arXiv:2512.23344v1 [cs.LO] 29 Dec 2025

验证 g 反应式系统中的异步超属性

RAVEN BEUTNER，CISPA 亥姆霍兹信息安全中心，德国BERND FINKBEINER，CISPA 亥姆霍兹信息安全中心，德国

超属性是关联多个执行轨迹的系统属性，在描述信息流和安全策略时十分常见。HyperLTL 等逻辑通过对执行轨迹进行显式量化，来表达反应式系统中的时态超属性，即沿无限执行轨迹推理时序行为的超属性。这类逻辑一个通常不受欢迎的副作用是，它们会同步地比较量化轨迹。这使得这些逻辑无法表达异步比较多条轨迹的属性，例如 Zdancewic 和 Myers 的观察确定性、McLean 的非推断或停顿精化。我们研究一种异步 *HyperLTL*（A-HLTL）的变体的模型检测问题。A-HLTL 是一种时序逻辑，能够表达跨时间步比较多条轨迹的超属性。除了对系统轨迹进行量化之外，A-HLTL 还支持对这些轨迹的停顿进行二级量化。因此，A-HLTL 能够简洁地规约许多广泛使用的异步超属性。A-HLTL 的模型检测需要找到合适的停顿；迄今为止，这仅适用于非常受限的片段或终止的系统。在本文中，我们针对任意 $\forall^*\exists^*$ A-HLTL 公式在反应式系统中的验证，提出一种新型的基于博弈的方法。在我们的方法中，我们将验证视为验证者与反驳者之间进行的博弈：双方通过控制底层轨迹和停顿的部分内容相互挑战。验证者的获胜策略随后对应于存在量化轨迹的具体见证，以及存在量化停顿的异步对齐。我们确定了若干片段，对于这些片段，我们基于博弈的解释是完备的，因而构成一种有限状态判定过程。我们为有限状态系统实现了一个原型，并报告了令人鼓舞的实验结果。

计算理论：• **模态逻辑和时态逻辑**；**逻辑与验证**；**通过模型检测进行验证**；• **安全与隐私** → **逻辑与验证**。

附加关键词与短语：时序逻辑、HyperLTL、异步 HyperLTL、模型检测、基于博弈的语义、观察确定性、细化、超活性

ACM 引用格式：
Raven Beutner 和 Bernd Finkbeiner。2025。验证反应式系统中的异步超属性。《*ACM* 程序设计语言会议录》9，OOPSLA2，文章 352（2025年10月），38 页。 https://doi.org/10.1145/3763130

1 引言

2008 年，Clarkson 和 Schneider [2008]创造了超属性这一术语，用来指称一类丰富的系统需求，这些需求关联多次计算。与传统的轨迹属性（即针对单次执行进行推理的属性，形式化定义为轨迹集合）不同，超属性（形式化定义为轨迹集合的集合）刻画了多次计算之间的交互。这涵盖了范围广泛的需求，包括信息流策略 [Goguen 和 Meseguer 1982；Guarnieri 等人 2020], 鲁棒性 [Biewer 等人 2022],连续性 [Chaudhuri 等人 2012],知识 [Beutner 等人 2023；Bozzelli 等人 2015], 以及线性化 [Herlihyand Wing 1990]。例如，Zdancewic 和 Myers [2003] 对观察确定性(OD)的开创性定义

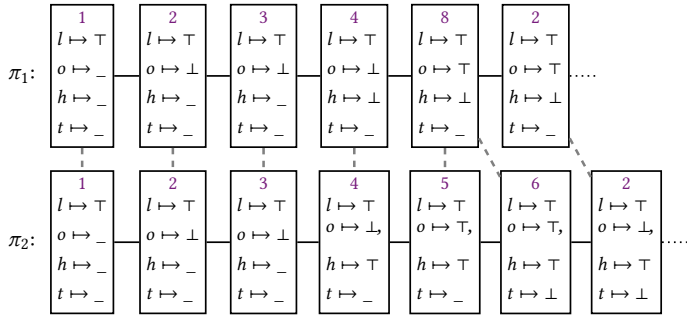
作者联系信息：Raven Beutner，raven.beutner@cispa.de，CISPA 亥姆霍兹信息安全中心，德国；Bernd Finkbeiner，finkbeiner@cispa.de，CISPA 亥姆霍兹信息安全中心，德国。



本作品根据知识共享署名 4.0 国际许可协议授权。©2025 版权所有归所有者/作者所有。ACM 2475-1421/2025/10-ART352https://doi.org/10.1145/3763130

```
1 o = ⊥
2 重复
3   h = 读取输入_H()
4   如果 h 那么
5     t = o 异或 l
6     o = t
7   else
8     o = o 异或 l
```

(a)



(b)

图1。图 1a 展示了一个涉及以下变量的布尔程序： o、 l、 h和 t。在图 1b中，我们展示了两条执行轨迹π_1、 π_2。每个状态都包含当前程序行以及所有变量的当前值（初始时，我们将低安全级输入 l 设为 $\top$）。在 π_1上，第 3 行的读取操作将 h 设为 $\bot$；而在 π_2 上，它将 h 赋值为 $\top$。

要求所有具有相同低安全级输入的执行对都具有相同的低安全级观测序列。同样，McLean [1994]的非推断要求，对于系统的每次执行，都存在一次匹配的执行：即使高安全级输入固定为“虚拟”输入，其低安全级观测（序列）也相同；两者都是超属性。

然而，Clarkson 和 Schneider [2008]的通用定义缺少一种用于描述超属性的具体规范语言。HyperLTL 的引入 [Clarkson 等人 2014] 推动了用于表达和推理反应式系统（即持续与环境交互并因此产生无限 执行轨迹的系统）中超属性的时序 逻辑的广泛发展。HyperLTL 扩展了线性时序逻辑（LTL） [Pnueli 1977]，增加了对系统中轨迹的显式量化。例如，我们可以将观察确定性的一种简化形式表示为 $\varphi_{OD}^{syn} := \forall \pi_1.\forall \pi_2.(l_{\pi_1} = l_{\pi_2}) \rightarrow (o_{\pi_1} = o_{\pi_2})$，这表示所有初始低安全级输入相同（通过变量 l 建模）的执行对 π_1、 π_2 都应当产生相同的输出（通过 o建模），此性质在全局范围内成立（通过 LTL 的全局算子表示）。

关键在于，HyperLTL 的语义——以及大多数其他用于时态超属性的逻辑的语义[Ábrahám 等人 2020； Ábrahám 和 Bonakdarpour 2018； Bajwa 等人 2023； Beutner 和 Finkbeiner 2023b； Beutner 等人 2023；Coenen 等人 2019a；Dimitrova 等人 2020；Finkbeiner 等人 2020；Giacomo 等人 2021；Gutsfeld 等人 2020；Rabe 2016] ——是同步的。也就是说，这种逻辑可以关联系统中的多条轨迹，但在对 LTL 主体求值期间，所有轨迹上的时间都步调一致地推进。对于许多属性而言，这种限制过于严格。举个简单的例子，考虑图1a中的程序。程序根据（布尔型）高安全级输入 h 进行分支，并在两个分支中都将 o 更新为 o 异或 l。因此， l 的初始值相等就意味着输出序列相同；该程序满足 Zdancewic 和 Myers [2003]提出的 OD 原始定义。然而，该程序违反了 φ_{OD}^{syn} OD 中表达的同步 HyperLTL 属性；在第一个分支中， o 的更新延迟了一步（因为第5行中间写入了 t），因此 LTL 主体在同步求值过程中被违反。图1b展示了图1a的两个示例执行，说明不同的执行会如何延迟输出发生变化的时间点。

异步超属性。 因此，许多安全和信息流属性——尤其是在研究 分布式 程序时——无法用 HyperLTL 严格的同步语义来表达。这鲜明地揭示了两者之间的差异：按照 Clarkson 和 Schneider [2008]的定义，超属性从来就不是“同步的”，而是被宽泛地定义为 轨迹集合的集合，且重要属性 [McCullough 1988； McLean 1994； Zdancewic 和 Myers 2003] 唯一原因

涉及事件序列，且不受任何形式的同步时间步限制。然而，大多数用于表达时态超属性的现有逻辑都会强制对所有轨迹进行同步遍历。

异步 *HyperLTL*。事实证明，我们可以借助一个简单的想法，大幅扩展 HyperLTL 的表达能力，从而精确刻画许多同步超逻辑无法表达的安全属性：停顿。形式上，异步 HyperLTL（简称 A-HLTL）通过显式量化系统轨迹的停顿，对 HyperLTL 进行了扩展 [Baumeister 等人 2021]。本文采用 A-HLTL 的一种新颖（但等价）变体，它直接对停顿轨迹进行量化，而不使用所谓的轨迹序列；Baumeister 等人 [2021] 所用的正是这种轨迹序列（详见第 4 节）。给定系统中的一条轨迹 π，我们用 $\beta \blacktriangleright \pi$ 表示轨迹 β 是轨迹 π 的一种公平停顿。由此，我们可以在 A-HLTL 中表述

$$\forall \pi_1.\forall \pi_2.\exists \beta_1 \blacktriangleright \pi_1.\exists \beta_2 \blacktriangleright \pi_2.\ (l_{\beta_1} = l_{\beta_2}) \rightarrow \square(o_{\beta_1} = o_{\beta_2}) \qquad (\varphi_{OD})$$

要求对于系统中的每一对轨迹 π_1, π_2，都存在 π_1 的停顿 β_1 和 π_2 的停顿 β_2，使得停顿后的轨迹 β_1, β_2 满足 $(l_{\beta_1} = l_{\beta_2}) \rightarrow (o_{\beta_1} = o_{\beta_2})$。$\varphi_{OD}$ 因而要求所有初始 l 值相等的轨迹对 π_1，π_2 在 o 上停顿等价，从而精确刻画了 Zdancewic 和 Myers [2003] 最初对观察确定性的定义。图1a中的程序满足 φ_{OD}：任意两条轨迹都经过相同的输出值序列，因此可以通过适当停顿使它们对齐。在图1b中，我们用虚线展示了一种可能的停顿方式，它能使输出对齐。

验证异步 *HyperLTL*。Baumeister 等人 [2021]和Hsu 等人 [2023]证明，A-HLTL 是一种表达力强大的逻辑，能够简洁地对许多常用的信息流策略和异步细化属性进行高层规约（更多详情请参见第8节）。特别是，我们可以使用 A-HLTL 直接量化停顿，而无须手动识别对齐点（例如系统输出发生变化的位置，或细化属性中的不可见步骤）；详见第8 节的详细讨论。尽管 A-HLTL 看起来只是对 HyperLTL 的简单扩展，但对停顿的量化显著增加了模型检测的复杂性，因此我们无法应用为 HyperLTL 开发的现有算法（这些算法通常严重依赖 HyperLTL 的同步求值，例如使用自动机概括轨迹组合 [Beutner 和 Finkbeiner 2023a；Finkbeiner 等人 2015]）。相反，现有的 A-HLTL 方法要么采用系统的有界展开，因此仅适用于终止系统（即在经过固定步数后到达最终状态的系统）[Hsu 等人 2023]，要么考虑 A-HLTL 的受限片段，这些片段可以手动化简为 HyperLTL [Baumeister 等人 2021]。

A-HLTL 的基于博弈的验证。本文提出了一种用于验证反应式系统中 A-HLTL 的新方法，其可验证的相关性质远超以往方法所支持的片段。在我们的方法中，我们将 A-HLTL 公式的验证解释为验证者与反驳者之间的一场博弈；这与针对同步 HyperLTL 的成功近似方法类似 [Beutner 和 Finkbeiner 2022a；Coenen 等人 2019b]。在我们的博弈中，我们逐步构造轨迹和停顿，并将轨迹的控制权——以及其停顿的控制权——交给玩家。直观而言，在每一步中，反驳者都可以扩展所有全称量化轨迹，并决定是否推进或停顿全称量化停顿。验证者可以对所有存在量化轨迹和停顿做出类似的决定。我们在第 2 节概述了这一方法。我们证明，基于博弈的近似是可靠的：如果验证者赢得博弈，那么 A-HLTL 公式就成立。由于 A-HLTL 的模型检测通常不可判定（即使在有限状态系统中也是如此！）[Baumeister 等人 2021], 我们的方法必然是不完备的。我们找出了若干 A-HLTL 片段，对于这些片段，我们基于博弈的解释是完备的，因此

构成有限状态判定过程。特别是，我们证明，对于以往方法所支持的片段，我们的方法是完备的 [Baumeister 等人 2021；Hsu 等人 2023]。

实现。 我们的基于博弈的验证方法适用于有限状态与无限状态系统（见第8节）。为了与现有方法和工具进行比较，我们将面向有限状态迁移系统的验证方法实现为原型工具，并将其与有界模型检测方法进行比较 [Hsu 等人 2023]。

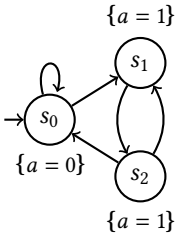


图 2. 示例系统

结构与贡献。 简而言之，我们的贡献包括以下内容： **(1)** 我们提出了 A-HLTL 的一个更简单（但等价）的变体，它直接对停顿进行量化（第 4节）； **(2)** 我们为验证以 A-HLTL 表示的异步超性质提出了一种新型的基于博弈的语义（第 5节）； **(3)** 我们确定了一些片段，在这些片段中，我们基于博弈的解释是完备的，因此可为有限状态系统提供判定过程。尤为重要的是，这包括所有可容许公式 [Baumeister 等人 2021]，，对于这些公式，我们证明验证者可以遵循一种规范的（“最大”）停顿策略（第6节）； **(4)** 我们将方法应用于有限状态系统，并在现有基准测试上进行评估（第7节）。

2 概述

在本节中，我们将说明如何利用博弈来验证对停顿进行量化的超属性。考虑图 2 中的 3 状态迁移系统 T，其中每个状态都会为变量a 赋予某个整数值。考虑以下 A-HLTL 公式

$$\forall \pi_1.\exists \pi_2.\exists \beta_1 \blacktriangleright \pi_1.\exists \beta_2 \blacktriangleright \pi_2.\bigcirc\Box(a_{\beta_1} \neq a_{\beta_2}). \qquad (\varphi_{fair})$$

该公式要求，对于任意执行 π_1，都存在某个执行 π_2 以及某些停顿β_1， β_2 （分别针对 π_1 和 π_2），使得从第二步开始（使用线性时序逻辑的下一个算子）， a 在 β_1 与 β_2中的取值不同。[1] 该属性在 T 上成立。例如，考虑轨迹$\pi_1 = \{a=0\}\{a=0\}(\{a=1\})^\omega$。我们可以通过选择 $\pi_2 = \{a=0\}\{a=1\}\{a=1\}(\{a=0\})^\omega$来匹配这一执行，并通过选取停顿 $\beta_1 = \{a=0\}\{a=0\}\{a=0\}(\{a=1\})^\omega$和 $\beta_2 = \pi_2$来对齐两条轨迹（注意，“显而易见”的选择 $\pi_2 = \{a=0\}\{a=1\}(\{a=0\})^\omega$并非 T 中的一条轨迹）。

直接验证此属性时，对于 π_1 的每一种选择（量化范围为 T 中的无限轨迹），我们都需要提供一条见证轨迹 π_2 并 为 π_1， π_2的每种组合找到恰当的停顿。我们不直接着手解决这个问题，而是将验证近似为验证者与反驳者之间的一场博弈。[2] 我们这场博弈的主要思想是，将 π_1 和 π_2 的路径以及停顿 β_1， β_2 视为一场无限时长双人博弈的结果：反驳者控制全称 量化的轨迹和停顿，验证者控制存在 量化的轨迹和停顿。对于示例公式φ_{fair}，反驳者沿着 T 的转移逐步构造（全称量化的）轨迹 π_1 。验证者则可以通过遍历 T 的另一个副本作出回应，从而生成 π_2的轨迹，并控制停顿 β_1， β_2。因此，每一场无限博弈都会生成无限轨迹，分别对应

[1]我们在此考察的 A-HLTL 公式是一个简单示例，旨在展示我们验证方法的核心思想。不过，如果我们将 $a = v$(for $v \in \{0,1\}$) 解读为“向进程 v授予了授权”，就可以将 φ_{fair} 视为一种简化的公平性属性：它要求，对于每个执行，都存在另一个执行，使其产生完全相反的授权分配（允许停顿），也就是说，两个进程获得授权的次数相同。[2]此类近似方法已成功应用于同步 HyperLTL 的验证 [Beutner 和 Finkbeiner 2022a、2024、2025c；Coenen 等人 2019b]，，在这类场景中，有限状态模型检测是可判定的，而博弈则提供了一种计算成本更低（但不完备）的近似方法。在我们的 A-HLTL 场景中，模型检测是不可判定的，因此我们设计了一种新型博弈，作为 A-HLTL 的可靠近似方法。

π_1 和 π_2 以及停顿 β_1，β_2 的 π_1，π_2；随后，验证者的目标是确保这些停顿 β_1，β_2（合在一起）满足 $(a_{\beta_1} \neq a_{\beta_2})$，即 φ_{fair} 的 LTL 主体。

基于图的博弈。 为形式化反驳者与验证者之间的交互，我们构造了一个基于图的博弈。该博弈由一组通过边连接的顶点组成，每个顶点被分配给反驳者或验证者。博弈通过将一个标记从一个顶点移动到另一个顶点来推进，而控制当前顶点的玩家可以决定标记应沿哪条出边移动。因此，给定顶点的出边决定了玩家在特定情况下可以做出的选择（关于基于图的博弈的更多细节，请参见第 3节）。

验证博弈。 我们针对 A-HLTL 的验证博弈建立在两个基本思想之上：状态窗口和相对指针。反驳者（分别为验证者）通过在 T 中移动来控制轨迹 π_1（分别为 π_2）。鉴于 β_1 和 β_2 是 π_1 和 π_2 的停顿（即，在停顿的第 i步中，β_1 可以等于 π_1 的第 j步，具体取决于停顿的速度），为了适应这一事实，我们不维护单个状态，而是维护一个*状态窗口*，可以将其视为轨迹的*滑动窗口*。随后，我们将停顿 β_1、β_2 标识为指向 π_1 和 π_2的状态窗口的*相对指针*。直观地说，该指针标识停顿的当前位置，即，β_1（分别为 β_2）指向 π_1（分别为 π_2）的（状态窗口中的）一个位置。每当我们需要 β_1的当前状态时，也就是，我们获取 π_1的窗口中由 β_1 指针所指向的状态。

随后，每个顶点都属于三种博弈状态之一：$\cup$ 阶段（更新阶段）、$\forall$ 阶段或 $\exists$ 阶段。验证者控制 $\exists$ 阶段中的所有顶点，而反驳者控制 $\forall$ 阶段中的顶点（$\cup$ 阶段中的顶点只有一个唯一后继，因此由哪位玩家控制并不重要）。每当博弈处于 $\cup$ 阶段时，我们都会考察 β_1, β_2 的当前状态，并对 LTL 公式 $(a_{\beta_1} \neq a_{\beta_2})$ 求值一步。因此，要获胜，验证者必须确保每当博弈处于 $\cup$ 阶段时（首次访问除外），$a_{\beta_1} \neq a_{\beta_2}$ 成立。[3] 在 $\cup$ 阶段之后，博弈进入 $\forall$ 阶段。在 $\forall$ 阶段，反驳者可以推进（全称量化的）π_1；π_2 的状态窗口保持不变。由于我们跟踪的是一个状态窗口，这相当于沿着 T 的转移将一个状态追加到该窗口中。在 $\exists$ 阶段，验证者可以通过向 π_2 的窗口追加一个状态来作出响应。此外，验证者还可以决定（存在量化的）停顿 β_1 和 β_2 是否应推进到 π_1 和 π_2 的状态窗口中的下一个状态。因此，验证者可以通过控制相对指针来（隐式地）控制停顿 β_1, β_2。通过推进 β_1-指针在 π_1 上的位置（即，将 β_1-指针移动到分配给 π_1 的状态窗口中的下一个状态），停顿 β_1 就会执行一步正常的（非停顿）步；如果验证者保持 β_1-指针不变，则实际上是通过重复该指针在上一回合所指向的同一状态来执行停顿步。

获胜策略。 在我们的示例中，系统 T 由有限个状态组成，因此生成的基于图的博弈只包含有限个顶点。图 3 展示了验证者的一段获胜策略。[4] 每个顶点都将 π_1 和 π_2 映射到一个非空的状态窗口（序列），停顿 β_1, β_2 则用箭头表示。我们的博弈从初始顶点 v_0开始，在该顶点，π_1和 π_2 映射到包含 T 初始状态的单元素窗口 $[s_0]$，而 β_1 和 β_2 指针指向唯一的第一个位置。在 v_1 中（处于 $\forall$阶段），反驳者现在可以为 π_1 选择一个后继状态：要么将状态 s_0（即移动到 v_2）追加到 π_1的状态窗口中，要么将状态 s_1（即移动到 v_8）追加到其中。我们重点讨论前一种情况。在 v_2 中（处于 $\exists$阶段），验证者可以向[4] 的窗口追加一个状态

[3]为了跟踪这一时序性质，我们将 LTL 公式转换为确定性自动机，并在每个 $\cup$-阶段顶点中跟踪该自动机的状态。为简洁起见，本概述部分省略了这个自动机。[4]验证者的（位置）策略是一个函数：对于每个由验证者控制的顶点，该函数指定一个具体的后继顶点（见第 3 节）。我们将策略可视化为对博弈图的限制。也就是说，对于所有由反驳者控制的顶点，我们保留所有可能的后继顶点；而对于所有由验证者控制的顶点，我们固定一条特定的出边。

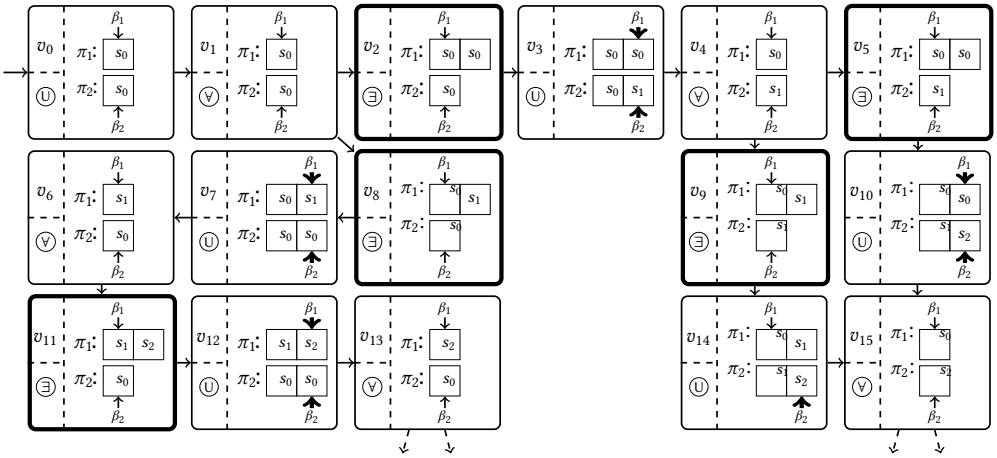


图3. 我们展示了图 2中系统的验证博弈里验证者的一种获胜策略（的部分）。每个顶点都映射到某个状态窗口 π_1, π_2 并跟踪{ ∪ ⓥ ∃}中的当前阶段。我们将这些顶点标记为 v_0, ⋯ v_{15}。验证者控制 ∃阶段中⓪所有顶点，这些顶点以粗边框标出。

并决定是否应推进 β_1 和 β_2 指针。此处展示的获胜策略会移动到 v_3，即追加状态 s_1 并同时推进 β_1 和 β_2。在 v_3 中（处于 U阶段），β_1 指向状态 s_0，而 β_2 指向 s_1；根据 $(a_{\beta_1} \neq a_{\beta_2})$ 的要求，它们在 a的求值上有所不同。从 v_3出发，我们（确定性地）移动到 v_4，方法是移除 π_1 和 π_2的状态窗口中的第一个位置：β_1 和 β_2 指针（给出停顿的当前位置）指向各自状态窗口中的第二个位置，因此可以安全地丢弃位于停顿当前位置之前的所有位置。在 v_4中，反驳者可以再次将状态 s_0（即移动到 v_5）或 s_1（即移动到 v_9）追加到 π_1的状态窗口中。我们重点讨论后一种情况。在 v_9中，π_2 的当前窗口以状态 s_1结尾，因此验证者别无选择，只能追加 s_2（因为必须遵循 T 的转移）。验证者可以选择同时推进 β_1 和 β_2 指针，使它们分别指向状态 s_1 和 s_2。然而，这会导致验证者输掉博弈，因为在下一个 U阶段，$a_{\beta_1} \neq a_{\beta_2}$ 会被违反。相反，验证者移动到顶点 v_{14}，也就是说，只推进 β_2 指针，并让 β_1-指针保持不变，从而使 β_1停顿（即 β_1 指针仍指向状态 s_0，重复与上一回合相同的状态）。在v_{14}中，停顿 β_1 和 β_2 因此分别指向状态 s_0 和 s_2，满足 $a_{\beta_1} \neq a_{\beta_2}$。在$v_{14}$中，我们再次（确定性地）移除 π_2 窗口中的第一个位置。此外，还可以缩短 π_1的窗口，移除所有位于 β_1-指针之后的位置，从而确保状态窗口长度不会超过 2 个状态（即我们确定性地移动到顶点 v_{15}）。

图3 中的初始草图可以扩展为一个完整的策略，该策略**(1)** 确保$(a_{\beta_1} \neq a_{\beta_2})$成立（在∪阶段时，每次都评估一步），并且**(2)** 使 β_1 和 β_2指针无限次前进（即，停顿 β_1 和 β_2 是公平的）。值得注意的是，我们的博弈经过专门设计，其能力低于存在量化的能力：如果验证者能够为 π_2，β_1，β_2构造恰当的见证，无论反驳者如何在 π_1上构造，我们也能在给定整个（无限）轨迹 π_1 时为 π_2，β_1，β_2 找到见证（如同 A-HLTL 的语义）；因此，我们利用该博弈证明了 T 满足φ_{fair}。

我们强调，尽管这个示例是刻意简化的（用于演示），我们的方法仍是首个能够自动φ_{fair} 验证的方法：图 2 中的系统显然不会终止 [Hsu 等人 2023], 并且φ_{fair} 不可容许（参见第 6.3 节） [Baumeister 等人 2021]。

3 预备知识

我们用 $\mathbb{N}$ 表示自然数集合，用 $\mathbb{B} := \{\top, \bot\}$ 表示布尔值集合。给定一个集合X，我们用 X^ω表示其上的无限序列集合，用 $X^{\le n}$表示长度至多为 $n \in \mathbb{N}$的有限序列集合。对于 $u \in X^{\le n}$、 $i \in \mathbb{N}$和 $x \in X$，我们定义： $|u|$为 u的长度， $u(i) \in X$为 u 中的第 i个元素（从第 0 个开始），而 $u \cdot x$ 表示将 x追加到 u后得到的序列。对于 i、 $j \in \mathbb{N}$，我们定义 $u[i,j] := u(i)u(i+1)\cdots u(j)$，当 $|u| > j$时；否则（若 $|u| \le j$），$u[i,j] := u(i)u(i+1)\cdots u(|u|-1)$。最后，我们定义 $u[i,j) := u[i,j-1]$。我们使用 λ 记法表示未命名函数：例如， $\lambda n.\, n+1$ 表示函数 $\mathbb{N} \to \mathbb{N}$，它将每个$n \in \mathbb{N}$ 映射到 $n+1$。

一阶理论。HyperLTL [Clarkson 等人 2014]和 A-HLTL [Baumeister 等人 2021；Hsu 等人 2023] 均在迁移系统上求值，并通过有限多个原子命题（AP）访问当前状态，我们可以将其视为布尔变量。然而，在许多情况下，系统的变量并非布尔变量，而是取自更丰富的论域（例如整数集）。我们通过将一阶公式（模某个固定的一阶理论）纳入原子表达式，扩展 A-HLTL。也就是说，我们不再在原子命题层面访问轨迹，而是可以对复杂数据类型上的关系公式进行推理，并使用诸如 $=$、 $\neq$、 $\le$、 .....[Beutner 和 Finkbeiner，2025a] 等解释谓词。形式上，我们假设存在某个一阶签名和某个固定的背景理论 $\mathfrak{T}$，其论域为 $\mathbb{V}$ （我们可以将其视为所有值的集合）。对于变量集合 X，我们用 $\mathrm{F}_{\mathcal{X}}$ 表示变量 X 上所有一阶公式的集合。给定一个公式$\theta \in \mathrm{F}_{\mathcal{X}}$ 和变量赋值 $A : \mathrm{X} \to \mathbb{V}$，若 θ 在 A （模 $\mathfrak{T}$）下成立，我们记作 $A \models^{\mathfrak{T}} \theta$[Barwise 1977]。

迁移系统。作为基本计算模型，我们使用基于状态的迁移系统。为此，设 X 是一个固定的有限系统变量集。迁移系统（TS）是一个元组 $\mathrm{T} = (S, s_0, \kappa, \ell)$，其中 S是一个（可能无限的）状态集，$s_0 \in S$是一个初始状态， $\kappa : S \to (2^S \setminus \{\emptyset\})$ 是一个转移函数，而 $\ell : S \to (\mathrm{X} \to \mathbb{V})$ 是一个函数，它将每个状态映射为 X 上的一个变量赋值。T 中的一条路径是一个无限序列 $p = s_0 s_1 s_2 \cdots \in S^\omega$（起始于 s_0），满足对所有 $i \in \mathbb{N}$， $s_{i+1} \in \kappa(s_i)$。我们用 $\mathrm{Paths(T)} \subseteq S^\omega$表示 T 中所有路径的集合。给定一条路径 $p = s_0 s_1 s_2 \cdots$，其对应的轨迹通过逐点应用 ℓ得到，即 $\ell(p) := \ell(s_0)\ell(s_1)\ell(s_2)\cdots \in (\mathrm{X} \to \mathbb{V})^\omega$。我们用 $\mathrm{Traces(T)} \subseteq (\mathrm{X} \to \mathbb{V})^\omega$表示 T 所生成的所有轨迹的集合。给定两条轨迹 $t, t' \in (\mathrm{X} \to \mathbb{V})^\omega$，如果 t' 可通过对 t的任意位置进行任意（但有限）步数的停顿而得到（记作 $t' \blacktriangleright t$），则称 t' 是 t的公平停顿。形式化地， $t' \blacktriangleright t$ 当且仅当存在一个函数 $f : \mathbb{N} \to \mathbb{N}$，使得 (1) $\forall i \in \mathbb{N}.\, t'(i) = t(f(i))$，(2) f单调递增（即， $i \ge j$蕴含 $f(i) \ge f(j)$），并且 (3) f是满射（即， $\forall i \in \mathbb{N}.\, \exists j \in \mathbb{N}.\, f(j) = i$）。

Büchi 自动机。 在博弈过程中，我们需要跟踪验证者构造的轨迹和停顿是否满足 A-HLTL 公式的 LTL 主体。为此，我们将 LTL 公式转换为一个确定性 *Büchi* 自动机 (DBA)。DBA 类似于处理有限字的确定性有限自动机，但它接受无限 字。因此，我们可以将此自动机与反驳者和验证者的（无限）行为并行运行，并判断他们的无限对局是否满足 LTL 公式。[5] 字母表为 Σ的一个 (DBA) 是一个元组 $\mathrm{A} = (Q, q_0, \delta, F)$，其中 Q是有限状态集， $q_0 \in Q$是初始状态，$\delta : Q \times \Sigma \to Q$ 是确定性转移函数， $F \subseteq Q$ 是接受状态集。给定一个字 $u \in \Sigma^\omega$，A 存在唯一的运行 $q_0 q_1 \cdots \in Q^\omega$ （从 q_0开始），其定义为：对所有 $i \in \mathbb{N}$，$q_{i+1} = \delta(q_i, u(i))$。如果该运行无限多次访问 F，则称其为接受运行。我们用 $\mathrm{L(A)} \subseteq \Sigma^\omega$ 表示 A 的唯一运行为接受运行的字集。

[5]注意，并非每个线性时序逻辑公式都能转换为确定性 Büchi 自动机。我们可以改用稍复杂的确定性奇偶自动机，它们可以刻画所有 ω-正则属性 [Piterman 2007]。不过，为了简化表述，本文始终使用更简单的 Büchi 自动机。

Büchi 博弈。作为一种具体的博弈形式化方法，我们使用基于图的博弈，其中两名玩家沿着图的边，将一个标记从一个顶点移至另一个顶点。我们后续设计的博弈将使图中的出边准确反映当前博弈情境下任一玩家的行动能力（例如，将一个状态追加到状态窗口、推进一次停顿等）。验证者的目标是确保无限博弈过程满足 A-HLTL 公式的 LTL 主体。由于我们用 Büchi 自动机跟踪该 LTL 主体，因此自然得到验证者的布奇取胜目标。形式化地说，一个 Büchi 博弈是一个元组 $G = (V_{\mathfrak{B}}, V_{\mathfrak{R}}, \alpha, F)$，其中 $V := V_{\mathfrak{B}} \uplus V_{\mathfrak{R}}$ 是一个（可能无限的）顶点集（$V_{\mathfrak{B}}$ 中的顶点由验证者控制，$V_{\mathfrak{R}}$ 中的顶点由反驳者控制），$\alpha : V \to (2^V \setminus \{\emptyset\})$ 是转移函数，$F \subseteq V$ 是接受顶点集。G 中的一次博弈是一个无限序列 $\rho \in V^\omega$，满足对所有 $i \in \mathbb{N}$，$\rho(i+1) \in \alpha(\rho(i))$。如果博弈过程无限多次访问 F 中的状态，则 ρ 由 $\mathfrak{B}$ 获胜。位置策略（对验证者而言）（参见 [Martin1975]）是一个函数 $\sigma : V_{\mathfrak{B}} \to V$，满足对所有 $v \in V_{\mathfrak{B}}$，$\sigma(v) \in \alpha(v)$。如果对于每个满足 $\rho(i) \in V_{\mathfrak{B}}$ 的 $i \in \mathbb{N}$，都有 $\rho(i+1) = \sigma(\rho(i))$，则博弈过程 $\rho \in V^\omega$ 与 σ 相容。和第 2 节中一样，我们可以将策略视为对博弈状态空间的限制：保留由反驳者控制的所有顶点的所有出边；对于每个由验证者控制的顶点 $v \in V_{\mathfrak{B}}$，仅保留通往顶点 $\sigma(v)$ 的边。如果存在一个策略 σ，使得所有 σ- 相容的博弈过程 ρ（即受限顶点空间中的所有无限路径）只要从 v（即 $\rho(0) = v$）开始，就都由验证者获胜，那么验证者便赢得顶点 $v \in V$。给定顶点集 $X \subseteq V$，若验证者从任意顶点 $v \in X$ 出发都能赢得 G，我们记为 $\text{wins}_{\mathfrak{G}}(\mathfrak{B}, X)$。

4 异步 HyperLTL

HyperLTL 扩展了线性时序逻辑 [Pnueli 1977]，使其能够显式地对轨迹进行量化 [Clarkson 等人 2014]。随后，A-HLTL 在轨迹的一级量化基础上，进一步对这些轨迹的停顿进行二级量化。

备注 1. 在 *A-HLTL* 的原始变体中 *[Baumeister 等人 2021; Hsu 等人 2023]*，通过对所谓的轨迹序列进行量化，实现了对停顿的量化。例如，$\forall \pi_1. \forall \pi_2. E \tau. (o_{\pi_1, \tau} = o_{\pi_2, \tau})$ 表示，对于所有轨迹对 π_1, π_2，都存在某个轨迹序列 τ，使得两条轨迹在输出 o 上对齐。这里，轨迹序列对 (π_1, τ) 和 (π_2, τ) 分别表示对 π_1 和 π_2 的独立停顿。通过轨迹序列迂回处理会导致语义繁复，因为每个轨迹序列都隐式地对所有轨迹的停顿进行量化，详情见 *[Baumeister 等人 2021; Hsu 等人 2023]*。本文提出一种新颖的 *A-HLTL* 变体，通过显式地对轨迹的停顿进行量化。我们的变体更易理解，语义也更简单，并且与原始变体等价（见附录 A）。上述示例可在新的 *A-HLTL* 变体中表示为 $\forall \pi_1. \forall \pi_2. \exists \beta_1 \blacktriangleright \pi_1. \exists \beta_2 \blacktriangleright \pi_2. (o_{\beta_1} = o_{\beta_2})$；也就是说，对于每个轨迹-轨迹对 (π_1, τ)、(π_2, τ)，我们都会显式地对该轨迹的停顿进行量化。

语法。令 $V = \{\pi_1, \pi_2, \ldots, \pi_n\}$ 为一组轨迹变量，$B = \{\beta_1, \beta_2, \ldots, \beta_k\}$ 为一组停顿变量。我们将 $X_{\mathcal{B}} := \{x_\beta \mid x \in X \ \beta \in B\}$ 定义为以停顿变量为索引的系统变量集合。A-HLTL 公式 φ 由以下语法生成

$$\varphi := \exists \pi. \varphi \mid \forall \pi. \varphi \mid \phi$$
$$\phi := \exists \beta \blacktriangleright \pi. \phi \mid \forall \beta \blacktriangleright \pi. \phi \mid \psi$$
$$\psi := \theta \mid \neg \psi \mid \psi \wedge \psi \mid \bigcirc \psi \mid \psi \, \mathcal{U} \, \psi$$

其中 $\pi \in V$ 是轨迹变量，$\beta \in B$ 是停顿变量，$\theta \in \mathrm{F}^{X_{\mathcal{B}}}$ 是关于以停顿变量为索引的系统变量（即 $X_{\mathcal{B}}$）的一阶公式。我们分别用 ψ 和 ψ 表示线性时序逻辑的派生算子最终 和全局 。

语义。 根据上述语法，A-HLTL 公式的形式为

$$\varphi = \mathbb{Q}\pi_1 \ldots \mathbb{Q}\pi_n.\ \mathbb{Q}\beta_1 \blacktriangleright \pi_{l_1} \ldots \mathbb{Q}\beta_k \blacktriangleright \pi_{l_l}.\ \psi$$

其中， $\pi_1, \ldots, \pi_n \in \mathrm{V}$ 是轨迹变量， $\beta_1, \ldots, \beta_k \in \mathrm{B}$ 是停顿变量，用于量化轨迹 $\pi_{l_1}, \ldots, \pi_{l_l}$ 的停顿（分别对应，其中 $l_1, \ldots, l_k \in \{1, \ldots, n\}$）； $\mathbb{Q} \in \{\forall, \exists\}$ 是量词，而 ψ 是一个由 $\mathrm{F}x_{\mathcal{B}}$ 中的原子构成的 LTL 公式（即关于 $\mathrm{X}_{\mathcal{B}}$ 中变量的一阶公式）。与 HyperLTL 类似，我们首先按照典型的一阶语义，对迁移系统 T 中的轨迹 $\pi_1, \ldots, \pi_n$ 进行量化。在 n 条轨迹确定后，我们继续对这些轨迹的停顿进行二级量化，即 $\mathbb{Q}\beta \blacktriangleright \pi$ 对（先前量化的）轨迹 π 的公平停顿进行量化。解析量词前缀后，我们便得到 k 个停顿 $\beta_1, \ldots, \beta_k$，并可对 LTL 公式 ψ 求值。在此，LTL 公式 ψ 中的每个原子都是关于 $\mathrm{X}_{\mathcal{B}}$ 中变量的一阶公式（模 $\mathfrak{T}$），也就是说，它是关于由停顿变量索引的系统变量的公式。在这样的公式中，每个变量 $x_\beta \in \mathrm{X}_{\mathcal{B}}$ 都指代 x 在停顿 β 上的当前值。例如，在 φ_{OD} 中，我们使用了原子公式 $o_{\beta_1} = o_{\beta_2}$，其中 “=” 是理论 $\mathfrak{T}$ 中的一个解释谓词符号，而 $o_{\beta_1}, o_{\beta_2} \in \mathrm{X}_{\{\beta_1, \beta_2\}}$ 是带索引的系统变量。

为正式定义语义，我们维护一个轨迹赋值 $\Pi : \mathrm{V} \rightharpoonup (\mathrm{X} \to \mathbb{V})^\omega$，它将轨迹变量映射到轨迹（用于计算轨迹量化），以及一个停顿赋值 $\Delta : \mathrm{B} \rightharpoonup (\mathrm{X} \to \mathbb{V})^\omega$，它将停顿变量映射到轨迹（用于计算停顿量化）。给定一个停顿赋值 Δ 和位置 $i \in \mathbb{N}$，我们将变量赋值写作 $\Delta_{(i)} : \mathrm{X}_{\mathcal{B}} \to \mathbb{V}$，表示第 i 步中 $\mathrm{X}_{\mathcal{B}}$ 的变量赋值，其定义为 $\Delta_{(i)}(x_\beta) := \Delta(\beta)(i)(x)$，即，$x_\beta$ 的值定义为停顿 $\Delta(\beta)$ 上第 i 步中 x 的值。给定一组轨迹 $\mathbb{T} \subseteq (\mathrm{X} \to \mathbb{V})^\omega$ 和位置 $i \in \mathbb{N}$，我们将 A-HLTL 的语义定义如下

$$\begin{aligned}
&\Pi, \Delta, i \models_{\mathbb{T}} \exists \pi.\, \varphi \text{ 当且仅当 } \exists t \in \mathbb{T}.\, \Pi[\pi \mapsto t], \Delta, i \models_{\mathbb{T}} \varphi \Pi, \Delta, i \models_{\mathbb{T}} \forall \pi.\, \varphi \text{ 当且仅当} \\
&\forall t \in \mathbb{T}.\, \Pi[\pi \mapsto t], \Delta, i \models_{\mathbb{T}} \varphi \Pi, \Delta, i \models_{\mathbb{T}} \exists \beta \blacktriangleright \pi.\, \phi \text{ 当且仅当 } \exists t' \in (X \to \mathbb{V})^\omega.\, t' \blacktriangleright \Pi(\\
&\pi) \wedge \Pi, \Delta[\beta \mapsto t'], i \models_{\mathbb{T}} \phi \Pi, \Delta, i \models_{\mathbb{T}} \forall \beta \blacktriangleright \pi.\, \phi \text{ 当且仅当 } \forall t' \in (X \to \mathbb{V})^\omega.\, t' \blacktriangleright \Pi(\pi) \\
&\Rightarrow \Pi, \Delta[\beta \mapsto t'], i \models_{\mathbb{T}} \phi \Pi, \Delta, i \models_{\mathbb{T}} \theta \text{ 当且仅当 } \Delta_{(i)} \models^{\mathfrak{T}} \theta \Pi, \Delta, i \models_{\mathbb{T}} \neg\psi \text{ 当且仅当} \\
&\Pi, \Delta, i \not\models_{\mathbb{T}} \psi \Pi, \Delta, i \models_{\mathbb{T}} \psi_1 \wedge \psi_2 \text{ 当且仅当 } \Pi, \Delta, i \models_{\mathbb{T}} \psi_1 \text{ 且 } \Pi, \Delta, i \models_{\mathbb{T}} \psi_2 \Pi, \Delta, \\
&i \models_{\mathbb{T}} \bigcirc\psi \text{ 当且仅当 } \Pi, \Delta, i+1 \models_{\mathbb{T}} \psi \Pi, \Delta, i \models_{\mathbb{T}} \psi_1 \ U \ \psi_2 \text{ 当且仅当 } \exists j \geq i.\, \Pi, \Delta, \\
&j \models_{\mathbb{T}} \psi_2 \text{ 且 } \forall i \leq k < j.\, \Pi, \Delta, k \models_{\mathbb{T}} \psi_1.
\end{aligned}$$

首先，我们沿着轨迹量词前缀，在 $\mathbb{T}$ 中的轨迹上填充轨迹赋值 Π，并将轨迹添加到 Π。对于每个量化的停顿 $\mathbb{Q}\beta \blacktriangleright \pi$，接着我们对轨迹 $\Pi(\pi)$ 的一个停顿 t' 进行量化，并将其添加到 Δ。注意， t' 是通过对 $\Pi(\pi)$ 进行停顿而得到的任意轨迹；它未必是 $\mathbb{T}$ 中的轨迹。最后，我们可以按照布尔运算符和时序运算符的通常求值方式，对 Δ 中的轨迹求值 LTL 主体。这里，若变量赋值 $\Delta_{(i)}$（它为 $\mathrm{X}_{\mathcal{B}}$ 中的带索引变量赋值）满足 θ（模理论 $\mathfrak{T}$），则原子公式 θ 在步骤 i 成立。迁移系统 T 满足 A-HLTL 性质 φ，记作 $\mathrm{T} \models \varphi$，当且仅当 $\emptyset, \emptyset, 0 \models_{Traces(\mathcal{T})} \varphi$，其中 $\emptyset$ 表示定义域为空的轨迹／停顿赋值。

例*4.1*。 在大多数 A-HLTL 公式中，我们会对每个轨迹变量的 一个 停顿进行量化（通常是存在量化）。然而，A-HLTL 也可以对同一轨迹的多个停顿进行量化 [Hsu 等人 2023]。例如，在 φ_{OD} 第 1 节中，低安全级输入被固定 在初始时，并且在求值过程中从不改变（如 Zdancewic 和 Myers [2003]的原始定义所述）。如果

低安全级输入可以随时间变化，那么规约就需要考虑低安全级输入的无限序列 。我们可以如下表示这种扩展

$$\forall \pi_1.\forall \pi_2.\ \forall \beta_1 \blacktriangleright \pi_1.\forall \beta_2 \blacktriangleright \pi_2.\ \exists \beta_3 \blacktriangleright \pi_1.\exists \beta_4 \blacktriangleright \pi_2.\ \square(l_{\beta_1} = l_{\beta_2}) \rightarrow \square(o_{\beta_3} = o_{\beta_4}). \qquad (\varphi_{NI})$$

该公式要求存在 一组 停顿 β_3， β_4 使得 π_1， π_2 能够对齐输出 o（类似于 φ_{OD}），前提是全称量化的 停顿 β_1， β_2 能够对齐低安全级输入 l。换言之，任意两条具有停顿等价输入的轨迹都应产生停顿等价的输出。 △

例*4.2*。分析不同速度下的轨迹时，也需要多个停顿。我们通过考察文献中观察确定性的各种变体来说明这一点。给定一个低安全级输入 $l \in \mathrm{X}$ 和两个低安全级输出 a， $b \in \mathrm{X}$，A-HLTL 公式

$$\forall \pi_1.\forall \pi_2.\exists \beta_1 \blacktriangleright \pi_1.\exists \beta_2 \blacktriangleright \pi_2.\ (l_{\beta_1} = l_{\beta_2}) \rightarrow \square\left(a_{\beta_1} = a_{\beta_2} \wedge b_{\beta_1} = b_{\beta_2}\right)$$

要求 a和 b在所有轨迹对上 同时停顿等价（类似于 Terauchi [2008]；Zdancewic 和 Myers [2003] 的定义）。然而，尤其是在考虑分布式系统时，轨迹上的各个变量可能来自具有独立时序的不同分布式组件。在这种情况下，我们可以表达

$$\forall \pi_1.\forall \pi_2.\exists \beta_1 \blacktriangleright \pi_1, \beta_2 \blacktriangleright \pi_2, \beta_3 \blacktriangleright \pi_1, \beta_4 \blacktriangleright \pi_2.\ (l_{\beta_1} = l_{\beta_2}) \rightarrow \square\left(a_{\beta_1} = a_{\beta_2} \wedge b_{\beta_3} = b_{\beta_4}\right),$$

要求变量 a和 b分别具有停顿等价性，即我们可以分别为两个输出选择独立的停顿（类似于Bartocci 等人 [2023]的定义；Huisman 等人[2006]的定义）。△

例*4.3*。McLean [1994]的非推断概念指出，对于每条轨迹 π_1，都应存在某条轨迹 π_2，它具有相同的低安全级事件序列，但使用固定的哑元高安全级输入。关键在于，McLean [1994]的定义只考虑低安全级事件的（有序）序列，也就是说，它不考虑绝对的（同步）时间步。在 A-HLTL 中，我们可以将非推断直接表示为

$$\forall \pi_1.\exists \pi_2.\ \exists \beta_1 \blacktriangleright \pi_1, \beta_2 \blacktriangleright \pi_2.\ \square(\bigwedge_{x \in L} x_{\beta_1} = x_{\beta_2}) \wedge \square(\bigwedge_{x \in H} x_{\beta_2} = \dagger).$$

也就是说，对于每条轨迹 π_1，都存在某条轨迹 π_2，使得某些停顿 β_1， β_2 能够对齐， π_2 从而低安全级变量（$L \subseteq \mathrm{X}$）在停顿意义下相等（因此低安全级事件的序列相同）。与此同时， π_2 （$H \subseteq \mathrm{X}$）上的高安全级输入应全局设置为某个哑元值（我们用†表示该值）。△

我们强调，虽然停顿本身并没有特别有趣，但它是将许多广泛使用的信息流属性统一到同一种逻辑中的关键技术手段（另见第 8 节）。也就是说，通过扩展 HyperLTL，使其能够对轨迹停顿进行量化，我们便能表达范围广得多的信息流策略。因此，任何针对 A-HLTL 的验证方法都可应用于所有这些属性（以及其变体，例如将某个额外变量声明为输出），而不必针对每种新的信息流属性（或其变体）都“重新发明轮子”。

5 基于博弈的 A-HLTL 验证

如果在存在量化轨迹或停顿的作用域内不出现全称轨迹和停顿量词，我们就称 A-HLTL 公式为 a $\forall^*\exists^*$ 公式（参见例如 φ_{OD}和 φ_{NI}）。我们的基于博弈的验证方法旨在将 A-HLTL 公式中的量词交替近似为验证者（$\mathfrak{V}$）与反驳者（R）之间的一场博弈。反驳者控制全称量化轨迹

和停顿，验证者则通过更新存在量化轨迹和停顿来响应反驳者的每一步行动（参见第 2 节）。由于所有全称量词都先于任何存在量化，因此验证者根据 A-HLTL 语义中可用信息的一个子集来更新存在量化轨迹和停顿：在 A-HLTL 语义中，对于一个 $\forall^*\exists^*$ 公式，我们可以在所有全称量化轨迹和停顿都确定之后，再选择存在量化轨迹和停顿。而在我们的博弈中，验证者逐步选择它们，因此只知道一个有限的前缀。Baumeister 等人 [2021] 和Hsu 等人 [2023] 研究的所有 $\forall^*\exists^*$ 公式都是。

在下文中，我们令 $\mathrm{T} = (S_{\mathcal{T}}, s_0, \tau, \kappa_{\mathcal{T}}, \ell_{\mathcal{T}})$ 为一个固定系统，并令

$$\varphi = \mathbb{Q}\pi_1 \ldots \mathbb{Q}\pi_n.\ \mathbb{Q}\beta_1 \blacktriangleright \pi_{l_1} \ldots \mathbb{Q}\beta_k \blacktriangleright \pi_{l_k}.\ \psi$$

为一个固定的 $\forall^*\exists^*$ A-HLTL 公式（其中 $l_1, \ldots, l_k \in \{1, \ldots, n\}$）。我们将 $\mathrm{V}_\forall \subseteq \{\pi_1, \ldots, \pi_n\}$（分别为 $\mathrm{V}_\exists$）定义为所有全称（分别为存在）量化的轨迹变量，并将 $\mathrm{B}_\forall \subseteq \{\beta_1, \ldots, \beta_k\}$（分别为 $\mathrm{B}_\exists$）定义为所有全称（分别为存在）量化的停顿变量。对于每个停顿变量 $\beta \in \mathrm{B}$，我们将 基迹$(\beta) \in \mathrm{V}$ 定义为其所基于的唯一轨迹，即 β 所基于的唯一 π，其中量词前缀包含 $\mathbb{Q}\beta \blacktriangleright \pi$。例如，在 φ_{NI} 中，基迹(β_1) = 基迹(β_3) = π_1 以及基迹(β_2) = 基迹(β_4) = π_2。

在本节中，我们将构造一个 Büchi 博弈——记作 $\mathrm{G}_{\mathcal{T},\varphi,Z}$——如果验证者赢得该博弈，则意味着 $\mathrm{T} \models \varphi$。正如我们已经讨论过的，该博弈的高层思路是让验证者（分别是反驳者）控制所有存在量化（分别是全称量化）的轨迹和停顿。为将其形式化，我们将该博弈建模为一个基于图的博弈，其中每个顶点都记录博弈的当前状态（例如，状态窗口、停顿的位置等）。随后，我们设计博弈的转移，使得所有可能的后继顶点恰好对应于玩家可以采取的所有可能行动（例如，向状态窗口追加一个状态、推进一次停顿等）。

5.1 跟踪接受与公平性

随着博弈的进行，我们需要跟踪玩家构造的路径和停顿是否满足 ψ（φ 的 LTL 主体）。我们通过将 ψ 转换为确定性 Büchi 自动机来实现这一点。此外，我们还需要确保所有停顿都是公平的，即进展无穷多次。为此，我们考虑以下经过修改的 LTL 公式

$$\psi_{mod} := \Big(\bigwedge_{\beta \in \mathcal{B}_\forall} \square\diamond\, moved_\beta \Big) \rightarrow \Big(\big(\bigwedge_{\beta \in \mathcal{B}_\exists} \square\diamond\, moved_\beta \big) \wedge \psi \Big),$$

其中包含针对每个 $\beta \in \mathrm{B}$ 的新鲜布尔变量 已移动$_\beta$。其直观含义是：只要停顿 β 在上一回合执行了非停顿步，已移动$_\beta$就成立。LTL 公式 ψ_{mod} 随后要求存在量化停顿无限多次地推进（通过线性时序逻辑算子组合来表达），并且只要所有全称量化停顿（将由反驳者控制）都是公平的，ψ就必须成立。

令 $\Theta \subseteq \mathrm{F}^{x_\mathcal{B}}$ 为 ψ 中使用的有限一阶公式集合。例如，在 φ_{OD} 中，$\Theta = \{l_{\beta_1} = l_{\beta_2},\ o_{\beta_1} = o_{\beta_2}\}$。下文假设 $\mathrm{A}_\psi = (Q_\psi,\ q_0,\ \psi,\ \delta_\psi,\ F_\psi)$ 是一个以字母表$2^{\Theta \cup \{moved_\beta \mid \beta \in \mathcal{B}\}}$为字母表的确定性 Büchi 自动机，它恰好接受满足 ψ_{mod} 的那些无限字。也就是说，在每一步中，我们需要告知 A_ψ 的转移函数，Θ 中哪些公式成立，以及哪些停顿已推进（通过 已移动$_\beta$ 变量）；A_ψ 则跟踪这种行为是否满足 ψ_{mod}。

5.2 博弈顶点

在异步设置中，不同的停顿可能指向同一条轨迹上的不同位置。如第2节所述，我们通过维护一个窗口来处理这一情况：对于每个轨迹变量，我们跟踪 $\mathcal{T}$ 中连续状态组成的有限序列，类似于滑动窗口。

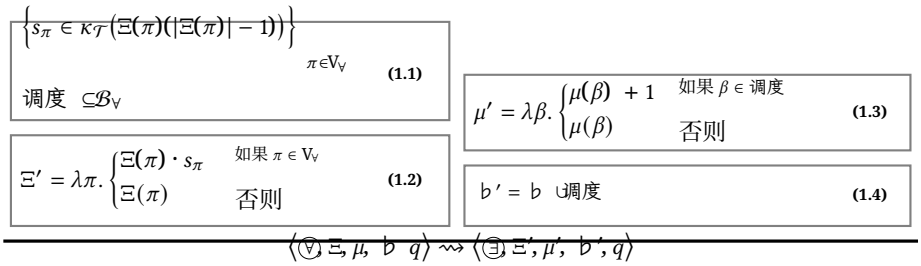


图 4. ∀ 阶段中顶点的转移。

每个停顿变量 β 都是指向分配给轨迹基迹(β)的状态窗口的（相对）指针，也就是说，该轨迹是 β 的停顿版本。我们用一个界 $Z \in \mathbb{N}$对博弈进行参数化，该界决定每个状态窗口的最大长度。如果每条轨迹只有一个停顿（大多数性质都是如此），那么大小为 $Z = 1$ 的窗口通常就足够了。如果考虑同一条轨迹上有多个停顿的性质，该界自然定义了博弈大小（较大的 Z 会生成更多顶点）与停顿的灵活性（较大的 Z允许同一条轨迹上的多个停顿彼此偏离得更远）之间的权衡。在许多情况下，我们可以静态推断出一个界 Z，同时保持完备性（参见第6节）。

现在可以定义用于验证 $T \models \varphi$ 的 Büchi 博弈 $G_{\mathcal{T},\varphi,Z}$。$G_{\mathcal{T},\varphi,Z}$ 中的每个常规博弈顶点都具有以下形式：阶段 $\Xi, \mu, \flat, q$，其中 (1) 阶段 $\in \{\forall, \exists, U\}$ 跟踪博弈的当前阶段；(2) $\Xi : V \to (S_\mathcal{T})^{\leq Z+1}$ 将每个轨迹变量映射到一个长度至多为 $Z+1$ 的状态窗口；(3) $\mu : B \to \{0, \ldots, Z\}$ 将每个停顿变量 β 映射到窗口 Ξ(基迹(β)) 内的一个相对位置；(4) $\flat \subseteq B$ 记录上一步中哪些停顿取得了进展（用于评估 moved$^\beta$）；以及 (5) $q \in Q_\psi$ 跟踪 A_ψ 的当前状态。除这些常规博弈顶点外，我们还添加了一个专用的错误顶点 v_{error}；当窗口界 Z 不足以容纳所有停顿时，博弈就会到达该顶点。

示例*5.1*。在本节中，我们将使用图3 中的示例来说明转移规则。在图3中，我们将顶点阶段 Ξ, μ, $\flat$ q表示如下：我们将$\Xi : \{\pi_1, \pi_2\} \to (S_\mathcal{T})^{\leq 2}$ 描绘为一个节点数组；将停顿指针 $\mu : \{\beta_1, \beta_2\} \to \{0, 1\}$描绘为箭头；当且仅当 $\beta \in \flat$时，将 β 对应的停顿指针加粗。例如，博弈顶点$\langle \forall, [\pi_1 \mapsto [s_0, s_1], \pi_2 \mapsto [s_1, s_2]], [\beta_1 \mapsto 0, \beta_2 \mapsto 1], \{\beta_2\}, _\rangle$表示为顶点 v_{14}。△

我们将最终的 Büchi 博弈定义为 $G_{\mathcal{T},\varphi,Z} := (V_\exists, V_\forall, \alpha, F)$，其中 $V_\exists$ 包含所有形如 $\exists, \Xi, \mu, \flat, q$的顶点，而$V_\forall$包含所有形如 $\forall, \Xi, \mu, \flat, q$、$U$ $\Xi, \mu, \flat, q$的顶点，以及错误顶点 v_{error}。对于接受状态，我们定义 $F := \{$阶段 $\Xi, \mu, \flat, q \mid q \in F_\psi\}$，也就是说，验证者通过无限次访问自动机状态为接受状态的顶点赢得一场博弈。因此，当且仅当 A_ψ 的模拟运行是接受的，也即当且仅当ψ_{mod} 在无限博弈中得到满足时，验证者才能赢得 $G_{\mathcal{T},\varphi,Z}$ 中的无限对局。转移函数 α 在第 5.3 节中定义。

5.3 转移规则

对于错误顶点 v_{error}，我们定义 $\alpha(v_{error}) := \{v_{error}\}$。由于 v_{error}是不包含在 F中的汇顶点，因此访问 v_{error} 会使验证者输掉博弈。对于每个常规博弈顶点 v，我们定义$\alpha(v) := \{v' \mid v \rightsquigarrow v'\}$，其中 $\rightsquigarrow$ 通过图 4 至 6 中的推理规则定义。此处，每条推理规则都会添加一个 $\rightsquigarrow$-转移，前提（位于推理规则上方）满足时方可添加。

$\forall$-阶段。对于每个顶点$\langle \forall, \Xi, \mu, \flat\, q\rangle$ 在 $\forall$-阶段中，反驳者可以推进所有全称量化的轨迹和停顿。从这类顶点出发的转移定义见图4。首先，

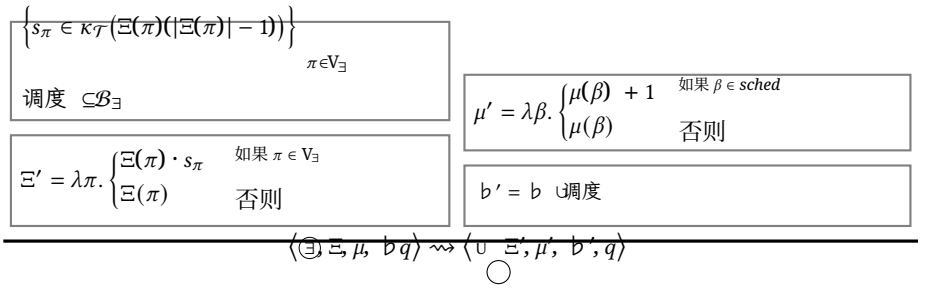


图 5. ∃ 阶段中顶点的转移。

对于所有全称量化轨迹，反驳者选择状态 $\{s_\pi \in \kappa_\mathcal{T}(\Xi(\pi)(|\Xi(\pi)|-1))\}_{\pi\in V_\forall}$（前提 (1.1)）。注意，每个 s_π 都是 T 中的后继状态，且是状态窗口$\Xi(\pi)$中的最后一个状态的后继（即 $\Xi(\pi)(|\Xi(\pi)|-1)$），因此，每个状态窗口始终对应于 T 中某条路径上的一个连续窗口。此外，反驳者可以通过选择一个子集 sched $\subseteq B_\forall$，决定要推进哪些全称量化停顿（前提 (1.1)）。固定 $\{s_\pi\}_{\pi\in\mathcal{V}_\forall}$和 sched 后，我们可以更新博弈顶点：对于每个 $\pi \in V_\forall$（前提 (1.2)），将 s_π 追加到状态窗口 $\Xi(\pi)$中。同时，sched 决定推进哪些（全称量化的）停顿（即通过移动到状态窗口中的下一个位置，执行一个非停顿步）。对于所有 $\beta \in$ sched，我们递增 $\mu(\beta)$ 指针（从而指向基迹(β)的状态窗口中的下一个状态），而对于所有其他停顿，则保持 $\mu(\beta)$ 不变（前提 (1.3)）。通过设置 $\flat' = \flat \cup$sched，记录哪些停顿已推进（前提 (1.4)）。注意，$\langle ⓥ, \Xi, \mu, \flat\ q\rangle$ 的每个可能后继顶点，都恰好对应于反驳者在各顶点中可以采取的预期操作（即扩展全称量化轨迹的轨迹窗口，并决定如何处理全称量化停顿）。

示例 5.2. 在图 3 的示例中，反驳者只控制轨迹变量 π_1（π_2, β_1, β_2 为存在量化）。例如，图 3 中的顶点 v_1 表示博弈顶点$\langle ⓥ, [\pi_1 \mapsto [s_0], \pi_2 \mapsto [s_0]], [\beta_1 \mapsto 0, \beta_2 \mapsto 0], \emptyset, _\rangle$。根据图 4 中的规则，该顶点有两个后继。在前提 (1.1) 中，我们可以选择 $s_{\pi_1} = s_0$ 或 $s_{\pi_1} = s_1$，它们是 s_0（π_1的窗口中的最后一个状态）的两个可能后继状态，参见图 2 中的迁移系统 T。随后将该状态追加到 π_1的状态窗口中，分别得到顶点 v_2 和 v_8。注意，停顿指针保持不变，因为 $B_\forall = \emptyset$。△

ⓔ-阶段。类似地，对于 ∃-阶段中的顶点，验证者可以推进图5中定义的所有存在量化轨迹和停顿。

示例5.3。考虑图 3 中的顶点 v_9，它表示博弈顶点 $\langle ⓔ, [\pi_1 \mapsto [s_0, s_1], \pi_2 \mapsto [s_1]], [\beta_1 \mapsto 0, \beta_2 \mapsto 0], \emptyset, _\rangle$。根据图 4 中的规则，该顶点有四个后继：$s_{\pi_2}$（即附加到 π_2 的状态窗口中的状态）只有一种选择，即 $s_{\pi_2} = s_2$（因为这是 T 中状态 s_1 的唯一后继，参见图 2）。对于调度，验证者可以选择任意 sched $\subseteq B_\exists = \{\beta_1, \beta_2\}$。我们选择 sched $= \{\beta_2\}$。根据图 5 中的转移规则，我们得到顶点 $\langle U\ [\pi_1 \mapsto [s_0, s_1], \pi_2 \mapsto [s_1, s_2]], [\beta_1 \mapsto 0, \beta_2 \mapsto 1], \{\beta_2\}\ _\rangle$，也就是说，我们将 s_2 附加到 π_2 的状态窗口，将 $\mu(\beta_2)$ 加 1（指向 π_2 状态窗口中的下一个状态），保持 $\mu(\beta_1)$ 不变（因此 β_1 仍指向 π_1 的状态窗口中与上一回合相同的状态，实际上使 β_1 停顿），并通过设置 $\flat = \{\beta_2\}$ 记录我们已推进了 β_2。在图 3 中，该顶点表示为 v_{14}。△

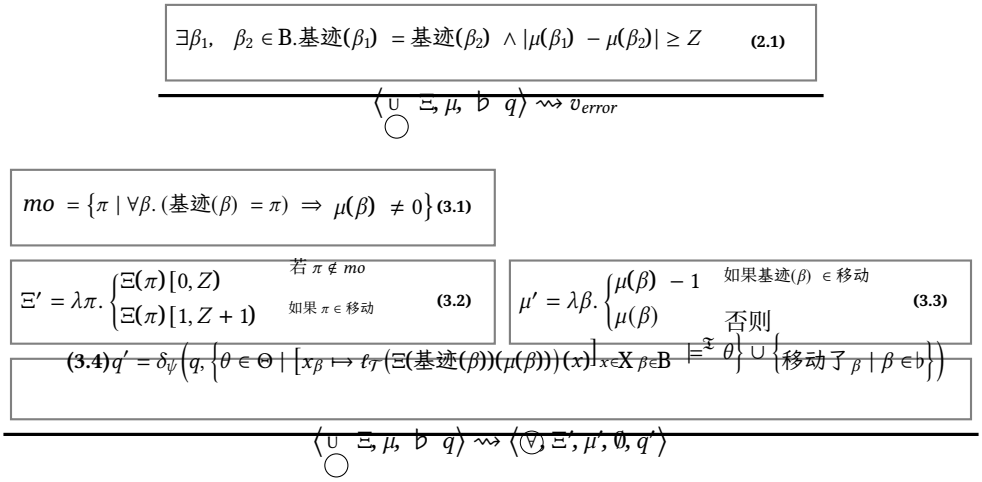


图 6. ∪ 阶段中顶点的转移。

∪阶段。∪阶段的转移定义见图 6。在 (2.1) 中，我们首先检查窗口大小 Z 是否不足以捕获某个轨迹变量的所有停顿，也就是说，是否存在两个停顿 β_1, β_2，它们指向同一个状态窗口（$\text{base}(\beta_1) = \text{base}(\beta_2)$），且相差至少 Z 步。如果是这样，我们就转到 v_{error}，从而使验证者失败。否则，我们重新进入 $\forall$ 阶段。同时，我们更新 Aψ 的自动机状态（前提 (3.4)）：为此，我们需要求值所有一阶公式 $\theta \in \Theta$（回想一下，$\Theta \subseteq \mathrm{F}^{\mathcal{X}_\mathcal{B}}$ 是用作 φ 的 LTL 主体 ψ 中原子的有限一阶公式集合）。在当前博弈顶点中，$\theta \in \Theta$ 成立，当且仅当 $\left[x_\beta \mapsto \ell_{\mathcal{T}}(\Xi(\text{base}(\beta))(\mu(\beta)))(x)\right] \models^{\mathfrak{T}} \theta$。也就是说，对于每个带索引变量 x_β，我们取对应于第 β 个停顿的当前状态，该状态是 $\text{base}(\beta)$ 的状态窗口中的第 $\mu(\beta)$ 个状态，即 $\Xi(\text{base}(\pi))(\mu(\beta))$，然后通过 T 的标记函数 $\ell_{\mathcal{T}}$ 查找 x 的值。为了对形式为 moved^β 的辅助命题求值（ψ_{mod} 用它来确保公平停顿，参见第 5.1 节），我们使用 $\flat$，并在 $\beta \in \flat$ 时令 moved^β 成立，也就是说，当且仅当 β 在前一个 $\forall$ 阶段或 $\exists$ 阶段中已推进。之后，我们将 $\flat$ 重置为空集。除了更新自动机状态之外，我们还会裁剪 Ξ 中的状态窗口，并相应地更新相对指针。为此，我们首先计算集合 $\text{mo} \subseteq \mathrm{V}$，其中包含所有轨迹变量 π：没有任何停顿将第一位置作为索引，也就是说，所有满足 $\text{base}(\beta) = \pi$ 所索引的位置大于 0 的停顿 β（前提 (3.1)）。随后，前提 (3.2) 将 mo 中所有轨迹的窗口向前移动一位，移除它们的第一位置。根据 mo 的构造，所有停顿都已越过第一位置，因此求值 ψ_{mod} 时将不再使用该位置上的状态。除了移动窗口之外，我们还会裁剪窗口末端，以确保窗口长度至多为 Z。对于所有属于 mo、其状态窗口已向前移动的停顿 $\beta \in$（即 $\text{base}(\beta) \in \text{mo}$），我们通过递减指针来修正 μ 中的相对位置（前提 (3.3)）。因此，对于每个停顿指针 β，$\text{base}(\beta)$ 的状态窗口中的当前状态都不会改变（我们通过递减相对位置抵消所有窗口移动）。

示例 5.4。考虑图 3 中的顶点 v_{14}，即顶点 $\langle \mathrm{U}\ \ [\pi_1 \mapsto [s_0, s_1], \pi_2 \mapsto [s_1, s_2]]$、$[\beta_1 \mapsto 0$、$\beta_2 \mapsto 1]$、$\{\beta_2\}\ _\rangle$。在前提 (3.1) 中，我们首先计算 $\text{mo} = \{\pi_2\}$。注意，π_1 不在 mo 中，因为 β_1 指向 π_1 的第一个状态，所以我们还不能丢弃 π_1 窗口中的第一个位置。然后，我们在前提 (3.2) 中更新状态窗口：我们将 π_1 重新映射到 $[s_0, s_1)[0, 1) = [s_0]$（因为 $\pi_1 \notin \text{mo}$），并将 π_2 映射到 $[s_1, s_2)[1, 2) = [s_2]$（回顾一下，我们设定了 $Z = 1$）。我们通过递减 $\mu(\beta_2)$ 来进行修正，因为基迹$(\beta_2) = \pi_2 \in \text{mo}$（前提 (3.3)）。因此，停顿 β_2 仍然映射到 π_2 窗口中的 s_2 状态

（该状态现在位于更新后的状态窗口中的位置 0）。此更新产生顶点 $\langle \forall, [\pi_1 \mapsto [s_0], \pi_2 \mapsto [s_2]], [\beta_1 \mapsto 0, \beta_2 \mapsto 0], \emptyset, _\rangle$，如图 3 中的顶点 v_{15} 所示。为了推进 DBA（我们在图 3 中省略了它），我们需要计算 $\Theta = \{a_{\beta_1} \neq a_{\beta_2}\}$ 中的一阶公式（在 φ_{fair} 中用作原子公式）以及移动的命题 β_1 和移动的命题 β_2（前提 (3.4)）。在顶点 v_{14} 中，我们通过查看状态窗口 Ξ（基迹$(\beta_1) = \Xi(\pi_1) = [s_0, s_1]$（分别为 $\Xi($基迹$(\beta_2)) = \Xi(\pi_2) = [s_1, s_2]$））中的 $\mu(\beta_1) = 0$ 个（分别为 $\mu(\beta_2) = 1$ 个）位置，获得 a_{β_1}（分别为 a_{β_2}）的当前值；这些位置对应于状态 s_0（分别为 s_2），因此 $a_{\beta_1} = \ell_{\mathcal{T}}(s_0)(a) = 0$（分别为 $a_{\beta_2} = \ell_{\mathcal{T}}(s_2)(a) = 1$）。由于 $\flat = \{\beta_2\}$，我们得到 $\{$移动$_\beta \mid \beta \in \flat\} = \{$移动$_{\beta_2}\}$，于是将 DBA 状态更新为 $q' = \delta(q, \{a_{\beta_1} \neq a_{\beta_2}$、移动$_{\beta_2}\})$。尽管我们在图 3 中省略了 DBA，但很容易看出，策略中的所有路径都满足 $\psi_{mod} = (a_{\beta_1} \neq a_{\beta_2}) \wedge$ 移动$_{\beta_1} \wedge$ 移动$_{\beta_2}$。△

5.4 初始顶点

首先，我们希望向验证者提供尽可能多的信息，因此对于全称量化轨迹，我们考虑所有长度为 Z 的可能状态窗口（参见第 6.2 节）。形式化地，我们将 V_{init} 定义为所有形如 $\mathrm{U}\ \Xi, \mu, \emptyset, q_0, \psi$ 的顶点，其中对于所有 $\beta \in \mathrm{B}$，$\mu(\beta) = 0$，并且 Ξ 中的状态窗口满足：对于所有 $\pi \in \mathrm{V}_\exists$，$\Xi(\pi) = [s_{0,\mathcal{T}}]$（即，所有存在量化轨迹都从长度为 1 的状态窗口开始）；并且对于所有 $\pi \in \mathrm{V}_\forall$，$\Xi(\pi) = [s_0, s_1, \ldots, s_{Z-1}]$，其中 $s_0 = s_0$、$\mathcal{T}$，且对于 $0 \leq i < Z - 1$，$s_{i+1} \in \kappa_{\mathcal{T}}(s_i)$（即，所有全称量化轨迹都从长度为 Z 的状态窗口开始，该窗口包含从 $s_{0,\mathcal{T}}$ 开始的 T 中的连续状态）。我们希望检查验证者是否能够从 V_{init} 中的所有顶点获胜，即 $\mathrm{wins}_{\mathcal{G}_{\mathcal{T},\varphi,Z}}(\mathfrak{V}, V_{init})$ 是否成立。

5.5 可靠性

定理 5.5（正确性）。考虑任意迁移系统 $\mathcal{T}$、一个 $\forall^*\exists^*$ *A-HLTL* 公式 φ 和一个界 $Z \in \mathbb{N}_{\geq 1}$。如果 $wins_{\mathcal{G}_{\mathcal{T},\varphi,Z}}(\mathfrak{V}, V_{init})$，那么 $\mathcal{T} \models \varphi$。

证明概要。我们利用验证者的获胜策略，为存在量化轨迹和停顿构造见证。为此，我们逐步模拟全称量化轨迹和停顿（站在反驳者的立场），并利用该策略的响应来构造见证。完整证明见第B节。□

此外，很容易看出，我们基于博弈的近似随窗口大小单调变化：

引理 *5.6*。对于任意 T φ，如果 $wins_{\mathcal{G}_{\mathcal{T},\varphi,Z}}(\mathfrak{V}, V_{init})$ 且 $Z' \geq Z$，那么 $wins_{\mathcal{G}_{\mathcal{T},\varphi,Z'}}(\mathfrak{V}, V_{init})$。对于任意$Z \in \mathbb{N}_{\geq 1}$，存在一个 T φ，使得 $\neg wins_{\mathcal{G}_{\mathcal{T},\varphi,Z}}(\mathfrak{V}, V_{init})$，但 $wins_{\mathcal{G}_{\mathcal{T},\varphi,Z+1}}(\mathfrak{V}, V_{init})$。

在第 2 节的示例中，我们针对每条轨迹处理了唯一的停顿，因此大小为 $Z = 1$ 的窗口足以容纳这一次停顿。如果我们考虑同一条轨迹上的多个 停顿，有时就需要更大的状态窗口，因此也需要更大的界 $Z > 1$。我们在第 C 节给出一个具体示例。

备注 *2*。一般而言，博弈 $G_{\mathcal{T},\varphi,Z}$ 包含无限多个状态。如果系统 T 以符号化方式表示（例如，使用一阶逻辑），我们可以推导出 $G_{\mathcal{T},\varphi,Z}$ 的符号化描述，并利用现有的无限状态博弈求解方法 *[Beyene* 等人 *2014*；*de Alfaro* 等人 *2001*；*Farzan* 和 *Kincaid 2018*；*Heim* 和 *Dimitrova 2024*；*Laveaux* 等人 *2022*；*Samuel* 等人 *2021*；*Schmuck* 等人 *2024]*，或者让用户交互式地求解该博弈 *[Correnson* 和 *Finkbeiner 2025]*（我们将在第 *8* 节进一步讨论）。如果 T 是有限状态迁移系统（即状态集合 $S_{\mathcal{T}}$ 有限），那么 $G_{\mathcal{T},\varphi,Z}$ 是有限状态 *Büchi* 博弈，我们可以直接计算。在这种情况下，$G_{\mathcal{T},\varphi,Z}$的顶点数量关于界 Z和其中的轨迹数量 φ 呈指数增长（自组合通常也是如此 *[Barthe* 等人 *2004]*），但关于 $|S_{\mathcal{T}}|$仅呈线性增长。由于 *Büchi* 博弈可以在多项式时间内求解 *[McNaughton1993]*，因此，判断 $wins_{\mathcal{G}_{\mathcal{T},\varphi,Z}}(\mathfrak{V}, V_{init})$ 是否成立，可在关于系统规模的多项式时间内完成。

6 完备性

一般而言，我们基于博弈的验证方法并不完备，即验证者可能会在 $\mathcal{G}_{\mathcal{T}、\varphi、Z}$中落败，即使 $\mathcal{T} \models \varphi$ 成立。因此，一般而言，我们无法利用该博弈来断定某个属性被违反；它只能证明属性成立。这种不完备性不可避免：对于有限状态系统 $\mathcal{T}$，判断 $wins_{\mathcal{G}_{\mathcal{T}、\varphi、Z}}(\mathfrak{V}、V_{init})$ 是否成立是可判定的（参见备注 2），但 $\mathcal{T} \models \varphi$ 一般而言是不可判定的[Baumeister 等人 2021]。然而，我们的方法对于许多片段是完备的，因此构成一种（有限状态）判定过程。特别是，我们的方法所适用的完备片段涵盖了许多先前已知的可判定类别 Baumeister 等人 2021；Hsu 等人 2023]。

6.1 交替自由公式

我们首先考虑这样的情况：每个轨迹变量至多有一次停顿，并且所有轨迹和停顿都采用存在量化或全称量化。在这种情况下，窗口大小为 $Z = 1$就足以保证完备性：

定理 6.1。 设 $\mathcal{T}$ 是任意迁移系统，并设 φ 是一个 $\exists^*$ 或 $\forall^*$ *A-HLTL* 公式，且每条轨迹至多有一次停顿。那么，$wins_{\mathcal{G}_{\mathcal{T},\varphi,1}}(\mathfrak{V}, V_{init})$ 当且仅当$\mathcal{T} \models \varphi$。

6.2 终止系统与前瞻

在我们的博弈构造中，我们确保全称量化轨迹的状态窗口——从初始状态开始——始终长度为 Z。回想一下，我们的博弈近似于 A-HLTL 语义：在 a $\forall^*\exists^*$ 公式的语义中，所有存在量化轨迹和停顿都是在已知全体全称量化轨迹和停顿的情况下固定的。在我们的博弈中，我们让验证者通过逐步响应反驳者的行动来构造存在量化轨迹和停顿。通过始终将全称量化轨迹的状态窗口保持为长度 Z（可能包括末尾未被任何停顿指向的状态），我们因此赋予验证者有限的预知能力，即验证者可以窥视全称量化轨迹上的下一个 Z 个状态，从而做出更充分知情的决策 [Abadi 和 Lamport 1991]。这在只需固定前瞻即可了解整个系统执行的系统中尤为有趣。若存在一个界 $D \in \mathbb{N}$（称为深度），使得所有长度至少为 D 的路径都会到达某个汇点状态，也就是说，存在一个状态 $s \in S_{\mathcal{T}}$，满足 $\kappa_{\mathcal{T}}(s) = \{s\}$，则称迁移系统 $\mathrm{T} = (S_{\mathcal{T}}, s_0, \tau, \kappa_{\mathcal{T}}, \ell_{\mathcal{T}})$ 是终止的（也称为树状 [Hsu 等人 2023]）。Hsu 等人 [2023] 提出了一种针对终止系统中 A-HLTL 的基于 QBF 的模型检测方法。我们可以证明，在选择 $Z = D$ 时，我们的方法对终止系统是完备的：

定理 6.2。 设 $\mathcal{T}$ 是一个深度为 D 的终止转移系统，且 φ 是一个 $\forall^*\exists^*$ *A-HLTL* 公式。则 $wins_{\mathcal{G}_{\mathcal{T}、\varphi、D}}(\mathfrak{V}、V_{init})$ 当且仅当 $\mathcal{T} \models \varphi$。

6.3 可容许公式

接下来，我们研究由 *Baumeister* 等人 提出的所谓 可容许 [2021]公式。在下文中，我们假设，在理论 $\mathfrak{T}$中，“=”被解释为 $\mathbb{V}$上的相等。

定义 *6.3*。 可容许公式的形式为 $\forall \pi_1 \ldots \forall \pi_n . \exists \beta_1 \blacktriangleright \pi_1 \ldots \exists \beta_n \blacktriangleright \pi_n . \psi$，其中 ψ 是以下内容的布尔组合：（**1**）任意数量的状态公式，即不使用时序运算符的公式；以及（**2**）一个单独的（正向出现的）阶段公式，其形式为 $\bigwedge_{i<j} \square\big(\bigwedge_{a\in P_{i,\ j}} (a_{\beta_i} = a_{\beta_j})\big)$，其中 $P_{i,\ j} \subseteq \mathrm{X}$。△

也就是说，我们对每条轨迹分别存在量化一个唯一的停顿，并限制 ψ 中允许的关系公式。我们可以将集合 $P_{i,j}$ 视为表示一组颜色，这些颜色由 $P_{i,j}$ 中变量的求值定义（因此有 $|\mathbb{V}|^{|P_{i,j}|}$ 种颜色）。于是，阶段公式断言

□ ($\bigwedge_{a\in P_{i,j}} a_{\beta_i} = a_{\beta_j}$) 对于每一对 $i < j$ 轨迹，因此要求 π_i 和 π_j 以相同的$P_{i,j}$-颜色序列前进，但速度可以不同（分别通过停顿 β_i和 β_j, 实现）。例如， φ_{OD} 是可容许的（更多示例见 [Baumeister 等人 2021] ）。我们可以证明，我们的方法对于可容许公式是完备的（甚至对 $Z = 1$）：

定理 6.4。 设 $\mathcal{T}$ 是任意迁移系统，且 φ 是一个可容许的 *A-HLTL* 公式。那么当且仅当 $\mathcal{T} \models \varphi$ 时， $\mathcal{G}_{\mathcal{T},\varphi,1}(\mathfrak{V}, V_{init})$ 获胜。

下面我们给出定理 6.4 的证明概要。第一个方向由定理 5.5 得出。对于另一个方向，假设 φ是可容许的，且 T $\models \varphi$。令$\psi_{phase} = \bigwedge_{i<j} \square\big(\bigwedge_{a\in P_{i,j}}(a_{\beta_i} = a_{\beta_j})\big)$为 ψ 中唯一的阶段公式。因此，验证者的目标是找到满足 ψ_{phase} 的停顿序列（只要存在这样的停顿序列）。

安全进展集。 由验证者 $\mathfrak{V}$ 控制的 $G_{\mathcal{T},\varphi,1}$ 中的每个顶点 v 都具有如下形式

$$v = \big\langle ⓔ, \big[\pi_i \mapsto [s_i, s_i']\big]_{i=1}^{n}, \big[\beta_i \mapsto 0\big]_{i=1}^{n}, \emptyset, q\big\rangle,$$

也就是说，轨迹 $\pi_1, \ldots, \pi_n$ 分别映射到长度为 2 的状态窗口 $[s_1, s_1'], \ldots, [s_n, s_n']$，且所有停顿序列 $\beta_1, \ldots, \beta_n$ 都指向窗口中的第 0 个位置。我们假设 v 在局部满足 ψ_{phase} 中的所有着色约束，即对于所有 $i < j$和 $a \in P_{i,j}$, $\ell_{\mathcal{T}}(s_i)(a) = \ell_{\mathcal{T}}(s_j)(a)$；否则， ψ_{phase} 已被违反。将此假设称为 (A)。在顶点 v处，验证者可以决定推进哪些停顿序列 $\beta_1, \ldots, \beta_n$。因此，我们可以用所谓的推进集 $M \subseteq \{1, \ldots, n\}$ 标识 v 的每个可能后继顶点。直观地说， M恰好包含这样的索引 i： β_i通过从状态 s_i 推进到 s_i' 执行了非停顿步。对于 $M \subseteq \{1, \ldots, n\}$，我们通过定义 $\text{next}_v^M(i) := s_i' \text{if } i \in M$，否则定义为 $\text{next}_v^M(i) := s_i$，从而定义状态 $\text{next}_v^M(1) \ldots, \text{next}_v^M(n) \in S_{\mathcal{T}}$。也就是说，在 M 中得到推进的每个停顿序列 β_i 都映射到 v 状态窗口中的第二个位置（即 s_i'），而所有未推进的停顿序列都停留在第一个状态（即 s_i）。如果对于每个 $i < j$ 和每个 $a \in P_{i,j}$, $\ell_{\mathcal{T}}(\text{next}_v^M(i))(a) = \ell_{\mathcal{T}}(\text{next}_v^M(j))(a)$，则称推进集 M 是安全的；也就是说， M 确保下一步在局部满足所有着色约束。

最大安全推进集。 关键在于， ψ_{phase} 中的着色约束是等式，因此具有对称性：例如，假设 **(1)**推进集{i}是安全的，并且**(2)**推进集{j}也安全。那么，根据**(1)**， s_i'与s_j具有相同的$P_{i,j}$-颜色；根据**(2)**， s_i与 s_j'具有相同的$P_{i,j}$-颜色；而根据**(A)**， s_i与s_j具有相同的$P_{i,j}$-颜色。这已经意味着 s_i'与 s_j'具有相同的$P_{i,j}$-颜色；推进集{i, j}也安全。更一般地，安全进展集构成一个并完备半格：

引理 6.5。 如果 M_1 和 M_2 是安全进展集，那么 $M_1 \cup M_2$ 也是安全进展集。

因此，我们可以将 M_v^{max}定义为顶点 v 中的唯一最大安全推进集（定义为所有安全进展集的并集）。现在，我们可以非正式地定义 $\mathfrak{V}$的获胜策略 σ^{max}： 对于每个满足**(A)** 且仍需满足 ψ_{phase}的顶点 v，我们定义 $\sigma^{max}(v) := v'$，其中

$$v' := \big\langle ⓤ, \big[\pi_i \mapsto [s_i, s_i']\big]_{i=1}^{n}, \big[\beta_i \mapsto ite(i \in M_v^{max}, 1, 0)\big]_{i=1}^{n}, \{\beta_i \mid i \in M_v^{max}\}, q\big\rangle.$$

这里，我们将 $ite(b, x, y)$ 记作 x若 b成立则取其值，否则取 y（if-then-else 的简称）。在 v'中，当且仅当 $i \in M^{max}$时，验证者才会推进 β_i 上的停顿（通过将 μ-指针从位置 0 增至 1）。如果 v 无需满足 ψ_{phase}（例如，状态公式已足以满足 ψ），验证者就会推进所有停顿 $\beta_1, \ldots, \beta_n$。

直观地说， σ^{max} 因此总是（局部地）尽可能推进更多停顿，同时确保 ψ_{phase}中的着色约束在下一步仍然成立。对于任意满足 π_1、……、 π_n的轨迹组合 $\exists \beta_1 \blacktriangleright \pi_1 \ldots \exists \beta_n \blacktriangleright \pi_n . \psi_{phase}$，策略 σ^{max} 会自动找到一个使ψ_{phase} 成立的停顿。因此很容易看出，验证者遵循策略$\mathcal{G}_{\mathcal{T},\varphi,1}$ 即可获胜，从而证明定理6.4。

6.4 一个新的可判定片段

借鉴第6.3节中的类似思路，我们可以证明，基于博弈的方法对于范围更广的一类性质也是完备的。核心思想是保留找到最大安全推进集的能力；也就是说，只要我们能够安全地推进某个停顿 β_i，并且能够安全地推进某个停顿 β_j，就也能够同时安全地推进 β_i 和 β_j。

给定赋值 $A : \mathrm{X} \to \mathbb{V}$ 和一个停顿变量 $\beta \in \mathrm{B}$，我们将 $A^\beta : \mathrm{X}_{\{\beta\}} \to \mathbb{V}$ 记作由 $A^\beta(x_\beta) := A(x)$ 定义的索引赋值。对于两个停顿变量 $\beta_i,\ \beta_j \in \mathrm{B}$，如果满足以下条件，我们称公式 $\theta \in \mathrm{F}^{X_{\{\beta_i,\beta_j\}}}$ 为矩形闭合：对于任意赋值$A,\ B,\ C,\ D : \mathrm{X} \to \mathbb{V}$，均有：

$$\left(A^{\beta_i} \cup B^{\beta_j} \models^{\mathfrak{T}} \theta\right) \wedge \left(C^{\beta_i} \cup B^{\beta_j} \models^{\mathfrak{T}} \theta\right) \wedge \left(A^{\beta_i} \cup D^{\beta_j} \models^{\mathfrak{T}} \theta\right) \Rightarrow \left(C^{\beta_i} \cup D^{\beta_j} \models^{\mathfrak{T}} \theta\right),$$

也就是说，如果 $(A,\ B)$、$(C,\ B)$ 和 $(A,\ D)$ 都满足 θ，那么 $(C,\ D)$ 也满足 θ。

在可容许公式中，我们处理的总是形如 $a_{\beta_i} = a_{\beta_j}$ $(\text{for } a \in P_{i,\ j})$ 的等式，它们显然是矩形闭合的。更一般地，所有形如 $\theta = (e_1 = e_2)$——其中 e_1 和 e_2是关于某个唯一（不一定相同的）停顿变量的一阶项（参见 [Barwise 1977]）——都是矩形闭合的。例如，$a_{\beta_1} = (b_{\beta_2} + c_{\beta_2})$ 是矩形闭合的，但 $a_{\beta_1} = (b_{\beta_2} + c_{\beta_3})$ 则不是。

定义 *6.6.* 一个矩形闭合不变式是一个公式 $\varphi = \forall \pi_1 \ldots \pi_n . \exists \beta_1 \blacktriangleright \pi_1 \ldots \exists \beta_n \blacktriangleright \pi_n . \psi$，其中 ψ 是以下内容的布尔组合：**(1)** 任意数量的状态公式；以及**(2)** 一个单个（正向出现的）公式 $\bigwedge_{\theta \in \Theta} \theta$，其中每个 $\theta \in \Theta$ 都是矩形闭合的。△

容易看出，任何可容许公式都是矩形闭合不变式（任何相等$a_{\beta_i} = a_{\beta_j}$都是矩形闭合的）。注意，可容许公式只允许形如 $a_{\beta_i} = a_{\beta_j}$ 的约束，即断言两条轨迹上同一个变量之间相等。矩形闭合不变式允许任意表达式，因此可以关联两条轨迹上的任意变量。采用与定理 6.4 相同的证明思路，我们可以证明更一般的结果：

定理 6.7. 设$\mathcal{T}$ 是任意迁移系统，且令 φ 为一个矩形闭合不变式。那么 $wins_{\mathcal{G}_{\mathcal{T},\ \varphi,\ 1}}(\mathfrak{V},\ V_{init})$ 当且仅当$\mathcal{T} \models \varphi$。

示例 *6.8.* 例如，假设一个系统由以下位（布尔变量）组成：$w1$、……、wn，但攻击者无法单独观察这些位。相反，攻击者可以通过探测攻击 [Wang 等人2017], 检测 某个 位当前是否已启用。我们可以将 Zdancewic 和 Myers [2003]的观察确定性推广到这种限制更严格的攻击者设置，定义如下：

$$\forall \pi_1 . \forall \pi_2 . \exists \beta_1 \blacktriangleright \pi_1 . \exists \beta_2 \blacktriangleright \pi_2 . \ (l_{\beta_1} = l_{\beta_2}) \to \square\left((w1_{\beta_1} \vee \cdots \vee wn_{\beta_1}) = (w1_{\beta_2} \vee \cdots \vee wn_{\beta_2})\right),$$

要求 $w1 \vee \cdots \vee wn$ 的值在两条轨迹上相同。 △

7 实现与求值

除了理论贡献之外，我们的博弈还可直接用于 A-HLTL 的（半）自动验证。值得注意的是，我们的整个框架适用于有限状态与无限状态系统。对于无限状态系统，我们设想采用交互式验证的方法（类似于 HyperLTL [Correnson 和 Finkbeiner 2025]中成功采用的方法）；我们将在第 8 节进一步讨论这一点。对于有限状态系统，我们可以直接构造并求解该博弈，从而实现全自动验证方法。需要注意的是，这类有限状态方法在无限状态环境中仍然具有意义。抽象（例如由一组谓词 [Graf 和 Saïdi 1997] 或一个抽象域 [Cousot 和 Cousot 1977]生成的抽象）通常会对无限变量域进行抽象，同时保持系统的时序行为；也就是说，无限状态系统的抽象会得到一个有限状态系统，其中无限状态系统的每一步都对应于有限状态抽象中的一个

表1。我们比较HyMCA 与HyperQB [Hsu 等人 2023] 在终止系统上的表现。我们展示系统的规模（$|S|$）、验证结果，以及各工具所用的总时间（t）。对于 HyMCA，我们还给出构造（t_{const}）和求解 $G_{\mathcal{T}}$ 所需的时间$t_{solve,\ \varphi,\ 1}$，以及 $|G|$ 的顶点数（|）。执行时间以秒为单位，超时（以“-”表示）设为 5 分钟。请注意，对于 HyMCA，总时间 t 可能略大于 t_{const} 与 t_{solve}之和，因为其中还包括解析和预处理等步骤。

				[Hsu 等人 2023]	HyMCA			
实例	**属性**	$\|S\|$	**结果**	t	t_{const}	$\|G\|$	t_{solve}	t
ACDB	φ_{OD}	110	✗	2.96	1.00	17,119	0.54	**1.86**
ACDB_{ndet}	φ_{OD}	679	✗	12.06	3.58	71,269	2.29	**7.18**
ConcLeak	φ_{OD}	577	✗	27.16	3.44	59,495	2.29	**6.51**
ConcLeak_{ndet}	φ_{OD}	2821	✗	-	62.1	899,014	45.9	**114.4**
SpecExec_{V1}	φ_{SNI}	29/57	✗	3.71	0.199	1,219	0.09	**0.69**
SpecExec_{V2}	φ_{SNI}	109/169	✓	8.15	0.15	355	0.05	**0.82**
SpecExec_{V3}	φ_{SNI}	88/169	✗	11.65	0.38	4,041	0.19	**1.69**
SpecExec_{V4}	φ_{SNI}	94/169	✗	10.45	0.71	11,797	0.34	**2.01**
SpecExec_{V5}	φ_{SNI}	94/169	✗	7.91	0.15	925	0.05	**0.96**
SpecExec_{V6}	φ_{SNI}	85/169	✗	10.80	0.62	9,573	0.29	**1.93**
SpecExec_{V7}	φ_{SNI}	109/169	✓	7.65	0.13	355	0.04	**0.82**
DBE	φ_{SC}	9/7	✓	1.22	0.13	469	0.06	**0.43**
LP	φ_{SC}	81/77	✓	3.22	0.78	18,705	0.63	**1.63**
EFLP	φ_{SC}	81/249	✓	15.34	2.6	68,015	2.4	**5.82**
CacheTA	φ_{OD}	49	✗	2.51	0.44	7,683	0.34	**1.03**
$\text{CacheTA}^{ndet+loops}$	φ_{OD}	59	✗	**3.79**	2.26	72,812	2.54	5.07
$\text{CacheTA}^{ndet+loops}$	φ_{OD}	87	✗	23.31	6.26	170,509	6.9	**13.57**

计算步骤。因此，在该抽象上验证类似于 Zdancewic 和 Myers[2003] 的观察确定性（OD）等性质，仍需要找到适当的停顿。A-HLTL 验证的核心计算难题在有限状态抽象中依然存在，因此有限状态验证的结果和工具仍然有用。

作为概念验证，我们在一个名为 HyMCA 的工具中实现了用于有限状态系统的基于博弈的方法。HyMCA 读取一个系统 T（以符号 NuSMV 系统的形式 [Cimatti 等人 2002]）、一个 A-HLTL 公式 φ 和一个界 $Z \in \mathbb{N}$，并自动构造和求解 $G_{\mathcal{T},\ \varphi,\ Z}$（编码为显式状态博弈）。如有需要，HyMCA 还可以解析不同系统上的轨迹 [Goudsmid 等人 2021]。在内部，我们使用 Spot [Duret-Lutz 等人 2022] 将线性时序逻辑公式转换为确定性自动机，并使用 Oink [van Dijk 2018] 求解 $G_{\mathcal{T},\ \varphi,\ Z}$。

终止系统。 首先，我们使用 HyMCA 在 Hsu 等人提供的基准测试上评估终止系统 [2023]。尽管理论上，Hsu 等人 [2023]提出的基于QBF 的有界模型检测方法适用于任意公式，但他们的实现（HyperQB）仅实现了少数固定的公式模板。我们采用以下可容许模板子集：观察确定性（φ_{OD}）、推测性非干扰 [Guarnieri 等人 2020]（φ_{SNI}）以及正确编译（φ_{SC}），并在多个 NuSMV 模型上对其进行检查。详情请参见 [Hsu 等人 2023] 。这三种属性均为可容许属性，因此属于 HyMCA 具有完备性的片段（参见定理 6.4）。

表2. 我们在非终止（反应式）系统上验证 A-HLTL 属性。我们给出了 HyMCA 的验证结果和执行时间。

系统	属性	结果	t_{HyMCA}
图 1a^{syn} （2 位）	φ_{OD}	✓	0.31
图 1a^{syn} （4 位）	φ_{OD}	✓	0.45
图 1a^{syn} （8 位）	φ_{OD}	✓	1.21
图 1a （2 位）	φ_{OD}	✓	0.33
图 1a （4 位）	φ_{OD}	✓	1.15
图 1a （8 位）	φ_{OD}	✓	109.43

系统	属性	结果	t_{HyMCA}
图 2	φ_{OD}	✓	0.25
缓冲区	φ_{OD}	✗	0.46
缓冲区	φ_{NI}	✓	0.54
缓冲区$_{delay,\ Z=3}$	φ_{NI}	✓	31.63
缓冲区$_{flipped}$	φ_{OD}	✗	0.76
缓冲区	φ_{NI}	✓	1.32

我们在表 1 中展示验证结果和耗时。我们采用固定的窗口大小 $Z = 1$，该大小足以保证完备性。若属性成立，则记为 ✓（若✗，则表示属性被违反）；根据完备性，我们可以通过求解博弈直接推断这一点。总体而言，我们观察到，当系统包含许多状态但深度很浅、因而 QBF 编码较小时，HyperQB 表现良好。不过，在大多数实例中，HyMCA 的速度仍快于HyperQB 。

反应式系统。我们的方法有一项特别优势：首次实现了对反应式（即非终止）系统上的 A-HLTL 进行自动验证。我们在表2中展示了若干测试案例。首先，我们检查 φ_{OD} 在图 1a 中的程序上（以及其同步版本）。为扩大系统规模，我们增加每个变量所存储的位数。我们还检查更复杂（不可容许）的 A-HLTL 属性，例如 φ_{NI} （参见示例4.1），并应用于非终止系统。例如，缓冲区实例描述了一个将低安全级输入（每一步都可能变化）传递到输出（可能存在延迟）的系统。因此，该系统不满足 φ_{OD}，但满足 φ_{NI}。需要强调的是，这些示例并非现实世界中的案例，而是现实世界组件的简单抽象。A-HLTL 属性针对这些程序的无限执行进行推理，因此验证极具挑战性（此前的方法均无法自动验证这些示例）。

超越有限状态系统。 我们强调，HyMCA 通过计算一个显式状态奇偶博弈，以尽可能简单的形式实现了我们新颖的基于博弈的验证方法。初步实验表明，与以往基于 QBF 的方法相比，即使是这种直接实现也表现良好。我们强调，A-HLTL 验证的主要复杂度源于逻辑本身的表达能力；系统的复杂度（规模）则处于次要地位。表1 和2 中检查的所有属性都运用了 A-HLTL 的高级异步推理，并将寻找适当异步停顿的任务交由验证工具处理（参见第8节）。我们的原型并非旨在成为功能完备的验证工具，而是用于展示基于博弈的方法在有限状态环境中运行良好。我们博弈的一项关键优势是适用于无限状态系统。例如，在同步 HyperLTL 领域，Co-enen 等人 [2019b] 表明，基于同步博弈的方法在有限状态环境中效果良好，因此得以成功应用于无限状态系统；具体而言，可以自动求解无限状态博弈 [Beutner 和 Finkbeiner 2022b；Itzhaky 等人 2024]，也可以通过交互方式求解 [Correnson 和 Finkbeiner 2025]。我们的方法能够处理更一般的异步性质（包括观察确定性等知名示例），并提供了有价值的抽象：我们无需对无限轨迹和停顿进行推理，只需在有限窗口内，对状态及该窗口中的指针进行局部推理。正如第8节所述，我们的博弈由此为将验证扩展到无限状态系统奠定了基础。

8 相关工作

关系程序逻辑。关系霍尔逻辑 [Benton 2004] ——以及相关的霍尔风格超属性规约 [Assaf 等人 2017；Beutner 2024；Dardinier 和 Müller 2024；Dickerson 等人 2022；D'Osualdo 等人 2022；Gladshtein 等人 2024；Maillard 等人 2020；Sousa 和 Dillig 2016]——刻画了多个程序运行的初始状态和最终状态，因此本质上是"异步"的（程序执行可能经过不同步数才终止）。然而，关系霍尔逻辑难以表达时序超属性，即针对可能无限的执行过程（例如无限协议交互或反应式系统中的执行）之时序行为进行推理的属性；而这类属性是描述诸如Zdancewic 和 Myers [2003]的观察确定性等性质所必需的。

超安全性与可交换性。许多工作通过利用可交换性研究了超安全属性的验证[Antonopoulos 等人 2023；Eilers 等人 2023、2020；Farzan 等人 2022；Farzan 和 Vandikas 2020；Shemer 等人 2019]。这些方法试图通过利用我们可以交错执行不同程序执行这一事实，寻找不同执行的对齐方式，以辅助验证。我们的工作与这些方法的不同之处在于，对齐（又称停顿）的量化是规约的一部分，而不仅仅是有助于验证的技术。让我们以Zdancewic 和 Myers [2003]的观察确定性（OD）为例。该属性指出，所有具有（初始）相同低安全级输入的轨迹对，就输出而言都是停顿等价的。为了验证这一属性，可以任意交错程序的中间步骤，即输出不发生变化的步骤。问题在于，验证挑战之一在于识别输出发生变化的点。

```
1 o = 0   2 c = 0
3 重复    4   o = ⌊ c /
  h ⌋     5   c = c + 1
```

图 7. 示例程序

示例 8.1。考虑图 7 中简单的基于整数的程序。这里，$\lfloor\cdot\rfloor$ 向下取整到最接近的整数。容易看出，该程序的输出序列（每次循环迭代对应一个输出）为$0^h1^h2^h\cdots$，也就是说，在最初的 h 次循环迭代中，o的值为 0，依此类推。无论 h取何值，该程序的输出都是停顿等价的；因此，该程序满足 OD。然而，使用基于可交换性的框架验证这一点却很困难。大多数现有框架 [Beutner 和 Finkbeiner 2022b；Eilers 等人 2020；Farzan 和 Vandikas 2020；Itzhaky 等人 2024；Shemer 等人 2019；Unno 等人 2021] 会寻找某种对齐，以便更容易地证明该属性。然而，为了表达 OD，我们确实希望在输出发生变化时比较两次执行的输出，也就是说，识别两次执行应当同步的点，是 OD 规约（隐式）包含的部分。在上述框架中，我们需要手动强制对齐（例如，定义显式观测点或添加同步断言）。只有这样，k-安全验证者才能找到某种对齐（位于两个输出发生变化的点之间），以帮助完成验证。在我们的示例中，我们无法通过语法推断输出发生变化的点。相比之下，A-HLTL 提供了一种更丰富、更高层次的规约语言，使我们能够通过对停顿进行一等量化来轻松表达 Zdancewic 和 Myers [2003]的 OD（见 φ_{OD}）；也就是说，需要对齐的点的识别本身就是规约的一部分。△

因此，我们的方法与现有验证方法的不同之处在于规约语言的表达能力：大多数超安全框架只关注由无限状态空间带来的验证挑战，而我们的工作则关注由规范逻辑本身带来的复杂度。

异步超属性的时序逻辑。 近年来，人们提出了许多用于规约异步超属性的逻辑。示例包括 A-HLTL（本文的主要研究对象） [Baumeister 等人 2021]， H_μ 以及 mumbling H_μ （对

多元 μ演算的扩展），[Gutsfeld 等人 2021, 2024],OHyperLTL [Beutner 和 Finkbeiner 2022b]（带有显式观测点的 HyperLTL 扩展）、HyperLTL$_S$ [Bozzelli 等人 2021]（允许算子跳过求值步骤的 HyperLTL 扩展）、HyperLTL$_C$ [Bozzelli 等人 2021]（仅部分轨迹推进的 HyperLTL 扩展）、团队语义的变体 [Gutsfeld 等人 2022；Krebs 等人 2018], 以及带有显式时间量化的一阶逻辑扩展 [Bartocci 等人 2022]。Bozzelli 等人 [2022] 研究了其中许多逻辑的表达能力，并发现它们大多互不可比。

A-HLTL 验证。Baumeister 等人 [2021] 研究了一类受限的 A-HLTL 属性，称为 可容许 (参见第6.3节）。他们证明，可容许公式的有限状态模型检测是可判定的，并给出了向（同步）HyperLTL 的手动归约。Hsu 等人 [2023] 研究了终止 系统（参见第 6.2节），并提出了一种基于 QBF 的有界模型检测方法。与以往针对 A-HLTL 的技术不同，我们的方法可可靠地应用于任意 $\forall^*\exists^*$ A-HLTL 公式，即它可能成功验证不属于任何已知可判定片段的属性。

$\forall^*\exists^*$ 与基于博弈的验证。近年来，已有许多技术被提出，用于验证超越 k-安全的超属性 [Antonopoulos 等人 2023；Beutner 和 Finkbeiner 2022b；Itzhaky 等人 2024；Unno 等人 2021]。这些技术将寻找合适对齐的过程与寻找存在量词见证轨迹的过程结合起来。例如，Unno 等人 [2021] 将 $\forall^*\exists^*$ 属性的验证编码为一种专门形式的 Horn 子句，并让求解器找出合适的对齐。目前，他们的编码只会寻找有助于验证的某种对齐。一个值得探讨的未来方向是，能否扩展该编码，以便显式地对对齐进行量化（如 A-HLTL 中那样）。Beutner 等人 [2024]；Correnson 等人 [2024]；Hsu 等人 [2023] 采用有界展开来验证或证伪 $\forall\exists$ 属性。我们的验证方法植根于对轨迹和停顿的基于博弈的解释。对于同步 HyperLTL 属性，此类基于博弈的解释已被成功用于近似处理代价高昂的量词交替 [Beutner 和 Finkbeiner 2022a；Coenen 等人 2019b]。我们的异步博弈在动态上与同步博弈有很大差异。我们的核心思想是使用窗口和相对指针，使玩家能够直接控制轨迹的停顿，这对于近似 A-HLTL 的语义至关重要。尤其是，我们的异步博弈允许玩家分别控制各条轨迹的速度（即停顿）。Beutner 和 Finkbeiner[2022b]；Itzhaky 等人 [2024] 针对以 OHyperLTL 表达的异步 $\forall^*\exists^*$ 属性采用了基于博弈的语义。在这种逻辑中，用户指定执行应在哪些点同步，而验证者则尝试在这些所谓的观测点之间找到对齐。如前所述，A-HLTL 允许显式地对停顿进行量化，这对于简洁地表达 OD 等属性是必要的；若要在 OHyperLTL 中表达 OD，我们需要在输出发生变化时手动放置观测点，但这在语法上往往无法实现（参见例 8.1）。一个值得探讨的问题是，Itzhaky 等人 [2024] 提出的 CHC 编码能否扩展为允许对停顿进行显式推理，或许可以采用我们提出的使用状态窗口和相对指针的新思路。

细化、停顿与模拟。超属性中一个特别有趣的类别是细化属性。给定两个系统 T_1 和 T_2, 以及一组可观测变量 Obs，我们可以使用（同步）HyperLTL 轻松表达锁步细化属性：φ^{syn} refine $:= \forall \pi_1.\, \exists \pi_2.\, \square\big(\bigwedge_{a\in \text{Obs}} a_{\pi_1} = a_{\pi_2}\big)$。证明细化属性（例如$\varphi^{syn}$ refine）的常用方法，是在 T_1 和 T_2 [之间建立模拟关系（Milner 1971]）。自动检查是否存在（同步）模拟关系的一种常用技术，是利用模拟关系与模拟博弈中的获胜策略之间的联系（Stirling 1995]）。

虽然 φ^{syn}_{refine} 要求锁步细化，但通常只需建立停顿细化属性（例如参见 [Leroy 2006]），我们可以用 A-HLTL 轻松地将其表达为

$$\forall \pi_1.\, \exists \pi_2.\, \exists \beta_1 \blacktriangleright \pi_1. \exists \beta_2 \blacktriangleright \pi_2.\, \square \Big(\bigwedge_{a \in Obs} a_{\beta_1} = a_{\beta_2} \Big). \qquad (\varphi_{refine})$$

也就是说，对于任意执行 π_1 在 $\mathcal{T}_1$ 中，都存在某个执行 π_2 在 $\mathcal{T}_2$ 中，使得除停顿之外，所有可观测变量的取值都一致。我们可以通过找到一个 φ_{refine} 停顿模拟[Namjoshi 1997；Namjoshi 和 Tabajara 2020]：只要任一系统执行“τ步”，即不改变 Obs 中任何变量的步，因此也不会发出“可见事件”，另一个系统就可以停顿。因此，我们基于博弈的技术可用作证明停顿精化的一种有原则的方法。从这个角度看，我们可以将基于博弈的验证方法视为同步模拟博弈的一种异步扩展；至少在 φ_{refine} 这一极其特殊的情况下是如此。值得注意的是，将我们的基于博弈的方法应用于 φ_{refine} 时，自然允许两个副本独立停顿（只要两个轨迹上的停顿都无限次地推进），也就是说，我们的技术不要求同步推进，即两个系统最终一同推进（参见 [Cho 等人 2023]）。

无限状态系统的交互式验证。 在我们的实现中，为了将我们的方法与现有方法进行评估，我们专注于有限状态系统。我们的所有结果（包括可靠性和完备性）也适用于无限状态系统。虽然我们可以尝试自动构造并求解由此产生的无限状态博弈（例如，参见 [Beyene 等人 2014；de Alfaro 等人 2001；Farzan 和 Kincaid 2018；Heim 和 Dimitrova 2024；Laveaux 等人 2022；Samuel 等人 2021；Schmuck 等人 2024]），更有前景的方向是采用交互式证明方法。Correnson 和 Finkbeiner [2025] 最近将基于博弈的方法用于 同步 HyperLTL [Beutner 和 Finkbeiner 2022a；Coenen 等人 2019b]，开发了一种 交互式证明系统，用于验证 ∀∃ 无限状态软件中的属性。直观来说，他们的方法通过余归纳逐步展开底层博弈，并让用户隐式地定义策略。我们的基于博弈的方法能够处理量化停顿的异步超属性，但仍可归结为一个普通的双人博弈。这个博弈为用户提供了明确的验证目标，并抽象掉了无限轨迹和停顿这一亟需解决的问题；用户只需推理当前状态中的停顿。我们的博弈还使我们能够采用与 Correnson 和 Finkbeiner [2025] 所用方法类似的框架，让用户通过逐步展开博弈场来交互式地证明 A-HLTL 属性。因此，我们的工作为开发统一的证明系统奠定了基础，该系统能够处理反应式系统中的许多重要信息流策略。

9 结论与未来工作

本文提出了一种严谨的方法，用于（自动或交互式）验证以 A-HLTL 表达的异步超属性。我们的方法将量词交替近似为博弈，并无缝支持对停顿的量化。这使得我们能够可靠地验证任意 $\forall^*\exists^*$ A-HLTL 属性，远超以往方法所支持的片段。此外，我们证明了该方法对许多片段是完备的，因此构成了一种有限状态判定过程。

我们的工作为未来的扩展留下了许多令人振奋的方向。在理论方面，例如，可以通过纳入不完全信息 $\forall^*\exists^*$，将我们基于博弈的视角扩展到支持超出 [Reif 1984]片段的 A-HLTL 公式；直观地说，验证者不应根据在其所控制的（存在量化）对象之后量化的全称轨迹和停顿来做出判断，而应依据这些对象 [Beutner 和 Finkbeiner 2025b，c]。在实践方面，我们认为

验证无限状态（软件）系统尤其值得关注。正如我们上文所述，我们基于博弈的方法（及证明）同样适用于（以符号化方式表示的）无限状态系统，从而得到无限状态博弈。探索求解由此产生的（此时为无限状态）博弈的有效技术，并为（可能无界的）状态窗口开发抽象技术，都是具有挑战性的开放问题。更具体地说，将Correnson 和 Finkbeiner [2025] 提出的交互式证明系统应用于我们的博弈，将很有意义；这将使用户能够以交互方式验证无限状态反应式系统中的异步超属性。

致谢
本研究得到欧洲研究委员会（ERC）HYPER 项目授权（101055412）以及德国研究基金会（DFG）TRR 248 项目（389792660）的支持。

参考文献
Martín Abadi 和 Leslie Lamport。1991。《细化映射的存在性》。《理论计算机科学》（1991）。doi:10.1016/0304- 3975(91)90224-P Erika Ábrahám、Ezio Bartocci、Borzoo Bonakdarpour 和 Oyendrila Dobe。2020。《带非确定性的概率超属性》。载于自动化验证与分析技术国际研讨会，ATVA 2020。doi:10.1007/978-3-030-59152-6_29Erika Ábrahám 和 Borzoo Bonakdarpour。2018。《HyperPCTL：一种用于概率超属性的时序逻辑》。载于定量系统评估国际会议，QEST 2018。doi:10.1007/978-3-319-99154-2_2 Timos Antonopoulos、Eric Koskinen、Ton Chanh Le、Ramana Nagasamudram、David A. Naumann 和 Minh Ngo。2023。《关系验证的对齐代数》。《ACM 程序设计语言会议录》POPL（2023）。doi:10.1145/3571213 Mounir Assaf、David A. Naumann、Julien Signoles、Eric Totel 和 Frédéric Tronel。2017。《超收集语义及其在信息流静态分析中的应用》。载于编程语言原理研讨会，POPL 2017。doi:10.1145/3009837.3009889 Ali Bajwa、Minjian Zhang、Rohit Chadha 和 Mahesh Viswanathan。2023。《栈感知超属性》。载于系统构造与分析工具和算法国际会议，TACAS 2023。doi:10.1007/978-3-031- 30823-9 16_Gilles Barthe、Pedro R. D'Argenio 和 Tamara Rezk。2004。《通过自组合实现安全信息流》。载于计算机安全基础研讨会，CSFW 2004。doi:10.1109/CSFW.2004.17 Ezio Bartocci、Thomas Ferrère、Thomas A. Henzinger、Dejan Nickovic 和 Ana Oliveira da Costa。2022。《信息流接口》。载于软件工程基础方法国际会议，FASE 2022。doi:10.1007/978-3- 030-99429-7 1_Ezio Bartocci、Thomas A. Henzinger、Dejan Nickovic 和 Ana Oliveira da Costa。2023。《超节点自动机》。载于并发理论国际会议，CONCUR 2023。doi:10.4230/LIPICS.CONCUR.2023.21 Jon Barwise。1977。《一阶逻辑导论》。载于《逻辑与数学基础研究》。Jan Baumeister、Norine Coenen、Borzoo Bonakdarpour、Bernd Finkbeiner 和 César Sánchez。2021。《一种用于异步超属性的时序逻辑》。载于国际计算机辅助验证会议，CAV 2021。doi:10.1007/978-3-030-81685-8 33_Nick Benton。2004。《用于静态分析和程序变换的简单关系正确性证明》。载于编程语言原理研讨会，POPL 2004。doi:10.1145/964001.964003 Raven Beutner。2024。《超活性的自动化软件验证》。载于系统构造与分析工具和算法国际会议，TACAS 2024。doi:10.1007/978-3-031-57249-4_10 Raven Beutner 和 Bernd Finkbeiner。2022a。《用于超属性验证的预言变量》。载于计算机安全基础研讨会，CSF 2022。doi:10.1109/CSF54842.2022.9919658 Raven Beutner 和 Bernd Finkbeiner。2022b。《超越 k-安全性的超属性软件验证》。载于国际计算机辅助验证会议，CAV 2022。doi:10.1007/978-3-031-13185-1_17 Raven Beutner 和 Bernd Finkbeiner。2023a。《AutoHyper：面向 HyperLTL 的显式状态模型检测》。载于系统构造与分析工具和算法国际会议，TACAS 2023。doi:10.1007/978-3-031- 30823-9 8_Raven Beutner 和 Bernd Finkbeiner。2023b。《HyperATL*：一种用于多智能体系统中超属性的逻辑》。《计算机科学逻辑方法》（2023）。doi:10.46298/LMCS-19(2:13)2023 Raven Beutner 和 Bernd Finkbeiner。2024。《用于超属性验证的非确定性规划》。载于自动化规划与调度国际会议，ICAPS 2024。doi:10.1609/ICAPS.V34I1.31457

Raven Beutner 和 Bernd Finkbeiner。2025a。《AutoHyper：利用语言包含性检查进行超属性模型检测》。《国际软件工具技术与转移杂志》（2025）。doi:10.1007/s10009-025-00801-5 Raven Beutner 和 Bernd Finkbeiner。2025b。《用于超属性验证的完全信息多人博弈》。载于国际自主智能体与多智能体系统会议，AAMAS 2025。doi:10.5555/3709347.3743896 Raven Beutner 和 Bernd Finkbeiner。2025c。《关于超属性验证、量词交替与部分信息博弈》。载于计算机辅助设计中的形式化方法，FMCAD 2025。Raven Beutner、Bernd Finkbeiner、Hadar Frenkel 和 Niklas Metzger。2023。《二阶超属性》。载于国际计算机辅助验证会议，CAV 2023。doi:10.1007/978-3-031-37703-7_15 Raven Beutner、Tzu-Han Hsu、Borzoo Bonakdarpour 和 Bernd Finkbeiner。2024。《面向超属性的语法引导自动程序修复》。载于国际计算机辅助验证会议 CAV 2024。doi:10.1007/978-3-031- 65633-0 1_Tewodros A. Beyene、Swarat Chaudhuri、Corneliu Popeea 和 Andrey Rybalchenko。2014。《一种求解无限图上博弈的基于约束的方法》。载于编程语言原理研讨会，POPL 2014。doi:10.1145/ 2535838.2535860 Sebastian Biewer、Rayna Dimitrova、Michael Fries、Maciej Gazda、Thomas Heinze、Holger Hermanns 和 Mohammad Reza Mousavi。2022。《用于时空兴奋剂检测的符合关系与超属性》。《计算机科学逻辑方法》（2022）。doi:10.46298/LMCS-18(1:14)2022 Laura Bozzelli、Bastien Maubert 和 Sophie Pinchinat。2015。《统一超时序逻辑与认知时序逻辑》。载于软件科学基础与计算结构国际会议，FoSSaCS 2015。doi:10.1007/978-3-662-46678- 0 11_Laura Bozzelli、Adriano Peron 和 César Sánchez。2021。《HyperLTL 的异步扩展》。载于计算机科学逻辑研讨会，LICS 2021。doi:10.1109/LICS52264.2021.9470583 Laura Bozzelli、Adriano Peron 和 César Sánchez。2022。《异步超属性时序逻辑的表达能力与可判定性》。载于并发理论国际会议，CONCUR 2022。doi:10.4230/LIPIcs.CONCUR. 2022.27 Swarat Chaudhuri、Sumit Gulwani 和 Roberto Lublinerman。2012。《程序的连续性与鲁棒性》。《ACM 通讯》（2012）。doi:10.1145/2240236.2240262 Minki Cho、Youngju Song、Dongjae Lee、Lennard Gäher 和 Derek Dreyer。2023。《免费停顿》。《ACM 程序设计语言会议录》OOPSLA（2023）。doi:10.1145/3622857 Alessandro Cimatti、Edmund M. Clarke、Enrico Giunchiglia、Fausto Giunchiglia、Marco Pistore、Marco Roveri、Roberto Sebastiani 和 Armando Tacchella。2002。《NuSMV 2：一种用于符号模型检测的开源工具》。载于国际计算机辅助验证会议，CAV 2002。doi:10.1007/3-540-45657-0_29 Michael R. Clarkson、Bernd Finkbeiner、Masoud Koleini、Kristopher K. Micinski、Markus N. Rabe 和 César Sánchez。2014。《超属性的时序逻辑》。载于安全与信任原理国际会议，POST 2014。doi:10.1007/978-3-642-54792-8 15_Michael R. Clarkson 和 Fred B. Schneider。2008。《超属性》。载于计算机安全基础研讨会，CSF 2008。doi:10.1109/CSF.2008.7 Norine Coenen、Bernd Finkbeiner、Christopher Hahn 和 Jana Hofmann。2019a。《超逻辑层次结构》。载于计算机科学逻辑研讨会，LICS 2019。doi:10.1109/LICS.2019.8785713 Norine Coenen、Bernd Finkbeiner、César Sánchez 和 Leander Tentrup。2019b。《验证超活性》。载于国际计算机辅助验证会议，CAV 2019。doi:10.1007/978-3-030-25540-4_7 Arthur Correnson 和 Bernd Finkbeiner。2025。《时序超活性的余归纳证明》。《ACM 程序设计语言会议录》POPL（2025）。doi:10.1145/3704889 Arthur Correnson、Tobias Nießen、Bernd Finkbeiner 和 Georg Weissenbacher。2024。《使用符号执行查找 ∀∃超漏洞》。《ACM 程序设计语言会议录》OOPSLA（2024）。doi:10.1145/3689761 Patrick Cousot 和 Radhia Cousot。1977。《抽象解释：通过构造或近似不动点实现程序静态分析的统一格模型》。载于编程语言原理研讨会，POPL 1977。doi:10.1145/512950.512973 Thibault Dardinier 和 Peter Müller。2024。《超霍尔逻辑：证明或证伪程序超属性》。《ACM 程序设计语言会议录》PLDI（2024）。doi:10.1145/3656437 Luca de Alfaro、Thomas A. Henzinger 和 Rupak Majumdar。2001。《无限状态博弈的符号算法》。载于并发理论国际会议，CONCUR 2001。doi:10.1007/3-540-44685-0_36 Robert Dickerson、Qianchuan Ye、Michael K. Zhang 和 Benjamin Delaware。2022。《RHLE：关系 ∀∃属性的模块化演绎验证》。载于亚洲编程语言与系统研讨会，APLAS 2022。doi:10.1007/978- 3-031-21037-2 4_Rayna Dimitrova、Bernd Finkbeiner 和 Hazem Torfah。2020。《马尔可夫决策过程的概率超属性》。载于自动化验证与分析技术国际研讨会，ATVA 2020。doi:10.1007/978-3-030-

59152-6 27_Emanuele D’Osualdo、Azadeh Farzan 和 Derek Dreyer。2022年。《组合式证明超安全性》。《 ACM 程序设计语言会议录》 OOPSLA （ 2022 ）。doi:10.1145/3563298 Alexandre Duret-Lutz 、 Etienne Renault、Maximilien Colange、Florian Renkin、Alexandre Gbaguidi Aisse、Philipp Schlehuber- Caissier、Thomas Medioni、Antoine Martin、Jérôme Dubois、Clément Gillard 和 Henrich Lauko。2022年。《从 Spot 2.0 到 Spot 2.10：有哪些新变化？》。载于国际计算机辅助验证会议，CAV 2022。doi:10.1007/978-3-031- 13188-2 9_Marco Eilers、Thibault Dardinier 和 Peter Müller。2023年。《 CommCSL：使用抽象交换性证明并发程序的信息流安全》。《ACM 程序设计语言会议录》PLDI（2023）。doi:10.1145/3591289 Marco Eilers、Peter Müller 和 Samuel Hitz。2020年。《模块化乘积程序》。《ACM 程序设计语言汇刊》（2020）。doi:10.1145/3324783 Azadeh Farzan 和 Zachary Kincaid。2018年。《线性算术博弈的策略合成》。《ACM 程序设计语言会议录》POPL（2018）。doi:10.1145/3158149 Azadeh Farzan、Dominik Klumpp 和 Andreas Podelski。2022年。《用于并发程序验证的可靠顺序化》。载于编程语言设计与实现国际会议，PLDI 2022。doi:10.1145/3519939.3523727 Azadeh Farzan 和 Anthony Vandikas。2020年。《安全性证明的归约》。《ACM 程序设计语言会议录》POPL（2020）。doi:10.1145/3371081 Bernd Finkbeiner、Christopher Hahn、Jana Hofmann 和 Leander Tentrup。2020年。《实现 ω-正则超属性》。载于国际计算机辅助验证会议，CAV 2020。doi:10.1007/978-3-030-53291-8_4 Bernd Finkbeiner、Markus N. Rabe 和 César Sánchez。2015年。《HyperLTL 与 HyperCTL 的模型检测算法*》。载于国际计算机辅助验证会议，CAV 2015。doi:10.1007/978-3-319-21690-4_3 Giuseppe De Giacomo、Paolo Felli、Marco Montali 和 Giuseppe Perelli。2021年。《HyperLDLf：用于检查有限轨迹过程日志属性的逻辑》。载于国际人工智能联合会议，IJCAI 2021。doi:10.24963/IJCAI. 2021/256 Vladimir Gladshtein、Qiyuan Zhao、Willow Ahrens、Saman P. Amarasinghe 和 Ilya Sergey。2024年。《关于结构化数据的机械化超安全性证明》。《 ACM 程序设计语言会议录》PLDI（2024）。doi:10.1145/3656403 Joseph A. Goguen 和 José Meseguer。1982年。《安全策略与安全模型》。载于安全与隐私研讨会，SP 1982。doi:10.1109/SP.1982.10014 Ohad Goudsmid、Orna Grumberg 和 Sarai Sheinvald。2021年。《多重属性的组合式模型检测》。载于验证、模型检测与抽象解释国际会议，VMCAI 2021。doi:10.1007/978-3-030- 67067-2 4_Susanne Graf 和 Hassen Saïdi。1997年。《使用 PVS 构造抽象状态图》。载于国际计算机辅助验证会议，CAV 1997。doi:10.1007/3-540-63166-6_10 Marco Guarnieri、Boris Köpf、José F. Morales、Jan Reineke 和 Andrés Sánchez。2020年。《Spectector：推测性信息流的原则化检测》。载于安全与隐私研讨会，SP 2020。doi:10.1109/SP40000.2020.00011 Jens Oliver Gutsfeld、Arne Meier、Christoph Ohrem 和 Jonni Virtema。2022年。《重新审视时序团队语义》。载于计算机科学逻辑研讨会，LICS 2022。doi:10.1145/3531130.3533360 Jens Oliver Gutsfeld、Markus Müller-Olm 和 Christoph Ohrem。2020年。《用于超属性的命题动态逻辑》。载于并发理论国际会议，CONCUR 2020。doi:10.4230/LIPICS.CONCUR.2020.50 Jens Oliver Gutsfeld、Markus Müller-Olm 和 Christoph Ohrem。2021年。《异步超属性的自动机与不动点》。《ACM 程序设计语言会议录》POPL（2021）。doi:10.1145/3434319 Jens Oliver Gutsfeld、Markus Müller-Olm 和 Christoph Ohrem。2024年。《递归程序的异步超属性判定》。《ACM 程序设计语言会议录》POPL（2024）。doi:10.1145/3632844 Philippe Heim 和 Rayna Dimitrova。2024年。《通过加速求解无限状态博弈》。《ACM 程序设计语言会议录》POPL（2024）。doi:10.1145/3632899 Maurice Herlihy 和 Jeannette M. Wing。1990年。《线性化：并发对象的正确性条件》。《 ACM 程序设计语言汇刊》（ 1990 ）。doi:10.1145/78969.78972 Tzu-Han Hsu 、 Borzoo Bonakdarpour 、 Bernd Finkbeiner 和 César Sánchez。2023年。《异步超属性的有界模型检测》。载于系统构造与分析工具和算法国际会议，TACAS 2023。doi:10.1007/978-3-031-30823-9 2_Marieke Huisman、Pratik Worah 和 Kim Sunesen。2006年。《观察确定性的时序逻辑刻画》。载于计算机安全基础研讨会，CSFW 2006）。doi:10.1109/CSFW.2006.6 Shachar Itzhaky、Sharon Shoham 和 Yakir Vizel。2024年。《超属性验证作为 CHC 可满足性问题》。载于欧洲编程研讨会，ESOP 2024。doi:10.1007/978-3-031-57267-8_9 Andreas Krebs、Arne Meier、Jonni Virtema 和 Martin Zimmermann。2018年。《用于超属性规约与验证的团队语义》。载于计算机科学数学基础国际研讨会，MFCS 2018。doi:10.4230/LIPICS.MFCS.2018.10

Maurice Laveaux、Wieger Wesselink 和 Tim A. C. Willemse。2022年。《符号奇偶博弈的即时求解》。载于系统构造与分析工具和算法国际会议，TACAS 2022。doi:10.1007/978- 3-030-99527-0 8_Xavier Leroy。2006年。《编译器后端的形式化认证，或：使用证明助手编写编译器》。载于编程语言原理研讨会，POPL 2006。doi:10.1145/1111037.1111042 Kenji Maillard、Catalin Hritcu、Exequiel Rivas 和 Antoine Van Muylder。2020年。《接下来的 700 个关系程序逻辑》。《ACM 程序设计语言会议录》POPL（2020）。doi:10.1145/3371072 Donald A Martin。1975年。《博雷尔确定性》。《数学年刊》2（1975）。Daryl McCullough。1988年。《非干扰与安全属性的可组合性》。载于安全与隐私研讨会，SP 1997。doi:10.1109/SECPRI.1988.8110 John McLean。1994年。《选择性交错函数封闭的轨迹集的组合一般理论》。载于安全与隐私研讨会，SP 1994。doi:10.1109/RISP.1994.296590 Robert McNaughton。1993年。《在有限图上进行的无限博弈》。《纯粹与应用逻辑年刊》（1993）。doi:10.1016/0168- 0072(93)90036-D Robin Milner。1971年。《程序间模拟的代数定义》。载于国际人工智能联合会议，IJCAI 1971。http://ijcai.org/Proceedings/71/Papers/044.pdf Kedar S. Namjoshi。1997年。《停顿双模拟的简单刻画》。载于软件技术与理论计算机科学基础会议，FSTTCS 1997。doi:10.1007/BFB0058037 Kedar S. Namjoshi 和 Lucas M. Tabajara。2020年。《见证安全编译》。载于验证、模型检测与抽象解释国际会议，VMCAI 2020。doi:10.1007/978-3-030-39322-9_1 Nir Piterman。2007年。《从非确定性 Büchi 自动机和 Streett 自动机到确定性奇偶自动机》。《计算机科学逻辑方法》（2007）。doi:10.2168/LMCS-3(3:5)2007 Amir Pnueli。1977年。《程序时序逻辑》。载于计算机科学基础研讨会，FOCS 1977。doi:10.1109/SFCS.1977.32 Markus N. Rabe。2016年。《信息流控制的时序逻辑方法》。博士学位论文。萨尔大学。John H. Reif。1984年。《不完全信息双人博弈的复杂度》。《计算机系统科学杂志》（1984）。doi:10. 1016/0022-0000(84)90034-5 Stanly Samuel、Deepak D’Souza 和 Raghavan Komondoor。2021年。《GenSys：用于在无限状态空间上进行最大控制器合成的可扩展不动点引擎》。载于联合欧洲软件工程会议与软件工程基础研讨会，ESEC/FSE 2021。doi:10.1145/3468264.3473126 Anne-Kathrin Schmuck、Philippe Heim、Rayna Dimitrova 和 Satya Prakash Nayak。2024年。《无限状态博弈的局部吸引子计算》。载于国际计算机辅助验证会议，CAV 2024。doi:10.1007/978-3-031- 65633-0 7_Ron Shemer、Arie Gurfinkel、Sharon Shoham 和 Yakir Vizel。2019年。《属性导向的自组合》。载于国际计算机辅助验证会议，CAV 2019。doi:10.1007/978-3-030-25540-4_9 Marcelo Sousa 和 Isil Dillig。2016年。《用于验证 k-安全属性的笛卡尔霍尔逻辑》。载于编程语言设计与实现会议，PLDI 2016。doi:10.1145/2908080.2908092 Colin Stirling。1995年。《进程的模态逻辑与时序逻辑》。载于《并发逻辑——结构与自动机》。doi:10.1007/3-540-60915-6_5Tachio Terauchi。2008年。《观察确定性的类型系统》。载于计算机安全基础研讨会，CSF 2008。doi:10.1109/CSF.2008.9 Hiroshi Unno、Tachio Terauchi 和 Eric Koskinen。2021年。《基于约束的关系验证》。载于国际计算机辅助验证会议，CAV 2021。doi:10.1007/978-3-030-81685-8_35 Tom van Dijk。2018年。《Oink：现代奇偶游戏求解器的实现与评估》。载于系统构造与分析工具和算法国际会议，TACAS 2018。doi:10.1007/978-3-319-89960-2_16 Huanyu Wang、Domenic Forte、Mark M. Tehranipoor 和 Qihang Shi。2017年。《集成电路探测攻击：挑战与研究机遇》。《IEEE 设计与测试》（2017）。doi:10.1109/MDAT.2017.2729398 Steve Zdancewic 和 Andrew C. Myers。2003年。《并发程序安全的观察确定性》。载于计算机安全基础研讨会，CSFW 2003。doi:10.1109/CSFW.2003.1212703

A A-HLTL 语法，出自 Baumeister 等人 [2021]

Baumeister 等人提出的 A-HLTL 原始变体 [2021],与本文研究的变体不同。具体而言，我们的变体直接对轨迹停顿进行量化，并避免使用所谓轨迹序列的记法。本节将回顾 Baumeister 等人 [2021] 的变体，并证明我们的变体具有同等的表达能力。

A.1 Baumeister 等人的 A-HLTL [2021]

设 V 为轨迹变量集合，W $=\{\tau, \tau_1, \tau_2, \ldots\}$ 为轨迹序列变量集合。我们定义 $X_{V,W} := \{x_{\pi,\tau} \mid x \in X\ \pi \in V\ \tau \in W\}$，即用一个轨迹变量和一个轨迹序列变量对系统变量进行索引。A-HLTL 公式由以下语法定义

$$\varphi := \exists \pi.\, \varphi \mid \forall \pi.\, \varphi \mid \phi$$
$$\phi := E\tau.\, \phi \mid A\tau.\, \phi \mid \psi$$
$$\psi := \theta \mid \neg\psi \mid \psi \wedge \psi \mid \bigcirc\psi \mid \psi\, \mathcal{U}\, \psi$$

其中 $\pi \in V$，$\tau \in W$ 是一个轨迹序列，而 $\theta \in F_{X_{V,W}}$。我们首先对轨迹进行量化（类别φ），然后对轨迹序列进行二级量化（类别 ϕ），最后是一个 LTL 主体ψ。直观而言，每个轨迹-轨迹对（π, τ）都描述了轨迹 π 的一次（公平）停顿。LTL 主体中的原子公式是一阶公式 $\theta \in F_{X_{V,W}}$，即关于带索引的系统变量的公式。与我们的定义不同，这里的变量不是由停顿变量索引，而是由轨迹和轨迹序列组成的一对索引。这里，每个变量 $x_{\pi,\tau}$ 指的是变量 x 在轨迹 π 上、轨迹序列（停顿）τ 下的取值。因此，与我们所采用的变体的主要区别在于，通过量化单个轨迹序列 τ，我们隐式地同时为所有轨迹量化停顿，也就是说，$(\pi_1, \tau) \ldots, (\pi_n, \tau)$ 都分别表示轨迹 $\pi_1, \ldots, \pi_n$ 的（独立）停顿。我们的 A-HLTL 变体通过为每条轨迹单独量化一个停顿，将这一点明确表示出来。

我们假设 A-HLTL 公式是闭合的，即，对于 LTL 主体ψ中使用的所有原子公式 θ，以及在 θ 中使用的任意变量 $x_{\pi,\tau}$，轨迹变量 π和轨迹序列变量 τ均由某个外层量词约束。

语义。Baumeister 等人 [2021] 通过显式的轨迹序列来定义 A-HLTL 语义，并引导轨迹的停顿。我们采用一种更简单（但等价）的语义，不涉及轨迹序列，而是直接对轨迹的停顿扩展进行推理，遵循 Bozzelli 等人 [2022,命题 1]。

设 $\mathbb{T} \subseteq (X \to \mathbb{V})^\omega$ 为一组轨迹。在 A-HLTL 的语义中，我们维护一个轨迹赋值$\Pi : V \rightharpoonup (X \to \mathbb{V})^\omega$，将轨迹变量映射到轨迹（用于求值轨迹量化）。若对于每个 $\pi \in V$，轨迹赋值 Π' 都是 Π 的停顿，当且仅当 $\Pi'(\pi) \blacktriangleright \Pi(\pi)$，我们就称 Π' 是 Π 的停顿。我们用 FSE(Π) 表示 Π的所有停顿构成的集合。为处理轨迹序列，我们使用高阶轨迹赋值$\Delta : W \rightharpoonup (V \rightharpoonup (X \to \mathbb{V})^\omega)$，将轨迹序列变量映射到轨迹赋值。直观地说，$\Delta(\tau)$是轨迹序列 τ下经过停顿处理的轨迹赋值。给定 $\Delta, x \in X$、$\pi \in V$、$\tau \in W$ 和 $i \in \mathbb{N}$，变量$x_{\pi,\tau}$ 在第 i步的值因此为 $\Delta(\tau)(\pi)(i)(x)$；也就是说，带索引变量$x_{\pi,\tau}$ 的值定义为轨迹赋值中绑定到 τ 的轨迹上、绑定到 π 的变量 x 在第 i步的值。给定轨迹赋值 Δ 和位置 $i \in \mathbb{N}$，我们将扩展变量求值定义为 $\Delta_{(i)} : X_{V,W} \to \mathbb{V}$，其定义为 $\Delta_{(i)}(x_{\pi,\tau}) := \Delta(\tau)(\pi)(i)(x)$。现在我们可以定义其语义：

$$\Pi, \Delta, i \models_{\mathbb{T}} \exists \pi.\, \varphi \quad \text{iff} \quad \exists t \in \mathbb{T}.\, \Pi[\pi \mapsto t], \Delta, i \models_{\mathbb{T}} \varphi$$
$$\Pi, \Delta, i \models_{\mathbb{T}} \forall \pi.\, \varphi \quad \text{iff} \quad \forall t \in \mathbb{T}.\, \Pi[\pi \mapsto t], \Delta, i \models_{\mathbb{T}} \varphi$$
$$\Pi, \Delta, i \models_{\mathbb{T}} E\tau.\, \phi \quad \text{iff} \quad \exists \Pi' \in \mathit{FSE}(\Pi).\, \Pi, \Delta[\tau \mapsto \Pi'], i \models_{\mathbb{T}} \phi$$

$$
\begin{array}{ll}
\Pi, \Delta, i \models_{\mathbb{T}} \mathsf{A}\tau.\, \phi & \text{iff} \quad \forall \Pi' \in \mathit{FSE}(\Pi).\, \Pi, \Delta[\tau \mapsto \Pi'], i \models_{\mathbb{T}} \phi \\
\Pi, \Delta, i \models_{\mathbb{T}} \theta & \text{iff} \quad \Delta_{(i)} \models^{\mathfrak{T}} \theta \\
\Pi, \Delta, i \models_{\mathbb{T}} \neg\psi & \text{iff} \quad \Pi, \Delta, i \not\models_{\mathbb{T}} \psi \\
\Pi, \Delta, i \models_{\mathbb{T}} \psi_1 \wedge \psi_2 & \text{iff} \quad \Pi, \Delta, i \models_{\mathbb{T}} \psi_1 \text{ and } \Pi, \Delta, i \models_{\mathbb{T}} \psi_2 \\
\Pi, \Delta, i \models_{\mathbb{T}} \bigcirc\psi & \text{iff} \quad \Pi, \Delta, i+1 \models_{\mathbb{T}} \psi \\
\Pi, \Delta, i \models_{\mathbb{T}} \psi_1 \,\mathcal{U}\, \psi_2 & \text{iff} \quad \exists j \geq i.\, \Pi, \Delta, j \models_{\mathbb{T}} \psi_2 \text{ and } \forall i \leq k < j.\, \Pi, \Delta, k \models_{\mathbb{T}} \psi_1.
\end{array}
$$

首先，遵循轨迹量词前缀，构造轨迹赋值 Π，并将轨迹添加到 Π中。然后，对于每条轨迹，我们考虑 Π 的 FSE（全称量化或存在量化），并将其添加到 Δ中。最后，我们可以按照通常的方式，对布尔运算符和时序运算符求值，从而求出 LTL 主体。此处，若变量赋值 $\Delta_{(i)}$（它为 X$\mathcal{V}$，$\mathcal{W}$中的带索引变量赋值）满足 θ（模理论 $\mathfrak{T}$），则原子公式 θ 在第 i 步成立。

A.2 从轨迹序列到停顿

不难看出，我们的 A-HLTL 变体等价于 [Baumeister 等人 2021；Hsu 等人 2023]所提出的基于轨迹的变体。直观而言，我们的定义更为冗长，因为它显式地为每条轨迹量化停顿，而基于轨迹的变体则使用单条轨迹，为所有轨迹上的停顿进行量化。我们分别证明两个方向。

给定一个基于轨迹的公式

$$\mathbb{Q}_1\pi_1 \ldots \mathbb{Q}_n\pi_n.\mathrm{Q}_1\tau_1 \ldots \mathrm{Q}_k\tau_k.\, \psi$$

其中 Q_1、……、$\mathrm{Q}_n \in \{\forall,\ \exists\}$ 和 Q_1、……、$\mathrm{Q}_k \in \{\mathrm{A}、\mathrm{E}\}$，我们可以将每个轨迹—轨迹序列对 $(\pi_i,\ \tau_j)$ 替换为一个新的停顿变量 $\beta_{\pi_i,\ \tau_j}$。形式化地，我们定义停顿变量集合 $\mathrm{B} := \{\beta_{\pi,\ \tau} \mid \pi \in \mathrm{V}\ \tau \in \mathrm{W}\}$，并考虑 A-HLTL 公式

$$\mathbb{Q}_1\pi_1 \ldots \mathbb{Q}_n\pi_n.\, \overline{\mathrm{A}_1}\beta_{\pi_1,\tau_1} \blacktriangleright \pi_1. \cdots \overline{\mathrm{A}_1}\beta_{\pi_n,\tau_1} \blacktriangleright \pi_n. \cdots \overline{\mathrm{A}_k}\beta_{\pi_1,\tau_k} \blacktriangleright \pi_1 \cdots \overline{\mathrm{A}_k}\beta_{\pi_n,\tau_k} \blacktriangleright \pi_n.\, \overline{\psi}$$

其中，我们定义 $\overline{\mathrm{A}} := \forall$和 $\overline{\mathrm{E}} := \exists$，以用停顿量化替代轨迹量化。我们将 $\overline{\psi}$定义为这样一个线性时序逻辑公式：它由 ψ通过将每个变量 $x_{\pi,\tau}$（由一个轨迹-轨迹对索引）替换为变量$x_{\beta_{\pi,\tau}}$（由一个停顿变量索引）得到。

示例 *A.1.* 考虑公式

$$\forall\pi_1.\forall\pi_2.\mathrm{E}\tau.\, \square(o_{\pi_1,\tau} = o_{\pi_2,\tau}).$$

使用我们的编码，我们将其转换为公式

$$\forall\pi_1.\forall\pi_2.\exists\beta_1 \blacktriangleright \pi_1.\exists\beta_2 \blacktriangleright \pi_2.\, \square(o_{\beta_1} = o_{\beta_2}),$$

也就是说，对于每个轨迹-轨迹对 $((\pi_1, \tau)$ 和 $(\pi_2, \tau))$，我们只需对该轨迹的一个停顿进行量化。注意，我们用 β_1 表示 $\beta_{\pi_1,\tau}$，用 β_2 表示 $\beta_{\pi_2,\tau}$。△

对于每种 $n \times k$ 轨迹-轨迹对组合，我们只需显式加入停顿。虽然这样写更冗长，但也能更清楚地突出实际发生的事情。事实上，我们认为，A-HLTL 难以理解的一个核心原因是，许多量化都是隐式进行的：每当我们对一条轨迹进行量化时，我们都会对所有轨迹的停顿进行量化，尽管这些停顿彼此完全独立。

A.3 从停顿到轨迹序列

对于另一个方向，我们只需将每个停顿量词替换为对一条唯一轨迹的量化。形式上，给定一个基于停顿的 A-HLTL 公式

$$\mathbb{Q}_1\pi_1 \dots \mathbb{Q}_n\pi_n.\mathbb{Q}'_1\beta_1 \blacktriangleright \pi_{k_1} \dots \mathbb{Q}'_m\beta_m \blacktriangleright \pi_{k_m}.\ \psi$$

我们定义等价的基于轨迹的公式

$$\mathbb{Q}_1\pi_1 \dots \mathbb{Q}_n\pi_n.\overline{\mathbb{Q}'_1}\tau_{\beta_1} \dots \overline{\mathbb{Q}'_m}\tau_{\beta_m}.\ \overline{\psi}$$

其中我们定义 $\forall = \overline{\mathrm{A}}$ 和 $\exists = \overline{\mathrm{E}}$。我们将新的线性时序逻辑主体 $\overline{\psi}$ 定义为 ψ，其中将每个变量 x_β 替换为 $x_{base(\beta),\ \tau_\beta}$。因此，每条轨迹 τ_β 都只用于唯一轨迹（即 基迹(β)）上；也就是说，我们不再对轨迹 π 的停顿 β 进行量化，而是对轨迹τ_β序列进行量化（因此是对所有 轨迹的停顿进行量化），但只在轨迹 π = 基迹(β)上使用轨迹τ_β。

示例 *A.2.* 考虑公式

$$\forall\pi_1.\forall\pi_2.\exists\beta_1 \blacktriangleright \pi_1.\exists\beta_2 \blacktriangleright \pi_2.\ \square(o_{\beta_1} = o_{\beta_2}),$$

使用我们的编码，我们将其转换为公式

$$\forall\pi_1.\forall\pi_2.\mathrm{E}\tau_{\beta_1}.\mathrm{E}\tau_{\beta_2}.\ \square(o_{\pi_1,\tau_{\beta_1}} = o_{\pi_2,\tau_{\beta_2}}).$$

请注意，这种编码并非最优（在此示例中，我们可以在 π_1和 π_2上使用相同的轨迹），但仍能得到等价的公式。△

B 正确性证明

本节中，我们通过证明定理 5.5，证明我们基于博弈的方法对于验证全称-存在 A-HLTL 性质是可靠的。

定理 5.5（正确性）。考虑任意迁移系统$\mathcal{T}$、一个 $\forall^*\exists^*$ *A-HLTL* 公式 φ和一个界 $Z \in \mathbb{N}_{\geq 1}$。若 $wins_{\mathcal{G}_{\mathcal{T},\varphi,Z}}(\mathfrak{V}, V_{init})$，则$\mathcal{T} \models \varphi$。

为证明定理 5.5，我们区分 φ 是否包含全称量化停顿。我们在命题B.1 中证明前一种情况，在命题B.2中证明后一种情况。

命题 B.1。 假设 φ 是一个不含全称量化停顿的 $\forall^*\exists^*$ *A-HLTL* 公式。如果 $wins_{\mathcal{G}_{\mathcal{T},\ \varphi,\ Z}}(\mathfrak{V}, V_{init})$，那么 $\mathcal{T} \models \varphi$。

证明。由于 φ 不包含全称量化停顿，因此它具有如下形式

$$\varphi = \forall\pi_1 \dots \pi_n.\ \exists\pi_{n+1} \dots \pi_{n+m}.\ \exists\beta_1 \blacktriangleright \pi_{l_1} \dots \beta_k \blacktriangleright \pi_{l_k}.\ \psi$$

其中 l_1、…、$l_k \in \{1, \dots, n+m\}$。我们假设 $wins_{\mathcal{G}_{\mathcal{T},\varphi,Z}}(\mathfrak{V}, V_{init})$，因此令 σ 为 $G_{\mathcal{T},\varphi,Z}$ 中验证者的获胜策略。为证明 $T \models \varphi$，任取全称量化轨迹变量 π_1、…、π_n 对应的 t_1、…、$t_n \in Traces(T)$ 中的轨迹，并令 p_1、…、$p_n \in Paths(T)$ 中的路径生成 t_1、…、t_n（即，对于所有 $1 \leq i \leq n$，$\ell_{\mathcal{T}}(p_i) = t_i$）。首先，我们将利用 σ 为存在量化的 π_{n+1}、…、π_{n+m}构造t_{n+1}、…、$t_{n+m} \in Traces(T)$ 中的轨迹。为此，考虑 $G_{\mathcal{T},\varphi,Z}$ 中的初始顶点

$$v_0 := \Big\langle ⓤ, \big[\pi_1 \mapsto p_1[0, Z-1], \dots, \pi_n \mapsto p_n[0, Z-1], \pi_{n+1} \mapsto [s_{0,\mathcal{T}}], \dots, \pi_{n+m} \mapsto [s_{0,\mathcal{T}}]\big],$$
$$\big[\beta_i \mapsto 0\big]_{i=0}^{k}, \emptyset, q_{0,\psi} \Big\rangle$$

其中，每条全称量化轨迹 π_1、……、π_n 分别从路径 p_1、……、p_n上长度为 Z 的初始状态窗口开始，而 π_{n+1}、……、π_{n+m}从长度为 1 的初始状态窗口开始。如$v_0 \in V_{init}$所示，策略 σ从 v_0出发赢得 $G_{\mathcal{T},\varphi,Z}$上的 $\mathfrak{V}$。令 $V = V_{\mathfrak{V}} \cup V_{\mathfrak{R}}$ 为 $G_{\mathcal{T},\varphi,Z}$中的顶点集合。

现在，我们通过模拟 σ，使用固定路径 p_1，……，p_n，以此迭代地构造博弈 $\vec{v} \in V^\omega$；也就是说，我们让反驳者始终按照p_1，……，p_n扩展状态窗口，从而向验证者发起挑战。然后，我们将从 σ的响应中提取轨迹 t_{n+1}，……，t_{n+m}。为了以形式化方式进行这一模拟，我们会跟踪索引 c_1，……，$c_n \in \mathbb{N}$。直观地说，每个 c_i都跟踪 p_i上的一个位置；在下次有机会时，应将该位置添加到 π_i的状态窗口中。

构造 $\vec{v}$

首先，我们设置 $\vec{v} = [v_0]$和 $c_1 = \cdots = c_n = Z$，即，应添加到 π_i 的状态窗口中的下一个位置是 p_i 上的第 Z 个位置。然后，我们重复以下操作（无限次）：

令 v 为 $\vec{v}$ 中的最后一个顶点。

（1）如果 $v = \langle \cup\ \Xi,\ \mu,\ \flat\, q \rangle$，我们将 v' 定义为 v的唯一后继状态（参见图 6）。此外，我们定义

$$mo = \{\pi \mid \forall \beta.\, (base(\beta) = \pi) \Rightarrow \mu(\beta) \neq 0\}$$

对于所有全称量化轨迹变量，其中没有任何停顿会 φ 指向第一个位置（类似于图 6 **(3.1)**）。对于每个 $\pi_i \in mo$，我们定义 $c_i := c_i + 1$。直观而言，这考虑到了这样一个事实：在下一个 $\forall$ 阶段中，我们可以向 π_i的状态窗口添加一个新状态，因为第一个状态将被移除。

(2) 如果 $v = \exists,\ \Xi,\ \mu,\ \flat\, q$，我们定义 $v' := \sigma(v)$，即查询验证者的获胜策略。

(3) 如果 $v = \forall,\ \Xi,\ \mu,\ \flat\, q$，我们定义 $v' := \exists,\ \Xi',\ \mu,\ \flat\, q$，其中

$$\Xi' := \left[\pi_1 \mapsto \Xi(\pi_1) \cdot p_1(c_1), \ldots, \pi_n \mapsto \Xi(\pi_n) \cdot p_n(c_n), \pi_{n+1} \mapsto \Xi(\pi_{n+1}), \ldots, \pi_{n+m} \mapsto \Xi(\pi_{n+m})\right].$$

也就是说，对于每条全称量化轨迹 π_i，我们将状态 $p_i(c_i)$ 追加到 π_i 的状态窗口中，即按照固定路径 p_i（使用索引 c_i）扩展状态窗口。之后，我们将 v' 追加到 $\vec{v}$，即定义 $\vec{v} := \vec{v} \cdot v'$，然后重复此过程。

构造 $\vec{v}$–结束

令 $\vec{v} \in V^\omega$ 为上述构造计算得到的无限对局。容易看出，$\vec{v}$ 是 $\mathrm{G}_{\mathcal{T},\varphi,Z}$ 中的一场博弈，且与 σ 兼容。由于 σ 对 $\mathfrak{W}$而言是获胜的，因此博弈 $\vec{v}$ 由 $\mathfrak{W}$获胜。根据 ψ_{mod} 的构造（参见第 5.1 节），这意味着验证者会无限多次推进所有停顿（它们均为存在量化）。现在，我们将使用 $\vec{v}$ 迭代地构造路径p_{n+1}，…，$p_{n+m} \in \mathrm{Paths(T)}$，为存在量化轨迹变量$\pi_{n+1}$，…，$\pi_{n+m}$定义见证轨迹。直观来说，每当验证者将某个状态添加到 π_i 的状态窗口中时，我们就将该状态添加到 $p_i \in \{p_{n+1}, \ldots, p_{n+m}\}$中。形式上，对于 $n < i \leq n + m$，我们按如下方式构造每个 p_i：

构造 $p_{n+1}, \ldots, p_{n+m}$

我们首先设置 $p_i := \epsilon$ 和 $j = 0$。然后重复执行以下操作（无限次）：

(1) 如果 $\vec{v}(j) \in \cup\ \Xi, \mu, \flat\, q$，并且对于所有具有 基迹$(\beta) = \pi_i$ 的 $\mu(\beta) \neq 0$，即在 $\Xi(\pi_i)$ 中，π_i 上的所有停顿都已越过第一个状态，则我们定义 $p_i := p_i \cdot \Xi(\pi_i)(0)$，也就是说，将 π_i 的状态窗口中的第一个位置添加到 p_i 中。

(2) 否则，我们保持 p_i 不变。

之后，我们递增 $j := j + 1$，并重复执行。

构造 $p_{n+1}, \ldots, p_{n+m}$– 结束

随着 σ 无限次地推进所有停顿，并且我们仅当所有停顿都越过第一个位置时才向 p_i 添加一个状态，我们会无限多次地向 p_i添加状态。令 $p_{n+1}, \ldots, p_{n+m} \in Paths(\mathrm{T})$表示所得的无限路径。现在为 $n < i \leq n + m$定义轨迹 $t_i := \ell_{\mathcal{T}}(p_i)$，并令 $\Pi := [\pi_i \mapsto t_i]_{i=1}^{n}$表示所得的轨迹赋值。我们声称 $\Pi, \emptyset, 0 \models_{\mathcal{T}} \exists \beta_1 \blacktriangleright \pi_{l_1} \ldots \beta_k \blacktriangleright \pi_{l_k}.\, \psi$，即轨迹$t_{n+1}, \ldots, t_{n+m}$是 φ中存在量化轨迹变量的有效选择。为证明这一点，对于每个停顿 $\beta \in \{\beta_1, \ldots, \beta_k\}$，我们需要构造轨迹$base(\beta)$的一个（公平）停顿，使其满足 ψ。为构造这些停顿，我们再次使用此前构造的顶点序列 $\vec{v}$。

对于每个 $\beta_i \in \{\beta_1, \ldots, \beta_k\}$，我们定义一个无限状态序列 $r_{\beta_i} \in S_{\mathcal{T}}^{\omega}$；对于每个 $i \in \mathbb{N}$，将其定义为$r_{\beta_i}(j) := \Xi(\text{base}(\beta_i))(\mu(\beta_i))$，其中 $\vec{v}(3 \cdot j) \in \mathrm{U} \quad \Xi, \mu, \flat \; q$。也就是说，我们始终查看每第三个位置（即U阶段中的所有预点），并使用状态窗口 $\Xi(\text{base}(\beta_i))$中由 $\mu(\beta_i)$指向的状态。首先，很容易看出，对于每个 $\beta_i \in \{\beta_1, \ldots, \beta_k\}$，$r_{\beta_i}$是路径 $\Pi(\text{base}(\beta_i))$（即 $r_{\beta_i} \blacktriangleright \Pi(\text{base}(\beta_i))$）的公平停顿。其次，我们声称，停顿状态序列$\{r_{\beta_i}\}$满足 ψ，即

$$\Pi, \left[\beta_1 \mapsto \ell_{\mathcal{T}}(r_{\beta_1}), \cdots, \beta_k \mapsto \ell_{\mathcal{T}}(r_{\beta_k})\right], 0 \models_{\mathcal{T}} \psi.$$

这源于这样的假设：$\vec{v}$ 由 $\mathfrak{V}$获胜，即每个 U阶段中的状态共同满足 ψ_{mod}，因此也满足 ψ（参见第 5.1 节）。请注意，在 $\mathrm{G}_{\mathcal{T},\varphi,Z}$中，对于每个 U阶段，我们都会更新 A_{ψ}：针对每个停顿 β_i，取 μ（β_i）在 Ξ（base（β_i））中的状态；这与r_{β_i}的定义相同。综合以上各点，$\{r_{\beta_i}\}$是 p_1、……、p_{n+m}的（公平）停顿，并满足 ψ。根据 A-HLTL 语义，这已经意味着 $\Pi, \emptyset, 0 \models_{\mathcal{T}} \exists \beta_1 \blacktriangleright \pi_{l_1} \ldots \beta_k \blacktriangleright \pi_{l_k}. \psi$。对于 φ中的全称量化轨迹变量 π_1,……, π_n的任意取值，这一构造均成立，从而按要求证明了 $\mathrm{T} \models \varphi$。□

命题 B.2. 假设 φ 是一个包含全称量化停顿的 $\forall^*\exists^*$ 公式。如果获胜$\mathcal{G}_{\mathcal{T},\varphi,Z}(\mathfrak{V}, V_{init})$，那么$\mathcal{T} \models \varphi$。

证明。我们假设获胜$\mathcal{G}_{\mathcal{T},\varphi,Z}(\mathfrak{V}, V_{init})$。首先，我们注意到，这意味着对于 φ中的每个轨迹变量，至多存在一个 全称量化停顿：如果同一轨迹上至少有两个停顿，验证者就永远无法获胜，因为反驳者可以轻易地迫使轨迹访问v_{error}，只需让两个全称量化停顿分歧超过 Z 步。因此，在下文中，我们假设每个轨迹变量恰好存在一个全称量化停顿，即 φ 具有如下形式

$$\varphi = \forall \pi_1 \ldots \pi_n. \forall \beta_1' \blacktriangleright \pi_1, \ldots, \beta_n' \blacktriangleright \pi_n. \exists \beta_1 \blacktriangleright \pi_{l_1} \ldots \beta_k \blacktriangleright \pi_{l_k}. \psi.$$

其中 $l_1, \ldots, l_k \in \{1, \ldots, n\}$。也就是说，$\beta_1', \ldots, \beta_n'$分别是$\pi_1, \ldots, \pi_n$的固定全称量化停顿，而 $\beta_1, \ldots, \beta_k$是存在量化停顿。令 $\mathrm{T} = (S_{\mathcal{T}}, s_{0,\mathcal{T}}, \kappa_{\mathcal{T}}, \ell_{\mathcal{T}})$ 和 $\mathrm{A}_{\psi} = (Q_{\psi}, q_{0,\psi}, \delta_{\psi}, F_{\psi})$。我们假设 σ是博弈 $\mathrm{G}_{\mathcal{T},\varphi,Z}$中验证者的一种获胜策略。

为证明 $\mathrm{T} \models \varphi$，令 $t_1, \ldots, t_n \in \mathit{Traces}(\mathrm{T})$ 为任意的轨迹，对全称量化的$\pi_1, \ldots, \pi_n$而言；并令 $p_1, \ldots, p_n \in S_{\mathcal{T}}^{\omega}$为 T 中生成 $t_1, \ldots, t_n$ 的路径，即 $\ell_{\mathcal{T}}(p_i) = t_i$。令$\Pi := [\pi_1 \mapsto t_1, \ldots, \pi_n \mapsto t_n]$。此外，令 $p_1', \ldots, p_n' \in S_{\mathcal{T}}^{\omega}$为 $p_1, \ldots, p_n$ 的（公平）停顿（即$p_i' \blacktriangleright p_i$），它们对应于全称量化停顿 $\beta_1', \ldots, \beta_n'$。我们可以用二进制序列 $b_1', \ldots, b_n' \in \mathbb{B}^{\omega}$ 表示这些停顿，以指示路径何时推进。也就是说，对于每个 $j \in \mathbb{N}$，$b_i'(j) = \top$ 当且仅当在第 j 步，p_i'执行了一个非停顿步。

为了证明 $\mathrm{T} \models \varphi$，我们需要为（存在量化的）停顿 β_1、……、β_k 找到见证。具体而言，对于每个 $\beta_i \in \{\beta_1$、……、$\beta_k\}$，我们将构造一个序列 $r_{\beta_i} \in S_{\mathcal{T}}^{\omega}$作为路径$\Pi$(基迹($\beta_i$) 的停顿，使得

$$\Pi, \left[\beta_1' \mapsto \ell_{\mathcal{T}}(p_1'), \ldots, \beta_n' \mapsto \ell_{\mathcal{T}}(p_n'), \beta_1 \mapsto \ell_{\mathcal{T}}(r_{\beta_1}), \ldots, \beta_k \mapsto \ell_{\mathcal{T}}(r_{\beta_k})\right], 0 \models \psi.$$

与命题 B.1的证明中一样，我们将模拟 σ 以构造这些停顿路径。考虑 $\mathcal{G}_{\mathcal{T}}$ 中的初始顶点，φ，Z

$$v_0 := \left\langle \mathrm{U}, \left[\pi_1 \mapsto p_1[0, Z-1], \ldots, \pi_n \mapsto p_n[0, Z-1]\right], \left[\beta \mapsto 0\right]_{\beta \in \{\beta_1', \ldots, \beta_n', \beta_1, \ldots, \beta_k\}}, \emptyset, q_{0,\psi} \right\rangle$$

其中每条全称量化轨迹 π_i 都从路径 p_i 上长度为 Z 的初始状态窗口开始。现在我们迭代地构造一个顶点序列 $\vec{v} \in V^{\omega}$。在此序列中，我们将让反驳者模拟 $p_1, \ldots, p_n$ 以及停顿 $p_1', \ldots, p_n'$（对应于停顿 $\beta_1', \ldots, \beta_n'$），然后利用 σ 的响应，为 $\beta_1, \ldots, \beta_k$构造见证。我们跟踪变量 $c_1, \ldots, c_n \in \mathbb{N}$。

同样，每个 c_i 跟踪 p_i 上当前处于 π_i 的状态窗口最后 位置的位置，我们据此按照 p_i 扩展状态窗口。我们将 $\vec{v}$ 的构造描述为以下算法：

构造 $\vec{v}$

最初，我们设置 $\vec{v} = [v_0]$ 和 $c_1 = \cdots = c_n = Z$。我们设置 $j := 0$。然后重复以下步骤（无限次）：

令 $v = \vec{v}(j)$ 为 $\vec{v}$ 中的最后一个顶点。

(1) 若 $v = \mathrm{U}\ \Xi, \mu, \flat\ q$，则将 v' 定义为 v 的唯一后继状态（参见图 6）。对于窗口发生移动更新的所有状态（即移除第一个位置，从而在窗口末尾腾出一个位置），我们将计数器 $c_1, \ldots, c_n$ 加一。形式化地，我们将 $\mathrm{mo} = \{\pi \mid \forall \beta.\, (\text{基迹}(\beta) = \pi) \Rightarrow \mu(\beta) \neq 0\}$ 定义为所有（全称量化的）轨迹变量的集合，其中 φ 中没有任何停顿指向第一个位置（类似于图 6）。对于每个 $\pi_i \in \mathrm{mo}$，我们定义 $c_i := c_i + 1$。

(2) 若 $v = \exists, \Xi, \mu, \flat\ q$，我们定义 $v' := \sigma(v)$。

(3) 若 $v = \forall, \Xi, \mu, \flat\ q$，我们首先定义

$$\Xi' := \left[\pi_1 \mapsto \Xi(\pi_1) \cdot p_1(c_1), \ldots, \pi_n \mapsto \Xi(\pi_n) \cdot p_n(c_n)\right]$$

也就是说，对于每条全称量化轨迹 π_i，我们追加状态 $p_i(c_i)$，即按照固定路径 p_i 追加状态窗口。根据构造，计数器 $c_1, \ldots, c_n$ 始终指向 $p_1, \ldots, p_n$ 上的位置，而这些位置是窗口 $\Xi(\pi_1) \ldots, \Xi(\pi_n)$ 的后继状态。因此，我们追加的状态是 T 中的有效转移。我们定义

$$\mu' := \lambda\beta. \begin{cases} \mu(\beta) + 1 & \text{if } \beta = \beta'_i \wedge b'_i(j) = \top \text{ for some } i \\ \mu(\beta) & \text{otherwise} \end{cases}$$

也就是说，当 p'_i（π_i 的固定停顿）在第 j 步（由 $b'_i \in \mathbb{B}^\omega$ 给出）执行了一个非停顿步时，我们才推进全称量化停顿 β'_i（其基路径为 π_i）。这确保我们更新停顿指针 $\beta'_1, \ldots, \beta'_n$，从而重现路径 $p_1, \ldots, p_n$ 的固定停顿 $p'_1, \ldots, p'_n$。类似地，我们定义

$$\flat' := \flat \cup \{\beta'_i \mid b'_i(j) = \top\}$$

最后，我们定义 $v' := \langle \exists, \Xi', \mu', \flat', q \rangle$。不难看出，$\mathcal{G}_{\mathcal{T}, \varphi, Z}$ 允许从 v 转移到 v'；我们向所有状态窗口中都追加了一个状态，并且（可能）推进了全称量化停顿 $\beta'_1, \ldots\ldots, \beta'_n$。（参见图 4）。

随后，我们将 v' 追加到 $\vec{v}$，即定义 $\vec{v} := \vec{v} \cdot v'$。然后将 $j := j + 1$递增，并重复此过程。

构造 $\vec{v}$–结束

设 $\vec{v} \in V^\omega$ 为上述构造计算出的无限对局。不难看出，$\vec{v}$ 是 $\mathrm{G}_{\mathcal{T}, \varphi, Z}$ 中的一场与 σ 相容的博弈。由于根据假设，σ 获胜，因此博弈 $\vec{v}$ 由 $\mathfrak{W}$ 获胜。现在，根据假设，$p'_1, \ldots, p'_n$ 是 $p_1, \ldots, p_n$ 的公平停顿，因此布尔序列 $b'_1, \ldots, b'_n$ 中会有 $\top$ 无穷多次。根据 $\vec{v}$ 的构造，$\beta'_1, \ldots, \beta'_n$ 的停顿指针也因此会被无限多次推进。这意味着 ψ_{mod} 的前提 $\mathrm{moved}_{\beta'_1} \wedge \cdots \wedge \mathrm{moved}_{\beta'_n}$（参见第 5.1 节）成立。由于 $\vec{v}$ 由 σ 获胜，这意味着 ψ_{mod} 的结论也成立。特别是，σ 会无限多次推进所有停顿 $\beta_1, \ldots, \beta_k$。

现在我们将使用 $\vec{v}$ 为 $\beta \in \{\beta'_1, \ldots\ldots, \beta'_n, \beta_1, \ldots\ldots, \beta_k\}$ 构造停顿（类似于命题 B.1 的证明）。对于每条路径 $\beta \in \{\beta'_1, \ldots\ldots, \beta'_n, \beta_1, \ldots\ldots, \beta_k\}$，我们定义如下序列 $r_\beta \in S^\omega_{\mathcal{T}}$：对于每个 $j \in \mathbb{N}$，令 $\vec{v}(3 \cdot j) = \mathrm{U}\ \Xi, \mu, \flat\ q$，并定义 $r_\beta(j) = \Xi(\mathrm{base}(\beta))(\mu(\beta))$。也就是说，我们始终查看每第三个位置（即 U阶段中的所有顶点），并使用状态窗口 $\Xi(\mathrm{base}(\beta))$ 中由 $\mu(\beta)$ 指向的状态。注意，对于每个 $1 \leq i \leq n$，$r_{\beta'_i} = p'_i$，即全称量化停顿 $\beta'_1, \ldots\ldots, \beta'_n$ 的状态序列，恰好就是固定停顿

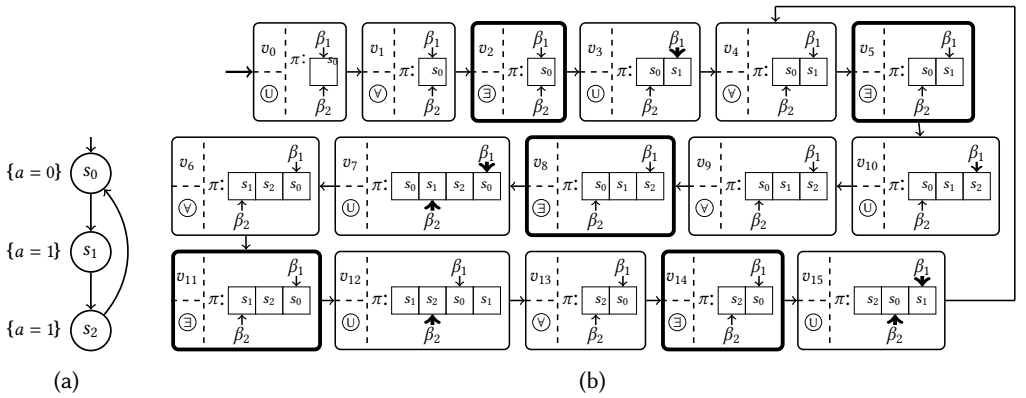


图 8. 图 8a展示了一个定义在系统变量 $\mathcal{X} = \{a\}$上的迁移系统。图 8b给出了验证者的一种获胜策略（详细内容请参见例 C.1）。与图 3类似，我们用箭头表示停顿指针；此处，一个表示停顿 β_1，另一个表示停顿 β_2。

p_1、……、p_n，我们在构造 $\vec{v}$ 时使用了该固定停顿（随着我们推进全称量化停顿，在 p'_i 对 p_i 执行非停顿步的那些步骤，即将 b'_i 设置为 true 的那些步骤中，恰好推进这些停顿）。

容易看出，对于每个 $\beta \in \{\beta'_1, \ldots, \beta'_n, \beta_1, \ldots, \beta_k\}$，$r_\beta$都是路径$\Pi(\text{基迹})(\beta)$ 的公平停顿（类似于命题 B.1的证明）。此外，我们声称路径 $\{r_\beta\}$ 满足ψ，即

$$\Pi, \left[\beta'_1 \mapsto \ell_{\mathcal{T}}(r_{\beta'_1}), \ldots, \beta'_n \mapsto \ell_{\mathcal{T}}(r_{\beta'_n}), \beta_1 \mapsto \ell_{\mathcal{T}}(r_{\beta_1}), \ldots, \beta_k \mapsto \ell_{\mathcal{T}}(r_{\beta_k})\right], 0 \models \psi.$$

这直接源于以下假设：$\vec{v}$由 $\mathfrak{V}$获胜：由于 $p'_1, \ldots, p'_n$都是公平停顿，ψ_{mod}的前提成立；而这又意味着（因为 $\mathcal{A}^\psi$跟踪 ψ_{mod}），ψ_{mod}的结论（尤其是 ψ）也成立（参见第 5.1节）。

因此 $\{r_\beta\}$ 是 $p_1, \ldots, p_n$ 的（公平）停顿，并满足 ψ；对于每个 $1 \leq i \leq n$，$r_{\beta'_i}$ 等于（全称量化的）p'_i。根据 A-HLTL 语义，这意味着

$$\Pi, \emptyset, 0 \models \forall \beta'_1 \blacktriangleright \pi_1, \ldots \beta'_n \blacktriangleright \pi_n. \exists \beta_1 \blacktriangleright \pi_{l_1} \ldots \beta_k \blacktriangleright \pi_{l_k}. \psi$$

正如所期望的那样。这一构造适用于对全称量化轨迹 $\pi_1, \ldots, \pi_n$ 的任意选择，以及对停顿 $\beta'_1, \ldots, \beta'_n$ 的所有选择；也就是说，对于所有此类选择，我们都可以使用 σ 构造停顿 $\beta_1, \ldots, \beta_k$，使其满足 ψ。这就按要求证明了 $\mathcal{T} \models \varphi$。□

C 窗口大小

在第 2 节的示例中，我们对每个轨迹变量只考虑了一种停顿，因此窗口大小为 2 就足以容纳所有停顿。如果同一轨迹变量对应多种停顿，情况就会有所不同。下面来看一个说明性示例。

例 C.1. 考虑图 8a 中的平凡迁移系统，它生成唯一轨迹 $([a \mapsto 0][a \mapsto 1][a \mapsto 1])^\omega$，并设 $\varphi := \exists \pi. \exists \beta_1 \blacktriangleright \pi. \exists \beta_2 \blacktriangleright \pi. (a_{\beta_1} \neq a_{\beta_2})$ 表示在该唯一轨迹上存在两个停顿，且它们在 a 的取值上始终不同。不难看出，验证者在 $G_{\mathcal{T},\varphi,1}$ 和 $G_{\mathcal{T},\varphi,2}$ 中失败。然而，验证者在 $G_{\mathcal{T},\varphi,3}$ 中获胜，这表明 $\mathcal{T} \models \varphi$（参见定理 5.5）。我们在图 8b 中描绘了 $G_{\mathcal{T},\varphi,3}$ 的一个获胜策略。每当博弈处于 $\cup$-阶段时，变量 a 的值在两个停顿指针处都不同。为确保这一点，两个停顿最多可以在 3 个位置上分叉（例如在顶点 v_7 处）。△

引理 *5.6.* 对于任意 T φ，如果 $wins_{\mathcal{G}_{\mathcal{T},\varphi,Z}}(\mathfrak{V}, V_{init})$ 且 $Z' \geq Z$，那么 $wins_{\mathcal{G}_{\mathcal{T},\varphi,Z'}}(\mathfrak{V}, V_{init})$。对于任意 $Z \in \mathbb{N}_{\geq 1}$，存在 T φ，使得 $\neg wins_{\mathcal{G}_{\mathcal{T},\varphi,Z}}(\mathfrak{V}, V_{init})$，但 $wins_{\mathcal{G}_{\mathcal{T},\varphi,Z+1}}(\mathfrak{V}, V_{init})$。

证明。我们分别证明这两部分：

- 假设 σ 是验证者在 $G_{\mathcal{T},\varphi,Z}$ 中的获胜策略。我们可以很容易地将 σ 扩展为 $G_{\mathcal{T},\varphi,Z'}$ 中的获胜策略：我们只需忽略更大的窗口大小，即该策略只考虑大小为 Z 的较小窗口，并忽略超出 Z 的所有状态，以作出决策。
- 我们可以扩展例 C.1 中的思想。对于每个 $Z \geq 1$，定义 $\mathrm{T}_Z := (S, s_0, \kappa, \ell)$，其中 $S = \{s_0, \ldots, s_Z\}$，$\kappa(s_i) := \{s_{(i+1)\%(Z+1)}\}$，以及 $\ell(s_1) = \cdots = \ell(s_{Z-1}) := [a \mapsto 1], \ell(s_0) := [a \mapsto 0]$。注意，$\mathrm{T}_Z$ 生成唯一轨迹 $(\{a \mapsto 0\}\{a \mapsto 1\}^Z)^\omega$。例如，T2 如图 8a 所示。考虑公式 $\varphi := \exists \pi. \exists \beta_1 \blacktriangleright \pi. \exists \beta_2 \blacktriangleright \pi. (a_{\beta_1} \neq a_{\beta_2})$。容易看出 $\mathrm{T}_Z \models \varphi$。然而，与例 C.1 类似，验证者在 $\mathrm{GT}_{Z,\varphi,Z}(\neg \mathrm{winsGT}_{Z,\varphi,Z}(\mathfrak{V}, V\mathrm{init}))$ 中并不获胜，但在 $\mathrm{GT}_{Z,\varphi,Z+1}(\mathrm{winsGT}_{Z,\varphi,Z+1}(\mathfrak{V}, V\mathrm{init}))$ 中获胜，正如所要求的那样。

□

D 无交替公式的完备性

定理 6.1。设 $\mathcal{T}$ 是任意迁移系统，且 φ 是一个 $\exists^*$ 或 $\forall^*$ *A-HLTL* 公式，每条轨迹至多包含一次停顿。当且仅当 $\mathcal{T} \models \varphi$ 时，$wins_{\mathcal{G}_{\mathcal{T},\varphi,1}}(\mathfrak{V}, V_{init})$。

证明。第一个方向可直接由定理 5.5. 得出。对于另一个方向，假设 $\mathcal{T} \models \varphi$。我们先考虑 φ 是一个 $\exists^*$ 公式的情况。由于 φ 每个轨迹变量至多包含一次停顿，因此不失一般性，我们可以假设它具有以下形式

$$\exists \pi_1 \ldots \pi_n. \exists \beta_1 \blacktriangleright \pi_1, \ldots, \beta_n \blacktriangleright \pi_n. \psi$$

也就是说，β_i 是轨迹 π_i 上唯一的停顿。由于 $\mathrm{T} \models \varphi$，我们得到见证轨迹 $t_1, \ldots, t_n \in Traces(\mathrm{T})$，它们对应于 $\pi_1, \ldots, \pi_n$，由路径 $p_1, \ldots, p_n \in Paths(\mathrm{T})$ 生成（即 $t_i = \ell_{\mathcal{T}}(p_i)$）。此外，根据语义，存在停顿状态序列 $p'_1, \ldots, p'_n \in S^\omega_{\mathcal{T}}$，使得对于所有 $1 \leq i \leq n$，$p'_i \blacktriangleright p_i$ 以及

$$[\pi_1 \mapsto t_1, \ldots, \pi_n \mapsto t_n], \left[\beta_1 \mapsto \ell_{\mathcal{T}}(p'_1), \ldots, \beta_n \mapsto \ell_{\mathcal{T}}(p'_n)\right], 0 \models_{\mathcal{T}} \psi$$

验证者现在可以轻松赢得 $G_{\mathcal{T},\varphi,1}$：只需根据 $p_1, \ldots, p_n$ 追加状态，并进行停顿，使得第 i 次访问 U阶段时的当前状态为 $p'_1(i) \ldots, p'_n(i)$。这与定理 5.5 的证明中使用的策略类似。请注意，反驳者对 $G_{\mathcal{T},\varphi,1}$ 没有影响，也就是说，每个由 R 控制的顶点都只有一个后继顶点。由于 $p'_1, \ldots, p'_n$ 是 $p_1, \ldots, p_n$ 的公平停顿，验证者会无限多次地推进 $\beta_1, \ldots, \beta_n$。由于 $p'_1, \ldots, p'_n$ 满足 ψ，因此验证者赢得由此产生的博弈，故按要求赢得 $\mathcal{G}_{\mathcal{T},\varphi,1}(\mathfrak{V}, V_{init})$。

φ 是 $\forall^*$ 公式的情况类似。为此，

$$\varphi = \forall \pi_1 \ldots \pi_n. \forall \beta_1 \blacktriangleright \pi_1, \ldots, \beta_n \blacktriangleright \pi_n. \psi$$

在此，验证者在 $\mathcal{G}_{\mathcal{T}、\varphi、1}$ 中别无选择，即由 $\mathfrak{V}$ 控制的每个顶点都有唯一的后继顶点。然而，反驳者的转移规则（参见图 4）只允许他在 p_1、……、p_n 中使用某种路径组合 p_1、……、p_n，并让这些路径在 $\mathcal{T}$ 中停顿。构造 ψ_{mod}（参见第 5.1节）确保：如果停顿不公平（即某条路径永远停顿），验证者就会获胜。如果停顿公平，则所有路径组合都满足 ψ，因为我们假设 $\mathcal{T} \models \varphi$。 □

E 终止系统的完备性

E.1 定理 6.2 的证明

定理 6.2。设 $\mathcal{T}$ 是一个深度为 D 的终止转移系统，且 φ 是一个 $\forall^*\exists^*$ *A-HLTL* 公式。当且仅当 $\mathcal{T} \models \varphi$ 时，$wins_{\mathcal{G}_{\mathcal{T},\ \varphi,\ D}}(\mathfrak{V}, V_{init})$。

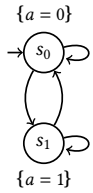


图 9

证明。“仅当”方向直接由定理 5.5 得出。对于另一方向，假设 $\mathrm{T} \models \varphi$ 且令 $\varphi = \forall \pi_1 \dots \pi_n. \exists \pi_{n+1} \dots \pi_{n+m}. \exists \beta_1 \blacktriangleright \pi_{l_1} \dots \beta_k \blacktriangleright \pi_{l_k}. \psi$。我们概述验证者在 $\mathrm{G}_{\mathcal{T},\varphi,D}$ 中的一个策略，证明 $\mathrm{wins}_{\mathcal{G}_{\mathcal{T},\varphi,D}}(\mathfrak{V}, V_{init})$。$\mathrm{G}_{\mathcal{T},\varphi,D}$ 中的每个初始顶点都将（全称量化的）轨迹 $\pi_1, \dots\dots, \pi_n$ 分配给一个长度为 D 的状态窗口，即 T 中 D 个连续状态的序列。令 $y_1, \dots\dots, y_n \in S_{\mathcal{T}}^{D}$ 为这些初始路径片段。根据 T 是深度为 D 的终止系统这一假设，每个前缀 y_i 已经定义了一条唯一的路径（即以汇点状态结束）。令 $p_1, \dots\dots, p_n \in \mathrm{Paths(T)}$ 为在 T 中通过扩展 $y_1, \dots\dots, y_n$ 得到的唯一路径。现在，将全称量化的轨迹变量 $\pi_1, \dots\dots, \pi_n$ 在 φ 中实例化为轨迹 $\ell_{\mathcal{T}}(p_1) \dots\dots, \ell_{\mathcal{T}}(p_n)$。由于 $\mathrm{T} \models \varphi$，我们因此得到对应于存在量化轨迹变量 $\pi_{n+1}, \dots\dots, \pi_{n+m}$ 的见证轨迹 $t_{n+1}, \dots\dots, t_{n+m} \in \mathrm{Traces(T)}$，以及相应的路径 $p_{n+1}, \dots\dots, p_{n+m} \in \mathrm{Paths(T)}$（即，对于 $n < i \leq n+m$，$\ell_{\mathcal{T}}(p_i) = t_i$）。此外，对于每个 $\beta_i \in \{\beta_1, \dots\dots, \beta_k\}$，存在路径 p_{l_i} 的一个停顿 $r_{\beta_i} \in S_{\mathcal{T}}^{\omega}$（即 $r_{\beta_i} \blacktriangleright p_{l_i}$），使得

$$\left[\pi_i \mapsto t_i\right]_{\pi_i \in \{\pi_1,\dots,\pi_{n+m}\}}, \left[\beta_i \mapsto \ell_{\mathcal{T}}(r_{\beta_i})\right]_{\beta_i \in \{\beta_1,\dots,\beta_k\}}, 0 \models_{\mathcal{T}} \psi.$$

验证者在 $\mathcal{G}_{\mathcal{T},\varphi,D}$ 现在只需根据 $\pi_{n+1}, \dots\dots, \pi_{n+m}$，按照 $p_{n+1}, \dots\dots, p_{n+m}$ 扩展状态窗口，并根据 β_i 进行停顿 r_{β_i}（类似于定理5.5证明中的反驳者）。注意，窗口大小为 D 即可满足要求，尽管我们在同一条轨迹上处理多个停顿：将一个停顿对推进 D 次后，状态将不再改变（因为至多经过 D 步就会到达汇点状态）。只要还有某个停顿指针尚未到达汇点状态，验证者就可以按照 $\{r_{\beta_i}\}$ 进行停顿，而不会让任何停顿指针严格大于 D。□

E.2 按窗口大小预知

考虑图 9 中的简单系统 T，它在变量 a 上生成轨迹。对于 $n \in \mathbb{N}$，考虑以下 HyperLTL 属性：

$$\varphi_n := \forall \pi_1. \exists \pi_2. \exists \beta_1 \blacktriangleright \pi_1. \exists \beta_2 \blacktriangleright \pi_2. \bigcirc\Box((a_{\beta_2} = 0) \leftrightarrow \bigcirc^n(a_{\beta_1} = 0))$$

其中，我们用 ^{n} 表示 $.$ 的 n 重迭代。该公式要求，对于每条轨迹 π_1，都存在一条轨迹 π_2 “预测” π_1，即 a 在 π_2 上的第 i 步成立，当且仅当它在 π_1 上的第（$i+n$）步成立。容易看出，$\mathrm{T} \models \varphi_n$。然而，对于 $n > 1$，$\mathfrak{V} \not\models \mathrm{G}_{\mathcal{T},\varphi_n,1}$，即验证者失去了 $\mathrm{G}_{\mathcal{T},\varphi_n,1}$。原因很简单：在我们的博弈中，我们逐步构造 π_1，π_2，因此验证者必须为 π_2 窗口选择一个后继状态，而此时它只知道 π_1 的一个前缀。然而，为了确保 φ_n 成立，验证者需要知道反驳者将在未来 n 步选择哪个状态。

由于 φ_n 每个轨迹变量在每条轨迹中只包含一次停顿，因此似乎没有必要使用大于1的窗口大小。然而，事实证明，窗口大小还有一个额外的好处（除了允许同一条轨迹上出现多次停顿之外）：它赋予验证者预知能力。在 $\mathcal{G}_{\mathcal{T},\varphi,Z}$ 中，我们始终为全称量化轨迹维持一个大小为 Z 的窗口（参见图6）。因此，验证者看到的不仅是状态窗口中的前几个位置，还能看到

接下来的 Z 步。对于属性 φ_n，了解接下来的 $Z = n$ 步恰好是此前缺失的信息：很容易看出，验证者在 $G_{\mathcal{T},\varphi_n,n}$ 中获胜，也就是说，了解接下来的 n 步提供了足够的信息来预测 π_1。

F 可容许公式的完备性

在本节中，我们证明定理 6.4：

定理 6.4。设 $\mathcal{T}$ 为任意迁移系统，且 φ 为一个可容许的 *A-HLTL* 公式。当且仅当 $\mathcal{T} \models \varphi$ 时，$\mathfrak{V}$ 在 $(\mathfrak{V}, V_{init})$ 中获胜。$_{\mathcal{T},\varphi,1}$。

我们假设 $\varphi = \forall \pi_1 \ldots \forall \pi_n. \exists \beta_1 \blacktriangleright \pi_1 \ldots \exists \beta_n \blacktriangleright \pi_n. \psi$ 在第6.3节中，我们已经给出了一个粗略的概述，并勾勒出在 $G_{\mathcal{T},\varphi,1}$ 中获胜的策略。我们回顾第6.3节中的一些定义：$G_{\mathcal{T},\varphi,1}$ 中由 $\mathfrak{V}$ 控制的每个顶点 v 都具有如下形式

$$v = \left\langle ⓔ, \left[\pi_i \mapsto [s_i, s_i']\right]_{i=1}^{n}, \left[\beta_i \mapsto 0\right]_{i=1}^{n}, \emptyset, q \right\rangle,$$

也就是说，每条轨迹 π_i 都映射到一个长度为 2 的状态窗口 $[s_i, s_i']$，并将 β_i-停顿点映射到该窗口中的第 0 个位置。我们假设 v 满足 ψ_{phase} 中的所有着色约束，即，对于所有 $i < j$ 和 $a \in P_{i,j}$，$\ell_{\mathcal{T}}(s_i)(a) = \ell_{\mathcal{T}}(s_j)(a)$（若不满足这一条件，则 ψ_{phase} 已被违反）。将此假设称为 (A)。对于 $M \subseteq \{1, \ldots, n\}$，我们定义状态 $\text{next}_v^M(1) \ldots, \text{next}_v^M(n) \in S_{\mathcal{T}}$：若 $i \notin M$，则 $\text{next}_v^M(i) = s_i$；若 $i \in M$，则 $\text{next}_v^M(i) = s_i'$。若对每个 $i < j$ 以及每个 $a \in P_{i,j}$，$\ell_{\mathcal{T}}(\text{next}_v^M(i))(a) = \ell_{\mathcal{T}}(\text{next}_v^M(j))(a)$，则推进集 M 是安全的；也就是说，M 确保下一步中局部满足所有着色约束。

引理 6.5. 若 M_1 和 M_2 是安全进展集，则 $M_1 \cup M_2$ 是安全进展集。

证明. 为了证明 $M_1 \cup M_2$ 是安全的，任取 $i < j$。我们声称，对于每个 $a \in P_{i,j}, \ell_{\mathcal{T}}(next_v^{M_1 \cup M_2}(i))(a) = \ell_{\mathcal{T}}(next_v^{M_1 \cup M_2}(j))(a)$，也就是说，$next_v^{M_1 \cup M_2}(i)$ 和 $next_v^{M_1 \cup M_2}(j)$ 具有相同的 $P_{i,j}$- 颜色。我们对 $(M_1 \cup M_2) \cap \{i, j\}$ 进行分类讨论：

- 若 $(M_1 \cup M_2) \cap \{i, j\} = \emptyset$：也就是说，在 $M_1 \cup M_2$ 中两个副本都没有推进，因此 $next_v^{M_1 \cup M_2}(i) = s_i$ 且 $next_v^{M_1 \cup M_2}(j) = s_j$，结论直接由假设 **(A)** 得出。
- 如果（$(M_1 \cup M_2) \cap \{i, j\} = \{i\}$（或者，类似地，（$(M_1 \cup M_2) \cap \{i, j\} = \{j\}$）：这意味着 $\text{next}_v^{M_1 \cup M_2}(i) = s_i'$ 且 $\text{next}_v^{M_1 \cup M_2}(j) = s_j$。现在，（$(M_1 \cup M_2) \cap \{i, j\} = \{i\}$ 意味着 $M_1 \cap \{i, j\} = \{i\}$ 或 $M_2 \cap \{i, j\} = \{i\}$。不失一般性，我们假设前者成立，因此 $\text{next}_v^{M_1 \cup M_2}(i) = s_i' = \text{next}_v^{M_1}(i)$ 且 $\text{next}_v^{M_1 \cup M_2}(j) = s_j = \text{next}_v^{M_1}(j)$。根据 M_1 是安全进展集这一假设，我们得到 s_i' 与 s_j 具有相同的 $P_{i,j}$-颜色。

- 若 $(M_1 \cup M_2) \cap \{i, j\} = \{i, j\}$：这可能出现在两种情况下（不包括对称情况）：– 若 $M_1 \cap \{i, j\} = \{i, j\}$（或类似地，$M_2 \cap \{i, j\} = \{i, j\}$）：根据 M_1 是安全的这一假设，可直接得出结论。– $M_1 \cap \{i, j\} = \{i\}$ 且 $M_2 \cap \{i, j\} = \{j\}$（或类似地，$M_1 \cap \{i, j\} = \{j\}$ 且 $M_2 \cap \{i, j\} = \{i\}$）：由于 M_1 是安全的，我们有 $\text{next}_v^{M_1}(i) = s_i'$ 与 $\text{next}_v^{M_1}(j) = s_j$ 具有相同的 $P_{i,j}$-颜色。同样，由于 M_2 是安全的，我们有 $\text{next}_v^{M_2}(i) = s_i$ 与 $\text{next}_v^{M_2}(j) = s_j'$ 具有相同的 $P_{i,j}$-颜色。此外，根据 (A)，s_i 与 s_j 具有相同的 $P_{i,j}$-颜色。综上可知，$\text{next}_v^{M_1 \cup M_2}(i) = s_i'$ 与 $\text{next}_v^{M_1 \cup M_2}(j) = s_j'$ 具有相同的 $P_{i,j}$-颜色，正如所需。□

现在，我们可以正式定义验证者的获胜策略。为简单起见，我们概述一种非位置策略，即不仅取决于当前顶点，还取决于此前有限博弈的策略。由于 Büchi（以及奇偶）博弈具有位置确定性 [Martin 1975]，，这已经意味着存在一种位置策略（参见第 3节）。

在 $G_{\mathcal{T}}$ 的 φ_1 初始顶点处，验证者可以观察所有轨迹的初始状态窗口 $\pi_1, \ldots\ldots, \pi_n$。这足以对所有状态公式进行求值 ψ（因为它们只取决于第一个状态）。基于这一求值，验证者可以判断要满足 ψ 是否必须满足 ψ_{phase}。

- 若情况并非如此，即 ψ 仅凭状态公式就已成立，验证者便会反复推进所有停顿 $\beta_1, \ldots, \beta_n$。这样便能确保某种公平停顿，而满足 $\psi_{\text{阶段}}$ 无论如何都不是必要的。

- 若不是这样，也就是说，验证者需要满足 $\psi_{\text{阶段}}$ 才能满足 ψ，那么它需要通过推进来使颜色序列对齐。正如第 6.3 节所述，我们通过引理 6.5 来实现这一点，并始终尽可能多地推进轨迹。对于每个顶点 v，我们将 $M_{max v}$ 定义为顶点 v 中唯一的最大安全推进集（定义为所有安全推进集的并集）；根据引理 6.5，该集合是安全的。对于每个顶点 v，验证者现在推进 $M_{max v}$ 中的所有轨迹，即我们定义

$$\sigma^{max}(v) := \left\langle \text{Ⓤ}, \left[\pi_i \mapsto [s_i, s_i']\right]_{i=1}^{n}, \left[\beta_i \mapsto ite(i \in M_v^{max}, 1, 0)\right]_{i=1}^{n}, \{\beta_i \mid i \in M_v^{max}\}, q \right\rangle.$$

  也就是说，验证者总是（局部地）尽可能多地推进停顿，同时确保 ψ_{phase} 中的着色约束在下一步成立。

令 σ^{max} 为验证者所采用的策略。不难看出，σ^{max} 赢得 $G_{\mathcal{T},\varphi,1}$：策略 σ^{max} 总是尽可能多地推进停顿，以满足 ψ_{phase} 中的着色约束。对于任意满足 $\exists \beta_1 \blacktriangleright \pi_1, \ldots, \beta_n \blacktriangleright \pi_n . \psi_{phase}$ 的路径 $p_1, \ldots, p_n$（其中 $\pi_1, \ldots, \pi_n$），策略 σ^{max} 会自动将它们对齐，使得 ψ_{phase} 成立。也就是说，每条轨迹都会被尽可能长时间地推进；如果在不违反 ψ_{phase} 的情况下无法继续推进，它就会自动“等待”，直到能够与其他轨迹一起改变其颜色。因此，我们得到：

引理 F.1。遵循策略 σ^{max}，验证者赢得 $\mathcal{G}_{\mathcal{T},\varphi,1}$。

接收日期：2025-03-25；接受日期：2025-08-12