超节点自动机

Ezio Bartocci 维也纳工业大学，
奥地利维也纳
Thomas A. Henzinger 奥
地利科学技术研究院，奥地利克洛斯特新堡
DejanNickovic 奥地利技术研究院（AIT），
奥地利维也纳
Ana Oliveira da Costa 维也纳工业
大学，奥地利维也纳

摘要

我们提出超节点自动机，将其作为一种新的规约形式，用于描述并发系统的超属性。超节点自动机是有限自动机，其节点标记有超节点逻辑 公式，转移则标记有动作。超节点逻辑公式用于描述不同系统执行中变量值序列之间的关系。与 HyperLTL 不同，超节点逻辑以异步 视角看待执行轨迹：它约束每个变量的取值及其变化顺序，但不关联这些变化发生的时间。不同执行轨迹仅通过超节点自动机的转移进行同步。超节点自动机自然地将节点层面的异步性与转移层面的同步性结合起来。我们证明，在动作标记的 Kripke 结构上，超节点自动机的模型检测问题是可判定的；这些结构中的动作会触发规约自动机的转移。因此，超节点自动机适合用于规约和验证异步超属性，例如多线程程序中通过去分类实现的观察确定性。

2012 年 ACM 学科分类安全与隐私 → 逻辑与验证

关键词与短语超属性、异步、自动机、逻辑

数字对象标识符10.4230/LIPIcs...

arXiv:2305.02836v2 [cs.FL] 8 Jan 2024

1 引言

线性时序逻辑（LTL）或自动机等形式化方法常用于规约和验证并发系统的轨迹属性。信息流策略等安全需求需要同时对多条执行轨迹进行推理；因此，它们无法表示为轨迹属性。超属性通过规约轨迹集合的属性来克服这一局限 [10]。HyperLTL [9], 是对带有轨迹量词的 LTL 的扩展，已成为规约和验证一类重要超属性的常用形式化方法。HyperLTL 及相关超逻辑的时序运算符会在所有绑定到某个轨迹变量的轨迹上同步推进；它们规约的是同步超属性。因此，例如，如果系统模式转移在不同系统执行中的发生时间可能不同，HyperLTL 就无法规约一种会随系统模式变化的信息流策略 [3]。近年来， [11, 8, 4] 等人多次独立观察到这一局限，并提出了异步版本的超逻辑来解决这一问题。

我们采用不同的思路，提出一种用于描述超属性的规范语言，称为超节点自动机，它通过结合自动机与逻辑，将同步性和异步性融为一体。超节点自动机 是有限自动机，其节点标记为来自一种完全异步、非时序超逻辑的公式，这种超逻辑称为超节点逻辑；其转移则标记为用于同步不同执行轨迹的动作。此前，基于自动机的语言已用于描述同步超属性 [6], 超节点自动机是

© Ezio Bartocci 和 Thomas A. Henzinger 和 Dejan Nickovic 和 Ana Oliveira da Costa ;
依据知识共享署名 4.0 国际许可协议授权。莱布尼茨国际信息学会议论文集，达格施图尔城堡—莱布尼茨信息学中心，达格施图尔出版社，德国。
LIPICS

XX：2 超节点自动机

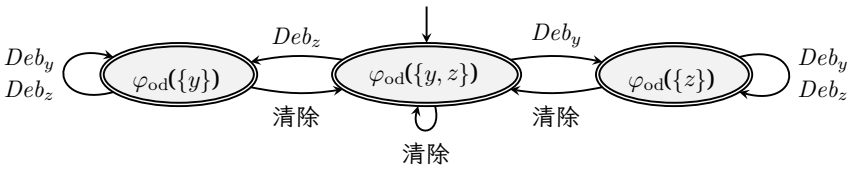
第一种在超属性规约中系统地区分轨迹同步性与轨迹异步性的语言：在超节点内（即超节点自动机的状态中），不同执行轨迹以各自独立的速度推进，只有在转移到下一个超节点时才会“彼此等待”。这种区分带来了自然的规约，也发挥了基于自动机和基于逻辑的形式体系各自的优势。

超节点的规约对有限轨迹片段采用最大程度的异步视角：每个程序变量都可以独立推进。我们引入超节点逻辑来规约此类异步超属性。超节点逻辑包括对有限轨迹的量化，以及二元关系 $x(\pi) \precsim y(\pi')$，其中轨迹变量为 π 和 π'，系统（或程序）变量为 x 和 y。该关系断言：在分配给 π 的轨迹中，程序变量 x 经历的有序值变化与变量 y 在 π' 中经历的变化相同，但这些变化可能发生在不同时间（停顿），并且 y 在 π' 中还可能发生额外变化（前缀化）。有限轨迹之间的无停顿前缀化关系 $\precsim$ 是超节点逻辑唯一的非逻辑算子，由此产生了一种新颖、优雅且强大的方法，用于规约有限轨迹上的异步超属性。

对于每个有限或无限动作序列，超节点自动机都会规约一个来自超节点逻辑的相应公式序列。本文的主要贡献是针对超节点自动机的模型检测算法。我们的算法检查给定的超节点自动机是否接受由动作标记的克里普克结构所定义的（可能是）无限轨迹集合。克里普克结构（即“模型”）中的转移上的动作标签，会在超节点自动机（即“规约”）中诱导出带有相同标签的转移。用于对超节点逻辑公式进行模型检测的子程序在技术上具有创新性：它在一种称为无停顿自动机的新概念上引入了自动机理论构造，并将其用于过滤和自组合等熟悉的逻辑语境中。尽管超节点逻辑具有存在轨迹量词，因而能够规约变量独立性等非安全超属性 [3]，但本文着重研究安全超节点自动机，以规约无限执行的超属性。据我们所知，这是针对一种简单但通用的形式体系所得到的首个可判定性结果，该形式体系能够规约重要的异步超属性。

或许最著名的异步超属性是观察确定性 [16]。在第 2 节中，我们通过对 [16] 所定义的观察确定性进行形式化规约，来阐明超节点逻辑的动机。我们还通过规约一种信息流策略，进一步阐明超节点自动机的动机；该策略要求并发程序的不同模式之间能够动态改变观察确定性。此类动态变化的一个例子是信息的去密级化，它可以用超节点之间的转移来表示。第 3 节定义超节点逻辑和超节点自动机。我们为文献 [12, 15] 中讨论过的若干不同版本的观察确定性提供形式化规约，以展示超节点逻辑强大的表达能力。第 4 节中，我们首先使用经停顿约简的自动机，解决 Kripke 结构上的超节点逻辑模型检测问题；随后解决动作标记的 Kripke 结构上的一般性超节点自动机模型检测问题。与此前提出的异步超属性规约形式体系 [11, 8, 4]，不同，后者通常不可判定，仅在特定片段上可判定，而我们的模型检测算法的复杂度对于 Kripke 结构中的变量数呈双指数增长。最后，在第 5 节中，我们论证了我们的形式体系与这些其他形式体系在表达能力上互不包含。因此，超节点自动机是规约异步超属性的一种很有前景的形式体系，也为安全属性的自动验证作出了重要贡献。

图 1 超节点自动机$\mathcal{H}$，用于说明安全信息的互斥去密级化，其中 $\varphi_{\text{od}}(L)$ 定义为 $= \forall\pi\forall\pi' \bigwedge_{l\in L}(l(\pi) \precsim l(\pi') \vee l(\pi') \precsim l(\pi))$。



贡献概述

我们提出了一种新的超属性规约形式体系，包含无状态规约（超节点逻辑）和有状态规约（超节点自动机），它具有以下特点：(1) 可用于规约具有代表性的异步信息流属性，例如观察确定性（无状态）和信息脱密（有状态）；(2) 在表达能力上与时序超逻辑彼此正交；(3) 其验证问题是可判定的，并可通过一种采用无停顿自动机的新型自动机理论方法求解。尽管超节点逻辑具有存在轨迹量词，因此可以规约变量独立性等非安全超属性 [3]，本文将重点讨论安全超节点自动机，以规约无限执行的超属性。我们预计，加入接受超节点（例如超节点 Büchi 自动机）将十分直接，但这留待未来工作。

2 动机示例

Zdancewic 和 Myers 的开创性工作 [16] 提出了观察确定性的概念，用于规约并发程序的信息流策略。观察确定性是一种非干扰属性，要求公开可见值不依赖于秘密信息。多线程程序的非干扰规约尤其具有挑战性，因为（i）多线程程序的执行取决于调度策略；（ii）从一种程序状态转变为另一种状态，在程序的每次执行中可能发生在不同时间。根据 [16]，，如果从任意两个低等价状态出发，那么任意两个执行轨迹中每个低变量的轨迹都在停顿 和前缀化意义下等价，则该程序具有观察确定性。Huisman、Worah 和 Sunesen [12]以及 Terauchi [15] 提出了观察确定性定义的若干变体，同样基于停顿与前缀扩展等价。该定义以异步方式看待执行轨迹，因此 HyperLTL 不足以对其进行规约 [3]。

在本节中，我们使用超节点逻辑来描述*Zdancewic-Myers* 观察确定性，并借助超节点自动机定义互斥解除保密策略。该解除保密策略不仅涉及观察确定性的异步要求，还涉及不同观察确定性要求之间的动态变化。具体而言，该策略要求在正常 运行期间，两个公开可见的变量 y 和 z 不得泄露秘密信息。该策略还允许两种调试 模式，在这两种模式下，y 或 z 可以泄露信息，但绝不能同时泄露。“模式操作”的灵感来自编程语言社区关于解除保密策略的示例（参见 [2, 14]）。

去密级化策略的超节点自动机规约$\mathcal{H}$ 如图 1 所示。超节点自动机定义在一组动作标记轨迹上，这些轨迹由程序变量的赋值和动作序列组成。自动机节点之间的转移以动作标记，表示程序何时改变其运行模式

XX:4 超节点自动机

算法 1 程序$\mathcal{P}_{\text{var}}$

```
1 do
2     变量 := 0;
3     <escape>read(x);</escape>
4     读取(status);
5     如果 (status = Deb_var) 则
6         var := x;
7     end
8     输出(var);
9 当true;
```

表1 $\mathcal{P}_y \parallel \mathcal{P}_z$的执行。

τ_1	x:	0	0	0	
	y:	0	0	0	
	z:	0	0	0	
	状态:	ε	Deb_y	Deb_z	
τ_2	x:	1	1	1	1
	y:	0	0	1	1
	z:	0	0	0	1
	状态:	ε	ε	Deb_y	Deb_z

，例如从正常模式切换到两种调试模式之一。请注意，规范自动机是完备且确定的：任意动作序列都能唯一确定自动机的一次运行。在此示例中，从正常模式转移到两种调试模式的转移分别标记为Deb_y 和Deb_z 动作；从任一种调试模式转移到正常模式的转移则标记为清除。

自动机节点标注有超节点逻辑的公式。在我们的示例中，三个公式都规定了 Zdancewic-Myers 观察确定性，但针对的是不同的程序变量。观察确定性要求，对于由 $\forall\pi\forall\pi'$ 指定的任意两次程序执行，它们在每个公开可见程序变量（l 公共变量集合 L 中的变量）上的投影，都必须在停顿和前缀化意义下等价（由 $l(\pi) \precsim l(\pi') \vee l(\pi') \precsim l(\pi)$ 指定）。可见程序变量会随模式变化，因此不同的规约节点使用同一公式的不同实例进行标注。例如，带有公式 $\varphi_{\text{od}}(\{y\})$ 的超节点仅要求对 y满足观察确定性。

算法 1 定义了一个反应式程序 $\mathcal{P}_{\text{var}}$ （其中 var 可以是 y 或 z），该程序在每次迭代中读取输入变量 x 和动作状态。例如，如果动作是 Deb_y，则变量 y 用于调试，程序会将 x 的内容复制到 y。

并行组合 $\mathcal{P}_y \parallel \mathcal{P}_z$ 不满足规约 $\mathcal{H}$。例如，考虑表1中所示的轨迹集合 $T = \{\tau_1, \tau_2\}$。我们对 τ_1 和 τ_2有两个重要观察：(1) 这些轨迹长度不同；(2) 它们呈现了发生在不同 时刻的相同 动作序列（Deb_y 后接Deb_z）（因此这些轨迹是异步的）。我们注意到，上述动作序列将每条轨迹划分为三个轨迹片段组成的序列，称为切片；在表1中，我们分别用白色、浅灰色和深灰色表示这些切片。我们将每个切片映射到超节点自动机中的一个唯一节点。首先可以看到，这两条从左向右读取的轨迹长度不同；它们是异步的，因为相同的动作序列发生在不同的时刻。我们还观察到，每个动作序列都会将轨迹集合划分为一系列轨迹集合片段，称为切片。每个切片都映射到超节点自动机中的一个唯一节点。T的白色切片映射到$\mathcal{H}$的初始节点。T 的浅灰色切片映射到从初始状态经由动作Deb_y 可到达的节点（即 y的调试模式），该节点由公式 $\varphi_{\text{od}}(\{z\})$ 标记。T的深灰色切片映射到同一个节点，因为动作 Deb_z 触发了自环转移。

动作序列定义了超节点自动机中的一条路径。然后，具有动作序列 p 的轨迹集合 T 满足超节点自动机 $\mathcal{H}$，当且仅当由 p 诱导的 T 的切片满足 $\mathcal{H}$ 中由 p 定义的路径上的超节点公式。我们的示例中并非如此，因为 T 的深灰色切片违反了与之关联的超节点公式 $\varphi_{\text{od}}(\{z\})$。更具体地说，程序变量 z 在轨迹 τ_1的深灰色片段中取值为 0，而在 τ_2 的深灰色片段中取值为 1。该规约

违反规约的原因是临界区（算法 1 中的第 $5-7$ 行）未受保护。动作 Deb_z 可能会在 $\mathcal{P}_z$ 中的 Deb_y 之后发生（第 4 行）。此后，输入值 x 被复制到 z （$\mathcal{P}_z$ 中的第 6 行），并且 y 和 z 都变为可观察（$\mathcal{P}_y$ 和 $\mathcal{P}_z$ 中的第 8 行）。因此，它们都会泄露有关 x 的信息，从而违反规约。可以通过引入适当的锁机制来保护临界区，确保在两次观察之间，x 不会同时被复制到 y 和 z，从而解决此问题。我们的模型检测算法使我们能够将本例中用于发现错误的推理过程完全自动化。

3 超节点自动机

本节中，我们定义超节点逻辑和超节点自动机。我们将程序执行表示为有限或无限序列，每个序列由有限轨迹片段及同步动作组成。设 X 为有限域上的有限个程序变量，Σ、A 为有限个动作，并且 $A_\varepsilon = A \cup \{\varepsilon\}$。

超节点逻辑是在有限轨迹片段上进行解释的。轨迹片段 τ 是赋值在 Σ^X中的有限序列，其中每个赋值 $v\colon X \to \Sigma$ 都将程序变量映射到域值。我们用 $(\Sigma^X)^*$表示 X 和 Σ 上的轨迹片段集合。片段性质 T 是轨迹片段的集合，即 $T \subseteq (\Sigma^X)^*$。超节点逻辑的公式描述轨迹片段集合的性质，称为片段超性质。形式上，片段超性质$\mathbf{T}$ 是片段性质的集合，即$\mathbf{T} \subseteq 2^{(\Sigma^X)^*}$。

超节点自动机是在有限和无限动作标记轨迹上进行解释的。一个动作标记轨迹 ρ 是一个有限或无限的有序对序列，每个有序对都由一个赋值和一个来自 A 的动作标签或空标签 ε 组成；即，$\rho \in (\Sigma^X \times A_\varepsilon)^*$ 或 $\rho \in (\Sigma^X \times A_\varepsilon)^\omega$。为简化技术处理，我们要求，对于无限动作标记轨迹，非空标签出现无穷多次。一个动作标记轨迹性质 是一个动作标记轨迹集合。超节点自动机接受动作标记轨迹性质，因此定义了一种动作标记轨迹超性质，即它所接受的所有动作标记轨迹性质的集合。

3.1 超节点逻辑

超节点逻辑 FO$[\precsim]$ 是一种一阶形式体系，用于描述程序变量的值在一组轨迹片段中的变化之间的关系。超节点逻辑的公式由以下文法定义：$\varphi ::= \exists \pi\, \varphi \mid \neg\varphi \mid \varphi \wedge \varphi \mid x(\pi) \precsim x(\pi)$，其中，一阶变量 π 的取值范围是$\mathcal{V}$，即轨迹变量的集合；一元函数符号 x 的取值范围是 X，即程序变量的集合。超节点逻辑仅通过二元经停顿约减的前缀化谓词 $\precsim$ 来表示时间。原子公式$x(\pi) \precsim y(\pi')$ 的预期含义是：x 在分配给 π 的轨迹片段中经历的有序值变化，与变量 y 在 π' 中经历的相同，之后 y 在 π' 中可能还会发生额外的值变化。换言之，超节点逻辑采用完全异步比较来比较不同的轨迹片段，其中所有变量都被分别考虑。

因此，我们在未压缩的轨迹片段上解释超节点公式，这些轨迹片段分别编码每个程序变量的演变。未压缩轨迹片段 $\tau\colon X \to \Sigma^*$是一个从程序变量映射到有限值串的函数。超节点逻辑的公式是在将轨迹变量赋值为未压缩轨迹片段的赋值下解释的。给定一组 $T \subseteq (\Sigma^*)^X$ 未压缩轨迹片段，赋值 $\Pi_T\colon \mathcal{V} \to T$将每个轨迹变量映射到 T中的一个未压缩轨迹片段。我们用 $\Pi_T[\pi \mapsto \tau]$ 表示对 Π_T的更新，其中 π 被赋值为 τ。在赋值 Π_T 下，超节点逻辑公式 φ 的满足关系按归纳方式定义如下：

$\Pi_T \models \exists\pi\varphi$ 当且仅当存在 $\tau \in T$: $\Pi_T[\pi \mapsto \tau] \models \varphi$;$\Pi_T \models \psi_1 \wedge \psi_2$ 当且仅当 $\Pi_T \models \psi_1$ 且 $\Pi_T \models \psi_2$; $\Pi_T \models \neg\psi_1$ 当且仅当 $\Pi_T \not\models \psi_1$;$\Pi_T \models x(\pi) \precsim y(\pi')$ 当且仅当 $\Pi_T(\pi)(x) \in \sigma_0^+ \dots \sigma_n^+$ 且 $\Pi_T(\pi')(y) \in \sigma_0^+ \dots \sigma_n^+\Sigma^*$，其中 $\sigma_i \neq \sigma_{i+1}$，对于 $0 \leq i < n$.

一组 T 未压缩轨迹片段是公式 φ 的模型，记作 $T \models \varphi$，当且仅当存在一个赋值 Π_T，使得 $\Pi_T \models \varphi$。我们采用通常的缩写 $\forall\pi\varphi \stackrel{\text{def}}{=} \neg\exists\pi\neg\varphi$ 和 $\varphi \vee \varphi' \stackrel{\text{def}}{=} \neg\,(\neg\varphi \wedge \neg\varphi')$。从现在起，除非另有说明，程序变量和轨迹变量均用自然数作为下标，即 $X = \{x_1, \dots, x_m\}$ 和 $\mathcal{V} = \{\pi_1, \dots, \pi_n\}$。

▶ **示例 1.** 我们通过对程序变量之间的四种不同非干扰变体进行形式化说明，来展示如何使用超节点逻辑。Zdancewic 和 Myers 在 [16] 中引入了观察确定性的第一个概念，用以刻画并发程序的非干扰。他们要求，在每次程序执行中，集合 L中的每个公开可见变量都必须在前缀化意义下停顿等价（即，其中一次执行可以有更多次取值变化）：

$$\forall\pi\forall\pi' \bigwedge_{l\in L} (l(\pi) \precsim l(\pi') \vee l(\pi') \precsim l(\pi)).$$

后来，Huisman、Worah 和 Sunesen [12] 通过要求每个公开可见变量在所有执行中都具有停顿等价性，强化了先前的定义：

$$\forall\pi\forall\pi' \bigwedge_{l\in L} l(\pi) \precsim l(\pi').$$

该公式中的全称轨迹量化迫使 $\precsim$ 具有对称性，因此两条轨迹对于每个可见的公共变量都必须具有相同的有序变化。

我们的第三种观察确定性变体来自 Terauchi[15], ，其要求所有公开可见变量的集合在前缀化意义下具有停顿等价性：

$$\forall\pi\forall\pi' \; (L(\pi) \precsim L(\pi') \vee L(\pi') \precsim L(\pi)).$$

请注意，由于我们将超节点公式解释在任意有限域上，因此可以将有限变量集合的值编码到单个变量中，该变量称为L 。

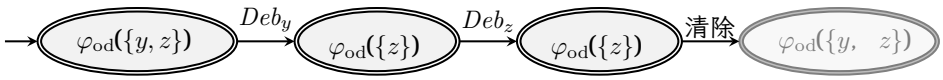
最后，我们将独立性（也称为广义无干扰 [10]）定义为 [3] 中所定义的形式。两个程序变量 x 和 y 是独立的，当且仅当：如果对于 x，某条轨迹 π中可能出现一系列值变化；对于 y，某条轨迹 π'中也可能出现一系列值变化，那么它们的组合（$x(\pi)$ $y(\pi')$）也可能出现在某条轨迹中。独立性公式规定，对于任意两条轨迹（π 和 π'），都存在第三条轨迹（$\pi_\exists$），该轨迹在停顿和前缀化意义下见证了组合（$x(\pi)$ $y(\pi')$）：

$$\forall\pi\forall\pi'\exists\pi_\exists \; (x(\pi) \precsim x(\pi_\exists) \wedge y(\pi') \precsim y(\pi_\exists)).$$ ◁

停顿约简轨迹片段

我们关注无停顿的未压缩轨迹片段，即在连续的时间点上不会重复相同变量值的轨迹片段。对于程序变量 x 和未压缩轨迹片段 τ，其中 $\tau(x) \in \sigma_0^+ \dots \sigma_n^+$，且 $\sigma_i \neq \sigma_{i+1}$对于 $i < n$，其 停顿约简为 $\lfloor\tau(x)\rfloor = \sigma_0 \dots \sigma_n$。我们自然地将这一概念扩展到通过$\lfloor\tau\rfloor(x) = \lfloor\tau(x)\rfloor$ 对 τ 进行停顿约简，其中所有程序变量 $x \in X$均适用；也扩展到通过 $\lfloor T\rfloor = \{\lfloor\tau\rfloor \mid \tau \in T\}$对一组未压缩轨迹片段 T 进行停顿约简。我们证明，超节点逻辑的公式无法区分一组未压缩轨迹片段 T 与其停顿约简 $\lfloor T\rfloor$。

图 2 超节点自动机中由动作序列 $Deb_y Deb_z$清除 定义的运行，见图 1。



▶ **命题 2**。 设 $T \subseteq (\Sigma^*)^X$ 为一组未压缩轨迹片段， φ 为一个超节点逻辑公式。那么，$T \models \varphi$ 当且仅当 $\lfloor T \rceil \models \varphi$。

3.2 超节点自动机

超节点自动机是有限自动机，其状态（称为超节点）带有超节点逻辑公式标记，转移带有动作标记。超节点自动机读取一组$R \subseteq (\Sigma^X \times A_\varepsilon)^\omega$动作标记轨迹，并接受其中的一些集合。

▶**定义 3**。 设动作集合为 A、程序变量集合为 X，则在这两个集合上定义的确定性有限超节点自动机（*HNA*）是一个元组 $H = (Q, \hat{}q, \gamma, \delta)$，其中 Q 是一个有限状态集合，$\hat{}q \in Q$ 为初始状态；状态标记函数 γ 为每个状态 Q指定一个关于程序变量 X 的超节点逻辑闭公式；转移函数δ： $Q \times A \to Q$ 是一个全函数，为每个状态和动作指定唯一的后继状态。

我们假设转移函数是全定义且确定的，这只是为了让技术性表述更简单。HNA$\mathcal{H}$ 的一次运行 是一个有限或无限序列$r = q_0 a_0\, q_1 a_1\, q_2 a_2 \ldots$，由超节点和动作交替组成，从初始超节点 $q_0 = \hat{}q$开始，并遵循转移函数，即对于所有 $i \geq 0$， $\delta(q_i,\ a_i) = q_{i+1}$。我们将相应的动作序列 $p = a_0 a_1 a_2 \ldots$称为 r的动作序列 。注意，每个动作序列都定义了$\mathcal{H}$的一次唯一运行。我们用$\mathcal{H}[p]$ 表示由动作序列 p定义的运行；图 2 展示了一个例子。

动作标记轨迹 $\rho = (v_0,\ a_0)(v_1,\ a_1)(v_2,\ a_2)\ldots$的动作序列，其中对于所有 $i \geq 0$， $v_i \in \Sigma^X$且 $a_i \in A_\varepsilon$，是将该轨迹投影到其动作后，并移除所有空标签 ε所得的结果；也就是说， $\rho[A] = a'_0 a'_1 \ldots$，其中 $a_0 a_1 \ldots \in a'_0 \varepsilon^* a'_1 \varepsilon^* \ldots$且对于所有 $i \geq 0$， $a'_i \in A$。给定一个动作标记轨迹集合 R，关于有限动作序列$p \in A^*$的投影 R 为$\{\rho \in R \mid \rho[A] = p\,p'$，其中某个后缀为 $p' \in A^* \cup A^\omega\}$。

超节点自动机运行中的每一步都会在一组动作标记轨迹上定义一个新的切片。设 $p = a_0 a_1 \ldots a_n$ 为有限动作序列， $\rho = (v_0, a'_0)(v_1, a'_1)\ldots$为一条以 p为前缀的动作标记轨迹，并设 R 为此类轨迹的集合。我们用 $\rho(\varnothing, a_0)$ 表示 ρ的初始轨迹片段，该片段以动作标签 a_0 结束。形式化地说， $\rho(\varnothing, a_0) = v_0 \ldots v_k$使得 $a'_k = a_0$，并且对所有 $0 \leq i < k$，都有 $a'_i = \varepsilon$。此外，我们用 $\rho(a_0 a_1 \ldots a_i, a_{i+1})$ 表示 ρ 后续的轨迹片段，这些片段在经历动作序列 $a_0 a_1 \ldots a_i$之后以动作标签 a_{i+1} 结束。归纳地，如果 $\rho(a_0 a_1 \ldots a_{i-1}, a_i) = v_k \ldots v_l$，对 $m > l$有 $a'_m = a_{i+1}$，并且对所有 $l < j < m$都有 $a'_j = \varepsilon$，那么 $\rho(a_0 a_1 \ldots a_i, a_{i+1}) = v_{l+1} \ldots v_m$。相应地，切片操作也扩展到动作标记轨迹集合；例如， $R(\varnothing, a) = \{\rho(\varnothing, a) \mid \rho \in R\}$。

由于超节点逻辑的公式是在未压缩轨迹片段上解释的，因此最后一步需要解压每个切片。对变量集合 $X = \{x_1, \ldots, x_m\}$ 中的轨迹片段进行解压 $\tau = v_0 \ldots v_n$，即 $\mathsf{unzip}(\tau) = \{x_0 \colon v_0(x_0)..v_n(x_0) \ldots,$ $x_m \colon v_0(x_m)..v_n(x_m)\}$。我们自然地将轨迹片段集合的解压定义为 $\mathsf{Unzip}(T) = \{\mathsf{unzip}(\tau) \mid \tau \in T\}$。

▶ **定义 4.** 设 $H = (Q, \hat{}q, \gamma, \delta)$ 是一个 *HNA*，且 R 为一组动作标记轨迹。令 p 为 A^*中的一个有限动作序列。集合 R 被 H 关于模式 p接受，记作$R \models_p H$，当且仅当对于运行 $H[p] = q_0 a_0\, q_1 a_1\, \ldots\, q_n a_n$，由 p 诱导的 R 的所有切片都是模型。

分别标记相应超节点的公式；也就是，解压$(R[p](\varnothing, a_0)) \models \gamma(q_0)$，以及解压$(R[p](a_0 \ldots a_{i-1}, a_i)) \models \gamma(q_i)$，对所有 $0 < i \leq n$ 均成立。

动作标记轨迹的集合 R 被 HNA$\mathcal{H}$ 接受，当且仅当对于所有有限动作序列$p \in A^*$，如果 $R[p] \neq \emptyset$，那么 $R \models_p \mathcal{H}$。由 $\mathcal{H}$ 接受的语言是所有被 $\mathcal{H}$ 接受的动作标记轨迹集合的集合，记作$\mathcal{L}(\mathcal{H})$。请注意，该定义假设 HNA 的所有有限和无限运行都是可行的；此类自动机通常称为安全自动机。也可以进行细化，例如要求有限运行必须以接受状态结束，或者要求无限运行必须无限次访问接受状态。

4 模型检测

我们针对转移标记有动作的克里普克结构，解决超节点自动机模型检测问题。换言之，我们给出一种算法，用于检查 HNA 是否接受由动作标记的 Kripke 结构生成的所有动作标记轨迹的集合。这一结果令人惊讶，因为对于许多允许规约异步超属性的形式化方法而言，验证问题是不可判定的。此外，我们的算法与此前发表的同步超性质模型检测算法有很大不同。

4.1 动作标记的 Kripke 结构

Kripke 结构是一个元组 $K = (W, \Sigma^X, \Delta, V)$，由有限的世界集 W、定义在有限域 Σ 上的变量集 X、转移关系 $\Delta \subseteq W \times W$以及赋值 $V : W \times X \to \Sigma$组成。赋值 V: $W \times X \to \Sigma$将有限域 Σ中的一个值分配给每个世界 [1]中的每个变量。给定一个具有转移关系 Δ的 Kripke 结构以及一个动作集 A，针对 K、定义在 A上的动作标记是一个函数 $\mathbb{A}$: $\Delta \to 2^{A_\varepsilon}$，它为每个转移分配一组动作标签（可能包括空标签 ε）。带初始状态的克里普克结构是一个 Kripke 结构，其中一个世界是初始世界，记作 (K, w_0)，其中 $w_0 \in W$。

带有动作标记 $\mathbb{A}$ 的 Kripke 结构 K 中的路径，是一个由世界和动作交替组成的有限或无限序列$w_0 a_0 w_1 a_1 w_2 a_2 \ldots$，并且对所有 $i \geq 0$，都符合转移关系 $(w_i, w_{i+1}) \in \Delta$和动作标记 $a_i \in \mathbb{A}(w_i, w_{i+1})$。我们用 Paths$(K, \mathbb{A})$ 表示所有此类路径的集合。路径 $\varrho = w_0 a_0 w_1 a_1 \ldots$定义了动作标记轨迹 zip$(\varrho) = V(w_0) a_0 V(w_1) a_1 \ldots$。我们用 Zip$(K, \mathbb{A})$ 表示由 Paths$(K, \mathbb{A})$ 中的路径定义的动作标记轨迹集合。Paths$(K, \mathbb{A}, w_0)$ 表示 Paths$(K, \mathbb{A})$ 中所有从世界 w_0出发的路径构成的集合。与前文一样，Zip$(K, \mathbb{A}, w_0)$ 指由 Paths$(K, \mathbb{A}, w_0)$ 中的路径定义的所有动作标记轨迹的集合。

现在，我们可以正式定义本文要解决的核心验证问题，即针对以带动作标记的带初始状态的克里普克结构表示的模型，进行以超节点自动机表示的规约的模型检测问题。将并发程序（例如第二节中的程序）转换为带动作标记的带初始状态的克里普克结构非常简单；出于篇幅考虑，此处省略其形式化过程。

[1] 这个克里普克结构的定义与通常基于命题变量的定义等价，因为我们可以轻松地将有限域上的变量集合编码为一组命题变量。

超节点自动机模型检测问题

设 (K, w_0) 是变量集合 X 上、定义于有限域Σ的带初始状态的克里普克结构，且 $\mathbb{A}$ 是 K 上取值于动作集合 A 的动作标记。设 $\mathcal{H}$ 是定义于相同变量集合 X 、域 Σ 和动作集合 A 上的超节点自动机。由 $(K, \mathbb{A}, w_0)$ 生成的动作标记轨迹集合是否被 $\mathcal{H}$ 接受；也就是说，$\mathrm{Zip}(K, \mathbb{A}, w_0) \in \mathcal{L}(\mathcal{H})$?

4.2 超节点模型检验

我们首先针对 Kripke 结构，阐述并解决超节点逻辑（而非自动机）的模型检测问题。该算法构成了超节点自动机模型检测的关键子程序。为了在 Kripke 结构上解释超节点逻辑的公式，我们为 Kripke 结构配备两组世界：入口世界，轨迹片段从此处开始；以及 出口世界，轨迹片段在此处结束。形式上，一个开放 *Kripke* 结构 由一个 Kripke 结构 $K = (W, \Sigma^X, \Delta, V)$ 以及一对 $\mathbb{W} = (W_{\mathrm{in}}, W_{\mathrm{out}})$ 构成，其中包含一个由 $W_{\mathrm{in}} \subseteq W$ 个入口世界组成的集合，以及一个由 $W_{\mathrm{out}} \subseteq W$ 个出口世界组成的集合。

开放 Kripke 结构 $(K, \mathbb{W})$ 的一条路径是 K 中从一个入口世界出发、 $w_0 \in W_{\mathrm{in}}$ 并在一个出口世界结束的路径 $w_0 \dots w_n$， $w_n \in W_{\mathrm{out}}$。与 Kripke 结构的情况一样，我们用 $\mathrm{Paths}(K, \mathbb{W})$ 表示开放 Kripke 结构 $(K, \mathbb{W})$ 的路径集合。由开放 Kripke 结构 $(K, \mathbb{W})$ 生成的未压缩轨迹片段集合为 $\mathrm{Unzip}(K)(x) = \{V(w_0, x) \dots V(w_n, x) \mid w_0 \dots w_n \in \mathrm{Paths}(K, \mathbb{W})\}$，对所有变量 $x \in X$ 均成立。

超节点逻辑模型检验问题

设 $(K, \mathbb{W})$ 是一个开放 Kripke 结构， φ 是定义在相同变量集合 X 和有限域 Σ上的超节点逻辑公式。由 $(K, \mathbb{W})$ 生成的未压缩轨迹片段集合是否是 φ 的模型；也就是说，$\mathrm{Unzip}(K, \mathbb{W}) \models \varphi$?

无停顿自动机

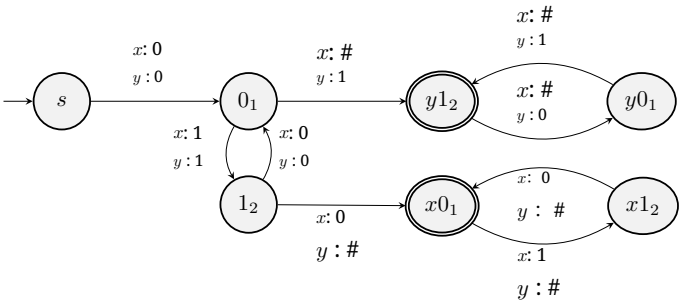
由命题 2 可知，要解决超节点逻辑模型检验问题，只需考虑 $\mathrm{Unzip}(K, \mathbb{W})$ 的停顿约简。我们引入无停顿自动机这一形式体系，用于指定无停顿的未展开轨迹片段集合。我们使用无停顿自动机的布尔算子来定义一种过滤：将其应用于超节点公式φ 以及定义在变量 φ 上的无停顿自动机后，会返回一个语言非空的自动机，当且仅当输入自动机的语言是 φ 的模型。最后，我们根据给定的开放 Kripke 结构 $(K, \mathbb{W})$，构造一个无停顿自动机；当某个未展开轨迹片段是由 $(K, \mathbb{W})$ 生成的轨迹片段的停顿约简时，该自动机便接受该片段。将这三个部分结合起来，即可定义超节点逻辑的模型检测算法。

无停顿自动机是非确定有限自动机（NFA）的一种受限形式，它读取未压缩轨迹片段，并保证对于每个状态，其输入转移和输出转移中不存在重复的变量赋值。我们用 Σ^X 表示将 X 中的所有变量赋值为 Σ 中的值或终止符号 #。形式化地，对于 $X = \{x_0, \dots, x_m\}$，令 $\Sigma^X = \{x_0 : \sigma_0, \dots, x_m : \sigma_m \mid \forall 0 \le i \le m\ \sigma_i \in \Sigma \cup \{\#\}\} \setminus \{x_0 : \#, \dots, x_m : \#\}$。

▶**定义 5**。 设 X 是 Σ上的有限变量集合。字母表 Σ^X 上的非确定无停顿自动机（*NSFA*）是一个元组 $A = (Q, \hat{Q}, F, \delta)$，其中包含有限集合 Q 的

XX:10 超节点自动机

图 3 无停顿自动机 $\mathcal{A}$，其中 x-轨迹片段的长度为奇数，y-轨迹片段的长度为偶数，并且 x 和 y 的首个值为 0。



ˆ状态、初始状态集合 $Q \subseteq Q$、终态集合 $F \subseteq Q$ 和转移关系 δ：$Q \times \Sigma^X \to 2^Q$，满足以下条件：对于所有状态 $q \in Q$ 和变量 $x \in X$，无停顿要求 $In(q, x) \cap Out(q, x) \subseteq \{\#\}$；终止要求，若 $\# \in In(q, x)$，则 $Out(q, x) = \{\#\}$，其中 $In(q, x)$ 是所有进入状态 q 的 x-赋值集合，而 $Out(q, x)$ 是所有从状态 q 出去的 x-赋值集合；形式上，$In(q, x) = \{v(x) \mid q \in \delta(q', v)$，其中某个 $q' \in Q$ 满足条件；$Out(q, x) = \{v(x) \mid \delta(q, v) \neq \emptyset\}$。

无停顿自动机 A 的一次运行是一个有限序列 $q_0 v_0 q_1 v_1 \ldots v_{n-1} q_n$，由交替出现的ˆ状态和变量赋值组成，以初始状态 $q_0 \in Q$ 开始，并且对于所有 $i < n$ 都满足转移函数 $q_{i+1} \in \delta(q_i, v_i)$。如果该运行以终止状态 $q_n \in F$ 结束，则称其为接受运行。对于变量集合 X（其域为 Σ）上的未压缩轨迹片段 τ，当且仅当存在一个接受运行 $q_0 v_0 \ldots v_{n-1} q_n$，使得对于所有 $x \in X$，$\tau(x) = v_0(x) \ldots v_{n-1}(x)$，该未压缩轨迹片段才会被无停顿自动机 A 接受。A 的语言记作 L(A)，是所有被 A 接受的未压缩轨迹片段所组成的集合。有时我们会讨论不含终止符号的无停顿自动机的语言：$\mathrm{L(A)}|_{\#} = \{\tau|_{\#} : X \to \Sigma^* \mid \tau \in \mathrm{L(A)}\}$，其中 $\tau|_{\#}$ 会移除轨迹片段 τ 中的所有 #。注意，由于 A 是无停顿的，因此 $\mathrm{L(A)}|_{\#} = \lfloor \mathrm{L(A)}|_{\#} \rfloor$，其中 $\lfloor \cdot \rfloor$ 表示未压缩轨迹片段的停顿约简。

图 3 展示了一个无停顿自动机的示例，该自动机接受布尔变量 $\{x, y\}$ 的所有未压缩轨迹片段，其中所有 x-轨迹片段的长度均为奇数，而所有y-轨迹片段的长度均为偶数，并且 x 和 y 的首个值均为 0。于是，图 3 中的自动机$\mathcal{A}$定义了语言 $\mathcal{L}(\mathcal{A}) = \{x: \tau_x, y: \tau_y \mid t_x \in (01)^*0$ 和 $t_y \in (01)^*01\}$。

NSFA 的并集、交集和确定化与 NFA 的相应操作相同。形式上，对于字母表为 Σ^X 的无停顿自动机 $\mathrm{A}_1 = (Q_1, \hat{}Q_1, F_1, \delta_1)$ 和 $\mathrm{A}_2 = (Q_2, \hat{}Q_2, F_2, \delta_2)$，它们的并集为定义在同一字母表 Σ^X 上的 $\mathrm{A}_1 \cup \mathrm{A}_2 = (Q_1 \dot{\cup} Q_2, \hat{}Q_1 \dot{\cup} \hat{}Q_2, F_1 \dot{\cup} F_2, \delta_\cup)$，其中当 $q \in Q_i$ 时，$\delta_\cup(q) = \delta_i(q)$，且 $i \in \{1, 2\}$；它们的交集为定义在同一字母表 Σ^X 上的 $\mathrm{A}_1 \cap \mathrm{A}_2 = (Q_1 \times Q_2, \hat{}Q_\cap, F_\cap, \delta_\cap)$，其中ˆ $Q_\cap = \{(q_1, q_2) \mid q_1 \in \hat{}Q_1$ 且 $q_2 \in \hat{}Q_2\}$，$F_\cap = \{(q_1, q_2) \mid q_1 \in F_1$ 且 $q_2 \in F_2\}$，并且 $\delta_\cap((q_1, q_2)\ l)$ 当且仅当 $= (q_1', q_2')$、$\delta_1(q_1, l) = q_1'$ 且$\delta_2(q_2, l) = q_2'$。而 A_1 的确定化定义为 $\mathrm{det(A}_1) = (2^{Q_1}, \hat{}Q_1, F_d, \delta_d)$，其中 $F_d = \{S \in 2^{Q_1} \mid S \cap F_1 \neq \emptyset\}$，且 $\delta_d(S, v) = \bigcup \delta_1(q, v)$，满足 $v \in \Sigma^X$。下文将证明$q \in S$无停顿自动机对上述运算封闭。

▶**命题 6.** 设 A_1 和 A_2 是定义在命题变量 X上的两个无停顿自动机。那么，$A_1 \cup A_2$ 和 $A_1 \cap A_2$ 都是定义在 X 上的无停顿自动机，且 $L(A_1 \cup A_2) = L(A_1) \cup L(A_2)$，并且 $L(A_1 \cap A_2) = L(A_1) \cap L(A_2)$。此外，对于一个非确定性

无停顿自动机 A，其确定化 $det(A)$ 是定义在 X 上的确定性无停顿自动机，并且具有相同的语言，即 $L(det(A)) = L(A)$。

证明。 第一部分可直接通过将无停顿自动机转换为非确定有限自动机得出。并集和析取不会影响与无停顿相关的限制。

考虑任意一个非确定无停顿自动机 $\mathcal{A}_1$。我们首先证明 $\det(\mathcal{A}_1)$ 满足无停顿条件，即对于它的所有状态 S 和变量 $x \in X$，$In(S, x) \cap Out(S, x) = \emptyset$。根据定义：

$$In(S,x) = \{v(x) \mid S \in \delta_d(S', v) \text{ for some } S' \in Q_d\} \Leftrightarrow$$
$$In(S,x) = \{v(x) \mid S = \bigcup_{q_1 \in S_1} \delta(q_1, v) \text{ for some } S' \in Q_d\} \Leftrightarrow$$
$$In(S,x) = \{v(x) \mid \forall q \in S \ \exists q_1 \in S' \text{ s.t. } q \in \delta(q_1, v) \text{ for some } S' \in Q_d\}.$$

因此，对于所有 $q \in S$，$(\star)$ $\mathrm{In}(q,x) = \mathrm{In}(S,x)$。由于 A_1 是一个非确定无停顿自动机，我们知道，对于所有 $q \in S$，$\mathrm{In}(q,x) \cap \mathrm{Out}(q,x) = \emptyset$。反设存在一个值同时属于 S 针对变量 x 的入转移和出转移，即 $l \in \mathrm{In}(S,x) \cap \mathrm{Out}(S,x)$。那么，根据 $\mathrm{Out}(S,x)$ 的定义，S，$q \in S$ 中存在一个状态，使得 $\delta(q, x: l) \neq \emptyset$。这与我们的结论 $(\star)$ 矛盾。现在证明 $\det(\mathrm{A}_1)$ 满足 # 结尾要求。根据 $(\star)$，如果 $\# \in \mathrm{In}(x)$，那么对于所有 $q \in S$，$\# \in \mathrm{In}(x)$。因此，由于 A 是一个非确定无停顿自动机，对于所有 $q \in S$，我们有 $\mathrm{Out}(x) = \{\#\}$。因此 $\mathrm{Out}(x) = \{\#\}$。

最后，$\mathcal{L}(\det(\mathcal{A}_1)) = \mathcal{L}(\mathcal{A})$。可直接由非确定有限自动机的同一结果得出。 ◀

无停顿自动机的补运算采用与非确定有限自动机相同的方法：首先将自动机确定化，然后对其进行完备化，最后交换终态和非终态。对于非确定无停顿自动机，唯一需要特别注意的操作是完备化，因为我们需要谨慎确保满足无停顿条件。

在 Σ^X 上的无停顿自动机 $\mathrm{A} = (Q, \hat{Q}, F, \delta)$ 是完备的，当且仅当 $\mathrm{In}(q) \cup \mathrm{Out}(q)$ 是根据定义 5 中的条件从 Σ^X 得出的极大子集，其中 $\mathrm{In}(q) = \{v(x) \mid x \in X \text{ 和 } v(x) \in \mathrm{In}(q,x)\}$，而 $\mathrm{Out}(q) = \{v(x) \mid x \in X \text{ 和 } v(x) \in \mathrm{Out}(q,x)\}$。下文定义的、在 Σ^X 上的通用无停顿自动机 U_{Σ^X} 是一个确定且完备的自动机，其语言为 $\mathrm{L}(\mathrm{U}_{\Sigma^X})|_{\#} = \lfloor (\Sigma^*)^X \rfloor$，也就是说，它包含 Σ^X 上所有无停顿未解压轨迹。我们将通用无停顿自动机用作完备化其他自动机时的“汇”区域。

▶ **定义 7.** 令 $X = \{x_0, \ldots, x_m\}$ 为有限域 Σ 上的变量集合。Σ^X 上的通用无停顿自动机 为 $U_{\Sigma^X} = (Q_{\mathcal{U}}, Q_{\mathcal{U}}, Q_{\mathcal{U}}, \delta_{\mathcal{U}})$，其中 $Q_{\mathcal{U}} = \Sigma^X$ 和

$$\delta_{\mathcal{U}}(\{x_i : \sigma_i\}_{i\in[0,m]}, \{x_i : \sigma'_i\}_{i\in[0,m]}) = \begin{cases} \{x_i : \sigma'_i\}_{i\in[0,m]} & \text{if } \forall 0 \le i \le m, \text{ if } \sigma_i = \# \text{ then } \sigma'_i = \# \text{ else } \sigma_i \neq \sigma'_i; \\ \emptyset & \text{otherwise.} \end{cases}$$

图4展示了布尔变量集合 $X = \{x, y\}$上的通用无停顿自动机 $\mathcal{U}_X$ 。我们使用通用自动机的状态和转移来完备化其他无停顿自动机。形式化地，给定变量 X 的域为 Σ 上的无停顿自动机 $\mathcal{A} = (Q, \hat{Q}, F, \delta)$，$\mathcal{A}$ 的完备化是同一组变量上的无停顿自动机 $\mathrm{complete}(\mathcal{A}) = (Q \cup Q_{\mathcal{U}}, \hat{Q}, F, \delta')$，其中 $Q_{\mathcal{U}}$ 是通用无停顿自动机 $\mathcal{U}_{\Sigma^X}$ 的状态，并且对于所有状态 $q \in Q \cup Q_{\mathcal{U}}$ 和所有赋值 $v \in \Sigma^X$：

$$\delta'(q,v) = \begin{cases} \delta_{\mathcal{U}}(q,v) & \text{如果是 } q \in Q_{\mathcal{U}}, \\ \delta(q,v) \cup \{v' \mid \text{for all } x \in X \text{ if } v(x) = \# \text{ then } v'(x) = \# & \\ \qquad \text{otherwise } v'(x) \in (\Sigma \setminus In(q,x) \cup Out(q,x))\} & \text{如果 } q \in Q. \end{cases}$$

XX：12 超节点自动机

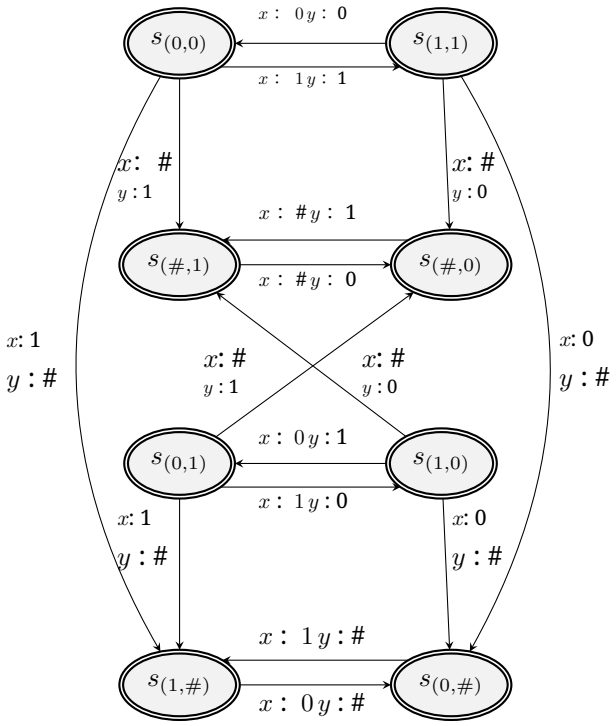


图4 通用无停顿自动机 $\mathcal{U}_{\{x,y\}}$ 基于布尔变量 $\{x, y\}$。它接受关于 $\{x, y\}$ 的所有无停顿的未展开轨迹片段。所有状态既是初始状态也是最终状态。

其中，输入和输出是相对于$\mathcal{A}$的转移关系 δ 定义的，而 $\delta_{\mathcal{U}}$ 是通用无停顿自动机$\mathcal{U}_{\Sigma x}$的转移关系。

▶**命题 8**。 设 A 是一个无停顿自动机。那么，$\mathrm{complete}(A)$ 是一个与 A 具有相同语言的完备无停顿自动机，即 $L(\mathrm{complete}(A)) = L(A)$。

证明。考虑任意无停顿自动机 A。A 和通用自动机都是无停顿自动机，满足无停顿和终止要求。要证明 complete(A) 是一个无停顿自动机，只需证明扩展 A 的转移关系（指向通用自动机中的状态）同时保持这两个要求。根据定义，如果变量轨迹已终止，则它将保持终止（如果 $v(x) = \#$，那么 $v'(x) = \#$），因此满足终止要求；如果它尚未终止，我们只添加标记为状态的入边和出边中未出现的值的转移（对于所有 $x \in X$, $v'(x) \notin \mathrm{In}(q) \cup \mathrm{Out}(q)$），因此满足无停顿要求。现在我们证明 complete(A) 是完备的。根据通用自动机的定义可知，通用自动机是完备的。对于 A 中所有不完备的转移（即尚未终止的变量），我们添加指向匹配的通用自动机状态的缺失转移；因此 complete(A) 是完备的。

最后，我们证明这两个自动机定义了相同的语言。由于我们保留了所有 $\mathcal{A}$转移，因此可得 $\mathcal{L}(\mathrm{complete}(\mathcal{A})) \supseteq \mathcal{L}(\mathcal{A})$。现在我们证明 $\mathcal{L}(\mathrm{complete}(\mathcal{A})) \subseteq \mathcal{L}(\mathcal{A})$。首先，我们注意到，没有转移将通用自动机中的状态与 $\mathcal{A}$ 中的状态连接起来。随后，当一次运行到达通用自动机中的某个状态时，之后的所有步骤都在通用自动机内进行。此外，终态不包括通用自动机中的状态。因此，接受运行只包含 $\mathcal{A}$ 中的转移。

◀

确定性且完备的无停顿自动机 $\mathcal{A}=(Q,\hat{}Q,F,\delta)$ 在 Σ^X 上的补自动机是 $\mathcal{A}=(Q,\hat{}Q,Q\setminus F,\delta)$，其中终态和非终态互换。

▶ **命题 9.** 设 A 是定义在 Σ^X 上的确定性且完备的无停顿自动机。那么，A 是无停顿自动机，并且 $L(A)=\lfloor(\Sigma^*)^X\rfloor\setminus L(A)$。

证明。设 A 是任意一个无停顿自动机。由 A 是无停顿自动机可直接得出，A 也是无停顿自动机（对自动机取补不会改变其转移函数）。出于同样的原因，A 也是一个确定且完备的无停顿自动机。于是，对于所有解压后的轨迹 $\tau\in\lfloor(\Sigma^*)^X\rfloor$，A 中都存在一个对应于 τ 的运行 r（由完备性可知）；并且这是 τ 唯一的运行（由确定性可知）。由于 A 和 A 具有相同的转移函数，因此 r 也是 A 中对应于 τ 的唯一运行。最后，由于终态已翻转，可知 r 是 A 中对应于 τ 的接受运行，当且仅当 r 不是 A 中对应于 τ 的接受运行。因此 $L(A)=\lfloor(\Sigma^*)^X\rfloor\setminus L(A)$。◀

从超节点逻辑公式到无停顿自动机

定义了在并集、交集和补集运算下封闭的无停顿自动机，为后续工作奠定基础后，我们现在转向超节点逻辑模型检验问题。给定一个无停顿自动机和一个超节点公式，我们沿着超节点公式的结构定义归纳过滤（即在每一步中生成一个子自动机），并将其应用于要进行模型检测的自动机。如果过滤返回的自动机语言非空，则输入自动机就是输入公式的一个模型。针对布尔算子的归纳过滤可自然地转换为自动机算子。对于原子超节点公式（即谓词 $\precsim$），我们定义一个无停顿自动机来刻画 $\precsim$ 的含义。图 5 展示了模型检测算法的概览。

▶ 定义 *10.* 令 $U_X=(Q_U,Q_U,Q_U,\delta_U)$ 为域 Σ^X 上的通用无停顿自动机，并令 $x,y\in X$。超节点逻辑原子公式 $x\precsim y$ 对应的无停顿自动机是定义在相同变量和域上的无停顿自动机 $A^{x\precsim y}=(Q,Q,Q,\delta)$，其中 $Q=\{v\in Q_U\mid v(x)=v(y)\text{ 或 }v(x)=\#\}$，并且对于所有 $q\in Q$ 和 $v\in\Sigma^X$，$\delta(q,v)=\delta_U(q,v)$。

为了处理超节点公式中的轨迹变量，我们通过 $X_{\mathcal{V}}=\{x_\pi\mid x\in X\text{ 和 }\pi\in\mathcal{V}\}$ 将对 $\mathcal{V}$ 中轨迹变量的引用扩展到变量集合 X。对于变量集合 $X_{\mathcal{V}}$ 上的解压轨迹片段 τ，我们推导出轨迹赋值 $\Pi_\tau(\pi,x)=\tau(x_\pi)$，其中适用于所有 $x\in X$ 和 $\pi\in\mathcal{V}$。现在我们证明，对于 $x(\pi)\precsim y(\pi')$，无停顿自动机所接受的所有单词都定义了满足该超节点原子公式的赋值。

▶ **引理 11.** 若解压后的轨迹片段 τ 位于 $(\Sigma^{X_{\mathcal{V}}})^*$，则它被 $A^{x_\pi\precsim y_{\pi'}}$ 在相同变量和域上接受，当且仅当 $\Pi_\tau\models x(\pi)\precsim y(\pi')$.

证明。考虑任意解压后的轨迹片段 τ over $(\Sigma^{X_{\mathcal{V}}})^*$。我们首先考虑⇒-方向，并假设 $\tau\in L(A^{x_\pi\precsim y_{\pi'}})|_\#$。那么，存在一个接受运行 $q_0\sigma_0q_1\sigma_1\dots\sigma_nq_{n+1}$，其中，对于所有 $x_\pi\in X_{\mathcal{V}}$，如果 $0\le i<|\tau(x_\pi)|$，则 $\sigma_i(x_\pi)=\tau(x_\pi,i)$，否则 $\sigma_i(x_\pi)=\#$。根据通用无停顿自动机和 $A^{x_\pi\precsim y_{\pi'}}$ 的定义，可得对于所有 $0\le j\le n$，$\sigma_j(x_\pi)=\sigma_j(y_{\pi'})$ 或 $\sigma_j(x_\pi)=\#$。此外，根据无停顿自动机的终止要求，如果存在 j s.t. $\sigma_j(x)=\#$，那么 x 的值将不再发生变化（即，对于所有 $k\ge j$，$\sigma_k(x)=\#$）。随后，我们可以通过对轨迹 τ 的大小进行归纳法，证明 $\Pi_\tau\models x(\pi)\precsim y(\pi')$。对于 ⇐-方向，我们假设 τ 不被 $A^{x_\pi\precsim y_{\pi'}}$ 接受。由于 $A^{x_\pi\precsim y_{\pi'}}$ 中的所有状态都是终态，因此存在一个步骤 $0\le i\le n$，其中 $\sigma_i(x_\pi)\neq\sigma_i(y_{\pi'})$ 且 $\sigma_i(x_\pi)\neq\#$。鉴于 τ 是无停顿的，因此 $\Pi_\tau\not\models x(\pi)\precsim y(\pi')$。◀

XX:14 超节点自动机

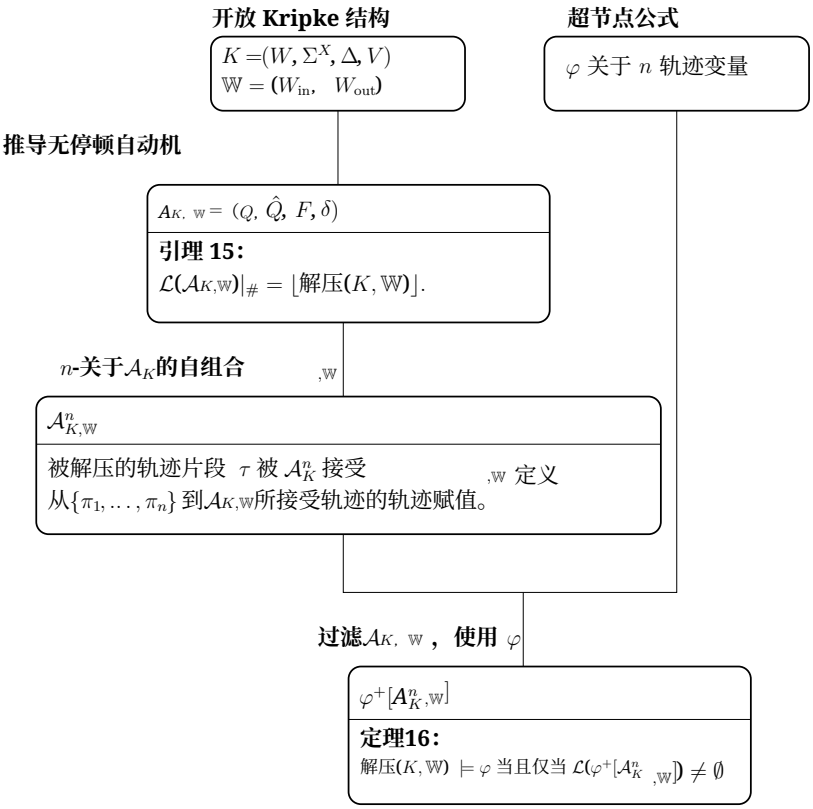


图 5 超节点公式的模型检测算法及相关结果。

接下来定义的归纳过滤是针对超节点逻辑公式的模型检测算法的主要组成部分。

▶**定义 12**。 设 A 是一个无停顿自动机， φ 是一个超节点逻辑公式。我们按如下方式，根据 φ 的结构归纳地定义 A 的正过滤和负过滤，分别记为 φ、 $\varphi^+[A]$ 和 $\varphi^-[A]$,:

$$(x(\pi) \precsim y(\pi'))^+[\mathcal{A}] = \mathcal{A} \cap \mathcal{A}_{x_\pi \precsim y_{\pi'}} \qquad (x(\pi) \precsim y(\pi'))^-[\mathcal{A}] = \mathcal{A} \cap \overline{\mathcal{A}_{x_\pi \precsim y_{\pi'}}}$$
$$(\varphi_1 \wedge \varphi_2)^+[\mathcal{A}] = \varphi_1^+[\mathcal{A}] \cap \varphi_2^+[\mathcal{A}] \qquad (\varphi_1 \wedge \varphi_2)^-[\mathcal{A}] = \varphi_1^-[\mathcal{A}] \cup \varphi_2^-[\mathcal{A}]$$
$$(\neg\varphi)^+[\mathcal{A}] = \varphi^-[\mathcal{A}] \qquad (\neg\varphi)^-[\mathcal{A}] = \varphi^+[\mathcal{A}]$$
$$(\exists\pi\varphi)^+[\mathcal{A}] = \varphi^+[\mathcal{A}] \qquad (\exists\pi\varphi)^-[\mathcal{A}] = \mathcal{A} \setminus \varphi^+[\mathcal{A}].$$

我们将对无停顿自动机 $\mathcal{A}$ 关于含有 n 个轨迹变量的公式φ 进行模型检测的问题，转化为通过 φ过滤 $\mathcal{A}$ 的 n 自组合。无停顿自动机 $\mathcal{A}^n$ 是通过标准同步积构造组合 n 个 $\mathcal{A}$ 副本的结果；对于每个副本 $\mathcal{A}_i$，其中 $i \leq n$，所有程序变量 $x \in X$ 都重命名为 x_{π_i}。请注意，由 $\mathcal{A}^n$ 接受的解压后轨迹片段 τ 所导出的赋值，定义了一个从 $\{\pi_1$、.....、 $\pi_n\}$ 到 $\mathcal{A}$ 所接受轨迹的轨迹赋值。

▶**定理 13.** 设 A 是一个无停顿自动机，且 φ 是一个带有n 轨迹变量的超节点逻辑公式。那么，$L(\varphi^+[A^n]) \neq \emptyset$ 当且仅当 $L(A) \models \varphi$，且 $L(\varphi^-[A^n]) \neq \emptyset$ 当且仅当 $L(A) \not\models \varphi$。

证明。 这可由下文针对开放公式证明的引理推出:

设 τ 是 $(\Sigma^{X_V})^*$ 上的一个解压后的轨迹片段。那么， $\tau \in L(\varphi^+[A^n])|_\#$ 当且仅当 $\Pi_\tau \models \varphi$; 且 $\tau \in L(\varphi^-[A^n])|_\#$ 当且仅当 $\Pi_\tau \not\models \varphi$。

我们通过对超节点公式的结构进行归纳法来证明该引理。为证明基础情形， $x_\pi \precsim y_{\pi'}$我们使用引理 11。特别地， $\tau \in \mathcal{L}(x_\pi \precsim y^+_{\pi'}[\mathcal{A}^n])|_\#$ 当且仅当 $\tau \in \mathcal{L}(\mathcal{A}^n)|_\#$且 $\tau \in \mathcal{L}(\mathcal{A}^{x_\pi \precsim y_{\pi'}})|_\#$ ⇔。由于 $\tau \in \mathcal{L}(\mathcal{A}^n)|_\#$，因此 Π_τ定义了一个从$\{\pi_1, \ldots \pi_n\}$到 A 接受的轨迹的轨迹赋值，并且根据引理 11， $\Pi_\tau \models x_\pi \precsim y_{\pi'}$。我们对 $\tau \in \mathcal{L}(x_\pi \precsim y^-_{\pi'}[\mathcal{A}^n])|_\#$也以类似方式进行证明。

接下来讨论归纳步骤，并假设归纳假设（IH）对任意超节点公式 φ 和 φ'均成立。归纳情形 $\varphi \wedge \varphi'$ 由 IH 和命题 6 得出。而归纳情形 $\neg\varphi$ 则由 IH 和命题 9 得出。

存在量词 – $\exists\pi\varphi$ – 的情形更具挑战性。我们先证明正过滤，并从 ⇒ 方向入手。考虑任意 $\tau \in \mathrm{L}((\exists\pi\varphi)^+[\mathrm{A}^n])|_\#$。根据过滤的定义，这等价于 $\tau \in \mathrm{L}(\varphi^+[\mathrm{A}^n])|_\#$。根据 IH， $\Pi_\tau \models \varphi$, 因此$\Pi_\tau \models \exists\pi\varphi$。现在证明 ⇐ 方向。假设存在一个轨迹赋值，使得 $\Pi \models \exists\pi\varphi$。那么，根据超节点公式满足关系的定义，存在一条在 Σ^X 上的未解压轨迹 τ'，且该轨迹被 A 接受（即 $\tau' \in \mathrm{L(A)}|_\#$），使得 $\Pi[\pi \mapsto \tau'] \models \varphi$。注意，从未解压轨迹片段 τ（其位于 $(\Sigma^{X_V})^*$ 上）推导出轨迹赋值（将轨迹变量 V 映射到 Σ^X 上的未解压轨迹）的函数是可逆的。于是，根据 IH，由赋值扩展 $\Pi_\pi = \Pi[\pi \mapsto \tau']$ 推导出的轨迹 τ_π 属于 $\mathrm{L}(\varphi^+[\mathrm{A}^n])|_\#$。并且根据过滤的定义， $\tau_\pi \in \exists\pi\varphi^+[\mathrm{A}^n]$。

现在证明负向过滤。我们先证明 ⇒-方向，并通过逆否命题来证明。假设对于任意轨迹赋值 Π，我们有 $\Pi \models \exists\pi\varphi$，那么存在一个扩展 $\Pi_\pi = \Pi[\pi \mapsto \tau\prime]$，满足 $\tau' \in \mathcal{L}(\mathcal{A})|_\#$，使得 $\Pi_\pi \models \varphi$。根据归纳假设，该轨迹

τ_π 由 Π_π 导出，并被 $\varphi^+[A^n]$接受。因此， τ_π 不被 $A^n \setminus \varphi^+[A^n]$接受；根据定义， $\tau_\pi \notin L((\exists\pi\varphi)^-[A^n])$。对于 $\Leftarrow$ 方向，考虑任意满足$\Pi \models \neg\exists\pi\varphi$的轨迹赋值。于是， $\Pi \models \forall\pi\neg\varphi$。等价地，对于 Π 的所有扩展，也就是说， $\Pi_\pi = \Pi[\pi \mapsto \tau']$ 对于所有 $\tau' \in L(A)|_\#$，我们都有 $\Pi_\pi \not\models \varphi$。考虑其中任意一个扩展 Π_π，根据归纳假设，对于由它导出的轨迹 τ_π，我们有 $\tau_\pi \in L(A^n)|_\#$ 且 $\tau_\pi \notin L(\varphi^+[A^n])$。等价地，$\tau_\pi \in L(A^n \setminus \varphi^+[A^n])$，因此 $\tau_\pi \in L((\exists\pi\varphi)^-[A^n])$。◀

从开放 Kripke 结构到无停顿自动机

现在我们展示如何将开放 Kripke 结构转换为一个无停顿自动机，使其接受与该 Kripke 结构生成的集合相同的未压缩轨迹片段。由于未压缩轨迹表示每个变量各自的演变过程，因此在根据开放 Kripke 结构构建无停顿自动机时，我们需要为每个变量保留一个独立的进度指针（以跳过停顿状态）。

首先，我们定义一个函数，该函数针对给定克里普克结构中状态 w 的每个变量 x，返回下一个无停顿转移。形式上，对于定义在变量集合 X 上、域为 Σ 的克里普克结构 $K = (W, X, \Delta, V)$，所有从 $w \in W$ 可达且通过一次转移到达的世界中， $x \in X$ 未被标记为 $\sigma \in \Sigma \cup \{\#\}$ 的世界所组成的集合为：

$$N(w, \sigma, x) = \{w' \mid \mathrm{Unzip}(K, w, w')[x] \in \sigma^+, (w', w'') \in \Delta \text{ and } V(w'', x) \neq \sigma\},$$

其中，$\mathrm{Unzip}(K, w, w')$ 是由 K 中从 w 到 w' 的路径所定义的解压轨迹集合。为了处理变量不存在具有不同值的更多可达世界的情况，在派生的无停顿自动机中，我们添加一组终止状态： $W^\# = \{w^\# \mid w \in W\}$。最后，转换后的无停顿自动机转移关系使用下文定义的函数 $\mathrm{next}/3$，并根据需要添加到终止状态的转移。形式化地，我们将 $\mathrm{next}(w \in W \cup W^\#, \sigma \in \Sigma \cup \{\#\}\ x \in X)$ 定义如下：

$$\mathrm{next}(w, \sigma, x) = \begin{cases} \{w^\#\} & \text{if } V(w, x) = \sigma \text{ and } N(w, \sigma, x) = \emptyset, \\ \{w\} & \text{if } \sigma = \# \text{ and } w \in W^\#, \\ N(w, \sigma, x) & \text{otherwise.} \end{cases}$$

▶ **定义 14.** 设 $(K, \mathbb{W}) = ((W, \Sigma^X, \Delta, V), (W_{\mathrm{in}}, W_{\mathrm{out}}))$ 是一个带有 $X = \{x_1, \ldots, x_m\}$ 的开放 *Kripke* 结构。由 $(K, \mathbb{W})$ 诱导的无停顿自动机 为 $A_{K,\mathbb{W}} = (Q, \hat{}Q, F, \delta)$，其中：

- $Q = (W \cup W^\#)^m$; $\hat{}Q = \{(w_1, \ldots, w_m) \mid w_i \in W_{\mathrm{in}}$ 对所有 $1 \le i \le m\}$;
- $F = \{(w_1, \ldots, w_m) \mid w_i \in W_{\mathrm{out}} \cup W^\#$ 输出 对所有 $1 \le i \le m\}$; $\delta((w_1, \ldots$
- $, w_m)\ v) = \{(w'_1, \ldots, w'_m) \mid w'_i \in$ 下一个 $(w_i, v[i], x_i)$ 对所有 $i\}$。

我们证明，无停顿自动机$A_{K,\mathbb{W}}$ $(K, \mathbb{W})$的语言与由 $(K, \mathbb{W})$ 定义的未压缩轨迹片段集合的停顿约简相同。

▶ **引理 15.** 对于所有开放克里普克结构$(K, \mathbb{W})$，均有 $L(A_{K,\mathbb{W}})|_\# = \lfloor$解压$(K, \mathbb{W})$ $\rfloor$.

证明。$L(A_{K,\mathbb{W}})|_\# \subseteq \lfloor \mathrm{Unzip}(K, \mathbb{W}) \rfloor$：设 τ 是 $A_{K,\mathbb{W}}$ 接受的、定义在 $(\Sigma^X)^*$ 上的解压后轨迹片段。那么，存在一个接受运行 $A_{K,\mathbb{W}}$ $q_0\sigma_0q_1\sigma_1 \ldots \sigma_n q_{n+1}$，其中 $\sigma_i = \tau[i]$，对于 $i \ge n_\circ$ 对于每个满足 $i \le m$ 的变量 x_i，我们按运行的大小进行归纳，证明该性质成立。根据 next 的定义，我们证明对于当前步骤 $j \le n-$ $q_j[i]$ $\sigma_j[i]$ $q_{j+1}[i]$，以下情况之一成立：(i) $N(q_j[i], \sigma_j[i], x_i) \neq \emptyset$，此时存在一条从 $q_j[i]$ 到 $q_{j+1}[i]$ 的路径 ϱ，其中 x_i 的值保持为 $\sigma_j[i]$; (ii) $N(q_j[i], \sigma_j[i], x_i) = \emptyset$

并且下一个状态已终止（$q_{j+1}[i] = q_j^{\#}[i]$）；或 (iii) $q_j[i]$ 是一个已终止状态，因此 $\sigma_j[i] =$ # 且 $q_{j+1}[i] = q_{j+1}^{\#}[i]$。据此，当情况 (i) 成立时，我们通过加入 ϱ 来延长无停顿自动机给出的部分路径，从而在克里普克结构中构造一条路径；当情况 (ii) 和 (iii) 成立时，则不做任何处理。请注意，我们确保将已终止的退出路径纳入无停顿自动机的终态中，从而保证接受条件与开放 Kripke 结构上完整路径的定义相符。对新路径进行停顿约简后，我们得到 $\tau[x_i]$。

$\mathcal{L}(\mathcal{A}_{K,\mathbb{W}}) \supseteq \lfloor$解压$(K,\mathbb{W})\rfloor$：设 τ 是$\lfloor$解压$(K,\mathbb{W})\rfloor$中一个解压后的无停顿轨迹片段。那么，存在一条路径属于 $(K,\mathbb{W})$，它生成了该轨迹片段。基于这条路径，对于每个变量，我们构造一个 wolds 序列，以移除所有停顿转移。然后，我们通过归纳法证明，该序列定义了一个接受运行$\mathcal{A}_{K,\mathbb{W}}$。◀

克里普克结构上的超节点逻辑模型检测

通过将开放 Kripke 结构规约为无停顿自动机，我们现在可以使用定义12中的过滤来解决超节点逻辑的模型检测问题。

▶ **定理16.** 设 $(K,\mathbb{W})$ 是一个开放 *Kripke* 结构，且 φ 是基于同一变量集合的超节点逻辑公式。设 n 为 φ 中轨迹变量的数量。那么，解压$(K,\mathbb{W}) \models \varphi$ 当且仅当 $L(\varphi^{+}[A^{n}_{K,\mathbb{W}}]) \neq \emptyset$。

该证明由定理1和定理2以及引理3得出。这给出了我们的主要结果。

▶**定理 17**。 开放克里普克结构上的超节点逻辑模型检测是可判定的。

使用我们的算法，对开放 Kripke 结构上的超节点逻辑公式进行模型检测时，其运行时间与变量数量呈双重指数关系，与 Kripke 结构中的世界数量呈单指数关系，并与公式长度呈单指数关系。

▶**推论18**。 在包含 k 个世界的开放 *Kripke* 结构上，对具有n 个轨迹变量和 m 个变量的超节点逻辑公式 φ 进行模型检测的时间复杂度为 $O(2^{n\cdot k^m})$。

证明。 开放 Kripke 结构的无停顿自动机编码有$\mathcal{O}(k^m)$个状态。确定化无停顿自动机有$\mathcal{O}(2^{k^m})$ 个状态。补全确定性无停顿自动机 2^m 个状态。所得自动机的 n-自组合有$\mathcal{O}(2^{n\cdot k^m})$ 个状态。◀

4.3 超节点自动机模型检测

我们定义了超节点自动机在给定动作序列 p 下的运行，其中每次运行都会导出一组与 p 一致的动作标记轨迹的切片。为了针对带有动作标记的带初始状态的克里普克结构 $(K,\mathbb{A},w_0)$ 对超节点自动机 $\mathcal{H}$ 进行模型检测，我们构建了一个称为 Slice$(K,\mathbb{A},w_0)$ 的有限自动机，它编码了由 $(K,\mathbb{A},w_0)$ 生成的动作标记轨迹的所有切片。随后，我们将模型检测问题归约为检查 Slice$(K,\mathbb{A},w_0)$ 与规范自动机 $\mathcal{H}$ 的组合所定义的语言是否非空，该组合称为 Join$(\mathcal{H}\ K, \mathbb{A}, w_0)$。我们在图 6 中展示了模型检测算法的概览。

我们首先定义给定动作标记 $\mathbb{A}$ 下，给定克里普克结构 $K=(W,\ \Sigma^X,\ \Delta,\ V)$ 的切片。切片的构成模块是克里普克子结构。克里普克结构 $K'=(W',\Sigma^X,\ \Delta',\ V')$ 是 K 的子结构，记作 $K' \leq K$，当且仅当 $W' \subseteq W$，并且对于

XX:18 超节点自动机

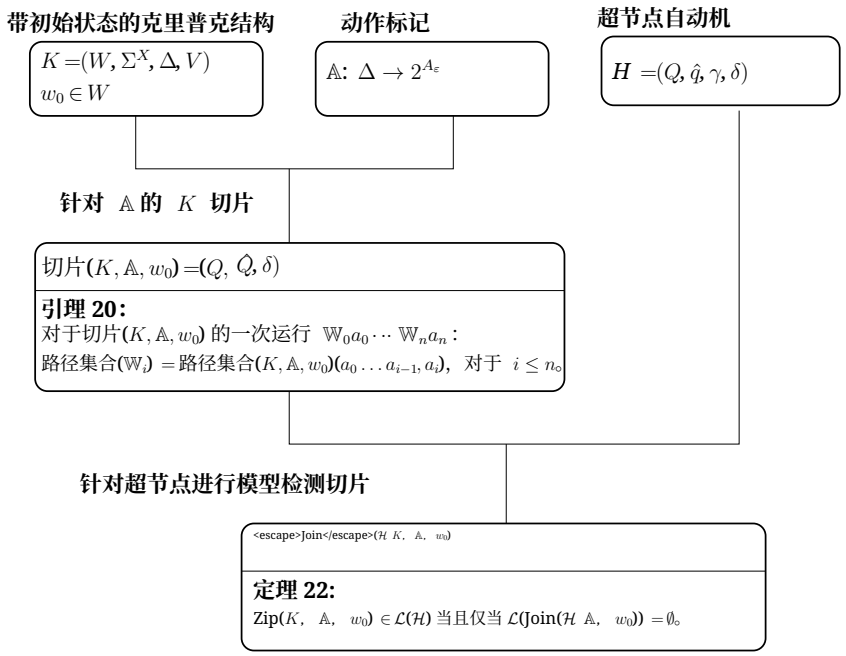


图 6 超节点自动机的模型检测算法及相关结果。

所有世界 $w\in W'$，我们有 $\Delta'(w)\subseteq\Delta(w)$，且 $V'(w)=V(w)$。由转移关系诱导的子结构 $\Delta'\subseteq\Delta$ 为 $K[\Delta']=(W',\ X,\ \Delta',\ V(W'))$，其中 $W'=\{w,\ w'\mid(w,\ w')\in\Delta'\}$。从 $W_{\text{in}}\subseteq W$ 中的入口世界出发，在动作标记的 Kripke 结构 $(K,\ \mathbb{A})$ 中，路径上的所有转移至第一个标记为 $a\in A$ 的步骤所定义的转移关系为：

$$(K,\mathbb{A},W_{\text{in}})\downarrow a=\{(w_j,w_{j+1})\mid w_0\varepsilon\ \ldots\ w_{n-1}\varepsilon\, w_n a\in\text{Paths}(K,\mathbb{A}),w_0\in W_{\text{in}}\text{ for all }j<n\}.$$

由 $(K,\ \mathbb{A},\ W_{\text{in}})\downarrow a$ 所诱导的开放子结构，记作 $\mathbb{K}[(K,\ \mathbb{A},\ W_{\text{in}})\downarrow a]$，是一个开放 Kripke 结构，其中 Kripke 结构为 $K[(K,\ \mathbb{A},\ W_{\text{in}})\downarrow a]$，集合 W_{in} 是入口世界，而集合 $\{w\mid w\in W$ 和 $\mathbb{A}(w,\ a)\neq\emptyset\}$ 是出口世界，包含动作 a 所有可能的出口点。

我们定义有限自动机 Slice$(K,\ \mathbb{A},\ w_0)$，并在下文的引理 20 中证明，每个有限动作序列 p 都定义了该自动机中的唯一一条路径，且这条路径的切片包含相同的轨迹片段；这些轨迹片段也可通过将动作序列 p 直接应用于原始的带有指定状态的动作标记 Kripke 结构而得到。自动机 Slice$(K,\ \mathbb{A},\ w_0)$ 的状态是由从任意选定的入口世界到某个动作 $a\in\mathbb{A}$ 的路径所诱导的所有开放子结构。请注意，这类状态只有有限多个。对于所有动作 a，Slice$(K,\ \mathbb{A},\ w_0)$ 的转移关系都会连接具有出口 a 的开放子结构与入口世界相匹配的开放子结构。

▶**定义 19**。设 (K,w_0) 是一个世界为 W 的带初始状态的克里普克结构，并设 $\mathbb{A}$ 是 K 的动作标记，其动作集合为 A。切片 $\text{Slice}(K,\mathbb{A},w_0)=(Q,\hat{Q},\delta)$ 是一个有限自动机，其中：

- $Q=\{\mathbb{K}[(K,\mathbb{A},W_{\text{in}})\downarrow a]\mid a\in A$ 和 $W_{\text{in}}\subseteq W\}$ 是状态集合，包含初始状态 $\hat{Q}=\{(K,(\{w_0\}\ W_{\text{out}}))\in Q\}$;$\delta$：$Q\times A\to Q$ 是一个转移函数，使得 $\delta((K,(W_{\text{in}},W_{\text{out}}))\ a)=(K',(W'in\ W'_{\text{out}}))$ 当且仅当：$(K,(W_{\text{in}},W_{\text{out}}))$ 执行动作 a 后退出；也就是说，对于所有 $w\in W_{\text{out}}$，存在 $w'\in W$ 使得 $a\in\mathbb{A}(w,w')$; 并且

- 入口世界集合 W'_{in} 定义了一个最大子集，该子集由在 $\mathbb{W}$ 中从出口世界执行动作 a 后可到达的世界组成；也就是说，对于所有不同于 $(K', (W'_{\text{in}}, W'_{\text{out}}))$ 的 $(K'', (W''_{\text{in}}, W''_{\text{out}})) \in Q$：如果 $W''_{\text{in}} \subseteq \{w \mid a \in \mathbb{A}(w', w)$ 对某个 $w' \in W_{\text{out}}\}$，那么 $W'_{\text{in}} \not\subseteq W''_{\text{in}}$。

我们指出，转移函数 $\delta : Q \times A \to Q$ 定义良好，因为对于给定动作 a，下一个入口世界存在唯一的极大子集。对于任意两个开放克里普克子结构，$(K, (W_{\text{in}}, W_{\text{out}}))$ 和 $(K', (W_{\text{属于}} W'_{\text{out}}))$，它们的并集定义了 $\mathbb{K}[(K, \mathbb{A}, W_{\text{in}} \cup W'_{\text{in}})) \downarrow a]$，而这再次是切片中的一个状态。

▶ **引理 20**。令 (K, w_0) 为一个带初始状态的克里普克结构，且 $\mathbb{A}$ 为 K 的动作标记，其动作集合为 A。对于 A^* 中的每个有限动作序列 $p = a_0 \dots a_n$，如果 $\text{Zip}(K, \mathbb{A}, w_0)[p] \neq \emptyset$，则 p 定义了 $\text{Slice}(K, \mathbb{A}, w_0)$ 的唯一运行 $\mathbb{K}_0 a_0 \cdots \mathbb{K}_n a_n$，使得对于所有 $0 \leq i \leq n$，路径集合$(\mathbb{K}_i)$ = 路径集合$(K, \mathbb{A}, w_0)(a_0 \dots a_{i-1}, a_i)$。

证明。考虑任意克里普克结构 $K = (W, \Sigma^X, \Delta, V)$、世界 $w_0 \in K$ 和动作标记 $\mathbb{A} : (W \times A) \to W$。由于 $\text{Slice}(\mathbb{A}(K, w_0))$ 的转移函数是确定性的，因此 (K, w_0) 中带有标记 $\mathbb{A}$ 的所有动作序列 $p \in A^*$，即 $\text{Zip}(\mathbb{A}(K, w_0))[p] \neq \emptyset$，在 $\text{Slice}(\mathbb{A}(K, w_0))$ 中定义了唯一的路径。

我们仍需证明，由切片定义的路径与对 $\mathbb{A}(K, w_0)$ 生成的路径进行切片所得的结果相同。为此，我们对序列的大小进行归纳证明。对于基础情形，对于所有大小为 1 的序列动作 a_0，$\text{Slice}(\mathbb{A}(K, w_0))$ 中所导出的路径为 $\mathbb{K}_0$，因此我们需要证明 $\text{Paths}(\mathbb{K}_0) = \text{Paths}(\mathbb{A}(K))(\varnothing, a_0)$。根据定义 19，$\text{Paths}(\mathbb{K}_0) = \text{Paths}((\mathbb{K}[(K, \{w_0\}) \downarrow_{\mathbb{A}} a_0])$。此外，根据由 a 导出的开放子结构的定义，$\text{Paths}(\mathbb{K}[(K, \{w_0\}) \downarrow_{\mathbb{A}} a_0]) = \{w_0 \dots w_n \mid (w_0, \varepsilon) \dots (w_{n-1}, \varepsilon)(w_n, a_0) \in \text{Paths}(K)\}$。因此，$\text{Paths}(\mathbb{K}[(K, \{w_0\}) \downarrow_{\mathbb{A}} a_0]) = \text{Paths}(\mathbb{A}(K))(\varnothing, a_0)$。

现在进行归纳步骤，我们假设作为归纳假设（IH），该命题对大小为 n 的序列成立。现在考虑大小为 $n+1$，$a_0 \dots a_n$ 的序列。根据 IH，我们知道，对于所有 $0 \leq i < n$，$\text{Paths}(\mathbb{K}_i) = \text{Paths}(\mathbb{A}(K))(a_0 \dots a_{i-1}, a_i)$。我们只需证明 $\text{Paths}(\mathbb{K}_n) = \text{Paths}(\mathbb{A}(K))(a_0 \dots a_{n-1}, a_n)$。根据 IH，我们知道 $\text{Paths}(\mathbb{K}_{n-1})$ 和 $\text{Paths}(\mathbb{A}(\mathbb{K}))(a_0 \dots a_{n-2}, a_{n-1})$ 具有相同的终止状态。由此可得，$\text{Paths}(\mathbb{K}_n)$ 和 $\text{Paths}(\mathbb{A}(\mathbb{K}))(a_0 \dots a_{n-1}, a_n)$ 具有相同的初始状态 W_{in}。并且，根据与基础情形类似的推理，$\text{Paths}(\mathbb{K}_n) = \text{Paths}(\mathbb{K}[(K, W_{\text{in}}) \downarrow_{\mathbb{A}} a_n]) = \text{Paths}(\mathbb{A}(K))(a_0 \dots a_{n-1}, a_n)$ ◀

最后，我们定义上述切片自动机与给定超节点自动机$\mathcal{H}$的同步组合。该组合的状态是由开放克里普克子结构（源自给定的带指针、动作标记的 Kripke 结构）和超节点逻辑公式（源自$\mathcal{H}$的超节点标签）组成的有序对。我们将所有开放克里普克子结构不是超节点公式模型的有序对标记为终态。

▶定义 21。设 $\text{H} = (Q_h, \hat{}q, \gamma, \delta_h)$ 是一个超节点自动机。H 与带指针、动作标记的 Kripke 结构 $(K, \mathbb{A}, w_0)$ 的切片 $\text{Slice}(K, \mathbb{A}, w_0) = (Q_s, \hat{}Q_s, F_s, \delta_s)$ 的交集，是有限自动机 $\text{Join}(\text{H } K, \mathbb{A}, w_0) = (Q, \hat{}Q, F, A, \delta)$，其

- 状态 $Q = \{(\mathbb{K}, q) \mid \mathbb{K} \in Q_s$ 和 $q \in Q_h$ 和 $\mathbb{W} \models \gamma(q)\} \cup \{(\mathbb{K}, q) \mid \mathbb{K} \in Q_s$ 和 $q \in Q_h$ 和 $\mathbb{K} \not\models \gamma(q)\}$；初始状态 $\hat{} Q = \{(\mathbb{K}, \hat{}q) \in Q \mid \mathbb{K} \in \hat{}Q_s\} \cup \{(\mathbb{K}, \hat{}q) \in Q \mid \mathbb{K} \in \hat{}Q_s\}$；终止状态 $F = \{(\mathbb{K}, q) \mid (\mathbb{K}, q) \in Q\}$；转移函数 δ：$Q \times A \to Q$，其中对于所有 $(\mathbb{K}, q) \in Q$，都有 $\delta((\mathbb{K}, q)\ a) = \{(\mathbb{K}', q') \in Q \mid \delta(q) = (q', a)$ 且 $\mathbb{K}' \in \delta_s(\mathbb{W}, a)\}$。

有限自动机 Join($\mathcal{H}$ $K, \mathbb{A}, w_0$) 读取动作序列。运行的概念按通常方式定义；若一次运行以终止状态结束，则称其为接受运行。当且仅当自动机没有接受运行时，自动机语言为空。

▶ **定理 22.** 设(K, w_0) 是一个带动作标记 $\mathbb{A}$ 的带初始状态的克里普克结构。设 H 是定义在与$(K,$ $\mathbb{A})$相同的命题和动作集合上的超节点自动机。那么，$\mathrm{Zip}(K, \mathbb{A}, w_0) \in L(H)$ 当且仅当有限自动机 $\mathrm{Join}(H\ K, \mathbb{A}, w_0)$ 的语言为空。

证明。考虑域 Σ、世界 $w_0 \in W$ 和动作标记 $\mathbb{A}\colon (W \times A) \to W$ 上任意的 $K = (W, \Sigma^X,$ $\Delta, V)$。我们要证明，Zip(𝔸(K, w_0)) $\notin$ L(H) 当且仅当 L(Join(H $\mathbb{A}(K, w_0)$)) $\neq \emptyset$.

Zip(𝔸(K, w_0)) $\notin$ L(H) 当且仅当存在一个动作序列 $p = a_0 \dots a_n$，它属于 Zip(𝔸(K, w_0))，即 Zip(𝔸(K, w_0))$[p] \neq \emptyset$，且 Zip(𝔸(K, w_0))$[p] \not\models$ H$[p]$。不失一般性，我们可以假设只有最后一个切片不满足 H 中对应的节点。令 H$[p] = q_0 a_0 \dots q_n a_n$。那么，Zip(𝔸$(K, w_0)$)$[p] \not\models$ H$[p]$ 当且仅当对于所有 $0 \leq j < n$，$(\star)$，Zip(𝔸(K, w_0))$(a_0 \dots a_{j-1}, a_j) \models q_j$，而 Zip(𝔸$(K, w_0)$)$(a_0 \dots a_{n-1},$ $a_n) \not\models q_n$。根据引理 20，Zip(𝔸(K, w_0))$[p] \neq \emptyset$ 在 Slice(𝔸(K, w_0)) 中定义了一条唯一的路径，$\mathbb{K}_0 a_0 \dots \mathbb{K}_n a_n$ 它保留了由𝔸(K, w_0)定义的路径切片。因此，根据 $(\star)$、Join(H $\mathbb{A}(K, w_0)$) 的定义以及引理 20，$(\mathbb{K}_0, q_0) a_0 \dots (\mathbb{K}_n, q_n) a_n$ 在 Join(H $\mathbb{A}(K, w_0)$) 中定义了一次接受运行。注意，$($ $\mathbb{K}_n, q_n)$ 属于 Join(H $\mathbb{A}(K, w_0)$)（且它是终态），因为 Zip(𝔸(K, w_0))$(a_0 \dots a_{n-1}, a_n) \not\models q_n$。◀

以下定理将本节中的所有结果汇总在一起。

▶**定理 23.** 带动作标记的带初始状态的克里普克结构上的超节点自动机模型检测是可判定的。

证明。 我们已经看到，在开放 Kripke 结构上对超节点逻辑进行模型检测是可判定的（定理 17）。计算 $\mathrm{Join}(\mathcal{H}\ K, \mathbb{A}, w_0)$ 也是可判定的。主要挑战在于对 $\mathbb{A}(K, w_0)$进行切片。注意，$\mathbb{A}(K, w_0)$中可能出现的状态数目有限，因为它们都是克里普克结构 K的子结构。◀

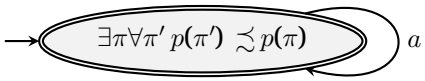
在动作标记的 Kripke 结构上对超节点自动机进行模型检测，最困难的部分是检查所有超节点的公式。因此，与超节点逻辑一样，超节点自动机的模型检测运行时间也主要取决于命题变量的数量，其依赖关系为双指数级。此外，我们的模型检测算法对克里普克结构的大小和超节点自动机的大小均呈单指数级依赖。

▶**推论 24。** 设 A 为一个动作集合，X 为一个 m 命题变量集合。设(K, w_0) 为定义在 X上的一个带初始状态的克里普克结构，$\mathbb{A}$ 为 K 在 A上的一个动作标记。设 H 为定义在 X和 A上的一个超节点自动机。检查$\mathrm{Zip}(K, \mathbb{A}, w_0)) \in L(H)$ 是否成立的时间复杂度为 $O(|H| \cdot 2^{|A|+n \cdot |K|^m})$，其中 n 是 H 中任意超节点公式所包含的轨迹量词的最大数量。

5 相关工作

用于表达异步超属性的首个逻辑，是对 μ- 演算的扩展，加入了对轨迹的显式量化，称为 Hμ [11]。Hμ 中不含轨迹量词的公式，在表达能力上等价于 [11] 中引入的奇偶多带交替异步字自动机（AAWA）。这两种形式体系的模型检测问题都具有高度不可判定性。不可判定性源于比较不同轨迹中的位置。

图 7 超节点自动机，用于说明：对于由重复动作 a 所诱导的轨迹集合中的每个切片，都存在一条（可能不同的）轨迹，用以描述命题变量 p 在当前切片内的异步进展。



轨迹之间的距离可以任意远，轨迹数量也不受限制。当两个维度之一（被比较位置之间的距离，或轨迹数量）有了明确的有限界限时，模型检测就变为可判定 [11]。相比之下，我们通过完全不同的方式实现可判定性：将不同程序变量的进程解耦（异步性），同时允许通过自动机层面的转移重新同步。

在 Hμ 公式中，轨迹量词总是位于时间算子之前，而超节点自动机则允许时间算子与轨迹量词之间进行受限形式的量词交替。具体而言，自动机层级的转移对应于最外层时间算子；这些算子位于超节点逻辑公式的轨迹量词之前，而公式中经停顿约简的前缀化关系则对应于最内层时间量词。正如 [3]，所示，改变轨迹量词与时间量词的顺序，可能会使同步超属性变为异步，反之亦然。因此，我们推测 Hμ 与超节点自动机的表达能力互不可比。以图 7 所示的超节点自动机为例，它规定：在由重复动作 a 所诱导的每个切片内，命题变量 p 的异步进展都由一条有限轨迹 π 完整描述。每个新切片都可以有一条不同的轨迹 π，用以见证 p 的异步进展。每个切片中的轨迹长度无界，而且由于我们不知道 a 会重复多少次，切片数量也无界。因此，我们不知道需要多少个最外层存在轨迹量词，才能保证每个切片都有单独的轨迹见证。因此，我们推测，图 7 中超节点自动机所指定的超属性无法用 Hμ 表达。

近期也有人探索 HyperLTL 的各种扩展，以支持异步超属性。停顿HyperLTL（HyperLTL$_S$）和上下文HyperLTL（HyperLTL$_C$）均在 [8]，中提出，通过引入新算子扩展了 HyperLTL。停顿 HyperLTL 允许使用描述不可区分时间序列的线性时序逻辑公式来标注时序算子。在这种语义下，轨迹中的下一个可观测步，是标注当前时间算子的线性时序逻辑公式真值发生变化的下一个时间点。因此，不同轨迹上的时间并非同步推进，因为时间推进取决于线性时序逻辑公式的赋值在一条轨迹中如何变化。上下文 HyperLTL 引入了一元模态，并以一组轨迹变量作为参数，称为上下文。上下文中的轨迹变量指定了在求值时序子公式时哪些轨迹会推进（不在上下文中的轨迹将停顿）。异步HyperLTL（A-HyperLTL）[4] 通过对轨迹进行量化扩展了 HyperLTL。与上下文类似，轨迹指定了每个求值步骤中推进的轨迹。尽管这些 HyperLTL 扩展的模型检测问题均不可判定，但作者找到了能够支持某些异步超属性的语法片段。这些可判定片段采用了与 Hμ 的可判定部分类似的限制。HyperLTL$_S$、HyperLTL$_C$ 和 A-HyperLTL 均包含于 Hμ [7]。正如我们所论证的，图 7 中的超节点自动机无法用 Hμ 表示，因此也无法用所提出的三种 HyperLTL 异步扩展中的任何一种表示。

Krebs 等人 [13] 提出在所谓的团队语义下重新诠释线性时序逻辑。团队语义处理的是变量赋值集合，作者还提出了同步和异步两种形式。他们证明，在异步团队语义下，线性时序逻辑的表达能力与全称 HyperLTL 相同（其中所有轨迹量词都是全称量词）。由于超节点逻辑允许对轨迹进行存在量化，因此我们的方法与之正交，且表达能力互不可比。

在 [5] 中，作者介绍了 HyperATL*。它扩展了交替时间时序逻辑 [1]，加入了将策略绑定到轨迹变量的策略量词，以及一种用于并行求解博弈的显式结构。HyperATL* 可用于规约策略超属性。由于我们关注的是线性时间异步超属性，而非策略超属性，因此这项工作与我们的研究方向正交。

尽管已有许多基于逻辑的规范语言被提出，用于表达异步超属性，但目前缺少基于自动机的方法来规约这类属性。请注意，AAWA [11] 不支持对轨迹变量进行显式量化。 [6] 的有限字超自动机通过对有限自动机添加轨迹显式量化的前缀化，在这一方向上迈出了一步，但它们仅适用于同步超属性。

6 结论

我们提出了一种用于规约并发系统超属性的新形式体系。该形式体系将不同执行轨迹之间的同步与异步比较结合起来：前者通过规范自动机中带动作标记的转移来表达，后者则通过标记规范自动机状态的超节点逻辑公式来表达。由此，超节点自动机的规范语言能够在不同长度的轨迹片段上交替表达异步要求，并设置同步点。与以往用于规约异步超属性的形式体系不同，超节点自动机完全支持自动验证。我们针对超节点自动机的模型检测算法基于一种全新的技术，该技术引入了无停顿自动机及其上的操作，从而很好地展现了自动机理论方法在验证中的强大能力。

除了具有可判定验证问题之外，超节点自动机还代表了一种真正新颖且实用的规范语言。我们通过以下方式证明了这一点：使用超节点逻辑规约观察确定性的若干已发表变体；使用超节点自动机规约信息脱密；并论证超节点自动机形式体系在表达能力上与近年来为规约异步超属性而提出的各种超逻辑不可比。

通过引入复合类型的变量（例如布尔数组），可以微调轨迹比较中的异步性与同步性之间的界限；例如，这类变量可用于耦合多线程程序中各线程的变量。此类字母表变体对超节点逻辑和超节点自动机的影响有待未来研究。当前工作直接引出了许多其他研究课题；即使这些课题看似简单，也仍需进一步探究，包括研究具有部分且非确定性转移关系的超节点自动机，以及具有无穷接受条件的超节点自动机（例如超节点 Büchi 自动机）；还包括扩展针对超性质规范的形式表达能力研究，将超节点逻辑和自动机纳入其中，并提出用于求解超节点逻辑和自动机的经典判定问题的算法

而不仅限于模型检测（例如可满足性和空性）。此外，无停顿自动机在其他异步验证场景中的适用性（不一定涉及超属性）也是一个值得探讨的问题。

参考资料

1 Rajeev Alur、Thomas A. Henzinger 和 Orna Kupferman。交替时间时序逻辑。《ACM杂志》，49(5):672–713，2002年。 doi:10.1145/585265.585270。2 Aslan Askarov 和 Andrei Sabelfeld。渐进式发布：统一去密级化、加密和密钥发布策略。载于《2007年IEEE安全与隐私研讨会》（SP '07），第207–221页，2007年。 doi:10.1109/SP.2007.22。3 Ezio Bartocci、Thomas Ferrère、Thomas A. Henzinger、Dejan Nickovic 和 Ana Oliveira da Costa。顺序信息流的不同类型。载于 VMCAI 2022 论文集：第23届验证、模型检测与抽象解释国际会议，计算机科学讲义第13182卷，第1–19页。施普林格，2022年。 doi:10.1007/978-3-030-94583-1_1。4 Jan Baumeister、Norine Coenen、Borzoo Bonakdarpour、Bernd Finkbeiner 和 César Sánchez。异步超属性的时序逻辑。载于 Alexandra Silva 和 K. Rustan M. Leino 编，《计算机辅助验证》（CAV），第694–717页。施普林格国际出版，2021年。 doi:10.1007/978-3-030-81685-8_33。5 Raven Beutner 和 Bernd Finkbeiner。策略超属性的时序逻辑。载于第32届并发理论国际会议（CONCUR 2021）论文集，莱布尼茨国际信息学会议论文集（LIPIcs）第203卷，第24:1–24:19页，德国达格施图尔，2021年。达格施图尔城堡—莱布尼茨信息学中心。 doi:10.4230/LIPIcs.CONCUR.2021.24。6 Borzoo Bonakdarpour 和 Sarai Sheinvald。有限字超语言。载于 LATA 2021 论文集：第15届语言与自动机理论及应用国际会议，计算机科学讲义第12638卷，第173–186页。施普林格，2021年。 doi:10.1007/978-3-030-68195-1。7 Laura Bozzelli、Adriano Peron 和 Cesar Sanchez。异步超属性时序逻辑的表达能力与可判定性。载于第33届并发理论国际会议（CONCUR 2022）论文集（即将出版），2022年。8 Laura Bozzelli、Adriano Peron 和 César Sánchez。HyperLTL的异步扩展。载于2021年第36届ACM/IEEE计算机科学逻辑年会（LICS），第1–13页，2021年。 doi:10.1109/LICS52264.2021.9470583。9 Michael R. Clarkson、Bernd Finkbeiner、Masoud Koleini、Kristopher K. Micinski、Markus N. Rabe 和 César Sánchez。超属性时序逻辑。载于 POST 2014 论文集：第三届安全与信任原理国际会议，计算机科学讲义第8414卷，第265–284页。施普林格，2014年。 doi:10.1007/978-3-642-54792-8。10 Michael R. Clarkson 和 Fred B. Schneider。超属性。《计算机安全杂志》，18(6):1157–1210，2010年。 doi:10.3233/JCS-2009-0393。11 Jens Oliver Gutsfeld、Markus Müller-Olm 和 Christoph Ohrem。异步超属性的自动机与不动点。《ACM程序设计语言论文集》，5(POPL):1–29，2021年。 doi:10.1145/3434319。12 M. Huisman、P. Worah 和 K. Sunesen。观察确定性的时序逻辑刻画。载于第19届IEEE计算机安全基础研讨会（CSFW'06），第13页–3，2006年。 doi:10.1109/CSFW.2006.6。13 A Krebs、A Meier、J Virtema 和 M Zimmermann。用于超属性规约与验证的团队语义。莱布尼茨国际信息学会议论文集，LIPIcs，117，2018年。14 Andrei Sabelfeld 和 David Sands。去密级化：维度与原则。《计算机安全杂志》，17(5):517–548，2009年。

15 Tachio Terauchi。观察确定性的类型系统。载于 *2008*年*IEEE*计算机安全基础第*21*届研讨会，第287–300页，2008年。 doi:10.1109/CSF.2008.9。16 S. Zdancewic 和 A.C. Myers。并发程序安全中的观察确定性。载于第*16*届*IEEE*计算机安全基础研讨会，*2003*年。会议论文集。，第29–43页，2003年。doi:10.1109/CSFW.2003.1212703。