数为 $\boldsymbol{x}$，可靠性条件 $\mathcal{C}$ 定义如下：

$$\tau_>(\boldsymbol{x}, \boldsymbol{x}') = \tau_\geq(\boldsymbol{x}, \boldsymbol{x}') \wedge \bigvee_i \tau^i_>(\boldsymbol{x}, \boldsymbol{x}') \quad \tau_\geq(\boldsymbol{x}, \boldsymbol{x}') = \bigwedge_i \tau^i_\geq(\boldsymbol{x}, \boldsymbol{x}')$$
$$\kappa_{\min}(\boldsymbol{x}) = \bigwedge_i \kappa^i_{\min}(\boldsymbol{x}) \qquad \mathcal{C} = \bigcap_i \mathcal{C}^i$$

构造器 7。字典序构造器针对 $i = 1, \ldots, m$ 接收隐式排序 $\text{Rk}^i = (\tau^i_>, \tau^i_\geq, \kappa^i_{\min})$，每个排序都带有参数 $\boldsymbol{x}$ 和可靠性条件 $\mathcal{C}^i$。它返回一个隐式排序 $\text{Lex}(\text{Rk}^1, \ldots, \text{Rk}^m) = (\tau_>, \tau_\geq, \kappa_{\min})$，其参数为 $\boldsymbol{x}$，可靠性条件 $\mathcal{C}$ 定义如下：

$$\tau_>(\boldsymbol{x}, \boldsymbol{x}') = \bigvee_i (\tau^i_>(\boldsymbol{x}, \boldsymbol{x}') \wedge \bigwedge_{j<i} \tau^j_\geq(\boldsymbol{x}, \boldsymbol{x}')) \quad \tau_\geq(\boldsymbol{x}, \boldsymbol{x}') = \tau_>(\boldsymbol{x}, \boldsymbol{x}') \vee \bigwedge_i \tau^i_\geq(\boldsymbol{x}, \boldsymbol{x}')$$
$$\kappa_{\min}(\boldsymbol{x}) = \bigwedge_i \kappa^i_{\min}(\boldsymbol{x}) \qquad \mathcal{C} = \bigcap_i \mathcal{C}^i$$

构造器 8。论域置换逐点构造器接收一个隐式排序 $\mathrm{Rk}^\circ = (\tau^\circ_>, \tau^\circ_\geq, \kappa^\circ_{\min})$，其参数为 $\boldsymbol{x} = \boldsymbol{y}{\cdot}\boldsymbol{z}$，可靠性条件为 $\mathcal{C}$，以及 $k \in \mathbb{N}$。它返回一个隐式排序 $\mathrm{DomPerm}(\mathrm{Rk}^\circ, \boldsymbol{y}, k) = (\tau_>, \tau_\geq, \kappa_{\min})$，其参数为 $\boldsymbol{z}$，可靠性条件 $\mathcal{C}$ 定义如下：

$$\tau_>(\boldsymbol{z}, \boldsymbol{z}') = \tilde{\exists}\sigma.\ \forall \boldsymbol{y}.\ \tau^\circ_\geq(\boldsymbol{y}{\cdot}\boldsymbol{z}, \boldsymbol{y}_\sigma{\cdot}\boldsymbol{z}') \wedge \exists \boldsymbol{y}.\ \tau^\circ_>(\boldsymbol{y}{\cdot}\boldsymbol{z}, \boldsymbol{y}_\sigma{\cdot}\boldsymbol{z}')$$
$$\tau_\geq(\boldsymbol{z}, \boldsymbol{z}') = \tilde{\exists}\sigma.\ \forall \boldsymbol{y}.\ \tau^\circ_\geq(\boldsymbol{y}{\cdot}\boldsymbol{z}, \boldsymbol{y}_\sigma{\cdot}\boldsymbol{z}') \qquad \kappa_{\min}(\boldsymbol{z}) = \forall \boldsymbol{y}.\kappa^\circ_{\min}(\boldsymbol{y}{\cdot}\boldsymbol{z})$$
$$\mathcal{C} = \{s \in \mathrm{struct}(\Sigma) \mid s \in \mathcal{C}^\circ \text{ and } \forall v \in \mathrm{assign}(\boldsymbol{z}).\ \mathrm{finite}_{\boldsymbol{y}}(\neg\kappa^\circ_{\min}(\boldsymbol{y}{\cdot}\boldsymbol{z}), s, v)\}$$

其中：$\boldsymbol{y}_\sigma = \mathsf{ite}(\boldsymbol{y} = \boldsymbol{y}^1_\rightarrow,\ \boldsymbol{y}^1_\leftarrow,\ \mathsf{ite}(\boldsymbol{y} = \boldsymbol{y}^1_\leftarrow,\ \boldsymbol{y}^1_\rightarrow,\ \ldots\ldots,\ \mathsf{ite}(\boldsymbol{y} = \boldsymbol{y}^k_\rightarrow,\ \boldsymbol{y}^k_\leftarrow,\ \mathsf{ite}(\boldsymbol{y} = \boldsymbol{y}^k_\leftarrow,\ \boldsymbol{y}^k_\rightarrow,\ \boldsymbol{y}))))$

$\tilde{\exists}\sigma.\ \alpha := \exists \boldsymbol{y}^1_\rightarrow,\ \boldsymbol{y}^1_\leftarrow,\ \ldots\ldots,\ \boldsymbol{y}^k_\rightarrow \boldsymbol{y}^k_\leftarrow.\ \bigwedge_{i<j} (\boldsymbol{y}^i_\rightarrow \neq \boldsymbol{y}^j_\rightarrow \wedge \boldsymbol{y}^i_\leftarrow \neq \boldsymbol{y}^j_\leftarrow \wedge \boldsymbol{y}^i_\rightarrow \neq \boldsymbol{y}^j_\leftarrow) \wedge \alpha$

C 示例说明

我们补充介绍一些评估示例。对于每个示例，我们简要描述系统及其时序性质。

Paxos。 Paxos 共识协议 [26] 允许节点在多个轮次（投票轮次）中提出值并进行投票，以就某个一致值达成共识。我们验证的时序性质是：在假设存在某个最大轮次 r_0，且此后没有节点再提出值的情况下，存在一组响应及时的节点构成法定人数；若在 r_0 中提出了某个值，则最终有一个法定人数的节点就该值达成一致。

ABP。 交替位协议 [3] 是一种用于在发送方和接收方之间通过有损先进先出信道进行消息传递的协议。发送方反复发送数据消息，其中包含相应的数据和发送方的位；接收方则反复发送带有接收方位的确认消息。当接收方收到的消息所带的位与自身位不同时，就会记录数据并翻转自身位。同样，当发送方收到的消息所带的位与自身位不一致时，就会翻转自身位并转到下一个数据条目。我们验证的时序性质是：在通信信道满足弱公平性且消息丢失次数无上限的情况下，发送方的所有数据最终都会被接收方收到。

HRB-Correct 与 ***HRB-Relay*** 混合可靠广播协议 [40] 允许节点在混合可靠性保证下广播消息。节点有 4 种故障模式：对称遗漏故障、非对称遗漏故障、对称故障或任意故障。在对每种故障模式的故障数量作出适当假设后，遵循协议的节点只能接受已向网络发起的消息。我们验证两种时序性质。正确性：如果一条消息已发送给网络中的所有非故障节点，那么最终会有某个遵循协议的节点接受该消息。中继：如果某个遵循协议的节点接受了一条消息，那么最终所有非故障节点都会接受该消息。这两种性质均在以下假设下得到验证：收到初始消息的非故障节点最终会发送该消息；收到发送给自己的消息的非故障节点最终会接收该消息。

阿克曼。 阿克曼函数 [2] 是一种快速增长的递归函数，定义为 $A(0,\ n) = n+1$, $A(m,\ 0) = A(m-1,\ 1)$ $A(m,\ n) = A(m-1,\ A(m,\ n-1))$。我们考虑的迁移系统是阿克曼函数 [19] 的一种栈实现。我们验证的时序性质是该迁移系统的终止性，这等价于阿克曼函数的全定义性。

队列示例。 我们证明了文献 [29] 中的三个启发性示例。在每个示例中，我们都证明了这样一项活性性质：任何发送的消息最终都会被接收。在 TimestampedQueue 中，一串带时间戳的消息通过一个 FIFO 队列传递，并假设该队列会被无限次轮询。在 CascadingQueue 中，这样一串消息在两个连续队列之间传递，并假设这两个队列都会被无限次轮询。在 ReorderingQueue 中，两个发送队列合并为一个到达队列，并假设每个队列都会被无限次轮询。

MutexRing。 一种用于环形排列节点的互斥算法，其中临界区的访问由是否持有令牌决定（摘自 [16]）。节点可处于以下状态：中立、等待或临界；只有持有令牌时，节点才能从中立状态转为临界状态；只有未持有令牌时，才能从中立状态转为等待状态。退出临界区时，节点会将令牌传递给右侧邻居。我们在公平调度下验证了互斥性。

LeaderRing. 环上的领导者选举协议 [8]。节点按序构成一个环，每个节点都有一个唯一标识符。初始时，每个节点都会发送自己的标识符，然后等待接收左侧邻居发送的标识符。收到标识符后，如果该标识符大于自己的标识符，节点就会将其传递给右侧邻居。如果节点收到自己的标识符，就可以宣布自己是领导者。我们要验证的时序性质是：在对所有节点进行公平调度的情况下，最终会选出一位领导者。

ToyStabilization. 这是对 Dijkstra k状态协议的简化版本，仅体现特权的移动（参见示例 [28]）。我们验证最终每个节点都会获得调度。

Dijsktra k-State. Dijkstra 的 k状态自稳定协议 [13]。该协议的状态由一个包含 n 个节点的环构成，每个节点都持有 $\{1, \ldots, k\}$中的一个值，其中 $k > n$。每个节点可能具有特权，也可能不具有特权，其定义取决于自身值与左侧邻居值之间的关系。每一步中，某个具有特权的节点执行一步操作，从而失去特权，并可能使右侧邻居获得特权。我们要验证的是，无论如何调度具有特权的节点，系统最终都会进入一个稳定状态，其中恰好有一个节点具有特权。证明分为三个时序性质，依据 [23]。

二进制计数器。 一种用数组实现的二进制计数器，从$2^n - 1$ 递减至 0，其实现方式确保每次操作只修改一个单元。我们验证该程序会终止。

SAT 回溯。 一种用于求解命题可满足问题的确定性回溯算法。该算法按固定顺序将每个变量先设为真；在因某个子句不满足而回溯后，再将其设为假。如果当前赋值满足所有子句，算法就返回可满足；否则回溯。当从第一个已设为假的变量处回溯时，算法返回不可满足。我们验证该算法会终止。

SAT 冲突驱动子句学习。 命题可满足性问题的 CDCL 算法 [4]。该算法非确定性地执行变量赋值决策、单元传播、发生矛盾时的回跳以及蕴含子句的学习。我们验证该算法会终止，并返回可满足或不可满足。