使用 Maude 和 SMT 求解对时间 Petri 网进行符号分析与参数综合*

Jaime Arias[1], Kyungmin Bae[2], Carlos Olarte[1], Peter Csaba Ölveczky[3], Laure Petrucci[1], 和 Fredrik Rømming[4]

[1] LIPN，CNRS UMR 7030，巴黎北索邦大学，法国[2] 浦项科技大学，韩国[3] 奥斯陆大学，挪威[4] 剑桥大学，英国

arXiv:2303.08929v1 [cs.LO] 15 Mar 2023

摘要。 带抑制弧的参数化时间 Petri 网（PITPN）通过允许在发射界限中使用参数，为定时系统提供灵活性。本文提出并证明了 PITPN 的一种具体重写逻辑语义和一种符号重写逻辑语义的正确性。我们展示了如何结合 Maude 与 SMT 求解，对 PITPN 进行可靠且完备的形式化分析。我们提出了一种用于符号可达性的新通用折叠方法；只要 PITPN 的参数化状态类图是有限的，该方法就会终止。我们说明，当前先进的 PITPN 工具 Roméo 所支持的几乎所有形式化分析和参数综合，都可以在 Maude 中结合 SMT 完成。此外，我们还支持从参数化 初始标识出发的分析和参数综合，以及完整的线性时序逻辑模型检验和使用用户自定义执行策略的分析。针对三个基准测试的实验表明，我们的方法在许多情况下优于 Roméo。

关键词： 参数化定时 Petri 网 · 语义 · 重写逻辑 · Maude · 可满足性模理论 · 参数综合 · 符号可达性分析

1 引言

时间(d) Petri 网 [27,49,19] 已被广泛用于实时系统建模。在时间 Petri 网中，发射条件由时间区间给出，使能变迁必须在该区间内发射。然而，在系统设计中，我们往往事先不知道关键系统参数的具体值，并希望找到能使系统按预期行为的参数值。带抑制弧的参数化时间 *Petri* 网 （PITPNs） [47,18,2,25]将时间 Petri 网扩展到这样一种情形：变迁可发射时间界限未知或仅部分已知。

PITPNs 的建模和形式化分析（包括综合能使系统满足期望性质的参数值）可借助先进工具 Roméo [26], 实现。该工具已应用于多种领域，包括振荡生物系统 [3], 、航空视频跟踪系统 [44], 和分布式软件调试 [17]。Roméo 支持对可达性（某个标识是否可达？）、活性（某个标识是否会在所有行为中到达？）、有时间界限的“直到”以及有界响应（每个 P-标识是否会在时间 Δ内后接一个 Q-标识？）进行分析和参数综合，且均从具体初始标识出发。Roméo 不支持若干期望功能，包括：

– 更广泛的系统属性，例如完整（即嵌套）时序逻辑。– 从参数化 初始标识开始，并综合出使系统满足期望性质的初始标识。– 使用用户定义的执行策略进行分析。例如，当变迁 t 和 t' 同时使能时，如果我总是选择触发 t，会发生什么？通常可以手动更改模型 来分析系统在此类情形下的行为，但这样做既费力又容易出错。

* 由 CNRS INS2I 项目 ESPRiTS 和 PHC Aurora AESIR 资助。

– 为 PITPN 提供一个“试验平台”，以便快速开发、测试和评估不同的分析方法与算法。Roméo 对此支持不足；它是一款高性能工具，使用 C++ 实现了专用算法。

PITPN 不支持大型分布式系统所需的许多特性，例如用户定义数据类型和函数，如彩色Petri网 [19]。

重写逻辑 [28,29]——由 Maude 语言和工具 [16]，以及面向实时系统的 Real-Time Maude [40,34] 支持——是一种适用于分布式系统和实时系统的表达力丰富的逻辑。在重写逻辑中，任何可计算数据类型都可以表示为（代数）等式规范，而系统的动态行为则通过对项（表示状态）应用重写规则来描述。凭借其表达力，Real-Time Maude 已成功应用于许多大型且复杂的实时系统，范围涵盖 50 页的主动网络和 IETF协议 [41,24],、涉及区域、角度等内容的先进无线传感器网络算法 [42],、采用无界队列的调度算法 [36], 、飞机转弯算法 [7]，等，超出了大多数流行实时系统形式化方法的适用范围。其表达力也使 Real-Time Maude 成为工业建模语言（子集）的实用语义框架和形式化分析后端 [35,1,8,33]。

这种表达能力是有代价的：大多数分析问题通常都是不可判定的。Real-Time Maude 使用显式状态分析，只访问一些 时间点。因此，并非所有可能的系统行为都会得到分析 （对于稠密时间定义域而言），所以在许多情况下，这种分析并不可靠 [38]。

本文利用近期将 SMT求解 集成到 Maude 中的成果，来解决上述第一个问题（为 PITPN 提供更多功能），并朝着解决第二个问题（开发适用于基于重写逻辑的实时系统的可靠且完备的分析方法）迈出第二步。

Maude 与 SMT求解 相结合（例如 Maude-SE 工具 [51]，中实现的方式），使我们能够对“状态” $\phi \mid\mid t$ 进行符号重写，其中，项 t 是包含变量的状态模式，而 ϕ 是用于限制这些变量可能取值的 SMT约束。

第 3 节以“Real-Time Maude 风格” [39]为（实例化的）PITPN 提供了“具体的”重写逻辑语义。在 PITPN 这类稠密时间设置中，该模型不可执行。第 4 节展示了如何对具体网进行（通常并不可靠的）基于时间采样的分析，其中时间以离散步递增，从而快速试验参数的不同取值。

第 5 节为参数化PITPN 给出了 Maude-with-SMT 语义，并展示了如何使用 Maude with SMT 对这类网进行（可靠的）符号分析。然而，现有的符号可达性分析方法（包括符号状态的“折叠”）即使在 PITPN 的状态类图有限（因此 Roméo 分析能够终止）时，也可能无法终止。因此，我们开发并实现了一种新的方法，用于在 Maude-with-SMT 中为可达性分析“折叠”符号状态，并证明只要 PITPN 的状态类图有限，这种新的可达性分析方法就会终止。

在第 5 节和第 6 节中，我们展示了如何使用 Maude-with-SMT 执行多种形式化分析和参数综合，包括无界可达性分析和有时限可达性分析。我们在第 6 节中展示了 Roméo 支持的所有分析方法（有一个小小的例外：某些时序公式中的时间界限不能是参数）也都可以在 Maude-with-SMT 中执行。此外，我们支持针对标识和“变迁时钟”的状态性质、对参数化初始标识进行分析和参数综合、检验完整（即嵌套）时序逻辑公式，以及按照用户定义的执行策略进行分析，如第 6 节所示。我们直接使用 Maude 的元编程功能，在 Maude 中对这些方法进行形式化和实现。这使得为 PITPN 开发新的分析方法变得非常容易。

这项工作也是我们探索的第二步，旨在为 Real-Time Maude 中的稠密时间实时系统开发可靠且完备的形式化分析方法。在第3节中介绍“标准”的 Real-Time Maude 风格的具体语义，以及在第5节中介绍符号语义，其中一个原因是探索如何将 Real-Time Maude 模型转换为 Maude-with-SMT 模型，以进行符号分析。在这一探索的第一步中，我们研究了适用于简单得多的参数化定时自动机（PTA）的符号重写方法 [4]。在 [4] 中，我们为每个自动机都指定一种新的重写理论，而在本文中，我们为所有PITPN 指定一个单一的重写理论（“解释器”）。此外，与本文中的模型不同，PTA 不需要方程或用户定义函数。最后，已知的折叠方法足以处理 PTA，而对于 PITPN，我们则必须开发更强的折叠方法。

在第7节中，我们针对三个 PITPN 对 Roméo 和我们结合SMT的Maude方法进行了基准测试。颇令人意外的是，在许多情况下，我们的高层原型性能优于 Roméo。我们还发现，在某些情况下，Roméo 给出的答案是“可能”，而 Maude 找到了可行解；此外，即使存在解，Roméo 有时也无法合成参数。

所有包含分析命令的可执行 Maude 文件、将 Roméo 文件转换为 Maude 的工具，以及基准测试数据均可在 [5] 获取。

2 预备知识

本节介绍迁移系统、双模拟 [14], 带抑制弧的参数化时间 Petri 网[47],重写逻辑 [28], 模SMT重写 [45] 以及 Maude/Maude-SE [15,51]。

一个 迁移系统 $\mathcal{A}$ 是一个三元组 $(A, a_0, \to_{\mathcal{A}})$，其中 A 是一个状态集合， $a_0 \in A$ 是初始状态，而$\to_{\mathcal{A}} \subseteq A \times A$ 是一个迁移关系。如果从 a_0 出发，经由 $\to_{\mathcal{A}}$ 可达的状态集合有限，则称 $\mathcal{A}$ 是有限的。关系 $\sim \subseteq A \times B$ 是从 $\mathcal{A}$ 到 $\mathcal{B} =$ 的双模拟 [14] $(B, b_0, \to_{\mathcal{B}})$，当且仅当：(i) $a_0 \sim b_0$；且 (ii) 对所有a, b ，若满足 $a \sim b$：如果 $a \to_{\mathcal{A}} a'$ ，则存在一个 b' 满足 $b \to_{\mathcal{B}} b'$ 且 $a' \sim b'$；反之，如果 $b \to_{\mathcal{B}} b''$，则存在一个 a'' 满足 $a \to_{\mathcal{A}} a''$ 且 $a'' \sim b''$。

2.1 带抑制弧的参数化时间Petri网（PITPN）。

我们回顾 [47] 中的定义。 $\mathbb{N}$、 $\mathbb{Q}_+$ 和 $\mathbb{R}_+$ 分别表示自然数、非负有理数和非负实数。在本文中，我们假设存在一个有限的时间参数集 $\Lambda = \{\lambda_1, \ldots, \lambda_l\}$。参数赋值 π 是一个函数 $\pi\colon \Lambda \to \mathbb{R}_+$。$\Lambda$ 上的一个（线性）不等式是形如 $\sum_{1 \le i \le l} a_i \lambda_i \prec b$ 的表达式，其中 $\prec \in \{<, \le, =, \ge, >\}$ 和 $a_i,\ b \in \mathbb{R}$。约束是此类不等式的合取。 $\mathcal{L}(\Lambda)$ 表示 Λ 上所有约束构成的集合。若参数赋值 π 满足约束 $K \in \mathcal{L}(\Lambda)$ ，记作 $\pi \models K$，则将约束中每个参数 λ 替换为 $\pi(\lambda)$ 后，所得表达式的计算结果为真。 $\mathbb{R}_+$ 的区间 I 称为 $\mathbb{Q}_+$ 区间，当且仅当其左端点 ${}^{\uparrow}I$ 属于 $\mathbb{Q}_+$ ，且右端点 $I^{\uparrow}$ 属于 $\mathbb{Q}_+ \cup \{\infty\}$。我们用 $\mathcal{I}(\mathbb{Q}_+)$ 表示 $\mathbb{Q}_+$ 区间的集合。参数化时间区间是一个函数 $I\colon {\mathbb{Q}_+}^{\Lambda} \to \mathcal{I}(\mathbb{Q}_+)$ ，它为每个参数赋值关联一个 $\mathbb{Q}_+$ 区间。 Λ 上的参数化时间区间集合记作 $\mathcal{I}(\Lambda)$。

定义 1（PITPN）。一个带抑制弧的参数化时间Petri网是一个元组$\mathcal{N} = \langle P, T, \Lambda, {}^{\bullet}(\circ), (\circ)^{\bullet}, {}^{\circ}(\circ)\ M_0,\ J,\ K_0 \rangle$ 其中

– $P = \{p_1, \ldots, p_m\}$ 是一个非空有限集合（由库所组成），–
$T = \{t_1, \ldots, t_n\}$ 是一个非空有限集合（由变迁组成），且 $P \cap T = \emptyset$，–
$\Lambda = \{\lambda_1, \ldots, \lambda_l\}$ 是一个有限参数集合，– ${}^{\bullet}(.) \in [T \to \mathbb{N}^P]$ 是后向关联函数，–
$(.)^{\bullet} \in [T \to \mathbb{N}^P]$ 是前向关联函数，– ${}^{\circ}(.) \in [T \to \mathbb{N}^P]$ 是抑制函数，–
$M_0 \in \mathbb{N}^P$ 是初始标识，– $J \in [T \to \mathcal{I}(\Lambda)]$ 为每个变迁指定一个参数化时间区间，且– $K_0 \in \mathcal{L}(\Lambda)$是关于 Λ的初始约束。

如果 $\Lambda = \emptyset$ 那么 $\mathcal{N}$ 是一个（非参数化的）带抑制弧的时间Petri网（*ITPN*）。

$\mathcal{N}$的标识是一个元素 $M \in \mathbb{N}^P$，其中 $M(p)$表示库所 p中的令牌数。 $\pi(\mathcal{N})$表示 ITPN，其中 PITPN $\mathcal{N}$ 中每个出现的 λ_i 都已替换为 $\pi(\lambda_i)$，对应的参数赋值为 π。例如，图1b中的 ITPN 对应于图 1a中的 PITPN，其中参数被赋值为 $\pi = \{\lambda_1^- \to 5, \lambda_1^+ \to 6, \lambda_2^- \to 3, \lambda_2^+ \to 4, \lambda_3^- \to 1, \lambda_3^+ \to 2\}$。

PITPN $\mathcal{N}$ 的具体语义 是根据具体 ITPN $\pi(\mathcal{N})$ 来定义的，其中 $\pi \models K_0$。如果变迁 t 在 M 中使能，则称其为使能变迁： $M \ge {}^{\bullet}t$ （对于 t 的每个输入库所， M 中的令牌数都大于或等于该库所与 t之间的弧上的值）。如果库所



(a) 一个 PITPN $\mathcal{N}$。 (b) ITPN $\pi(\mathcal{N})$。

图1：一个 PITPN 及其赋值。

连接到其某条抑制弧，且其标识中的令牌数至少等于该抑制弧的权重，则称该变迁受抑制。若变迁 t 已使能且未受抑制，则称其为使能 状态。标识 M中使能变迁和受抑制变迁的集合分别记为$Enabled(M)$和$Inhibited(M)$。如果变迁 t已连续使能至少 $^{\uparrow}J$时间（t），且不计其受抑制期间的时间，则称其为可发射 。如果变迁 t在 M中通过变迁 t_f的发射，在所得标识 $M' = M - {}^{\bullet}t_f + t_f^{\bullet}$中使能，而在 $M - {}^{\bullet}t_f$中未使能，则称其为新近使能：

$$NewlyEnabled(t, M, t_f) = ({}^{\bullet}t \leq M - {}^{\bullet}t_f + t_f^{\bullet}) \wedge ((t = t_f) \vee \neg({}^{\bullet}t \leq M - {}^{\bullet}t_f)).$$

新使能$(M,\ t_f)$ 表示在 M中触发 t_f后新使能的变迁。

ITPN 的语义定义为一个迁移系统，其状态为 $(M,\ I)$，其中 M 是一个标识，I 是一个函数，它将 M 中每个使能的变迁映射到一个时间区间；该迁移系统有两类变迁：时间 变迁（时间流逝）和离散变迁（网中的变迁被触发时发生）。

定义 *2*（*ITPN* **[47]**的语义）。*ITPN* $\pi(\mathcal{N})$ 的动态行为由迁移系统 $\mathcal{S}_{\pi(\mathcal{N})} = (\mathcal{A}, a_0, \rightarrow)$ 定义，其中：$\mathcal{A} = \mathbb{N}^P \times [T \to \mathcal{I}(\mathbb{Q})]$，$a_0 = (M_0, J)$ 且 $(M, I) \rightarrow (M', I')$，当且仅当存在 $\delta \in \mathbb{R}_+$、$t \in T$ 和状态 (M'', I'')，使得 $(M, I) \xrightarrow{\delta} (M'', I'')$ 且 $(M'', I'') \xrightarrow{t} (M', I')$，对于以下关系：

- *the* time transition relation, *defined* $\forall \delta \in \mathbb{R}_+$ *by:*
  $(M, I) \xrightarrow{\delta} (M, I')$ *iff* $\forall t \in T$*:*
  $$\begin{cases} I'(t) = \begin{cases} I(t) \ \textit{if } t \in Enabled(M) \ \textit{and } t \in Inhibited(M) \\ {}^{\uparrow}I'(t) = \max(0, {}^{\uparrow}I(t) - \delta), \ \textit{and } I'(t)^{\uparrow} = I'(t)^{\uparrow} - \delta \ \textit{otherwise} \end{cases} \\ M \geq {}^{\bullet}(t) \implies I'(t)^{\uparrow} \geq 0 \end{cases}$$
- *the* discrete transition relation, *defined* $\forall t_f \in T$ *by:* $(M, I) \xrightarrow{t_f} (M', I')$ *iff*
  $$\begin{cases} t_f \in Enabled(M) \wedge t_f \notin Inhibited(M) \wedge M' = M - {}^{\bullet}t_f + t_f^{\bullet} \wedge {}^{\uparrow}I(t_f) = 0 \\ \forall t \in T, I'(t) = \begin{cases} J(t) \ \textit{if } NewlyEnabled(t, M, t_f) \\ I(t) \ \textit{otherwise} \end{cases} \end{cases}$$

PITPN 的符号语义在 [2] 中定义为一个基于状态类的迁移系统（$\mathbb{N}^P \times \mathcal{L}(\Lambda), (M_0, K_0) \Rightarrow$），即由成对元素 $c = (M, D)$ 构成，其中包括一个标识 M 和关于 Λ的约束 D。变迁的发射会像具体语义中那样产生新的标识，同时也会捕获因等待该变迁发射而经过的时间所引入的新约束。例如，对于图 1a 中的 PITPN，初始状态类为 $(\{A, B\}, \lambda_1^- \leq \lambda_1^+ \wedge \lambda_2^- \leq \lambda_2^+ \wedge \lambda_3^- \leq \lambda_3^+)$。发射变迁 t_1时，等待 t_1 变为可发射所经过的时间必须确保其他变迁（本例中为 t_3 ）不会错过其截止时间。因此，我们得到一个附加不等式$\lambda_1^- \leq \lambda_3^+$，而发射 t_1 后得到的新状态类为 $(\{C, B\}, \lambda_1^- \leq \lambda_1^+ \wedge \lambda_2^- \leq \lambda_2^+ \wedge \lambda_3^- \leq \lambda_3^+ \wedge \lambda_1^- \leq \lambda_3^+)$。详情请参见 [2] 。

2.2 使用 SMT 和 Maude 进行重写

重写理论。一个 重写理论 [28] 是一个元组 $\mathcal{R} = (\Sigma, E, L, R)$，满足 以下条件

– Σ 是一个函数签名，声明了排序、子排序和函数符号；– E 是一个由以下形式的方程组成的集合：$t = t'$若 ψ，其中 t 和 t' 是相同排序的项，ψ 是方程的合取；– L 是一个标签集合；以及– R 是一个由以下形式的重写规则组成的集合：$l : q \longrightarrow r$ 若 ψ，其中 $l \in L$ 是一个标签，q 和 r 是相同排序的项，且 ψ 是方程的合取。

$T_{\Sigma,\, s}$ 表示排序为 s 的基项（即 不含变量的项）集合，而 $T_\Sigma(X)_s$ 表示在变量集 X 上排序为 s 的项集合。$T_\Sigma(X)$ 和 T_Σ 分别表示所有项和基项。替换 $\sigma : X \to T_\Sigma(X)$ 将每个变量映射到相同排序的项，而 $t\sigma$ 表示通过同时将项 t 中的每个变量 x 替换为 $\sigma(x)$ 所得到的项。替换 σ的定义域为 $dom(\sigma) = \{x \in X \mid \sigma(x) \neq x\}$，并假定该定义域是有限的。

若存在一条规则 $l : q \longrightarrow r$ 若 ψ、t 的一个子项 u 和一个替换σ，使得 $u = q\sigma$（模方程），且 t' 是将 u 替换为 $r\sigma$后由 t 得到的项，并且对于 ψ中的每个 $v = v'$，$v\sigma = v'\sigma$都成立，则称单步重写 $t \longrightarrow_{\mathcal{R}} t'$ 成立。我们用 $\longrightarrow^*_{\mathcal{R}}$ 表示 $\longrightarrow_{\mathcal{R}}$的自反传递闭包。

若重写理论 $\mathcal{R}$ 满足以下条件，则称其为顶层：存在一个排序 状态，它位于子排序偏序某个连通分量的顶端，并且对于每条规则 $l : q \longrightarrow r$ 若 ψ，q 和 r 的顶层排序都是状态，且没有算子以状态 或其任何子排序作为参数排序。

基于 *SMT* 的重写 *[45]*。对于函数签名 Σ 和一组方程 E，内置理论 $\mathcal{E}_0$ 是一种函数签名为 $\Sigma_0 \subseteq \Sigma$的一阶理论，其中：(1) Σ_0 中的每个排序 s 在 Σ中都是极小的；(2) 对于 $\Sigma \setminus \Sigma_0$中的每个算子$f$：$s_1 \times \cdots \times s_n \to s$，$s \notin \Sigma_0$；(3) f在 Σ_0中没有其他子排序重载类型。假定可使用与 (Σ, E) 一致的 SMT 理论 $\mathcal{T}_{\mathcal{E}_0}$，判定约束 $\mathcal{E}_0$ 的可满足性；也就是说，对于 Σ_0项 t_1 和 t_2，若 $t_1 = t_2$ 模 E成立，则 $\mathcal{T}_{\mathcal{E}_0} \models t_1 = t_2$。

一个受约束项是一个二元组 $\phi \parallel t$，由约束 ϕ（属于 $\mathcal{E}_0$）和项 t（属于 T_Σ）组成，其中（X_0）的变量 $X_0 \subseteq X$属于 $\mathcal{E}_0$ [45,10] 中的内置排序。受约束项 $\phi \parallel t$ 以符号方式表示模式t 的所有实例，且这些实例满足 ϕ：$[\![\phi \parallel t]\!] = \{t' \mid t' = t\sigma$（模 E），并且 $\mathcal{T}_{\mathcal{E}_0} \models \phi\sigma$ 对于基项 σ：$X_0 \to T_{\Sigma_0}\}$.

对于一个 Σ-项 $t \in T_\Sigma(X)$，其内建项抽象是一个二元组（t°，σ°），由项 $t^\circ \in T_{\Sigma\setminus\Sigma_0}(X)$ 和替换 σ°：$X_0 \to T_{\Sigma_0}(X_0)$ 组成，且满足 $t = t^\circ\sigma^\circ$，并且 t° 在 X_0中不包含重复变量。t 中任何非变量的内置子项都会被替换为互不相同的内置变量。$\Psi_{\sigma^\circ} = \bigwedge_{x \in dom\,(\sigma^\circ)} x = x\sigma^\circ$。设 $\phi \parallel t$ 为一个受约束项，(t°, σ°) 为 t的内建项抽象。若$dom(\sigma^\circ) \cap$ 变量集合$(\phi \parallel t) = \emptyset$，则 $[\![\phi \parallel t]\!] = [\![\phi \wedge \Psi_{\sigma^\circ} \parallel t^\circ]\!]$ [45]

设 $\mathcal{R}$ 是一个顶层理论，对于每条规则 $l : q \longrightarrow r$若 ψ，左侧 q 中未出现的额外变量均属于 X_0，且 ψ 是内置理论 $\mathcal{E}_0$ 中的约束。当且仅当存在一条规则 $l : q \longrightarrow r$ 若 ψ 和一个替换 $\sigma : X \to T_\Sigma(X_0)$，使得 (1) $t = q\sigma$ 和 $t' = r\sigma$（模方程），(2) $\mathcal{T}_{\mathcal{E}_0} \models (\phi \wedge \psi\sigma) \Leftrightarrow \phi'$，且 (3) ϕ' 是 $\mathcal{T}_{\mathcal{E}_0}$-可满足的，单步符号重写 $\phi \parallel t \rightsquigarrow_{\mathcal{R}} \phi' \parallel t'$ 成立。我们用 $\rightsquigarrow^*_{\mathcal{R}}$ 表示 $\rightsquigarrow_{\mathcal{R}}$ 的自反传递闭包。

受约束项上的符号重写 以符号方式表示一组（可能无限的）系统转换。若 $\phi_t \parallel t \rightsquigarrow^* \phi_u \parallel u$ 是一次符号重写，则存在一次“具体”重写 $t' \longrightarrow^* u'$，满足 $t' \in [\![\phi_t \parallel t]\!]$ 和 $u' \in [\![\phi_u \parallel u]\!]$。反过来，对于任意满足 $t' \in [\![\phi_t \parallel t]\!]$ 的具体重写 $t' \longrightarrow^* u'$，都存在一次符号重写 $\phi_t \parallel t \rightsquigarrow^* \phi_u \parallel u$，满足 $u' \in [\![\phi_u \parallel u]\!]$。

Maude。Maude [16] 是一种支持规约和分析重写理论的语言和工具。下面总结其语法：

导入 理论 R。 --- 导入理论 R排序 S ... Sk。 --- 排序 S1,..., Sk 的声明子种
类 S1 < S2。 --- 子种类关系变量集合 X1 ... Xm: S。 排序 S 的逻辑变
量算子 f: S1 ... Sn -> S。 --- 算子 S1 x... x Sn -> S

算子 c: -> T。 --- 排序 T 的常量 c方程 t =t'。 --- 方程条件方程 t =t' 如果 c。 --- 条件方程条件重写规则 [l] : q => r 如果 c。 --- 条件重写规则

Maude 提供多种分析方法，包括计算项 t (命令red t) 的范式、通过重写进行仿真（rew t），以及按照给定策略进行重写（srew t using str）。基本策略包括 $r[\sigma]$ （使用可选的基项代换 σ 应用规则 r 一次）、all（应用任意规则一次），以及 match P s.t. C，用于检查当前项是否匹配模式 P，并满足约束 C。复合策略可以通过连接（α; β）、析取（$\alpha \mid \beta$）、迭代（$\alpha*$）、 α or-else β（如果 α 执行失败，则执行 β）、归一化 $\alpha!$（持续执行 α，直到无法继续应用）等方式定义。

Maude 还提供从基项 t 进行显式状态可达性分析（search [n,m] t =>* t' such that Φ），并可检验线性时序逻辑公式 F （red modelCheck(t, F)）。 F 中的原子命题是用户定义的 Prop 排序项，而函数 op _|=_ : State Prop -> Bool 用于指定哪些状态满足给定命题。线性时序逻辑公式由状态公式、布尔联结词和时序逻辑算子 [] （“始终”）、 <> （“最终”）以及 U（“直到”）构成。对于符号可达性分析，命令

smt-search [n, m]: t =>* t' 满足以下条件 Φ --- n 和 m 为可选项

以符号方式搜索 n 个状态，这些状态可在 m 步内从 $t \in T_\Sigma(X_0)$到达，且与模式$t' \in T_\Sigma(X)$匹配，并满足 $\mathcal{E}_0$ 中的约束 Φ。更准确地说，它会搜索受约束项 $\phi_u \parallel u$，使得 $true \parallel t \rightsquigarrow^* \phi_u \parallel u$，并且对于某个 $\sigma: X \to T_\Sigma(X)$、 $u = t'\sigma$ （模方程）以及 $\mathcal{T}_{\mathcal{E}_0} \models \phi_u \Rightarrow \Phi\sigma$。

Maude 提供了内置排序 布尔值、整数和实数，用于布尔值、整数和实数的 SMT 理论。排序为 实数的有理数常量写作 n/m （例如，0/1）。Maude-SE [51] 扩展了 Maude，增加了模 SMT 重写的功能，包括为 smt-search 生成见证的功能。它使用两种理论转换来实现符号重写 [45]。本质上，重写规则 $l: q \longrightarrow r$**如果** ψ 会被转换为受约束项规则

$$l : \texttt{PHI} \parallel q^\circ \longrightarrow (\texttt{PHI}\ and\ \psi\ and\ \Psi_{\sigma^\circ}) \parallel r\ \textbf{if}\ \texttt{smtCheck}(\texttt{PHI}\ and\ \psi\ and\ \Psi_{\sigma^\circ})$$

其中 PHI 是一个布尔变量，(q°, σ°) 是对 q 的内建项抽象，而 smtCheck 会调用底层 SMT 求解器来检查一个 SMT 条件的可满足性。若将 $(vars(r) \cup vars(\psi) \cup vars(\Psi_{\sigma^\circ})) \setminus vars(q^\circ)$ 中额外的 SMT 变量视为常量，则此规则可执行。

3 ITPN 的重写逻辑语义

本节使用一个（不可执行的）重写理论 $\mathcal{R}_0$，给出（非参数化）ITPN 的重写逻辑语义。我们给出一种双模拟，将网 $\mathcal{N}$ 的具体语义与 $\mathcal{R}_0$ 中的重写关系联系起来，并讨论 $\mathcal{R}_0$ 的变体，以避免连续的 tick 步，并支持有界时间分析。

3.1 在 Maude 中形式化 ITPN：理论 $\mathcal{R}_0$

我们固定 $\mathcal{N}$ 为 ITPN $\langle P, T, \emptyset, {}^\bullet(.), (.)^\bullet, {}^\circ(.)\ M_0,\ J,\ true\rangle$，并说明如何将 ITPN 及此类网的标识表示为 Maude 项。

我们首先在 Maude 中定义用于表示变迁标签、库所和时间值的排序。通常的做法是将每个变迁 t_i 和每个库所 p_j 分别表示为排序 Label 和 Place 中的一个常量（例如，ops p_1 p_2 ... p_m : -> Place [ctor]）。为了避免这种简单的参数化，并只使用一个重写理论 $\mathcal{R}_0$ 来定义所有 ITPN 的语义，我们假设库所和变迁（标签）可以表示为字符串。形式上，我们假设存在一个单射命名函数$\eta: P \cup T \to$ String；为避免在文中充斥下标，我们通常不显式提及 η。

保护 字符串 . 保护有理数 .排序 Label 库所 . --- 变迁和库所的标识符子排序 字符串 < Label 库所 . --- 为简便起见，我们使用字符串排序 时间 TimeInf . --- 时间值子排序 Zero 正有理数 < 时间 <TimeInf .运算符 inf :-> TimeInf [构造器] 。变量集合 T T1 T2 : 时间 .等式T <=inf =真 .

排序 TimeInf 将一个“无穷大”值 inf 加入由时间值构成的 Time 排序中，这些时间值是非负有理数（PosRat）。

在重写逻辑中形式化 Petri网的“标准”方式（例如，参见 [28,46]）会将标识表示为 Maude 项。例如，库所 p 中有两个令牌、库所 q 中有三个令牌的标识表示为 Maude 项 $p\ p\ q\ q\ q$。这对于支持网中变迁的并发发射至关重要。然而，由于 PITPN 的语义是交错语义，并且为了支持从参数化初始标识出发进行基于SMT的重写分析（示例 8），我们改为将标识表示为从库所到该库所中令牌数量的映射，因此上述标识表示为 Maude 项 $\eta(p)$ |-> 2； $\eta(q)$ |-> 3。

以下声明将排序 Marking 定义为由分号分隔的若干对 $\eta(p)$ |-> n 所组成的集合。时间区间表示为项 [下界: 上界]，其中上界 上界 的排序为 TimeInf，也可以取无穷大值 inf。Maude 项 $\eta(t)$: pre -> $post$ 抑制条件 $inhibit$ in $interval$ 表示一个变迁 $t \in T$，其中 pre、$post$ 和 $inhibit$ 分别是表示 •(t)、(t)•、◦(t) 的标识；而 $interval$ 表示区间 $J(t)$。网由此类变迁组成，并以分号分隔（第 11–12 行）：

排序 标识 。 ---
标识 op empty : -> 标识 [ctor] . op _|->_ : 库所 自然数 -> 标识 [ctor] . op _;_ : 标识 标识 -> 标识 [ctor assoc comm id: empty] . 排序 Interval . --- 时间区间（上界可以是无穷大） op '[_:_']: 时间 TimeInf -> 区间 [ctor] . 排序 网 变迁 . --- 变迁和网 subsort 变迁 <网 . op _':_-->_抑制条件_in_ : Label 标识标识 标识 区间-> 变迁 [ctor] . op emptyNet : -> 网 [ctor] . op _;_ : 网 网 -> 网 [ctor assoc comm id: emptyNet] .

示例*1*。 假设显而易见的命名函数 η 将 A 映射为“A”等，图1中的网可表示为以下 Net 排序的项：

“t1” : (“A” |-> 1) --> (“C” |-> 1) in [5 : 6] ; “t2” : (“B” |-> 1) —> (“D” |-> 1) inhibit (“A” |-> 1) in [3 : 4]; “t3” : (“B” |-> 1) --> (“E” |-> 1) in [1 : 2].

我们定义一些关于标识的有用运算，例如 _+_ 和 _-_：

变量集合 N1 N2 : 自然数 . 变量集合 M M1 M2 : 标识 . 变量 P: 库所 .运算 _+_ : 标识 标识 -> 标识 . 等式 ((P |-> N1) ; M1) + ((P |-> N2) ; M2) =(P |-> N1 + N2); (M1 + M2) .等式 M1 +空 = M1 .

（此定义假设 $M2$中的每个库所只出现一次。）标识上的函数 _-_ 也以类似方式定义。以下函数用于比较标识，并检查变迁在某个标识中是否使能：

op _<=_ : 标识 标识 -> 布尔值 . --- 比较标识 eq ((P |-> N1) ; M1) <= ((P |-> N2) ; M2) = N1 <= N2and (M1 <= M2) . eq empty <=M2 = true . ceq M1 <= empty = false if M1 =/= empty . op 使能 : 标识 变迁 -> 布尔值 . --- 使能变迁 eq 使能(M, L : 前置条件 --> 后置条件 抑制条件 INHIBIT in 区间) =(前置条件 <= M)and not 受抑制(M, INHIBIT) .

```
op 受抑制: 标识 标识 ->布尔值 . --- 受抑制变迁eq受抑制(M, empty) =
false.eq 受抑制((P |-> N2) ; M, (P |-> N) ; INHIBIT) =((N > 0)
and (N2 >= N)) or 受抑制(M, INHIBIT) .
```

动态。 我们将 ITPN 的动态定义为用于此类网的 Maude“解释器”。 [47] 中的具体 ITPN语义会在时间流逝时，动态调整未受抑制变迁的“时间区间”。遗憾的是， [47] 中的定义似乎略有矛盾：一方面，时间区间的端点应为非负数，而且状态中只有使能变迁才具有区间；另一方面， [47] 中关于时间和离散变迁的定义提到了 $\forall t \in T, I'(t) = \ldots$ and $M \geq {}^\bullet(t) \implies I'(t)^\uparrow \geq 0$，若所有端点均为非负数，这些内容似乎便是多余的。采用 [47] 中关于时间和变迁步的定义（即我们的定义 2），会导致时间区间的右端点可能为负 值，对应的是禁用变迁。这会带来一些缺点：(i)“时间值”可能是负数；(ii) 会出现反直觉的“区间” $[0, -r]$ ，其右端点小于左端点；(iii) 在适当的离散化下，可达“状态空间”可能是无限的，因为这些负值可能没有下界。

为避免这些“不一致”，并采用简单且定义明确的语义，我们使用“时钟”而非“递减区间”；时钟表示相应变迁已使能（但未受抑制）的时长。此外，为缩减状态空间，禁用变迁的时钟始终为零。由此得到的语义等价于 [47] 中（最自然解释下的）语义，具体定义见定理 1。

排序 ClockValues 表示由分号分隔的项集合 $\eta(t)$ -> τ，其中 t 是变迁的（标签），而 τ 表示 t 的“时钟”当前值。

```
排序 时钟值 。 --- 时钟值操作 空集 : -> 时钟值 [构造子]。操作 _-->_ : Label 时间 -> 时钟
值 [构造子] 。操作 _;_ : 时钟值 时钟值 -> 时钟值 [构造子 assoc comm id:empty] 。
```

$\mathcal{R}_0$ 中的状态是以下项 m： 时钟： 网 属于 State 类型，其中 m 表示当前标识，时钟表示变迁时钟的当前值，而网 表示 Petri 网：

```
排序 状态。
操作 _: _: _ : 标识 时钟值 网 -> 状态 [构造子] 。
```

以下重写规则对网中变迁 L 的应用进行建模（L : PRE –-> POST inhibit INHIBIT in INTERVAL）；NET’。由于 _;_ 被声明为结合且交换的，网中的任意变迁 L 都可以使用此重写规则进行应用：

```
条件重写规则 [applyTransition] :M : (L-> T);CLOCKS :(L :
PRE--> POST inhibit INHIBIT in INTERVAL) ; NET=> (M -PRE)
 + POST : L -> 0 ; updateClocks(CLOCKS, M-PRE, NET) :(L :
PRE--> POST inhibit INHIBIT in INTERVAL) ; NET’若 使能(M,L :
PRE --> POSTinhibit INHIBIT in INTERVAL)and (T
inINTERVAL) 。

操作 _属于_ : 时间区间 -> 布尔值。等式 T 属于 [T1 : T
2] = (T1 <= T) 且 (T <= T2) 。等式 T 属于 [T1 : inf] =
T1 <= T 。
```

变迁 L 在标识 M 中处于使能状态（已使能且未受抑制），其时钟值 T 位于该区间内。执行该变迁后，标识变为 (M - PRE) +POST，L 的时钟被重置[5]，其他时钟则使用以下函数进行更新：

```
eq更新时钟((L’ -> T’) ; 时钟, INTERM-M, (L’ : 前置条件–-> 后置条件 抑制条件 抑制条件 在 区间)
; 网 )
```

[5] 由于在 o 中 根据语义，禁用变迁的时钟应为零，因此我们可以安全地在此规则中将 L 设为 0 .

= 如果 前置条件 <= INTERM-M 那么 (L' -> T') 否则 (L' -> 0) 结束 ;
更新时钟(CLOCKS, INTERM-M,NET).等式 更新时钟(empty, INTERM-M,
NET) = empty .

$\mathcal{R}_0$ 中的第二条重写规则规定了时间如何推进。时间可以推进任意值 T，只要时间推进不超过必须触发某个活动变迁的时刻。时钟会根据经过时间 T 进行更新，但被禁用或受抑制的变迁除外：

条件重写规则 [tick] : M : CLOCKS : NET => M : increaseClocks(M,CLOCKS,NET,T) : NET如果
T <= mte(M, CLOCKS, NET) [不可执行] 。

这条规则不可执行（[不可执行]），因为表示时间推进量的变量 T 只出现在规则的右侧。因此，将规则与待重写的状态进行匹配的替换不会为 T 赋予任何值。这个时间推进量 T 必须小于或等于标识 M 中使能变迁的上界的最小值：

op mte : 标识 时钟值 网 -> TimeInf . eq mte(M, (L -> T) ; CLOCKS, (L : 前置条
件 --> 后置条件 ... in [T1 : inf]) ; NET)=mte(M, CLOCKS, NET) . eq mte(M,
(L -> T) ; CLOCKS, (L : 前置条件 --> ... in [T1 : T2]) ; NET)= 如果 active(M,
L : ...) then min(T2 - T, mte(M, CLOCKS, NET)) else mte(M, CLOCKS, NET)
fi . eq mte(M, empty, NET) =inf .

函数 increaseClocks 根据经过时间增加变迁的时钟，但不包括未使能或受抑制的变迁：

算子increaseClocks : 标识 时钟值 网 时间 ->时钟值。方程 increaseClocks(M, (L ->
T1) ; 时钟, (L : 前置条件 --> ...) ; 网, T)=如果active(M, L : 前置条件 -->...)则 (L -> T
1 + T) 否则 (L -> T1) 结束 ; increaseClocks(M,CLOCKS, 网, T)。方程
increaseClocks(M, empty, 网, T) = empty。

下面的函数 $[\![_]\!]_{\mathcal{R}_0}$ 形式化地描述了如何将标识和网分别表示为重写逻辑中的项，其排序分别为 Marking 和 Net。[6]

定义 *3*。设 $\mathcal{N} = \langle P, T, \emptyset, \bullet(.), (.)\bullet, \circ(.)\ M_0, J, \mathit{true}\rangle$ 为一个 *ITPN*。则 $[\![_]\!]_{\mathcal{R}_0} : \mathbb{N}^P \to \mathcal{T}_{\mathcal{R}_0,\mathtt{Marking}}$ 定义为 $[\![\{p_1 \mapsto n_1, \ldots, p_m \mapsto n_m\}]\!]_{\mathcal{R}_0} = \eta(p_1)$ |-> n_1 ; ... ; $\eta(p_m)$ |-> n_m，其中可以省略条目 $\eta(p_j)$ |-> 0。网 $\mathcal{N}$ 的 *Maude* 表示 $[\![\mathcal{N}]\!]_{\mathcal{R}_0}$ 是排序为 *Net* 的项 $[\![t_1]\!]_{\mathcal{R}_0}$; $\cdots$; $[\![t_n]\!]_{\mathcal{R}_0}$，其中对于每个 $t_i \in T$，$[\![t_i]\!]_{\mathcal{R}_0}$ 为 $\eta(t_i)$: $[\![\bullet(t_i)]\!]_{\mathcal{R}_0}$ --> $[\![(t_i)\bullet]\!]_{\mathcal{R}_0}$ 抑制条件 $[\![\circ(t_i)]\!]_{\mathcal{R}_0}$ 在 $[{}^{\uparrow}J(t_i) : J(t_i)^{\uparrow}]$。

3.2 语义的正确性

本节将说明，我们的重写逻辑语义 $\mathcal{R}_0$ 能够正确仿真任意 ITPN $\mathcal{N}$。更具体地说，我们给出一个双模拟结果，将 $\mathcal{N}$ 中 $a_0 = (M_0, J)$ 的行为与 $\mathcal{R}_0$ 中从初始状态 $[\![M_0]\!]_{\mathcal{R}_0}$: initClocks($[\![\mathcal{N}]\!]_{\mathcal{R}_0}$) : $[\![\mathcal{N}]\!]_{\mathcal{R}_0}$ 开始的行为联系起来，其中 initClocks(网) 是一种时钟赋值，它为 网 中的每个变迁（时钟）$\eta(t)$ 以及每个变迁（标签）$\eta(t)$ 赋值 0。

由于 $\mathcal{N}$ 中的变迁由一段延迟和随后发生的离散变迁组成，我们定义相应的重写关系 $\mapsto$，将 tick 和 applyTransition 规则结合起来，并证明该关系满足双模拟。

定义 4. 设 t_1、t_2、t_3 是 $\mathcal{R}_0$ 中 *State* 类型的项。若存在一个 t_2，使得 $t_1 \longrightarrow t_2$ 是在 $\mathcal{R}_0$ 中应用时钟推进规则进行的一次单步重写，且 $t_2 \longrightarrow t_3$ 是在 $\mathcal{R}_0$ 中应用应用变迁规则进行的一次单步重写，则记作 $t_1 \mapsto t_3$。此外，若存在从 t_1 到 t_2 的一系列 $\mapsto$ 重写，则记作 $t_1 \mapsto^* t_2$。

[6] $[\![_]\!]_{\mathcal{R}_0}$ i由命名函数 η 参数化；不过，我们不显式显示此参数。

下面的关系将我们的基于时钟的状态与基于变化区间的状态联系起来；这种对应关系是一个简单的函数，但变迁的上界为 ∞ 时的情况除外：

定义 *5.* 设 $\mathcal{N} = \langle P, T, \emptyset, \bullet(.), (.)\bullet, \circ(.)\ M_0, J, true\rangle$ 为一个 *ITPN*， $\mathcal{S}_{\mathcal{N}} = (\mathcal{A}, a_0, \rightarrow)$ 为其具体语义。令 $T_{\Sigma, state}$ 表示 E 中 *State* 类型基项的 $\mathcal{R}_0$-等价类集合。我们定义关系 $\approx \subseteq \mathcal{A} \times T_{\Sigma, state}$，将具体语义中 $\mathcal{N}$ 的状态与 $\mathcal{R}_0$ 中（*State* 类型的）状态关联起来，其中对于所有状态 $(M, I) \in \mathcal{A}$、$(M, I) \approx m$：时钟：网，当且仅当 $m = [\![M]\!]_{\mathcal{R}_0}$ 且网 $= [\![\mathcal{N}]\!]_{\mathcal{R}_0}$，并且对于每个变迁 $t \in T$，

– 若 t 在 M 中未使能，则 *clocks* 中 $\eta(t)$ 的值为 *0*；– 否则：

- 如果 $J(t)^{\uparrow} \neq \infty$，那么 *clocks* 中时钟 $\eta(t)$ 的值为 $J(t)^{\uparrow} - I(t)^{\uparrow}$；
- 否则，如果 $^{\uparrow}I(t) > 0$ 那么 $\eta(t)$ 在时钟中的值为 $^{\uparrow}J(t) - ^{\uparrow}I(t)$；否则，$\eta(t)$ 在时钟中的值可以是任意值 $\tau \geq {}^{\uparrow}J(t)$。

定理 *1.* 设 $\mathcal{N} = \langle P, T, \emptyset, \bullet(.), (.)\bullet, \circ(.)\ M_0, J, true\rangle$ 是一个 *ITPN*，且 $\mathcal{R}_0 = (\Sigma, E, L, R)$。那么，$\approx$ 是迁移系统 $\mathcal{S}_{\mathcal{N}} = (\mathcal{A}, a_0, \rightarrow)$ 与 $(T_{\Sigma, state}, ([\![M_0]\!]_{\mathcal{R}_0}$: 时钟初始化 $([\![\mathcal{N}]\!]_{\mathcal{R}_0}) : [\![\mathcal{N}]\!]_{\mathcal{R}_0}) \mapsto)$ 之间的双模拟。

3.3 R 的若干变体$_0$

本节介绍理论 $\mathcal{R}_1$ 和 $\mathcal{R}_2$，它们是 $\mathcal{R}_0$ 的两种变体，旨在缩减（符号分析中的）可达状态空间，并支持有界时间分析。$\mathcal{R}_1$ 避免连续应用时钟推进规则。这对于符号分析很有用，因为在 $\mathcal{R}_1$的具体执行中，应用时钟推进规则可能无法充分推进时间，使某个变迁得以使能，从而导致死锁。$\mathcal{R}_2$ 添加了一个“全局时钟”，用于表示系统中已流逝的时间。（在 $\mathcal{R}_0$ 中，这样的全局时钟也可以用一个永远不会使能的“新”变迁的时钟来编码。）这使得能够分析有界时间性质（某个状态能否在某个时间区间内到达？）。

该理论 $\mathcal{R}_1$。为避免连续应用时钟推进规则，我们可以在全局状态中添加一个新组件，其值为 tickOk 或 tickNotOk。只有当全局状态的这个新组件的值为 tickOk 时，才能应用时钟推进规则。因此，我们为这些扩展后的全局状态添加一个新构造子 _:_:_:_ ，一个新排序 TickState（其值为 tickOk 和 tickNotOk），并修改（或添加）以下两个重写规则：

sort TickState . ops tickOk tickNotOk : -> TickState ［构造子］. op _:_:_:_ : TickState 标识 时钟值 网 -> 状态 ［构造子］. var TS : TickState . 条件重写规则 ［applyTransition］: TS : M : ((L -> T) ; 时钟值) : (L : 前置条件 -->...) ; 网) =>tickOk : ((M - 前置条件) + 后置条件) : ... 如果 active(...)且 (T 属于区间). 条件重写规则 ［tick］: tickOk : M : ... => tickNotOk :M : increaseClocks(...)... 如果 T <=最短到期时间(M, 时钟值, 网) ［不可执行］.

定理 2。 设 $t = m$ *: clocks : net* 是 $\mathcal{R}_0$ 中一个 *State* 类型的项。那么，$t \longrightarrow^*_{\mathcal{R}_0} m'$ *: clocks′ : net* 当且仅当 `tickOk` *:* m *: clocks : net* $\longrightarrow^*_{\mathcal{R}_1}$ `tickNotOk` *:* m' *: clocks′ : net*。

尽管可达性得以保留，但在 $\mathcal{R}_1$ 中“任意”应用时钟推进规则时，如果时间推进得不够，无法触发变迁，就可能导致 $\mathcal{R}_1$ 中发生死锁，而在 $\mathcal{R}_0$ 中则不会。

该理论 $\mathcal{R}_2$。为了回答某个状态能否在特定时间区间内到达等问题，并实现有界时间分析（不探索超出时间界限的行为），我们在全局状态中加入一个表示“全局时间”的新组件：

算子 _:_:_:_@_: TickState MarkingClockValues 网 Time -> 状态 [构造子] 。

tick 和 applyTransition 规则会按预期进行修改。例如，tick 规则变为：

条件重写规则 [tick]: tickOk : M : CLOCKS : NET @ GT
=> tickNotOk: M: increaseClocks(..., T): 网@ GT + T
如果 T <= mte(M, CLOCKS, NET) [不可执行] .

其中 GT 是 Time 排序的一个变量。对于时间界限 Δ，我们可以在此规则的条件中添加合取项 GT +T <= Δ，以阻止执行超出该时间界限的操作。

设 t 和 t' 是 $\mathcal{R}_0$中 State 类型的项。若 t' 在时间 d 内从 t到达，记作 $t \xrightarrow[\mathcal{R}_0]{d} t'$，其中 $t \longrightarrow^*_{\mathcal{R}_0} t'$ 和 d 是轨迹中多次应用 tick规则时变量 T 所取值的总和。

定理 3. 设 $t = m$ *: clocks : net* 是 $\mathcal{R}_0$中 *State* 类型的项。那么，$t \xrightarrow{d}{}^*_{\mathcal{R}_0} m'$ *: clocks′ : net* 当且仅当 tickOk *:* m *: clocks : net* @ $0 \longrightarrow^*_{\mathcal{R}_2}$ tickNotOk *:* m' *: clocks′ : net* @ d。

4 Maude 中 ITPN 的显式状态分析

理论 $\mathcal{R}_0$–$\mathcal{R}_2$ 无法直接在 Maude 中执行，因为时钟推进规则在其右侧引入了一个新变量 T。遵循 Real-Time Maude 分析稠密时间系统的方法，尽管我们无法覆盖所有时间点，但可以选择在某些时间点对系统执行进行“采样”。例如，在本节中，我们将时钟推进规则改为每次应用时使时间增加一个时间单位：

条件重写规则 [tickOne]: M: 时钟: 网 => M: increaseClocks(M, CLOCKS, NET, 1): NET如果 1 <=mte(M, CLOCKS, NET)。

采用这种时间采样进行分析通常既不健全也不完备，因为它无法涵盖所有可能的系统行为：例如，如果某个变迁的触发区间为 $[0.5, 0.6]$，我们就无法通过这种时间采样执行该变迁。尽管如此，如果所有区间界限都是自然数，那么“所有行为”都应当能够得到覆盖。

因此，我们可以快速构建规约原型，并尝试不同的参数值，然后再应用后续章节中提出的健全且完备的符号分析和参数综合方法。

项 net3(a,b) 表示图 2 中的网（或其更通用的版本），其中 a 和 b 分别是变迁 t_3的区间下界和上界：

op net3 : 时间 TimeInf -> 网. var LOWER : 时间 . var UPPER : TimeInf . eq net3(LOWER, UPPER)= "t1" : "p5" |-> 1 --> "p1" |-> 1 in [2 : 6] ; "t2" : "p1" |-> 1 --> "p2" |-> 1 ; "p5" |-> 1 in [2 : 4] ; "t3" : "p2" |-> 1 ; "p4" |-> 1 --> "p3" |-> 1 in [LOWER : UPPER]; "t4" : "p3" |-> 1 --> "p4" |-> 1 in [0 : 0].

图 2 中的初始标识用项 init3 表示：

操作 init3 :-> 标识 .等式 初始3 = "p1" |-> 0 ; "p2" |-> 0 ; "p3" |-> 0 ; "p4" |-> 1 ; "p5" |-> 1 .

我们可以使用不同的参数值对该网进行 2000 步模拟：[7]

Maude> 重写 [2000] *init3 :* 时钟初始化*(net3(3,5)) : net3(3,5)* .

[7] 部分 文中所有 Maude 代码和 Maude 执行的输出都将替换为“...” r.

结果状态:
"p1" |-> 0 ; "p2" |-> 1 ; "p3" |-> 0 ; "p4" |-> 1 ; "p5" |-> 1 : ... : ...

为了进一步分析系统，我们定义一个 k-安全函数：当且仅当标识 m 中不存在令牌数超过 n 的库所时，k-safe(n, m) 成立:

op k-safe：自然数 标识 -> 布尔值。

变量 M：标识。 变量集合 N N1 N2：自然数。 变量 P：库所。等式 k-safe(N,empty) = true。等式 k-safe(N1, P |-> N2; M) = N2 <= N1 且 k-safe(N1, M)。

我们可以将 n 快速（5 毫秒内）检查当变迁 t_3 的区间为 $[3, 4$ 时，该网是否为 1-安全]:

Maude> *search [1] init3 : 时钟初始化(net3(3,4)) : net3(3,4) =>*M : CLOCKS : NET 满足 not k-safe(1,M) .*

解 1(状态 27) M --> "p1" |-> 0 ; "p2" |-> 2 ; "p3" |-> 0; "p4" |-> 1 ; "p5" |-> 1时钟 --> "t1" -> 0 ; "t2" -> 0 ; "t3" -> 4; "t4" -> 0
网 --> ...

该网不是 1-安全：我们到达了一个状态，其中库所 p_2中有两个令牌。不过，如果将 t_3 的区间改为 $[2, 3]$，该网就是 1-安全的:

Maude> *search [1] init3 : 时钟初始化(net3(2,3)) : net3(2,3) =>*M : CLOCKS : NET 满足 not k-safe(1,M) .*

无解。

进一步分析表明，net3(3,4) 是 2-安全的，但 net3(3,5) 甚至不是 1000-安全的。

我们还可以通过在 Maude 中进行完整的线性时序逻辑（LTL）模型检验，分析网的具体实例。例如，我们可以定义一个参数化原子命题 place p has n tokens；当且仅当某个状态的标识中，库所 p 恰好有 n 个令牌时，该命题成立:

op 库所_拥有_令牌 : 库所 自然数 ->Prop [构造子] 。等式 (P |-> N1 ; M : 时钟 : 网) |= 库所 P 有 N2 个令牌 = (N1 == N2) 。

然后，我们可以检查这样的性质：在系统的每个 行为中，是否存在无穷多个状态使得 p_3 中没有令牌，并且存在无穷多个状态使其恰有一个令牌：[8]

Maude>*red modelCheck(init3 : 时钟初始化(net3(3,4)) : net3(3,4), ([] <> 库所 "p3" 中有 0 个令牌) /\ ([] <> 库所 "p3" 中有 1 个令牌)) .*

结果 布尔值：true

我们知道，net3(3,4) 可以到达 p_2 中有两个令牌的标识；但这是否必然会发生（即，这种情况是否会发生在所有行为中）？

Maude>*red modelCheck(init3 : 时钟初始化(net3(3,4)) : net3(3,4),<> 库所 "p2" 有 2 个令牌) 。*

结果 ModelCheckResult：反例(...)

结果是一个反例，展示了一条 p_2 永远不会有两个令牌的路径。

我们还可以通过在状态中添加一个全局时间分量，获得与 $\mathcal{R}_3$ 对应的“时间采样”规约:

[8] []、 <>、/\ 和 ~ 分别是对应（时态）逻辑算子 □ （“始终”）、 ◇ （“最终”）、合取和否定的 Maude 表示。

算子 _:_:_@_: 标识 时钟值 网 时间 -> 状态 [构造子] 。
并修改时钟推进规则，使全局时钟根据经过时间递增。此外，对于有界时间分析，我们添加一个约束，确保系统执行不会超出时间界限 Δ:

条件重写规则 [executableTick]: M : 时钟 : 网 @ 大于 => M :
increaseClocks(M,CLOCKS,NET,1) : NET @ 大于 + 1如果 并且 --- 删除此条件以进行无界分析1 <=mte(M, FT, NET) 。

通过设置 Δ 为 1000，我们可以模拟系统 net3(3,5) 的一种行为，直到时间 1000 :

Maude> 重写 *init3 :* 时钟初始化*(net3(3,5)) : net3(3,5)* @ 0 。

结果状态:
"p1" |-> 0 ; "p2" |-> 1 ; "p3" |-> 0 ; "p4" |-> 1 ; "p5" |-> 1 : ... : ... @ 1000

然后，我们可以通过将时钟推进规则中的 Δ 设为 10，检查 net3(3,4) 在时间区间 $[5, 10]$ 内是否为 1-安全，并执行以下命令:

Maude> *search [1] init3 :* 时钟初始化*(net3(3,4)) : net3(3,4)* @ 0 =>*M : CLOCKS : NET @ GT suchthat not k-safe(1, M)and GT >= 5 .*

解 1（状态 68）
标识 --> "p1" |-> 0 ; "p2" |-> 2 ; "p3" |-> 0 ; "p4" |-> 1 ; "p5" |-> 1
...
GT --> 8

这表明，非一安全标识可在八个时间单位内到达。

5 参数与符号执行

对理论 $\mathcal{R}_0$–$\mathcal{R}_2$ 进行标准的 Maude 显式状态分析，无法分析 PITPN 的所有可能行为，原因有二：(1) tick 规则在其右侧引入了一个新变量 T，反映出时间可以推进任意满足 T <= mte(...) 的值；以及(2) 通过 Maude 显式状态分析具体状态，无法分析含有未初始化参数的参数化网。（例如，如果 INTERVAL 不是具体区间，那么 applyTransition 规则中的条件 T in INTERVAL 永远不会求值为 true，因此该规则永远不会被应用。）

通过 Maude-SE 对含 SMT 变量的符号状态进行分析，可以符号化表示时间推进 T 和网中未初始化的参数，从而解决这两个问题。这使得我们能够使用分析和参数综合方法，分析具有未知参数的密集时间系统中的所有可能行为。

本节定义一种重写理论 $\mathcal{R}_1^S$，它能够忠实地对 PITPN 进行建模，并可使用 Maude-SE 进行符号执行。我们证明，$\mathcal{R}_1$ 中的（具体）执行均可由 $\mathcal{R}_1^S$中的（符号）执行表示，反之亦然。我们还表明，在模 SMT 重写中，标准折叠技术 [30] 不足以合并 $\mathcal{R}_1^S$中的等价符号状态。因此，我们提出一种新的折叠技术，当编码后的 PITPN 的状态类图有限时，该技术可保证 $\mathcal{R}_1^S$ 的可达性分析终止。

5.1 符号重写逻辑语义

我们使用重写理论 $\mathcal{R}_1^S$ 定义 PITPN 的“符号”语义。该理论是 $\mathcal{R}_1$ 的符号对应形式。我们不以 $\mathcal{R}_0$ 为基础，因为符号“tick”步表示从一个符号状态出发的所有可能时钟步。因此，我们不会引入对应 PITPN 中不可能出现的死锁。

$\mathcal{R}_1^S$ 是通过将标识中的排序 Nat 和时钟值的排序 sortPosRat 替换为相应的 SMT 排序 Integer 和 Real 得到的。（前者仅用于支持对符号初始状态进行推理，此类状态中某个位置的令牌数量未知。）此外，规则中的条件（例如 M1 <=M2）会替换为相应的布尔值排序 SMT 表达式。在 Maude-SE 中对 $\mathcal{R}_1^S$ 进行符号执行时，会累积并检查参数化变迁发生所需约束的可满足性。

我们首先如下声明 Time 排序：

```
排序 Time TimeInf . 子排序 实数 < Time < TimeInf.
算子 inf :-> TimeInf [构造子] 。
```

其中，Real 是 SMT 实数的排序。（我们在重写规则中添加约束，以保证仅将非负实数作为时间值。）

区间的定义与 $\mathcal{R}_0$中相同：算子 '[_:_'] : Time TimeInf -> 区间。由于 实数 是 Time 的子排序， $\mathcal{R}_1^S$ 中的区间可以包含 SMT 变量。这意味着，PITPN 中的参数化区间 $[a, b]$ 可以表示为项 [a:Real : b:Real], , 其中 a 和 b 是 Real 排序的变量。

标识、网和时钟值的定义与第 3.1 节中的定义相似。我们只需调整以下标识定义：

```
算子 _|->_ : 库所 整数 -> 标识 [构造子] .
```

因此，在一对 ir $\eta(p)$ |-> e_I, e_I 是一个可能包含 SMT 变量的 SMT 整数表达式).

对标识和区间的操作保持不变，不过要使用相应的 SMT 排序。由于 Maude 中 Nat 和 Rat 的运算符与 Integer 和 Real 的运算符具有相同的函数签名，因此只需对规约做少量调整。例如，M1 <= M2 的新定义为：

```
变量集合 N1 N2: 整数 .算子 _<=_ : 标识 标识 -> 布尔值 .等式 ((P |-> N1) ;
M1) <= ((P |-> N2) ; M2) =N1 <= N2 且 (M1 <= M2) 。等式 空 <=M2 =
真 。
```

其中， <= 在 N1 <= N2 中是一个函数 算子 _<=_ : 整数 整数 -> 布尔值。其中的符号状态定义如下：

```
排序 State。 算子 _:_:_:_ : TickState 标识 时钟值 网 -> State [构造子]
```

$\mathcal{R}_1^S$ 中的重写规则作用于可能包含 SMT 变量的符号状态。尽管这些规则与 $\mathcal{R}_1$中的规则相似，但其符号执行完全不同。回顾第2节，Maude-SE⁀定义了一种理论转换，用于实现符号重写。在生成的理论 $\mathcal{R}_1^S$中，应用规则时，右侧出现但左侧未出现的变量会被替换为新鲜的⁀变量。此外，$\mathcal{R}_1^S$ 中的规则作用于形式为 $\phi \parallel t$的受约束项，其中 t 在本例中是一个 State 类型的项，而 ϕ 是一个可满足的 SMT布尔表达式。约束 ϕ 通过累积规则中的条件获得，从而限制了 t中变量的可能取值。

$\mathcal{R}_1^S$ 中的时钟重写规则为

```
条件重写规则 [tick] : tickOk : M : 时钟 : 网=> tickNotOk : M :
increaseClocks(M, 时钟, 网, T) : 网如果 (T >= 0/1 并且 mte(M, 时钟,网, T)) 。
```

变量 T 被限制为非负实数，并且必须满足以下谓词最短到期时间（mte），该谓词汇总了以下约束，以确保时间不会前进到超过某个时刻：在该时刻，某个使能变迁必须触发：

```
op mte : Marking ClockValues Net Real -> 布尔值 . eq mte(M, empty, NET, T) = t
rue . eq mte(M, (L -> R1) ; CLOCKS, (L : 前置条件 --> ...in [T1 : inf]) ;NET , T)=
mte(M, CLOCKS, NET, T) . eq mte(M, (L -> R1) ; CLOCKS, (L : 前置条件 --> ... in [T1 :
T2]); NET, T)= active(M, L : ...) ? T <= T2 - R1 : true)and mte(M, CLOCKS, NET,T) .
```

这意味着，对于每个变迁 L，如果 L 中区间的上界为 inf，则不会对 T 添加任何限制。否则，如果 L 在标识 M 下处于使能状态，SMT 三元运算符 C ? E1 : E2（检查 C 并据此选择 E1 或 E2）会进一步约束 T，使其小于 T2 - R1。increaseClocks 的定义也使用这一 SMT 算子来表示时钟的新值：

等式 `increaseClocks(M, (L -> R1) ; CLOCKS, (L : 前置条件 --> ... ) ; 网, T)= (L -> (active(M, L : 前置条件 ...) ? R1 + T :R1 )) ; increaseClocks(M, CLOCKS, 网, T) .`

应用变迁的规则定义如下：

条件重写规则 `[applyTransition] :TS : M: ((L -> T) ; 时钟) : (L : 前置条件 --> ...) ; 网)=> tickOk : ((M - 前置条件) + 后置条件) : 更新时钟(...): (L : 前置条件 --> ... ; 网)` 若使能(...) and (T in 区间) .

应用此规则时，会添加新的约束，以断言变迁 L 可以触发（谓词 active 和 _in_），并更新时钟状态：

等式 `更新时钟((L' -> R1) ; 时钟, INTERM-M, (L':前置条件 --> ...); 网)= (L -> 前置条件 <= INTERM-M ?R1 : 0/1) ; 更新时钟(...) 。`

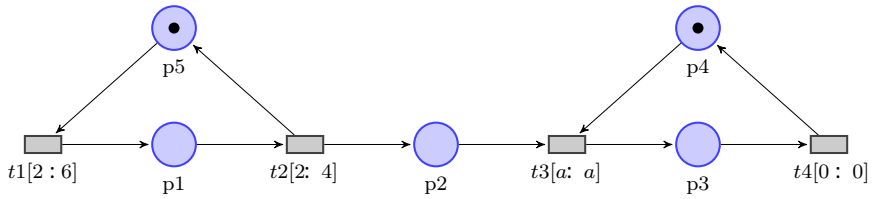


图 2：取自 [50] 的一个简单生产-消耗系统。

下文中，k-安全(k,m)是一个谓词，表示标识 m 在任何库所中都不超过 k个令牌。

示例 *2*。令 网 和 m_0 分别为 Maude 项，它们分别表示图 2 所示的 PITPN 和初始标识。项 网 包含一个变量 a:实数，表示参数 a。以下命令

`smt-search` tickOk: m_0 : 时钟初始化(net) : net `=>* TICK: M : CLOCKS : NET`满足条件 (a:实数 `>= 0/1` 并且 非 k-safe(1, M)) `= true` 。

回答了是否可能到达这样一个状态的问题：其标识中，某个库所里的令牌数超过一个。Maude 给出了肯定的回答，所得的累积约束表明，当 a:实数 `>= 4/1` 时，这样的状态是可达的（ p_2中有 2 个令牌）。

在 $\mathcal{R}_1^{\mathbf{S}}$ 中，Marking 排序的项可能包含 Integer 排序的参数（即变量）表达式。令 Λ_m 表示此类参数的集合，$\pi_m : \Lambda_M \to \mathbb{N}$ 为这些参数的赋值函数。我们用 m_s 表示从库所到包含参数变量的 Integer 表达式的映射。类似地，时钟$_s$ 表示从变迁到 Real 表达式（包括变量）的映射。我们用 $\pi_m(m_s)$ 表示这样一个基项：其标识里的参数已替换为对应的值 $\pi_m(\lambda_i)$。时钟$_s$ 的情况也类似。我们用 $[\![\mathcal{N}]\!]_{\mathcal{R}_1^{\mathbf{S}}}$ 表示上述在 $\mathcal{R}_1^{\mathbf{S}}$ 中对网的重写逻辑表示。

设 t_s 是 $\mathcal{R}_1^{\mathbf{S}}$ 中 State 类型的项，并假设 $\phi \parallel t_s \rightsquigarrow_{\mathcal{R}_1^{\mathbf{S}}} \phi' \parallel t'_s$。根据构造，如果对于所有 $t \in [\![\phi \parallel t_s]\!]$，所有标识（Integer 类型）、时钟和参数（Real 类型）均为非负数，那么对于所有 $t' \in [\![\phi' \parallel t'_s]\!]$，情况也同样如此。注意，基

在 $\mathcal{R}_1^{\mathbf{S}}$ 中满足该条件的项（Marking、ClockValues 等排序）与 $\mathcal{R}_1$中的（基）项之间存在一一对应关系。我们用$t \approx\in [\![\phi \parallel t_s]\!]$ 表示存在一个 $\mathcal{R}_1^{\mathbf{S}}$ 项 $t' \in [\![\phi \parallel t_s]\!]$，且 t 是它在 $\mathcal{R}_1$中对应的项。

以下定理表明，符号语义与由具体执行 $\mathcal{R}_1$ 在任意参数赋值 π 和 π_m下产生的所有行为相一致。此外，对于所有带参数的符号执行，都存在相应的具体执行，其中参数被实例化为与所得累积约束相一致的值。

定理 4（可靠性与完备性）。设 $\mathcal{N}$ 是一个 *PITPN*，m_s 是一个可能包含参数的标识。

(1) $L\ e^{t\ \phi\ b}\ e\ th^{e\ cons}\ train t\ \bigwedge_{\lambda_i \in \Lambda}(0 \le \lambda_i^- \le \lambda_i^+) \wedge \bigwedge_{\lambda_i \in \Lambda_m}(0 \le \lambda_i).\ If$ $\phi \parallel tickOk: m_s:$ 时钟$s: [\![\mathcal{N}]\!]_{\mathcal{R}_1^s} \rightsquigarrow^*_{\mathcal{R}_1^s} \phi' \parallel TS': m_s':$ 时钟$'_s: [\![\mathcal{N}]\!]_{\mathcal{R}_1^s}$则，存在 π和 π_m使得 $tickOk: \pi_m(m_s):$ 时钟$: [\![\pi(\mathcal{N})]\!]_{\mathcal{R}_0} \longrightarrow^*_{\mathcal{R}_1} TS': \pi_m(m_s'):$ 时钟$': [\![\pi(\mathcal{N})]\!]_{\mathcal{R}_0}$其中 $\phi' \wedge \bigwedge_{\lambda_i \in \Lambda} \lambda_i = \pi(\lambda_i) \wedge \bigwedge_{\lambda_i \in \Lambda_m} \lambda_i = \pi_m(\lambda_i)$ 可满足，时钟 $\approx\in [\![\phi \parallel$ 时钟$_s]\!]$ 且 时钟$' \approx\in [\![\phi' \parallel$ 时钟$'_s]\!]$。*(2* π π_m ϕ $\bigwedge_{\lambda_i \in \Lambda} \lambda_i =$*)* 设 为参数赋值，$\pi\ \lambda_i$ 为参数标记$\lambda_i = \pi_m\ \lambda_i$ 赋值。设约束为 $(())$ $\wedge \bigwedge_{\lambda_i \in \Lambda_m}(())$。如果

tick: $\pi_m(m_s):$ 时钟$: [\![\pi(\mathcal{N})]\!]_{\mathcal{R}_0} \longrightarrow^*_{\mathcal{R}_1} TS': m':$ 时钟$': [\![\pi(\mathcal{N})]\!]_{\mathcal{R}_0}$, 然后 $\phi \parallel$ tick: $m_s:$ 时钟$s: [\![\mathcal{N}]\!]_{\mathcal{R}_1^s} \longrightarrow^*_{\mathcal{R}_1^s} \phi' \parallel TS': m_s':$ 时钟$_s': [\![\mathcal{N}]\!]_{\mathcal{R}_1^s}$其中 $m' \approx\in [\![\phi' \parallel m_s']\!]$，时钟 $\approx\in [\![\phi \parallel$ 时钟$_s]\!]$ 且时钟$' \approx\in [\![\phi' \parallel$ 时钟$'_s]\!]$。

理论 $\mathcal{R}_2$ 的符号对应形式 $\mathcal{R}_2^{\mathbf{S}}$ 也可以用类似的方法定义。

5.2 符号可达性的一种新折叠方法

对于具有有限参数化状态类图的网，无论正向查询还是反向查询，可达性分析都应当终止。然而，即使对于这类网，smt-search生成的符号状态空间也是无限的，因此当目标状态不可达时，smt-search不会终止。问题在于，smt-search只有在已经访问过相同的状态时，才会停止从某个符号状态继续探索。由于每次应用时钟推进规则时都会在 $\mathcal{R}_1^{\mathbf{S}}$ 中创建新鲜变量，因此表示同一组具体状态的符号状态并不相同，尽管它们在逻辑上等价，如下例所示。

例 *3*。以下命令试图证明图 2 中的 PITPN 在 $0 \le a < 4$ 条件下是 1-安全的，但该命令不会终止。

smt-search tickOk: m_0 : 0-clock(net) : net =>* TICK : M : CLOCKS : NET满足条件 (a:实数 >= 0/1 且 a:实数 < 4 且 非 M <= k-safe(1,M)) = true。

此外，命令

smt-search tickOk : m_0 : 0-clock(net) : net =>* TICK : M : CLOCKS : NET 使得 (a:实数 >= 0/1 且 a:实数 < 4 且 M <= m_0 且 m_0 <= M) = true。

在搜索可达状态时，$M = m_0$ 将产生无限多个（等价的）解，例如包括以下约束:

解 1: #p5-9:整数 === 1 且 #t3-9:实数 + a:实数 - #t2-9:实数 <= 0/1 且..... 解 2: #p5-16:整数 === 1 且 #t3-16:实数 + a:实数- #t2-16:实数 <= 0/1且.....

其中，由 smt-search 创建的变量以 # 开头，并以从序列中取出的一个数字结尾，以保证新鲜性。设 $\phi_1 \parallel t_1$ 和 $\phi_2 \parallel t_2$ 分别是解 1 和解 2 中找到的受约束项。在此输出中，$\phi_2 \parallel t_2$ 是通过进一步重写 $\phi_1 \parallel t_1$得到的。表示标识和时钟状态的变量（例如，t_1 中的 #p5-9 和 t_2中的 #p5-16）显然不同，尽管它们表示相同的具体值集合（$[\![\phi_1 \parallel t_1]\!] = [\![\phi_2 \parallel t_2]\!]$）。由于应用规则时约束会不断累积，我们注意到，对于某个 ϕ_2'，ϕ_2 等于 $\phi_1 \wedge \phi_2'$，并且 变量集合$(\phi_1 \parallel t_1) \subseteq$ 变量集合$(\phi_2 \parallel t_2)$。

在模SMT重写中折叠等价符号状态的常用方法是包含[30]。基本上，如果在搜索过程中，我们已经遇到另一个包含某个符号状态的符号状态，就会停止从该符号状态继续搜索。更准确地说，设 $U = \phi_u \parallel t_u$ 和 $V = \phi_v \parallel t_v$ 为受约束项。那么，若存在一个替换 σ，使得 $t_u = t_v\sigma$ 且蕴含关系 $\phi_u \Rightarrow \phi_v\sigma$成立，则有 $U \sqsubseteq V$。在这种情况下，$[\![U]\!] \subseteq [\![V]\!]$。如果此前已经遇到另一个受约束项 V，且 $U \sqsubseteq V$，那么搜索就不会进一步探索受约束项 U。已知这种带折叠的可达性分析是可靠的（不会生成虚假的反例 [6]），但不一定是完备的（因为 $[\![U]\!] \subseteq [\![V]\!]$ 并不蕴含 $U \sqsubseteq V$）。

例 *4.* 设 ϕ_1 和 ϕ_2 分别是例 3 中第二条 `smt-search`命令找到的两个解中的约束。设 σ 是一个替换，它对每个库所 p_i 和变迁 t_j，都将 #pi-9 映射为 #pi-16，并将 #tj-9 映射为 #tj-16。SMT 求解器判定公式 $\neg(\phi_2 \Rightarrow \phi_1\sigma)$ 可满足（因此 $\phi_2 \Rightarrow \phi_1\sigma$ 无效）。因此，基于检查该蕴含关系的过程将无法判定第二个解中的状态可被第一个解中的状态包含。

$\neg(\phi_2 \Rightarrow \phi_1\sigma)$ 的可满足性见证可以为如何提高包含判定过程的精确性提供一些思路。假设 ϕ_1 携带了某个由 R表示的时钟的信息 $R = T_0$，且 T_0 是一个受 $\phi = (0 \leq T_0 \leq 2)$ 约束的时钟变量。再假设在 ϕ_2 中，同一时钟的值为 $R' = T_1 + T_2$，并受 $\phi' = (\phi \wedge T_1 \geq 0 \wedge T_2 \geq 0 \wedge T_1 + T_2 \leq 2)$ 约束。设 $\sigma = \{R \mapsto R'\}$。注意，$(R' = T_1 + T_2 \wedge \phi \wedge \phi')$ 并不蕴含 $(R = T_0 \wedge \phi)\sigma$（例如，赋值 $T_1 = T_2 = 0.5$ 和 $T_0 = 2$）。关键观察在于，即使 R 和 R' 都受约束而处于区间 $[0, 2]$ 内（因此表示该时钟的同一状态），蕴含关系前件中的 R' 赋值也不必与后件中的 R 赋值一致。

接下来，我们提出一种包含关系来解决上述问题。令 $\phi \parallel t$为一个受约束项，其中 t 是一个 State 类型的项。考虑针对 t的内建项抽象 (t°, σ°)，其中 t° 与 t 相同，但会将标识中的表达式 e_i（$p_i \mapsto e_i$）和时钟（$l_i \longrightarrow e_i$）替换为新鲜变量。相应地定义替换 σ°。令 $\Psi_{\sigma^\circ} = \bigwedge_{x \in dom(\sigma^\circ)} x = x\sigma^\circ$。我们用 $(\phi \parallel t) \Downarrow_{\texttt{now}}$ 表示受约束项 $\phi \wedge \Psi_{\sigma^\circ} \parallel t^\circ$。直观地说，$(\phi \parallel t) \Downarrow_{\texttt{now}}$ 会用新鲜变量替换时钟值和标识，而布尔表达式 Ψ_{σ° 则约束这些变量，使其取值与 t中的时钟和标识相同。由 [45] 可知，$[\![\phi \parallel t]\!] = [\![(\phi \parallel t) \Downarrow_{\texttt{now}}]\!]$。

注意，$(\phi \parallel t) \Downarrow_{\texttt{now}}$ 中出现的唯一变量是参数（如果有）以及$dom(\sigma^\circ)$ 中的新鲜变量（表示时钟和标识的符号状态）。对于受约束项 $\phi \parallel t$，我们用 $\exists(\phi \parallel t)$ 表示公式 $(\exists X)\phi$，其中 $X =$ 变量集合$(\phi) \setminus$ 变量集合(t)。

定义 6（关系 $\preceq$）。设 $U = \phi_u \parallel t_u$ 和 $V = \phi_v \parallel t_v$ 为受约束项，其中 t_u 和 t_v 是 *State* 类型的项。此外，设 $U \Downarrow_{now} = \phi'_u \parallel t'_u$ 和 $V \Downarrow_{now} = \phi'_v \parallel t'_v$，其中 $vars(t'_u) \cap vars(t'_v) = \emptyset$。我们在受约束项上定义关系 $\preceq$，使得当且仅当存在一个替换 σ 满足 $t'_u = t'_v\sigma$，且公式 $\exists(U \Downarrow_{now}) \Rightarrow \exists(V \Downarrow_{now})\sigma$ 有效时，$U \preceq V$ 成立。

公式 $\exists(U \Downarrow_{\texttt{now}})$隐藏了有关所有时钟变量的信息，以及先前时刻的时钟和标识信息。我们得到的是有关参数以及“当前”时钟和标识取值的信息。此外，如果上文的 t_u 和 t_v 都是 tickOk 状态（或都是 tickNotOk 状态），并且它们表示同一个 PITPN 的两个符号状态，那么 t'_u 和 t'_v 始终匹配（σ 在表示参数的变量上为恒等映射，并将 $V \Downarrow_{\texttt{now}}$ 和 $U \Downarrow_{\texttt{now}}$中创建的对应变量相互映射）。

定理 5（可靠性与完备性）。设 U 和 V 是 $\mathcal{R}_1^S$ 中的受约束项，表示同一个 *PITPN* 的两个符号状态。那么，当且仅当 $[\![U]\!] \subseteq [\![V]\!]$ 时，$U \preceq V$成立。

我们基于定义 6 中的折叠关系，实现了一种新的符号可达性分析。以 Maude-SE 中定义的理论转换为基础，我们将理论 $\mathcal{R}_1^{\mathbf{S}}$ 转换为重写理论 $\mathcal{R}_1^{f\mathbf{S}}$，该理论对形式为 $S : \phi \parallel t$ 的项进行重写，其中 S 是受约束项的集合（即已访问状态）。理论 $\mathcal{R}_1^{f\mathbf{S}}$ 定义了排序 SetState，用于表示以分号分隔的受约束项集合，以及一个

算子 subsumed($\phi \parallel t$， S)，当且仅当存在 $\phi' \parallel t' \in S$ 使得 $\phi \parallel t \preceq \phi' \parallel t'$ 时，该算子归约为 true。规则$l : q \longrightarrow r$ **如果** ψ 属于 $\mathcal{R}_1^{\mathbf{S}}$，则会在 $\mathcal{R}_1^{fS}$ 中转换为以下规则：

$$l : \ S : \texttt{PHI} \parallel q^{\circ} \longrightarrow (S; \phi_r \parallel r) : \phi_r \parallel r \ \textbf{if} \ \texttt{smtCheck}(\phi_r) \wedge \texttt{not subsumed}(\phi_r \parallel r, S)$$

其中，PHI 是一个布尔变量，S 是一个排序为 SetState 的变量，(q°， σ°) 是对 q和 ϕ_r =(PHI $and \psi and \Psi_{\sigma^{\circ}}$) 的内建项抽象。请注意，只有当新状态 $\phi_r \parallel r$不被 S中已访问的状态包含时，才会发生该变迁。理论 $\mathcal{R}_2^{fS}$ 也可由 $\mathcal{R}_2^{\mathbf{S}}$类似地得到。

在 $\mathcal{R}_1^{fS}$ 中，对于参数上的初始约束 ϕ，命令
search $[n,m]$: empty: $\phi \parallel t$ =>* S : $\phi' \parallel t'$ 满足以下条件 smtCheck($\phi' \wedge \Phi$) 会回答是否能够到达与 t' 匹配且满足条件 Φ 的符号状态。下文中，我们用 init(net， m_0， ϕ) 表示项 empty : $\phi \parallel$ tickOk : m_0 :时钟初始化(net) : net。

例 *5.* 考虑图 2 中的 PITPN。设图中的标识为 m_0，且 $\phi = 0 \leq a < 4$。命令

search init(net， m_0， ϕ) =>* S : $\phi' \parallel$ (TICK : M : CLOCKS : NET)
满足smtCheck(ϕ' and not k-safe(1,M)) .
终止并返回无解，这表明当 $0 \leq a < 4$ 时，该网是 1-安全的。

以下结果表明，如果 $\mathcal{N}$ 的符号语义中的可达状态类集合（见 [2]）是有限的，那么使用新的折叠技术得到的可达符号状态集合也有限。

推论 1. 对于任意 *PITPN* $\mathcal{N}$ 和状态类 (M, D)，如果迁移系统 $(\mathcal{C}, (M, D) \Rightarrow)$ 是有限的，那么 $\left(T_{\Sigma,State}, init(\mathcal{N}, M, D) \rightsquigarrow_{\mathcal{R}_1^{fs}}\right)$ 也是有限的。

值得注意的是，定义6和定理 5中的新折叠关系适用于任何满足第2.2节简要说明的使用SMT进行重写的要求的重写理论 $\mathcal{R}$ [45]。

6 参数综合与符号模型检测

本节展示如何使用 Maude-SE 进行多种超越可达性分析的形式化分析。我们将介绍如何使用 Maude-SE 求解参数综合问题、检验当前先进的 PITPN 工具 Roméo 所支持的非嵌套定时时序逻辑性质、针对不同库所中的令牌数量未知的参数化初始状态进行推理，以及分析带有用户定义执行策略的网。由此，我们提供了超越 Roméo 所支持范围的分析方法，同时支持 Roméo 提供的几乎所有形式的分析。

6.1 参数综合

状态谓词是一种布尔表达式，其原子命题包括对标识值（例如，k-安全(1, m)）和时钟（例如， $c_1 < c_2$）的测试。EF-综合是计算参数值π 的问题，使得 $\pi(\mathcal{N})$存在一条运行能够到达满足给定状态谓词 ϕ的状态。安全性综合问题 AG$\neg\phi$ 是计算使满足 ϕ 的状态不可达的参数值的问题。

在理论 $\mathcal{R}_1^{fS}$ 中进行search （参见第 5.2 节），可为求解这些参数综合问题提供半判定过程（这些问题通常不可判定）。如下所示，search 计算出的约束可用于综合出允许此类执行路径的参数值。安全性综合问题 AG$\neg\phi$ 可通过求出 EFϕ 的所有解，再对所得约束取反来解决。

示例 *6*。 示例 2 展示了一个 EF 综合问题：找出参数 a 的取值，使得能够到达一个至少有两个令牌的状态，且这些令牌位于某个 库所中。如果 $\phi = 0 \leq a$，命令

search [1] init(net, m_0, ϕ) =>* S : PHI' || (TICK :M : CLOCKS : NET)满足以下条件smtCheck(PHI' and not k-safe(1,M)) .
返回一个解以及相应的约束 ϕ'，实例化模式 PHI' 后，可按如下方式提取参数值。设 X 为 ϕ' 不表示参数的 SMT 变量集合。对公式 $\exists X.\phi'$ 调用 SMT 求解器 Z3 的量词消去过程（qe），可将其化简为 a:Real >= 4/1，从而得到所需的参数 a的值。

为了解决安全性综合问题 AG$\neg\phi$，我们使用了 Maude 的元编程功能 [16]来实现命令 safety-syn(net, m_0, ϕ_0, ϕ)，其中 m_0 是一个标识，ϕ_0 是关于参数的约束，ϕ 是涉及变量 M 和 CLOCKS 的约束，其形式与示例 6 中的 search 命令相同。该命令会迭代调用 search，以查找从 m_0可达且不满足 ϕ 的状态，其初始约束为 ϕ_0。如果找到这样的状态，且其累积约束为 ϕ'，则会再次调用 search 命令，并使用初始约束 $\phi_0 \wedge \neg\phi'$。当找不到更多不满足 ϕ 的可达状态时，该过程便会停止，从而解决 AG$\neg\phi$ 综合问题。

示例 7. 考虑图 3 中的 PITPN，取自 [48]，其中包含参数 a，以及三个区间分别为 $[a:a]$、$[2a:2a]$、$[3a:3a]$ 的参数化变迁。Roméo 可以综合出使该网满足 1-安全的参数 a 的取值，同时满足初始约束 $30 \le a \le 70$。在 Maude 中也可以回答同样的查询：
安全性综合(net, m_0, a:实数 >= 30/1 and a:实数 <= 70/1, k-安全(1,M))。
找到的第一个反例假设 $a \le 48$。如果 $a > 48$，search 则找不到任何反例。这与 Roméo 得出的答案相同。

Roméo 仅支持针对标识的性质。上述命令中的状态谓词还可以包含关于时钟值的条件。

我们的符号理论允许在初始标识中包含参数（整数排序的变量）。这使得我们可以使用 Maude-SE 求解涉及参数化初始标识的综合问题。例如，我们可以确定使该网满足 k-安全和/或活性的初始标识：

示例 8. 考虑图 2 中网的一个参数化初始标识 m_s，其中参数 x_1、x_2 和 x_3分别表示库所 p_1、p_2 和 p_3 中的令牌数，初始约束 ϕ_0 表明 $a \ge 0$ 和 $0 \le x_i \le 1$。执行命令 safety-syn(net, m_s, ϕ_0, k-safe(1, M)) 可确定，当 $x_1 = x_3 = 0$ 且 $0 \le x_2 \le 1$时，该网是 1-安全的。

基于策略的分析。 Maude 的策略功能 [15] 使我们能够分析执行遵循某种用户定义策略的 PITPN。如下例所示，这类策略可能会影响参数综合分析的结果。

示例9. 我们使用以下策略 t3-first执行图 2 中的网：每当变迁 t_3 与其他某个变迁同时使能时，t_3 就优先触发。该执行策略可以指定如下：

t3-first := (applyTransition[L <- "t3"] 否则 all)!
使用 t3-first 运行 srew init(net, m_0, $a \ge 0$)，在 $\mathcal{R}_1^{fS}$ 中找到了采用此策略可达的所有符号状态，而且它们全都是 1-安全。因此，所有参数值 $a \ge 0$ 在此执行策略下都能保证所需的性质。

6.2 分析时态性质

本节展示如何使用 Maude-SE 分析 Roméo [26],支持的时态性质，尽管在少数情况下，时序公式中不含参数界。Roméo 可以分析以下时态性质：

$$\mathbf{Q}\,\phi\,\mathsf{U}_J\,\psi \mid \mathbf{Q}\mathsf{F}_J\,\phi \mid \mathbf{Q}\mathsf{G}_J\,\phi \mid \phi \rightsquigarrow_{\le b} \psi$$

其中 **Q** $\in \{\exists, \forall\}$ 是存在/全称路径量词，ϕ 和 ψ 是作用于标识上的状态谓词，J 是一个时间区间 $[a, b]$，其中 a 和/或 b 可以是参数，而 b 可以是 ∞。例如，$\forall \mathrm{F}_{[a,b]}\, \phi$表示从初始状态出发的每一条 路径上，都能在 $[a, b]$中的某个时间到达满足 ϕ 的标识。有界响应 $\phi \rightsquigarrow_{\leq b} \psi$ 表示公式 $\forall \mathrm{G}(\phi \implies \forall \mathrm{F}_{[0,b]}\, \psi)$（每个 ϕ-标识必须 在时间 b内由一个 ψ-标识跟随）。

由于查询包含时间界限，我们使用理论 $\mathcal{R}_2^{fS}$，而 init(net, m_0, ϕ) 将表示项 empty : ϕ || tickOk : m_0 : 时钟初始化(net) : net @ 0/1。

状态谓词（包括关于标识和时钟的不等式，以及测试全局时钟是否处于给定区间内）定义如下：

ops _>=_ _>_ _<_ _<=_ _==_ : 库所 整数 -> Prop . ops _>=_ _>_ _<_ _<=_ _==_ : Clock 实数 -> Prop . op in-time : 区间 -> Prop . eq S : C || (TICK : M ; (P |-> N1) : CLOCKS : NET) @ G-CLOCK |= P >= N1'= smtCheck(Cand N1 >= N1'). --- 同理，对于 >、 <=、 < 和 ==, eq S : C || (TICK : M : CLOCKS : NET) @ G-CLOCK |= in-timeINTERVAL= smtCheck(Cand (G-CLOCK in INTERVAL)) .

原子命题（排序 Prop）在表示为受约束项 $S : \phi \,\|\, t$ 的符号状态上进行求值。由于它们可能包含变量，因此需要调用可满足性模理论来确定约束 ϕ 是否蕴含该命题。

Roméo 支持的某些时序公式可以使用上一节介绍的可达性命令轻松验证。性质 $\exists \mathrm{F}_{[a,b]}\, \psi$ 可以使用以下命令验证：

search [1] init(net, m_0, ϕ) =>* S : PHI' || TICK : M : CLOCKS : NET @ G-CLOCK使得 $STATE'$ |= ψ 且 G-CLOCK 属于 [a : b] 。

其中，ϕ 表示所有参数均为非负数，而状态′ 是 * 右侧的表达式。a 和 b 可以是表示待综合参数的变量；ψ 可以是涉及 CLOCKS 的表达式。例如，

search [1] init(net, m_0, ϕ) =>*S' : PHI' || TICK : (M ; "p1" |-> P1) : (CLOCKS ; "t2"-> C2) : NET @ G-CLOCK使得 $STATE'$ |= P1 > 1 且 C2 < 2/1 且 G-CLOCK 属于 [a : b] 。

检查是否能够在某个 $[a, b]$时间到达某个标识，使库所p_1中有多个令牌，且变迁 t_2 的时钟值为 < 2。

对偶性质 $\forall \mathrm{G}_{[a,b]}\, \phi$ 可以通过分析 $\exists \mathrm{F}_{[a,b]}\, \neg\phi$来检查。

例*10.* 考虑示例 7 中参数为 $\phi = 30 \leq a \leq 70$的 PITPN。性质$\exists \mathrm{F}^{[b,b]}$(¬*1*-安全)可通过以下命令验证，该命令确定参数 b 满足$60 \leq b \leq 96$。

search [1] init(net, m_0, ϕ) =>* S : PHI' || TICK : M : CLOCKS : NET @ G-CLOCK满足 $STATE'$ |= b:Real >= 0/1 and (G-CLOCK in [b:Real : b:Real]) and not(k-safe(1,M)).

有界响应 $\phi \rightsquigarrow_{\leq b} \psi$ 公式可以通过对 $\mathcal{R}_0^{\mathrm{S}}$进行简单的理论转换，然后进行可达性分析来验证。理论转换会为 State 类型添加一个新的构造子，以构造形如 C_ϕ: M: C $locks$: N et 的项，其中 C_ϕ要么是无时钟，要么是时钟(τ)；后者表示自访问某个 ϕ-状态以来经过的时间（τ），且之后尚未访问过 ψ-状态。重写规则会进行调整，以便按如下方式更新这个新组件。新的时钟推进规则将 clock(T1) 更新为 clock(T1 +T)，并保持无时钟不变。applyTransition 规则被拆分为两条规则：

条件重写规则 [applyTransition] : 时钟(T) : M ... => NEW-TP : M' ...若 NEW-TP := 若 $STATE'$ |= ψ 则 无时钟 否则 时钟(T) fi /\ ...条件重写规则 [applyTransition] : 无时钟 : M ... => NEW-TP :M' ...若 NEW-TP :=若 $STATE'$ |= ϕ且非 $STATE'$ |= ψ则 时钟(0/1) 否则 无时钟 fi /\...

在第一条规则中，如果遇到一个 ψ-状态，新的“ϕ-时钟”会重置为无时钟。在第二条规则中，如果新状态满足 ϕ 但不满足 ψ，这个“ϕ-时钟”就开始运行。通过搜索一个满足以下条件的状态，可以回答查询 $\phi \rightsquigarrow_{\leq b} \psi$ ：某个 ϕ-状态在截止时间 b之前尚未接续一个 ψ-状态：

```
搜索 [1]...  =>* S : PHI' || clock(T) : ... 满足 T > b 。
```

可达性分析无法用于分析 Roméo 支持的其他性质（$\mathbf{Q}\ \phi \mathrm{U}_J\ \psi$，以及$\forall \mathrm{F}_J\ \phi$ 和它的对偶 $\exists \mathrm{G}_J\ \phi$）。虽然开发一个基于可满足性模理论的完整定时时序逻辑模型检验器是未来的工作，但我们可以结合 Maude 的显式状态模型检验器与 SMT求解来处理这些（以及许多其他）查询。值得一提的是，除了 Roméo 所支持的功能之外，我们还可以使用完整的线性时序逻辑，并允许在状态命题中加入时钟条件。

定时时序算子可以基于 Maude 中的（非定时）线性时序逻辑时序算子（<>、 [] 和 U）定义：

```
op <_>_ : 区间 Prop ->Formula . op _U__ : Prop 区间 Prop ->
Formula . op [_]_ : 区间 Prop -> Formula . vars PR1 PR2 : Prop .
eq < INTERVAL > PR1 = <> (PR1 /\ in-time INTERVAL). eq
PR1 U INTERVAL PR2 = PR1 U (PR2 /\ in-time INTERVAL) . eq [
INTERVAL ] PR1 = ~ (< INTERVAL > (~ PR1)).
```

对于这类非嵌套定时时序逻辑公式，可以按如下方式对全称量化和存在量化公式进行模型检测：

```
op A-model-check : State Formula ->Bool . --- 全称查询op E-model-check : State
Formula ->Bool . --- 存在性查询eqA-model-check(STATE, F) = modelCheck(STATE, F)
 == true .eq E-model-check(STATE, F)  = modelCheck(STATE , ~ F)  =/=true .
```

7 基准测试

我们在三个案例研究中，将 Maude-with-SMT 分析的性能与 Roméo（版本 3.8.6）的性能进行了比较。我们比较不同重写理论求解综合问题 EF($p > n$) 所需的时间（即，库所 p 中的令牌数大于 n），针对不同的库所 p 和 $0 \leq n \leq 2$，并检查该网是否为 1-安全。实验中使用的模型包括：图 2 中的生产者-消费者 [50] 系统、图 3 中的调度 [48]系统，以及取自 Roméo 网站的图 4 中的教程系统。我们修改了教程模型，使变迁 startOver 中产生两个令牌，从而导致无限行为。各模型的详细信息见表 1。

所有实验均在 Dell Precision Tower 3430 上运行，配置为 Intel Xeon E-2136 六核处理器（3.3 GHz）、64 GiB 内存和 Ubuntu 20.04。每项实验均使用 Maude，并结合两种不同的 SMT 求解器：Yices 和 Z3。超时时间设为 10 分钟。

图5以对数刻度展示了 Roméo 和 Maude 在三个案例研究中的执行时间。（每项实验的数据见附录 A。）图中的每个点表示 Roméo 和 Maude 分析属性 EF($p > n$) 和 AG($\neg$ 1-safe) 所用的时间。在某些情况下，$\mathcal{R}_1^{\mathbf{S}}$ 的表现优于$\mathcal{R}_0^{\mathbf{S}}$；与 Z3 相比，使用 Yices2 所需的时间更短。对于否定查询（例如，在调度和生产者-消费者案例中，EF($p > 2$) 为假），如预期所料，$\mathcal{R}_0^{\mathbf{S}}$ 和 $\mathcal{R}_1^{\mathbf{S}}$会超时。在这些情况下，$\mathcal{R}_1^{f\mathbf{S}}$ 能在超时前完成分析。目前，Maude-SE 仅支持使用 Z3 进行存在量化查询，因此 $\mathcal{R}_1^{f\mathbf{S}}$ 只能通过该 SMT 求解器执行。近期，Maude-SE 将集成对 Yices2 中量词的支持，我们预计这将提升 $\mathcal{R}_1^{f\mathbf{S}}$ 的性能。最后值得注意的是，在某些可达性查询中，Maude-SE 的表现优于 Roméo。更值得关注的是，在 Roméo 无法终止的情况下，我们的方法能够终止。将我们的结果作为额外约束注入模型，并使用 Roméo 运行后，证明了这些结果的有效性。当搜索顺序导致 Roméo 探索具有无界标识的无限分支时，就会出现这种现象。

模型	参数	库所	变迁	弧	
生产者–消费者		1	5	4	10
调度		3	6	9	15
教程		2	6	5	12

表1：基准测试中使用的模型描述（图5）。

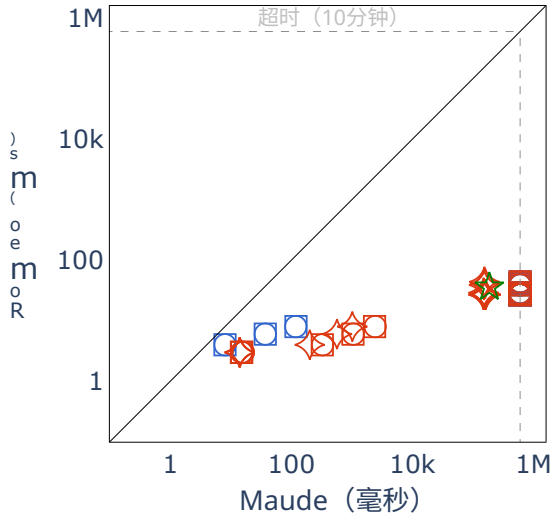


图 3： [48]中的案例研究。

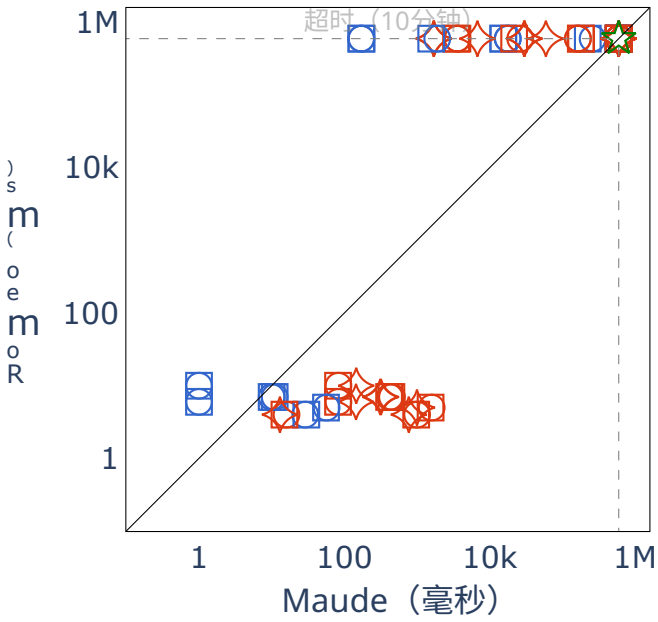
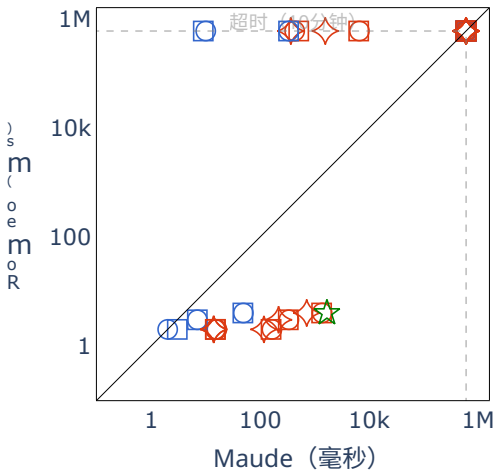
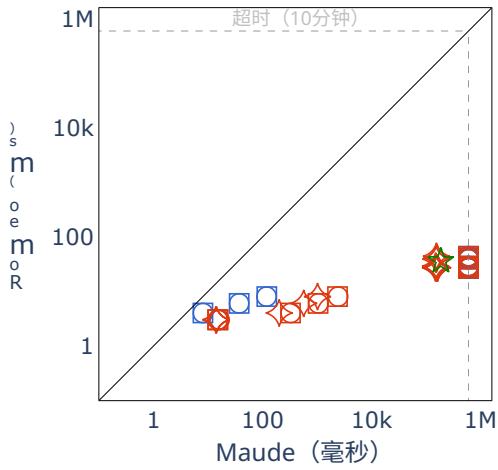


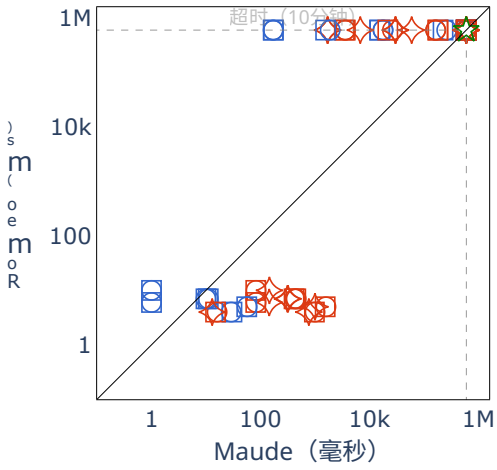
图4：案例研究教程。



(a) 生产者-消费者



(b) 调度



(c) 教程

图5：执行时间（对数刻度），比较 Roméo 和 Maude。理论 $\mathcal{R}_0^{\mathbb{S}}$ 使用 Z3（○），理论 $\mathcal{R}_0^{\mathbb{S}}$ 使用 Yices（○），理论 $\mathcal{R}_1^{\mathbb{S}}$ 使用 Z3（□），理论 $\mathcal{R}_1^{\mathbb{S}}$ 使用 Yices（□），以及 $\mathcal{R}_1^{f\mathbb{S}}$ 使用 Z3（◇）。点 ☆ 表示该性质为 1-安全。

8 相关工作

参数化时间 *Petri* 网工具支持。 据我们所知，没有其他工具可以分析参数化时间(d) Petri网，而 Roméo [26] 可以。

重写逻辑中的 *Petri* 网。 以代数方式形式化 Petri 网 [31] 是重写逻辑背后的灵感来源之一。文献 [46], 和 [37] 分别为定时网赋予了重写逻辑语义。与本文不同，这些论文侧重于此类网的语义，不考虑执行和分析；也不考虑抑制弧或参数。Capra [13,12],Padberg 和 Schultz [43],以及 Barbosa 等人 [11] 使用 Maude 形式化动态可重构 Petri 网（带有抑制弧）和输入/输出 Petri 网。与我们的工作不同，这些论文针对的是非定时非参数化网，不侧重于形式化分析，而只展示了标准（显式状态）`search` 和线性时序逻辑模型检验的示例。

Maude 中实时系统的符号方法。 我们在 [4] 中为参数化时间自动机（PTA）开发了一种符号重写语义和分析方法。与本文的区别包括：与 PITPN（带有抑制弧，且状态中的令牌数量没有上限）相比，PTA 是非常简单的结构，因此 PITPN 的语义比 PTA 的语义更复杂；PTA 的语义不使用“结构化”状态、方程或用户定义函数；对于 [4] 中的每个 PTA，都需要定义一个新的重写理论，而本文则为所有网使用单一的重写理论；对于 PTA，可通过对符号状态进行“标准”折叠来获得所需的符号可达性属性，而 PITPN 则需要开发一种新的折叠机制； [4] 中的分析不包括对时序逻辑公式进行模型检验；等等。

此外，许多实时系统也已使用基于 SMT 的重写进行形式化分析，包括 PLC ST程序 [23], 虚拟同步信息物理系统 [20,21,22],和软智能体 [32]。这些论文与我们的工作不同之处在于，它们使用受保护项 [9,10] 来缩减状态空间，而不是折叠，并且不考虑参数综合问题。

9 结论

我们为 PITPN 提供了一种“具体的”重写逻辑语义，并证明该语义与 [47]中此类网的语义双模拟。然而，这个模型不可执行；此外，使用 Real-Time Maude 式“时间采样”进行显式状态 Maude 分析，会导致对 PITPN 这类稠密时间系统的分析不健全。因此，我们系统地转换了这个模型，将其转化为一种“符号”重写模型，从而可以结合 Maude 和 SMT 求解进行可靠且完备的符号分析。

我们展示了如何使用 Maude-with-SMT 实现 PITPN 工具 Roméo 所支持的几乎所有形式化分析和参数综合。此外，我们还展示了 Maude-with-SMT 如何为 PITPN 提供额外功能，包括从参数化初始标识中综合初始标识（而不仅仅是发射界限），以满足期望性质；进行完整的线性时序逻辑模型检验；以及使用用户定义的执行策略进行分析。我们开发了一种针对符号状态的新“折叠”方法，因此，只要对应的 Roméo 分析能够终止，使用 Maude-with-SMT 进行的符号可达性分析也会终止。

我们在若干基准测试上比较了 Roméo 与我们结合SMT的Maude方法的性能。结果表明，在许多情况下，Maude 与 SMT 求解器 Yices 结合使用时优于 Roméo，而与 Z3 结合使用时则明显更慢。我们还发现，Roméo 有时无法找到（已存在的）解，并且某些执行的输出中包含消息“可能”，这表明 Roméo 正在计算一个近似值。如第7节所述，这可能是由 Roméo 中实现的 search 探索机制导致的。Maude 的 search 命令采用广度优先策略，因此能够保证完备性（如果给定状态可达，最终就会找到它）。此外，约束运算交由最先进的 SMT 求解器处理。我们还指出，Maude 的规约与相应的数学定义非常接近。因此，更容易检查实现的正确性；再加上 Maude 的元编程功能，开发、测试和评估不同的分析算法也很容易。

本文不仅为 PITPN 增添了新功能，还表明，即使是像我们这个受 Real-Time Maude 启发的 PITPN 解释器这样的模型——它包含函数、方程和无界标识——也可以轻松转化为符号重写理论，而 Maude-with-SMT 能够据此对稠密时间系统进行非常有用且可靠、完备的分析。

未来的工作应包括：实现 Maude-SE 在 Yices2 中处理量词所需的绑定，从而提升使用 $\mathcal{R}_1^{fS}$ 进行分析的性能；扩展 Maude 的 LTL 模型检查器，使其成为基于完整 SMT 的（含折叠）定时 LTL 和 CTL 模型检查器，从而涵盖 Roméo 提供的所有分析；开发更丰富的定时策略语言，以控制 PITPN 的执行；并探索理论变换，以对 Real-Time Maude 规约进行可靠且完备的符号分析。

参考文献

1. AlTurki, M., Dhurjati, D., Yu, D., Chander, A., Inamura, H.：软件系统时序属性的形式化规约与分析。载于：软件工程基础方法（FASE 2009）。计算机科学讲义，第5503卷，第262–277页。施普林格（2009） 2. André, E., Pellegrino, G., Petrucci, L.：含抑制弧的时间 Petri 网的精确鲁棒性分析。载于：定时系统的形式化建模与分析（FORMATS'13）。计算机科学讲义，第8053卷，第1–15页。施普林格（2013） 3. Andreychenko, A., Magnin, M., Inoue, K.：使用参数化模型检测分析振荡生物系统的韧性属性。《生物系统》**149**，50–58（2016） 4. Arias, J., Bae, K., Olarte, C., Ölveczky, P.C., Petrucci, L., Rømming, F.：参数化定时自动机的重写逻辑语义与符号分析。载于：第8届ACM SIGPLAN安全关键系统形式化技术国际研讨会论文集（FTSCS 2022）。第3–15页。ACM（2022） 5. Arias, J., Bae, K., Olarte, C., Ölveczky, P.C., Petrucci, L., Rømming, F.：pitpn2maude（2023），https://depot.lipn.univ-paris13.fr/arias/pitpn2maude 6. Bae, K., Escobar, S., Meseguer, J.：使用窄化对无限状态系统进行抽象逻辑模型检测。载于：重写技术与应用（RTA 2013）。LIPIcs，第21卷，第81–96页。Schloss Dagstuhl - Leibniz信息学中心（2013） 7. Bae, K., Krisiloff, J., Meseguer, J., Ölveczky, P.C.：使用多速率PALS设计和验证分布式信息物理系统：飞机转向控制系统案例研究。《科学计算程序》**103**，13–50（2015）。https://doi.org/10 011 8 1016 2014 09 . / j.scico. . . . 8. Bae, K., Ölveczky, P.C., Feng, T.H., Lee, E.A., Tripakis, S.：使用 Real-Time Maude 验证分层 Ptolemy II 离散事件模型。《科学计算程序》**77**（12），1235–1271（2012） 9. Bae, K., Rocha, C.：用于模SMT重写的受保护项。载于：组件软件形式化方面国际会议（FACS 2017）。第78–97页。施普林格（2017） 10. Bae, K., Rocha, C.：使用受保护项进行模SMT重写的符号状态空间约简。《科学计算程序》**178**，20–42（2019） 11. Barbosa, P.E.S., Barros, J.P., Ramalho, F., Gomes, L., Figueiredo, J., Moutinho, F., Costa, A., Aranha, A.：SysVeritas：用于在嵌入式系统设计中验证 IOPT 网及其执行语义的框架。载于：可持续发展技术创新——第二届IFIP WG 5.5/SOCOLNET计算、电气与工业系统博士会议（DoCEIS 2011）。IFIP信息与通信技术进展，第349卷，第256–265页。施普林格（2011） 12. Capra, L.：Maude中可重构PT网的规范化。载于：可达性问题（RP 2022）。计算机科学讲义，第13608卷，第160–177页。施普林格（2022） 13. Capra, L.：重写逻辑与 Petri 网：可重构分布式系统的自然模型。载于：分布式计算与智能技术（ICDCIT 2022）。计算机科学讲义，第13145卷，第140–156页。施普林格（2022） 14. Clarke, E.M., Grumberg, O., Peled, D.A.：《模型检测》。麻省理工学院出版社（2001） 15. Clavel, M., Durán, F., Eker, S., Escobar, S., Lincoln, P., Martí-Oliet, N., Meseguer, J., Rubio, R., Talcott, C.：《Maude手册》（3.2.1版）。SRI国际（2022），网址：http://maude.cs.illinois.edu 16. Clavel, M., Durán, F., Eker, S., Lincoln, P., Martí-Oliet, N., Meseguer, J., Talcott, C.L.：《Maude全解——高性能逻辑框架》，计算机科学讲义，第4350卷。施普林格（2007） 17. Coullon, H., Jard, C., Lime, D.：面向安全高效分布式软件调试设计的集成模型检测。载于：集成形式化方法（IFM 2019）。计算机科学讲义，第11918卷，第120–137页。施普林格，Cham（2019） 18. Grabiec, B., Traonouez, L., Jard, C., Lime, D., Roux, O.H.：使用参数化时间Petri网的展开进行诊断。载于：定时系统的形式化建模与分析（FORMATS 2010）。计算机科学讲义，第6246卷，第137–151页。施普林格（2010） 19. Jensen, K., Kristensen, L.M.：《有色Petri网——并发系统建模与验证》。施普林格（2009） 20. Lee, J., Bae, K., Ölveczky, P.C.：HybridSynchAADL的扩展及其在协作式自主无人机中的应用。载于：形式化方法、验证与确认的应用：适应与学习（ISoLA 2022）。第47–64页。施普林格（2022） 21. Lee, J., Bae, K., Ölveczky, P.C., Kim, S., Kang, M.：AADL中虚拟同步信息物理系统的建模与形式化分析。《软件技术转让国际期刊》第1–38页（2022） 22. Lee, J., Kim, S., Bae, K., Ölveczky, P.C.：HybridSynchAADL：AADL中虚拟同步信息物理系统的建模与形式化分析。载于：计算机辅助验证（CAV 2021）。计算机科学讲义，第12759卷，第491–504页。施普林格（2021）

23. Lee, J., Kim, S., Bae, K.：使用模SMT重写对PLC ST程序进行有界模型检测。载于：第8届ACM SIGPLAN安全关键系统形式化技术国际研讨会论文集（FTSCS 2022）。第56–67页。ACM（2022）24. Lien, E., Ölveczky, P.C.：IETF多播协议的形式化建模与分析。载于：第七届IEEE软件工程与形式化方法国际会议（SEFM 2009）。第273–282页。IEEE计算机学会（2009）25. Lime, D., Roux, O.H., Seidner, C.：参数化时间Petri网的代价问题。Fundam. Informaticae **183**(1-2), 97 2083 26 123 2021 10 3233 2021 2083, 10 3233 2021 – ( )。https://doi.org/ . /FI- - https://doi.org/ . /FI- - 。Lime, D., Roux, O.H., Seidner, C., Traonouez, L.：Romeo：带秒表的Petri网参数化模型检测器。载于：系统构造与分析的工具和算法（TACAS 2009）。计算机科学讲义，第5505卷，第54–57页。施普林格（2009）27. Merlin, P.M.：计算系统可恢复性研究。博士论文，加利福尼亚大学欧文分校，美国加利福尼亚州欧文（1974）28. Meseguer, J.：作为并发统一模型的条件重写逻辑。Theor. Comput. Sci. **96**(1), 73–155（1992）29. Meseguer, J.：重写逻辑的二十年。J. Log. Algebraic Methods Program. **81**(7-8), 721–781（2012）30. Meseguer, J.：广义重写理论、相干性完备化与符号方法。J. Log. Algebraic Meth- ods Program. **110** （2020）31. Meseguer, J., Montanari, U.：Petri网是幺半群。信息与计算 **88**(2), 105–155（1990）32. Nigam, V., Talcott, C.L.：使用模SMT重写自动证明信息物理系统的安全性。载于：重写逻辑及其应用（WRLA 2022）。计算机科学讲义，第13252卷，第212–229页。施普林格（2022）33. Ölveczky, P.C.：Real-Time Maude中嵌入式系统建模语言的语义、仿真与形式化分析。载于：形式化建模：参与者、开放系统、生物系统——纪念Carolyn Talcott七十寿辰论文集，计算机科学讲义，第7000卷，第368–402页。施普林格（2011）34. Ölveczky, P.C.：Real-Time Maude及其应用。载于：重写逻辑及其应用（WRLA 2014）。计算机科学讲义，第8663卷，第42–79页。施普林格（2014）35. Ölveczky, P.C., Boronat, A., Meseguer, J.：在Real-Time Maude中对行为AADL模型进行形式化语义分析。载于：分布式系统形式化技术，联合举办的第12届IFIP WG 6.1国际会议FMOODS 2010和第30届IFIP WG 6.1 FORTE 2010。计算机科学讲义，第6117卷，第47–62页。施普林格（2010）36. Ölveczky, P.C., Caccamo, M.：在Real-Time Maude中对CASH调度算法进行形式化仿真与分析。载于：软件工程基础方法（FASE 2006）。计算机科学讲义，第3922卷，第357–372页。施普林格（2006）37. Ölveczky, P.C., Meseguer, J.：重写逻辑中的实时系统与混合系统规约。Theor. Comput. Sci. **285**(2), 359–405（2002）38. Ölveczky, P.C., Meseguer, J.：Real-Time Maude的抽象与完备性。载于：第6届重写逻辑及其应用国际研讨会（WRLA 2006）。理论计算机科学电子札记，第174卷，第5–27页。Elsevier（2006）39. Ölveczky, P.C., Meseguer, J.：Real-Time Maude的语义与语用。High. Order Symb. Comput. **20**(1- 2), 161–196（2007）40. Ölveczky, P.C., Meseguer, J.：Real-Time Maude工具。载于：系统构造与分析的工具和算法（TACAS 2008）。计算机科学讲义，第4963卷，第332–336页。施普林格（2008）41. Ölveczky, P.C., Meseguer, J., Talcott, C.L.：在Real-Time Maude中对AER/NCA主动网络协议套件进行规约与分析。Formal Methods Syst. Des. **29**(3), 253–293（2006）42. Ölveczky, P.C., Thorvaldsen, S.：在Real-Time Maude中对无线传感器网络算法进行形式化建模、性能估计与模型检验。Theor. Comput. Sci. **410**(2-3), 254–280（2009）43. Padberg, J., Schulz, A.：使用Maude对可重构Petri网进行模型检验。载于：第9届图变换国际会议（ICGT 2016）。计算机科学讲义，第9761卷，第54–70页。施普林格（2016）44. Parquier, B., Rioux, L., Henia, R., Soulat, R., Roux, O.H., Lime, D., André, É.：应用参数化模型检测技术重用实时关键系统。载于：安全关键系统形式化技术（FTSCS 2016）。计算机与信息科学通信，第694卷，第129–144页。施普林格（2017）45. Rocha, C., Meseguer, J., Muñoz, C.A.：模SMT重写与开放系统分析。J. Log. Algebraic Methods Program. **86**(1), 269–297（2017）46. Stehr, M., Meseguer, J., Ölveczky, P.C.：重写逻辑作为Petri网的统一框架。载于：统一Petri网，Petri网进展。计算机科学讲义，第2128卷，第250–303页。施普林格（2001）47. Traonouez, L., Lime, D., Roux, O.H.：使用状态类图对带秒表的时间Petri网进行参数化模型检测。载于：Cassez, F., Jard, C.（编）定时系统的形式化建模与分析（FORMATS 2008）。计算机科学讲义，第5215卷，第280–294页。施普林格（2008）

48. Traonouez, L., Lime, D., Roux, O.H.：秒表 Petri 网的参数化模型检测。《J. Univers. Comput. Sci.》15(17), 3273–3304 (2009) 49. Vernadat, F., Berthomieu, B.：时间 Petri 网的状态空间抽象。载于：Son, S.H., Lee, I., Leung, J.Y. (编)，《实时与嵌入式系统手册》。Chapman and Hall/CRC (2007) 50. Wang, J.：时间 Petri 网。载于：《定时 Petri 网：理论与应用》，第 63–123 页。施普林格 (1998) 51. Yu, G., Bae, K.：Maude-SE：Maude 与可满足性模理论求解器的紧密集成。载于：WRLA@ETAPS 初步论文集。第 220–232 页 (2020)

A 基准测试数据

模型	到达的库所 Roméo (毫秒)		Maude（毫秒）				
			$\mathcal{R}_0^S$		$\mathcal{R}_1^S$		$\mathcal{R}_1^{fS}$
			Yices	Z3	Yices	Z3	Z3
教程	开始	4.0	**0.0**	16.0	**0.0**	15.0	13.0
教程	子开始	6.0	**1.0**	83.0	**1.0**	82.0	145.0
教程	父级续延	10.0	**1.0**	83.0	**1.0**	82.0	146.0
教程	子级完成	**7.0**	10.0	424.0	10.0	414.0	320.0
教程	父节点完成	**7.0**	11.0	453.0	11.0	445.0	313.0
教程	已加入	**4.0**	29.0	1005.0	30.0	977.0	773.0
生产者_消费者	项目就绪	**2.0**	**2.0**	164.0	3.0	158.0	117.0
生产者_消费者	缓冲区	**3.0**	7.0	336.0	7.0	327.0	217.0
生产者_消费者	已接收项目	**TO**	10.0	429.0	**9.0**	509.0	365.0
生产者_消费者	就绪消费者	2.0	**0.0**	16.0	**0.0**	15.0	14.0
生产者_消费者	就绪生产者	2.0	**0.0**	15.0	**0.0**	15.0	14.0
调度	就绪1	3.0	**0.0**	16.0	**0.0**	15.0	14.0
调度	就绪2	3.0	**0.0**	16.0	**0.0**	15.0	14.0
调度	就绪3	3.0	**0.0**	15.0	**0.0**	15.0	14.0
调度	结尾1	**4.0**	8.0	327.0	8.0	324.0	201.0
调度	结尾2	**6.0**	37.0	1054.0	37.0	1039.0	566.0
调度	结束3	**8.0**	118.0	2422.0	119.0	2364.0	1015.0

表2：EF($p > 0$)的时间

模型	到达的库所 Roméo (毫秒)		Maude（毫秒）				
			$\mathcal{R}_0^{\mathbf{S}}$		$\mathcal{R}_1^{\mathbf{S}}$		$\mathcal{R}_1^{fS}$
			Yices	Z3	Yices	Z3	Z3
教程	开始	**5.0**	57.0	1594.0	56.0	1541.0	1001.0
教程	子开始	TO	**172.0**	3584.0	**172.0**	3459.0	1705.0
教程	父级续延	TO	**171.0**	3604.0	172.0	3608.0	1710.0
教程	子级完成	TO	TO	TO	TO	TO	TO
教程	父节点完成	TO	262316.0	TO	224865.0	TO	**59594.0**
教程	已加入	TO	TO	TO	TO	TO	TO
生产者 _消费者 项		TO	TO	TO	TO	TO	TO
生产者 _消费者	缓冲区	**4.0**	49.0	1368.0	48.0	1338.0	713.0
生产者 _消费者 已		TO	TO	TO	TO	TO	TO
生产者 _消费者 就		TO	TO	TO	TO	TO	TO
生产者 _消费者 就		TO	TO	TO	TO	TO	TO
调度	ready1	**27.0**	TO	TO	TO	TO	154670.0
调度	就绪2	**27.0**	TO	TO	TO	TO	161968.0
调度	就绪3	**27.0**	TO	TO	TO	TO	150771.0
调度	结尾1	**41.0**	TO	TO	TO	TO	158852.0
调度	结尾2	**27.0**	TO	TO	TO	TO	161629.0
调度	结尾3	**27.0**	TO	TO	TO	TO	157329.0

表 3：EF($p > 1$) 的时间

B 结果证明

B.1 定理 1 的证明

证明。 (i) 根据定义 $a_0 = (M_0, J) \approx (\llbracket M_0 \rrbracket_{\mathcal{R}_0}$: 时钟初始化($\llbracket \mathcal{N} \rrbracket_{\mathcal{R}_0}$) : $\llbracket \mathcal{N} \rrbracket_{\mathcal{R}_0}$)，由于时钟初始化(...) 中所有时钟均为 0，因此这些时钟满足定义 5 中的所有约束，因为 $I = J$ 在初始状态中。(ii) 由以下两个引理得出。

引理 1。 若 $(M, I) \longrightarrow (M', I')$ 且 $(M, I) \approx (\llbracket M \rrbracket_{\mathcal{R}_0}$: 时钟 : $\llbracket \mathcal{N} \rrbracket_{\mathcal{R}_0})$，则存在一个时钟′，使得($\llbracket M \rrbracket_{\mathcal{R}_0}$: 时钟 : $\llbracket \mathcal{N} \rrbracket_{\mathcal{R}_0}) \mapsto (\llbracket M' \rrbracket_{\mathcal{R}_0}$: 时钟′: $\llbracket \mathcal{N} \rrbracket_{\mathcal{R}_0})$ 且 $(M', I') \approx (\llbracket M' \rrbracket_{\mathcal{R}_0}$: 时钟′: $\llbracket \mathcal{N} \rrbracket_{\mathcal{R}_0})$。

证明。 由于 $(M, I) \longrightarrow (M', I')$，因此存在一个中间状态对 $(M, I'') \in (T \cup \mathbb{R}_+)$，使得 $(M, I) \xrightarrow{\delta} (M, I'')$，且 $(M, I'') \xrightarrow{t_f} (M', I')$。

对于第一步 ($\xrightarrow{\delta}$)，由于 $(M, I) \xrightarrow{\delta} (M, I'')$，因此存在一个 δ 使得 $\forall t \in T$，要么是 $I''(t) = I(t)$，要么是 $I''(t) = \max(0, {}^{\uparrow}I(t) - \delta)$，并且 $I''(t)^{\uparrow} = I(t)^{\uparrow} - \delta$。两种情况下，我们都有 $\forall t \in T$，$I''(t)^{\uparrow} \geq 0$。现在，令 T $= \delta$，则必有 T <= mte($\llbracket M \rrbracket_{\mathcal{R}_0}$, *clocks*, $\llbracket \mathcal{N} \rrbracket_{\mathcal{R}_0}$)。这是因为，mte($\llbracket M \rrbracket_{\mathcal{R}_0}$, *clocks*, $\llbracket \mathcal{N} \rrbracket_{\mathcal{R}_0}$)定义为 $J(t)^{\uparrow}$ 与 t的时钟值之间的最小差值

模型	到达的库所	Roméo （毫秒）	Maude（毫秒）				
			$\mathcal{R}_0^S$		$\mathcal{R}_1^S$		$\mathcal{R}_1^{fS}$
			Yices	Z3	Yices	Z3	Z3
教程	开始	TO	1904.0	26445.0	**1572.0**	21787.0	6799.0
教程	子开始	TO	17760.0	164884.0	**15447.0**	180210.0	30043.0
教程	父级续延	TO	17813.0	164248.0	**15432.0**	181712.0	30186.0
教程	子级完成	TO	TO	TO	TO	TO	TO
教程	父节点完成	TO	TO	TO	TO	TO	TO
教程	已加入	TO	TO	TO	TO	TO	TO
生产者 _消费者 项		TO	TO	TO	TO	TO	TO
生产者 _消费者	缓冲区	TO	333.0	6618.0	**331.0**	6419.0	1564.0
生产者 _消费者 已		TO	TO	TO	TO	TO	TO
生产者 _消费者 就		TO	TO	TO	TO	TO	TO
生产者 _消费者 就		TO	TO	TO	TO	TO	TO
调度	就绪1	**44.0**	TO	TO	TO	TO	154580.0
调度	就绪2	**27.0**	TO	TO	TO	TO	157433.0
调度	就绪3	**27.0**	TO	TO	TO	TO	157285.0
调度	结尾1	**29.0**	TO	TO	TO	TO	149607.0
调度	结尾2	**27.0**	TO	TO	TO	TO	148758.0
调度	结尾3	**39.0**	TO	TO	TO	TO	151563.0

表4：EF($p > 2$) 的时间

模型	Roméo （毫秒）	Maude（毫秒）
教程	TO	TO
生产者 _消费者	**4.0**	1676.0
调度	**36.0**	186624.0

表5：1-安全时间验证

在所有 $t \in T$中。也就是说，这是一个使能变迁到达其区间右端点之前可以经过的最长时间。换言之，这是 δ的上限。因此，可以将时钟规则应用于（$[\![M]\!]_{\mathcal{R}_0}$：时钟：$[\![\mathcal{N}]\!]_{\mathcal{R}_0}$），并使所有使能时钟的时间前进 δ。

对于第二步（$\xrightarrow{t_f}$），由于（M，I''）$\xrightarrow{t_f}$（M'，I'），变迁 t_f处于使能状态，且 $^{\uparrow}I''(t_f) = 0$。由于$^{\uparrow}I(t_f) = 0$，根据（M，I''）的时钟 定义，变迁 t_f的时钟必须处于区间 $[^{\uparrow}J(t_f)\ J(t_f)^{\uparrow}]$内。这正是对前一个时钟规则应用所产生的状态应用 applyTransition-rule 的条件。

引理 2. 如果（$[\![M]\!]_{\mathcal{R}_0}$: 时钟: $[\![\mathcal{N}]\!]_{\mathcal{R}_0}$） $\mapsto b$并且
$(M, I) \approx ([\![M]\!]_{\mathcal{R}_0}$: 时钟 : $[\![\mathcal{N}]\!]_{\mathcal{R}_0})$，那么存在一个状态$(M', I') \in \mathcal{A}$，使得$(M, I) \longrightarrow (M', I')$且$(M', I') \approx b$。

证明。 由于 ($[\![M]\!]_{\mathcal{R}_0}$: 时钟 : $[\![\mathcal{N}]\!]_{\mathcal{R}_0}$) $\mapsto b$，我们知道存在一个中间状态 (M : CLOCKS : NET) $\in T_{\Sigma,\ \texttt{State}}$使得 ($[\![M]\!]_{\mathcal{R}_0}$: 时钟 : $[\![\mathcal{N}]\!]_{\mathcal{R}_0}$) tick $\longrightarrow$ (M : CLOCKS : NET) 并且 (M : CLOCKS : NET) applyTransition$\longrightarrow$
对于第一步 $\left(\xrightarrow{\texttt{tick}}\right)$，由于 ($[\![M]\!]_{\mathcal{R}_0}$: 时钟 : $[\![\mathcal{N}]\!]_{\mathcal{R}_0}$) tick $\longrightarrow$ (M : CLOCKS : NET)，存在一个 T <=mte($[\![M]\!]_{\mathcal{R}_0}$, $clocks$, $[\![\mathcal{N}]\!]_{\mathcal{R}_0}$)。现在，如同前一个引理中一样，由于最短到期时间是 δ 的上限，我们知道存在一个时间迁移 $(M, I) \xrightarrow{\delta} (M, I'')$，其中 δ 等于上述时钟规则应用中使用的 T，使得 (M : CLOCKS : NET) =($[\![M]\!]_{\mathcal{R}_0}$: 时钟 : $[\![\mathcal{N}]\!]_{\mathcal{R}_0}$)。
对于第二步 $\left(\xrightarrow{\texttt{applyTransition}}\right)$，由于 ($[\![M]\!]_{\mathcal{R}_0}$: 时钟 : $[\![\mathcal{N}]\!]_{\mathcal{R}_0}$) applyTransition$\longrightarrow$ b，因此必定存在一个变迁 t_f，它处于使能状态，且其时钟位于区间 $[\uparrow J(t_f)\ J(t_f)\uparrow]$内。根据 (M, I'') 上 时钟 的定义，这恰好是离散步 t_f到 (M, I'') 的条件。因此，$(M, I'') \xrightarrow{t_f} (M', I')$，并且 ($[\![M]\!]_{\mathcal{R}_0}$: 时钟 : $[\![\mathcal{N}]\!]_{\mathcal{R}_0}$) applyTransition$\longrightarrow$ ($[\![M']\!]_{\mathcal{R}_0}$: 时钟 : $[\![\mathcal{N}]\!]_{\mathcal{R}_0}$)。

B.2 定理 2 和 3 的证明

定理 2。

证明。 对于 ($\Leftarrow$) 这一方向，只需在 $\mathcal{R}_0$ 中遵循与 $\mathcal{R}_1$ 相同的执行策略。对于 ($\Rightarrow$)，只需对 $\mathcal{R}_0$ 轨迹进行如下（保持可达性的）修改：将连续应用两次时钟规则、分别使用 $T = t_1$ 和 $T = t_2$ 的操作，替换为一次使用 $T = t_1 + t_2$的 tick 操作。这样就足以证明，在 $\mathcal{R}_1$中可以得到相同的轨迹。

定理 3。

证明。 根据定理 2，我们知道可以在 $\mathcal{R}_0$ 中遵循 tickOk/tickNotOk 策略，从而生成一条等价轨迹。利用该轨迹，只需注意到，在 $\mathcal{R}_0$中使用 T= δ ($t \xrightarrow{\delta}_{\mathcal{R}_0} t'$) 的 tick 应用，与在 $\mathcal{R}_2$ 中使用同一 T 实例的 tick 应用相匹配，因此全局时钟恰好前进 δ 个时间单位，即可直接得出结果。

B.3 定理 4 和 5 的证明

定理 4

证明。 该结果可直接由模可满足性模理论重写 [45]的可靠性与完备性推出。更准确地说，根据 [45] ，我们知道：若 $\phi_t \parallel t \rightsquigarrow^* \phi_u \parallel u$ ，则存在某个 $t' \in [\![\phi_t \parallel t]\!]$ 和$u' \in [\![\phi_u \parallel u]\!]$，使得 $t' \longrightarrow^* u'$ ；若 $t' \longrightarrow^* u'$ 且满足 $t' \in [\![\phi_t \parallel t]\!]$，则存在 ϕ_u 和 t_u ，使得 $\phi_t \parallel t \rightsquigarrow^* \phi_u \parallel u$ 与$u' \in [\![\phi_u \parallel u]\!]$。

定理 5

证明。 设 $U \Downarrow_{\texttt{now}} = \phi'_u \parallel t'_u$和 $V \Downarrow_{\texttt{now}} = \phi'_v \parallel t'_v$，其中 变量集合($t'_u$)∩变量集合($t'_v$) $= \emptyset$。设 $X_u =$变量集合(ϕ'_u) \ 变量集合(t'_u)，且 $X_v =$ 变量集合(ϕ'_v) \ 变量集合(t'_v)。根据构造， t'_u, $t'_v \in T_{\Sigma\setminus\Sigma_0}(X_0)$， $\exists(U \Downarrow_{\texttt{now}}) = (\exists X_u)\phi'_u$， 且 $\exists(V \Downarrow_{\texttt{now}}) = (\exists X_v)\phi'_v$。只需证明 $[\![U \Downarrow_{\texttt{now}}]\!] \subseteq [\![V \Downarrow_{\texttt{now}}]\!]$ 当且仅当 $U \preceq V$。
($\Rightarrow$) 假设 $[\![U \Downarrow_{\texttt{now}}]\!] \subseteq [\![V \Downarrow_{\texttt{now}}]\!]$。那么， t'_u和 t'_v可通过 E 合一（见证为 $w \in [\![U \Downarrow_{\texttt{now}}]\!] \cap [\![V \Downarrow_{\texttt{now}}]\!]$）。由于 t'_v没有重复变量，且 E仅包含关于 $\Sigma\setminus\Sigma_0$的结构公理，根据匹配引理[45, 引理 5]，存在一个替换 σ，使得 $t'_u = t'_v\sigma$（模 ACU 相等）。由于 t'_u 的任何内置子项都是 X_0中的变量， σ 是一个重命名代换 σ： $X_0 \to X_0$，因此 $[\![V \Downarrow_{\texttt{now}}]\!] = [\![(V \Downarrow_{\texttt{now}})\sigma]\!]$。

假设 $\exists(U \Downarrow_{\texttt{now}}) \Rightarrow \exists(V \Downarrow_{\texttt{now}})\sigma$ 无效，也就是说，$((\exists X_u)\phi'_u) \wedge (\forall X_v)\neg\phi'_v\sigma$ 可满足。令 Y为 $((\exists X_u)\phi'_u) \wedge (\forall X_v)\neg\phi'_v\sigma$ 中自由变量的集合。注意， $Y =$ 变量集合(t'_u) $=$ 变量集合($t'_v\sigma$)。令 ρ: $Y \to T_{\Sigma_0}$ 为

一个表示 $(\exists X_u)\phi'_u \wedge (\forall X_v)\neg\phi'_v\sigma$ 的满足赋值的基项代换。那么， $t'_u\rho \in [\![U \Downarrow_{\texttt{now}}]\!]$但是 $t'_u\rho = t'_v\sigma\rho \notin [\![(V \Downarrow_{\texttt{now}})\sigma]\!] = [\![V \Downarrow_{\texttt{now}}]\!]$，这与假设矛盾。

(⇐) 假设 $U \preceq V$。存在一个替换 σ ： $X_0 \to X_0$ ，使得 $t'_u = t'_v\sigma$ 且 $\exists(U \Downarrow_{\texttt{now}}) \Rightarrow \exists(V \Downarrow_{\texttt{now}})\sigma$ 有效。令 Y 为 $\exists$ 中的自由变量集合。$(U \Downarrow_{\texttt{now}}) \Rightarrow \exists(V \Downarrow_{\texttt{now}})\sigma$。如上所述，$[\![V \Downarrow_{\texttt{now}}]\!] = [\![(V \Downarrow_{\texttt{now}})\sigma]\!]$ 且 $Y = \mathsf{vars}(t'_u) = \mathsf{vars}(t'_v\sigma)$。令 $w \in [\![U \Downarrow_{\texttt{now}}]\!]$。那么，对于某个基项代换 ρ_u， $w = t'_u\rho_u$ 和 $\phi'_u\rho_u$ 成立。根据 $\rho_u|_Y$ 中的赋值，我们可以构造一个赋值 $\mathcal{V}$ ，使 $\exists(U \Downarrow_{\texttt{now}})$ 为真；并且根据假设，也使 $\exists(V \Downarrow_{\texttt{now}})\sigma$ 为真。因此，存在一个基项代换 ρ_v（它在 $\mathcal{V}$ 中已赋值的取值上保持一致），使得 $\phi'_v\sigma\rho_v$ 成立且 $\rho_u|_Y = \rho_v|_Y$。注意， $w = t'_u\rho_u = t'_u(\rho_u|_Y) = t'_v\sigma(\rho_v|_Y) = t'_v\sigma\rho_v$。因此， $w \in [\![(V \Downarrow_{\texttt{now}})\sigma]\!]$。□

推论 1

证明。 假设 $(\mathcal{C}, (M, D) \Rightarrow)$ 是一个有限迁移系统，并为导出矛盾，假设从 $\mathsf{init}(\mathcal{N}, M, D)$ 可达的 $\rightsquigarrow_{\mathcal{R}_1^{fs}}$ 状态有无限多个。由于 $\left(T_{\Sigma,\texttt{State}}, \mathsf{init}(\mathcal{N}, M, D) \rightsquigarrow_{\mathcal{R}_1^{fs}}\right)$ 是有限分支的，因此必定存在一个如下形式的无限序列 $U_0 \rightsquigarrow_{\mathcal{R}_1^{fs}} U_1 \rightsquigarrow_{\mathcal{R}_1^{fs}} \cdots$，根据 $\rightsquigarrow_{\mathcal{R}_1^{fs}}$ 的定义， $U_j \not\preceq U_i$ 对于 $i < j$。由定理 5 可知 $[\![U_j]\!] \not\subseteq [\![U_i]\!]$。根据定理 4，这意味着每次变迁之后都会找到更多具体且不同的状态。因此，可达状态类不可能是有限的，从而导出矛盾。