

一种检测 TAL-freeness 的代数方法

彭 蓉¹⁾ 崔竞松²⁾ 曾祥勇³⁾

¹⁾(武汉大学软件工程国家重点实验室 武汉 430072)

²⁾(武汉大学计算机学院 武汉 430072)

³⁾(湖北大学数学与计算机科学学院 武汉 430062)

摘 要 时间动作锁(Time-Action-Lock, TAL)指的是实时系统处于一种时间无法继续同时又没有任何动作能够发生的状态. Behzad 和 Kozo 在时间自动机的几何学基础上提出了一种针对 TAL-freeness 的检测方法. 但该方法要求必须将需要检测的模型转化为一种逻辑语言 Rational Presburger Sentences 后才能进行检测, 因此使得验证过程比较繁琐. 文中提出了一种检测 TAL-freeness 的代数方法, 能够直接对系统模型进行直接验证, 并且能够定位死锁原因. 针对该方法, 文中还给出了相应算法并提供了正确性证明与性能分析.

关键词 时间自动机; 时间动作锁; 检测算法

中图法分类号 TP302

An Algebraic Approach for TAL-Freeness Detection

PENG Rong¹⁾ CUI Jing-Song²⁾ ZENG Xiang-Yong³⁾

¹⁾(State Key Laboratory of Software Engineering, Wuhan 430072)

²⁾(School of Computer Science, Wuhan University, Wuhan 430072)

³⁾(Faculty of Mathematics and Computer Science, Hubei University, Wuhan 430062)

Abstract A Time-Action-Lock (TAL) is a state of a real-time system at which neither time can progress nor an action can occur. Behzad and Kozo presented a TAL-freeness detection method based on the geometry of Timed Automata. It is realized by means of translating the problem to Rational Presburger Sentences that has its drawback of efficiency. In this paper, the authors present an algebraic approach for TAL-freeness detection, which can detect the TAL-freeness directly. Detailed correctness proofs and performance analysis are provided.

Keywords timed automata; time action lock; detection algorithm

1 引 言

死锁通常指的是系统中的某些进程或所有进程由于等待某些事件的发生而处于挂起状态. 死锁的存在常常意味着模型中存在错误或规范中存在不一致. 在实时系统中存在的各种死锁^[1-3,5-6,9]中, 时间锁(Time Lock)^[6]是一类比较重要的死锁. 系统中存在时间锁就意味着从某个特定时刻起系统时间

就无法继续; 而时间动作锁(Time-Action-Lock, TAL)^[6]则是一种特殊的时间锁, 它意味着系统在处于时间锁状态的同时没有任何动作能够发生. 因此, 时间动作锁指的是实时系统处于一种时间无法继续同时又没有任何动作能够发生的状态.

文献[3]提出了一种基于时间自动机(Timed Automata, TA)检测 TAL-freeness 的测试算法. 在时间自动机理论中, 时间是利用一组由 n 维欧氏空间 R^n 中的凸形区域构成的时间约束描述的^[6]. 因

此,作用于时间自动机的每个状态上的各类时间约束(例如时间不变式和警戒条件),都可以映射到 R^n 上的区域.因此,文献[3]提出的检测算法首先将每个状态对应的凸形区域转换为相应的一阶谓词逻辑的语句 Rational Presburger Sentences(RPS),然后利用 RPS 引擎^[7]验证这些语句.显然,由于此方法需要将时间约束转化为逻辑语言,再利用逻辑语言的推理功能进行测试,验证过程繁琐,同时当系统不满足 TAL-freeness 的条件时,也难以精确定位出错原因,因此有必要对此进行改进.

本文在研究时间自动机的几何意义与代数意义的基础上提出了一种检测 TAL-freeness 的测试方法.该方法首先利用几何学方法得到系统时间约束对应的欧氏空间,然后再利用其代数性质判定其是否满足 TAL-freeness 的条件.当系统不满足 TAL-freeness 的条件时,本方法能够对其冲突原因精确定位,从而为系统设计人员提供有效的系统修正指南.

本文将首先简单地介绍时间自动机的基本概念,然后引入在检测 TAL-freeness 时需要使用到的一些定义及定理;第 4 节主要描述检测 TAL-freeness 的代数方法,并对其进行正确性证明;根据该方法,第 5 节提出了相应的检测算法,并对其进行详细的性能分析;第 6 节给出一个相关实例;最后对全文进行总结并提出下一步工作的展望.

2 时间自动机

Alur 和 Dill 提出的时间自动机^[4]已经被广泛地应用于对实时系统各种行为属性的验证之中.根据本文的需要,我们在一定程度上对文献[3]中关于时间自动机的形式化定义进行扩展.

定义 1. 定义时钟变量集合 $\aleph = \{x_1, x_2, \dots, x_n\}$, $x_i \in R_+$, 其中时钟变量取值为非负实数.

定义 2. 定义 $c_1(\aleph)$ 表示一组形为 $x_i \sim q$ 的原子公式的集合,其中 $x_i \in \aleph$, 关系符 $\sim \in \{\leq, \geq, <, >, =\}$, $q \in Q_+$, Q_+ 为非负有理数集合.

定义 3. 定义 $c_2(\aleph)$ 表示一组形为 $x_i \sim q$, $x_i - x_j \sim q$ 的原子公式的集合,其中 $x_i, x_j \in \aleph$, $i \neq j$. q 与 \sim 的定义同上.

定义 4. 定义 $c(\aleph) = c_1(\aleph) \cup c_2(\aleph)$ 标识所有可能的约束的集合.此处,将 $x_i \sim q$ 与 $x_i - x_j \sim q$ 作为原子约束.

定义 5. 赋值(变量赋值)用映射 $v: \aleph \rightarrow R_+^n$ 描述,表示将一组实值赋给时钟集 \aleph . \aleph 中的所有时钟

都以相同速率增长.

时间自动机 TA 可以用如下 9 元组定义 $(S, E, G, A, \aleph, I, \mathfrak{R}, T, s_0)$:

1) $S = \{s_0, s_1, \dots, s_m\}$ 表示一个有限状态集,其中 $s_0 \in S$ 表示自动机 TA 的初始状态,其时钟的初始值全部置 0: $x_i = 0$;

2) E 为有限事件集;

3) G 为形为 $c(\aleph)$ 的有限警戒条件集;

4) A 为有限动作集;

5) \aleph 为有限的时钟变量集;

6) $I: S \rightarrow c(\aleph)$ 为将时间不变式绑定到具体状态的函数,意味着 TA 处于的任何状态必须满足该状态的时间不变式. TA 中任一状态的缺省不变式为 $\text{true}(x \geq 0)$. 此处,用 $\text{inv}(s)$ 表示状态 s 的时间不变式;

7) \mathfrak{R} 为形为 $x := e$ 的重置时钟集,其中 $x \in \aleph$;

8) $T \subseteq S \times E \times G \times A \times R \times S$ 为变迁关系集. T 中的每个元素形为 (s_i, e, g, a, r, s_j) , 其中, $s_i, s_j \in S$ 分别表示该变迁的原状态与目标状态, $e \in E$ 与 $g \in G$ 分别代表触发该变迁必须要满足的事件与警戒条件,而 $a \in A$ 与 $r \in R$ 则分别代表变迁发生后必须要执行的动作以及需要重置的时钟集. $s_i \xrightarrow{e, g, a, r} s_j$ 表示:当事件 e 发生同时警戒条件 g 为 true 时, TA 从状态 s_i 转换为新的状态 s_j , 同时执行动作 a , 并根据 r 重置时钟变量. 此处,用 $t = (e, g, a, r)$ 表示连接 s_i 与 s_j 的边,用 $\text{event}(t)$, $\text{guard}(t)$, $\text{action}(t)$ 和 $\text{reset}(t)$ 分别表示 e, g, a 和 r .

表示 1. 对任何状态 $s \in S$, 用 $^\circ s$ 表示以 s 为目标的状态集,用 s° 表示以 s 为起点的状态集.

由于本文重点在于 TAL-freeness 的检测方法分析,不妨假设 TA 能够得到其变迁发生的所有事件,从而使得 TA 能否继续仅仅取决于能否满足各状态的时钟约束.

3 TAL-freeness 定理

寻找并消除时间锁是实时系统分析和设计中所需解决的主要问题之一.

时间动作锁是时间锁的一种,具体系统状态实例如下.

例 1. 图 1 描述了一个在状态 s_0 具有 TAL 的时间自动机. 该自动机到达状态 $(s_0, 8)$ 后, 由于时间不变量 $x \leq 8$ 的约束使得其必须离开该状态, 但由于此时时间变量值为 8, 不能满足变迁上的时间约束 $6 \leq x < 7$ 的要求, 从而导致变迁无法发生, 满足

TAL 发生的条件.

在介绍本文提出的方法之前,我们需要简要地介绍一下相关的关键定义与结论^[3].

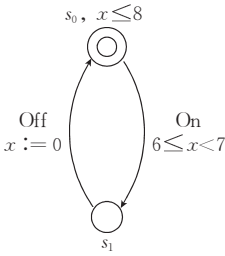


图 1 带有 TAL 的时间自动机

表示 2. 对每个限制 $c \in c(\mathbb{N})$, 在 R_+^n 的所有点中存在一个满足 c 的子集. 在后文中, 将用 \underline{c} 代表 $c \in c(\mathbb{N})$ 在 R_+^n 中对应的区域.

定义 6. 假设 $n \geq 1$, 对任意的, 定义函数 $\Gamma(x, \underline{c}) = \sup\{t \geq 0 \mid x + t \times \mathbf{1} \in \underline{c}\}$, 其中 $\mathbf{1} = (1, \dots, 1) \in R_+^n$. 如果存在 t 使得 $x + t \times \mathbf{1} \in \underline{c}$ 且 $\Gamma(x, \underline{c}) < \infty$, 则定义 $Fringe(x, \underline{c}) = x + \Gamma(x, \underline{c}) \times \mathbf{1}$; 如果 $\Gamma(x, \underline{c}) = \infty$, 则定义 $Fringe(x, \underline{c}) = \{\infty\}$; 如果没有 t 能使得 $x + t \times \mathbf{1} \in \underline{c}$, 则定义 $\Gamma(x, \underline{c}) = -\infty$ 且 $Fringe(x, \underline{c}) = \{-\infty\}$.

定义 7. 如果 $c_1, c_2 \in c(\mathbb{N})$ 且 $\underline{c_1} \subset \underline{c_2}$, 则定义 $Fringe(\underline{c_1}, \underline{c_2}) = \bigcup_{x \in \underline{c_1}} Fringe(x, \underline{c_2})$.

引理 1. 假设 $c_1 \in c(\mathbb{N})$ 且 c_2 是将 c_1 中所有原子公式中的 $<(>)$ 替换为 $\leq(\geq)$ 得到的时钟约束集, 则 $\underline{c_2}$ 是 $\underline{c_1}$ 的拓扑闭包, 即 $\underline{c_2} = \overline{\underline{c_1}} = \underline{c_1}$.

引理 2. 如果 $c \in c_1(\mathbb{N})$, \underline{c} 表示其对应的矩形区域, $(\alpha_1, \alpha_2, \dots, \alpha_n)$ 表示矩形区域 \underline{c} 的右上角坐标, 则对于 \underline{c} 中的任意点 $x = (x_1, x_2, \dots, x_n) \in \underline{c}$ 而言:

- 1) $\Gamma(x, c) = \min\{\alpha_i - x_i \mid 1 \leq i \leq n\}$;
- 2) $Fringe(x, c) = x + \min\{\alpha_i - x_i \mid 1 \leq i \leq n\} \times \mathbf{1}$.

引理 3. 如果 \underline{c} 是 $c \in c_2(\mathbb{N})$ 中的限制条件对应的非空区域, $c' \in c(\mathbb{N})$ 是将 c 中所有形如 $x_i - x_j \sim q$ 的约束去掉后形成的时钟集, 并假设 $\underline{c'}$ 表示 c' 所对应的矩形区域, 则 $\underline{c} \subset \underline{c'}$ 且对于任意的 $x \in \underline{c}$, 有 $Fringe(x, \underline{c}) = Fringe(x, \underline{c'})$.

定理 1. 假设 TA 是一个时间自动机, 则 TA 无时间自动锁的必要条件为 TA 中的每个状态 s 均需满足式(1):

$$Fringe(A_s, B_s) \subset \overline{C_s} \quad (1)$$

其中,

- 1) 若 $^\circ s \neq \emptyset$, 则 $A_s = \bigcup_{e \in ^\circ s} A_{s,e}$, 其中 $A_{s,e} = \underline{guard(t)} \cap \underline{inv(s)} \cap \underline{reset(t)}$; 否则, $A_s = \underline{inv(s)}$;
- 2) $B_s = \underline{inv(s)}$;
- 3) $C_s = \bigcup_{e \in s^\circ} C_{s,e}$, 其中 $C_{s,e} = \underline{guard(t)} \cap$

$\underline{inv(s)}$;

4) $\overline{C_s}$ 表示 C_s 的拓扑闭包.

详细证明请参见文献[3].

上述定理表明, 判定一个时间自动机是否无时间动作锁只需判定自动机中的每一个状态的输入变迁对应的警戒条件所确定的时间区域相对于该状态时间不变式所确定的时间区域的边界是否被该状态的输出变迁上的警戒条件对应的时间区域的拓扑闭包所包含.

针对上述定理, 文献[3]提出了一种检测 TAL-freeness 的几何方法. 但由于该方法不能支持直接计算, 使得设计者不得不引入 RPS 引擎来完成检测任务. 在此基础上, 本文提出了下述检测 TAL-freeness 的代数方法.

4 检测 TAL-freeness 的代数方法

本节介绍的检测 TAL-freeness 的代数方法建立在定理 1 的基础上. 考察定理 1 中的式(1), 不难发现, 关键在于如何验证 $Fringe(\underline{c_1}, \underline{c_2}) \subset \underline{c_3}$, 其中, $c_1, c_2, c_3 \in c(\mathbb{N})$ 且 $\underline{c_1} \subset \underline{c_2}$. 为了检验该属性, 首先必须引入以下定义与引理.

表示 3. 假设 \underline{c} 为形如 $I_1 \times I_2 \times \dots \times I_n$ 的矩形区域, 其中 I_i 表示非负实数区间, 则 \underline{c} 可以用 $\{[\beta_i, \alpha_i] \mid 1 \leq i \leq n\}$ 表示. 任意的 n 维向量 x 可以用 $x = (x_1, x_2, \dots, x_n)$ 表示.

定义 8. 定义区域 \underline{c} 的上边界为

$$Verge(\underline{c}) = \bigcup_{i=1}^n ((x_i = \alpha_i) \wedge (\bigcup_{1 \leq j \neq i \leq n} x_j \in [\beta_j, \alpha_j])).$$

引理 4. $x \in Verge(\underline{c})$ 当且仅当

$$\exists i \in [1, n] (x_i = \alpha_i) \wedge (\bigcup_{1 \leq j \neq i \leq n} x_j \in [\beta_j, \alpha_j]).$$

证明可以直接通过定义 8 推导得到.

引理 5. $x \in Fringe(\underline{c_1}, \underline{c_2})$ 当且仅当

$$x \in Verge(\underline{c_2}) \wedge (\exists x' \in \underline{c_1}, t \in \mathbb{R}_+, x = x' + t \times \mathbf{1}),$$

其中 $\underline{c_2} = \{[\beta_i, \alpha_i] \mid 1 \leq i \leq n\}$ 且 $\mathbf{1} = (1, \dots, 1) \in R_+^n$.

证明可以通过定义 6~8 直接得到. 注意, 证明充分性时需要事实: 系统中的所有时钟同步变化.

引理 6. $x \notin \underline{c}$ 当且仅当 $\exists k \in [1, n] x_k \notin [\beta_k, \alpha_k]$.

证明可以通过表示法 3 得到.

假设 $c_1, c_2 \in c_1(\mathbb{N})$ 且 $\underline{c_1} \subset \underline{c_2}$, 用 $\underline{c_1} = \{[\beta_i, \alpha_i] \mid 1 \leq i \leq n\}$ 与 $\underline{c_2} = \{[\beta_i, \alpha_i] \mid 1 \leq i \leq n\}$ 分别表示 $\underline{c_1}, \underline{c_2}$.

定义 9. 定义函数 $\delta_i = \{\min(\alpha_k - \alpha_i + \alpha_i - \beta_k) \mid 1 \leq k \leq n\}$.

定义 10. 定义函数 ϕ_i : 若 $\delta_i < 0$, 则定义 $\phi_i = \emptyset$; 否则定义 $\phi_i = \{(x_i = \alpha_i) \wedge \bigcup_{1 \leq j \neq i \leq n} x_j \in \sigma_j\}$, 其中: $\sigma_j = [\beta_j, \alpha_j] \cap A_j, A_j = [\alpha_i - a_i + b_j, \alpha_i - a_i + a_j + \delta_j]$.

定义 11. 定义函数 $F(\underline{c}_1, \underline{c}_2) = \bigcup_{i=1}^n \phi_i$.

定理 2. 假设 $\underline{c}_1, \underline{c}_2 \in c_1(\mathbb{N})$, 且 $\underline{c}_1 \subset \underline{c}_2, \underline{c}_3$ 表示 $c_1(\mathbb{N})$ 中有限个时钟约束对应的矩形区域, 并将 $\underline{c}_1, \underline{c}_2, \underline{c}_3$ 分别用 $\underline{c}_1 = \{[b_i, a_i] \mid 1 \leq i \leq n\}, \underline{c}_2 = \{[\beta_i, \alpha_i] \mid 1 \leq i \leq n\}$ 和 $\underline{c}_3 = \{[r_i, R_i] \mid 1 \leq i \leq n\}$ 表示, 则

$$Fringe(\underline{c}_1, \underline{c}_2) \subset \underline{c}_3 \Leftrightarrow F(\underline{c}_1, \underline{c}_2) \subset \underline{c}_3.$$

证明. 1) 必要性. 即证明 $F(\underline{c}_1, \underline{c}_2) \subset \underline{c}_3 \Rightarrow Fringe(\underline{c}_1, \underline{c}_2) \subset \underline{c}_3$.

若 $Fringe(\underline{c}_1, \underline{c}_2) \subset F(\underline{c}_1, \underline{c}_2)$, 则以上命题显然成立, 因此, 可以将其转化为证明:

$$\forall x \in Fringe(\underline{c}_1, \underline{c}_2) \Rightarrow x \in F(\underline{c}_1, \underline{c}_2).$$

假设 $x \in Fringe(\underline{c}_1, \underline{c}_2)$. 由引理 5 可得 $x \in Verge(\underline{c}_2) \wedge (\exists x' \in \underline{c}_1, x = x' + t \times \mathbf{1}, t \in \mathbb{R}_+)$.

由于 $x \in Verge(\underline{c}_2)$, 不妨假设 x 可以表示为 $(x_m = \alpha_m) \wedge (\bigcup_{1 \leq j \neq m \leq n} x_j \in [\beta_j, \alpha_j])$, 且存在 x' 与 t , 使得 $x' \in \underline{c}_1, t \in \mathbb{R}_+$ 且 $x = x' + t \times \mathbf{1}$. 此时, 需要证明 $x \in \phi_m, F(\underline{c}_1, \underline{c}_2)$ 的第 m 个子集. 由于 $x_m = \alpha_m$, 因此, 只需证明 $\forall 1 \leq j \neq i \leq n, x_j \in \sigma_j$. 由于 $x \in Verge(\underline{c}_2)$, 显然 $x_j \in [\beta_j, \alpha_j]$. 因此, 我们只需证明 $x_j \in A_j$:

由 $\{x_j = x'_j + t \mid 1 \leq j \leq n\}$ 有

$$x'_m = x_m - t = \alpha_m - t,$$

由于 $x' \in \underline{c}_1$, 则 $b_m \leq x'_m = \alpha_m - t \leq a_m$, 即

$$\alpha_m - a_m \leq t \leq \alpha_m - b_m,$$

有 $\forall 1 \leq j \leq n, b_j \leq x'_j \leq a_j$,

$$(\alpha_m - a_m) + b_j \leq x_j = x'_j + t \leq (\alpha_m - b_m) + a_j.$$

由于 $A_j = [(\alpha_m - a_m) + b_j, (\alpha_m - a_m) + a_j + \delta_m]$ 且 $\delta_m = \{\min(\alpha_i - a_m + a_m - b_i) \mid 1 \leq i \leq n\}$, 故而存在两种情况:

a) 当 $i = m$ 时, 有 $\delta_m = \alpha_m - b_m$. 因此, $A_j = [(\alpha_m - a_m) + b_j, (\alpha_m - b_m) + a_j]$, 即 $x_j \in A_j$.

b) 当 $i = k \neq m$ 时, 有 $\delta_m = \alpha_k - a_m + a_m - b_k$. 因此, $A_j = [(\alpha_m - a_m) + b_j, \alpha_k - b_k + a_j]$, 由于 $x_k - x_j = x'_k - x'_j \geq b_k - a_j$, 即 $x_j \leq x_k - b_k + a_j \leq \alpha_k - b_k + a_j$, 因此 $x_j \in A_j$.

故而, $x \in F(\underline{c}_1, \underline{c}_2)$. 必要性得证.

2) 充分性. 即证明 $Fringe(\underline{c}_1, \underline{c}_2) \subset \underline{c}_3 \Rightarrow F(\underline{c}_1, \underline{c}_2) \subset \underline{c}_3$.

反证法. 此命题等价于证明:

$$\exists y \in F(\underline{c}_1, \underline{c}_2) \wedge y \notin \underline{c}_3 \Rightarrow \exists x \in Fringe(\underline{c}_1, \underline{c}_2) \wedge x \notin \underline{c}_3,$$

假设 $y \in F(\underline{c}_1, \underline{c}_2) \wedge y \notin \underline{c}_3$,

由 $y \in F(\underline{c}_1, \underline{c}_2)$, 则 $\exists i \in [1, n] y \in \phi_i$,

有 $y_i = \alpha_i$ 且 $y_j \in \sigma_j$, 其中 $\sigma_j = ([\beta_j, \alpha_j] \cap A_j), 1 \leq j \neq i \leq n$.

又由于 $y \notin \underline{c}_3$, 则 $\exists k \in [1, n] y_k \notin [r_k, R_k]$.

此时, 可以将问题分为以下两种情况:

a) $i = k$

有 $y_i = y_k \notin [r_k, R_k] = [r_i, R_i]$, 即 $\alpha_i \notin [r_i, R_i]$.

构造向量 $x \in R^+$, 使得 $x_i = \alpha_i \wedge \{x_j = \min(\sigma_j) \mid j \neq i\}$, 则 $x \in F(\underline{c}_1, \underline{c}_2), x \in Verge(\underline{c}_2)$ 且 $x \notin \underline{c}_3$.

假设 $x' = x - (\alpha_i - a_i) \times \mathbf{1}$, 容易证明 $x' \in \underline{c}_1$ (使用假设条件 $\underline{c}_1 \subset \underline{c}_2$). 由引理 5 可知, $x \in Fringe(\underline{c}_1, \underline{c}_2)$.

b) $i \neq k$

由 $y \in \phi_i$, 则 $y_k \in [\beta_k, \alpha_k] \cap A_k$.

$$\begin{aligned} \text{又由于 } A_k &= [\alpha_i - a_i + b_k, \alpha_i - a_i + a_k + \delta_k] \\ &= [\alpha_i - a_i + b_k, \alpha_i - a_i + a_k] \cup \\ &\quad [\alpha_i - a_i + a_k, \alpha_i - a_i + a_k + \delta_k], \end{aligned}$$

可以分以下两种情况讨论:

a) $y_k \in [\beta_k, \alpha_k] \cap [\alpha_i - a_i + b_k, \alpha_i - a_i + a_k]$,

构造向量 $x \in R^+$, 使得 $x_i = \alpha_i \wedge x_k = y_k \wedge \{x_j = \min(\sigma_j) \mid j \neq i, k\}$, 则 $x \in F(\underline{c}_1, \underline{c}_2), x \in Verge(\underline{c}_2)$ 且 $x \notin \underline{c}_3$.

同样, 假设 $x' = x - (\alpha_i - a_i) \times \mathbf{1}$, 不难证明 $x' \in \underline{c}_1$. 由引理 5 可知, $x \in Fringe(\underline{c}_1, \underline{c}_2)$.

b) $y_k \in [\beta_k, \alpha_k] \cap [\alpha_i - a_i + a_k, \alpha_i - a_i + a_k + \delta_k]$.

构造向量 $x \in R^+$, 使得 $x_i = \alpha_i \wedge x_k = y_k \wedge \{x_j = \min(\alpha_j, a_j + y_k - a_k) \mid j \neq i, k\}$, 则 $x \in F(\underline{c}_1, \underline{c}_2), x \in Verge(\underline{c}_2)$ 且 $x \notin \underline{c}_3$.

假设 $x' = x - (y_k - a_k) \times \mathbf{1}$, 可以证明 $x' \in \underline{c}_1$. 根据引理 3 可知, $x \in Fringe(\underline{c}_1, \underline{c}_2)$.

因此, 命题 $\exists y \in F(\underline{c}_1, \underline{c}_2) \wedge y \notin \underline{c}_3 \Rightarrow \exists x \in Fringe(\underline{c}_1, \underline{c}_2) \wedge x \notin \underline{c}_3$ 得证, 即充分性得证. 证毕.

根据定理 2, 即可得到检测 TAL-freeness 的算法.

5 TAL-freeness 检测算法

本节首先介绍根据定理 1、2 提出的检测算法, 接着对该算法的复杂度进行详细分析.

5.1 算法

根据定理 1 与定理 2, 本文提出了 TAL-freeness 检测算法.

设计思想. 对时间自动机 TA 中的每个状态, 应用算法 JudgeTALFree($s, counterstate$) 来判定是否存在 TAL 的可能.

算法 1. JudgeTAL($s, counterstate$).

假设: $counterstate$ 的初值为空, 基本函数 $guard, inv$ 与 $reset$ 已经存在

1. 计算 B_s 与 C_s
 - a) 计算 B_s
 $B_s = \underline{inv}(s)$;
 - b) 计算 C_s
 - i) 针对 C_s 的每个输出边计算
 $C_{s,e} = \underline{guard}(e) \cap \underline{inv}(s)$;
 - ii) 计算所有 $C_{s,e}$ 的并集
 $C_s = \bigcup_{e \in s} C_{s,e}$;
2. 若 $^{\circ}s = \phi$, 则
 - a) 计算 $A_s = \underline{inv}(s)$;
 - b) 计算 $F(A_s, B_s)$;
 - c) 如果 $F(A_s, B_s) \subset C_s$, 则 return true, 否则, $\{counterstate = \{s\}; \text{return false}\}$
3. 否则 ($^{\circ}s \neq \phi$)
 - a) 对每个输入边 e
 - i) 计算 $A_{s,e} = \underline{guard}(e) \cap \underline{inv}(s) \cap \underline{reset}(e)$;
 - ii) 计算 $F(A_{s,e}, B_s)$;
 - iii) 如果 $F(A_{s,e}, B_s) \not\subset C_s$, 则 $\{counterstate = \{s\} \text{return false}\}$
 - b) return true

算法 2. $F(A, B)$.

假设: n 维向量 A, B 以及 $F(A, B)$ 的返回向量的上界、下界分别用 b_i, β_i, r_i 和 a_i, α_i, R_i ($1 \leq i \leq n$) 表示, 其中 n 表示时间自动机 TA 中的时钟数.

1. 循环
 - For $i = 1$ to n 计算 ϕ_i
 - a) 计算 δ_i :
 - i) $\delta_i = \infty$ (实际编程时可以使用可取值的最大实数表示 ∞);
 - ii) For $j = 1$ to n

$$\{temp = \alpha_j - \alpha_i + a_i - b_j;$$

$$\text{if } temp < \delta_i \text{ then } \delta_i = temp\}$$
 - iii) If $\delta_i < 0$ then $\{\phi_i = \emptyset; \text{continue};\}$
 - b) $r_i = \alpha_i; R_i = \alpha_i$;
 - c) For $j = 1, j \neq i$ to n

$$\{r_j = \max(\beta_j, \alpha_i - a_i + b_j);$$

$$R_j = \min(\alpha_j, \alpha_i - a_i + a_j + \delta_i); \}$$
2. 计算所有 ϕ_i 的并集, 并返回集合.

如果对 TA 中的每一个状态, JudgeTAL 函数的返回值为 true, 则可断定该 TA 无时间动作锁.

5.2 复杂度分析

下面, 我们对此算法进行复杂度分析.

令 $n_t = |T|$, $n_c = |\mathcal{S}|$ 表示分别时间自动机 TA 中变迁与时钟的个数.

由算法 1 可知, 求交集和并集的集合操作拥有时间复杂度 $O(n_c n_t)$, 而整个算法中的关键负载为根据算法 2 计算 $F(A_s, B_s)$. 由于 $F(A_s, B_s)$ 中对 n_c

进行了两次循环, 因此算法 2 的时间复杂度为 $O(n_c^2)$. 由于在算法 1 中针对每一条边都调用了一次 $F(A_s, B_s)$, 因此整个算法的时间复杂度为 $O(n_c^2 n_t)$

6 实 例

在我们开发的基于 MDA 的系统集成开发环境的模型验证器中已经实现了该方法, 如图 2 所示.

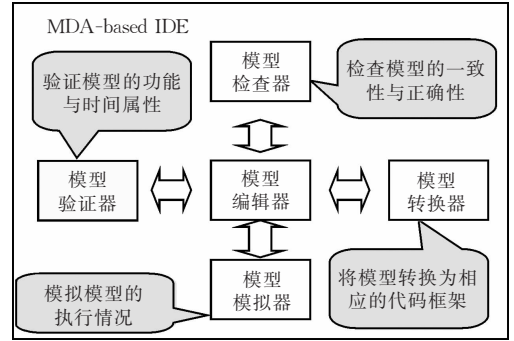


图 2 IDE 的体系结构

设计者首先利用模型编辑器对系统建模, 然后即可用模型验证器的该项死锁检测功能对模型进行测试, 从而帮助设计者发现模型的设计缺陷. 以下实例利用了算法 JudgeTAL 验证了一个简化的媒体同步协议^[8], 该协议用于减少由于网络延迟而带来的抖动.

图 3 描述了同步协议模块 (Synchronization Protocol Module, SPM) 对应的一个时间自动机模型. 该时间自动机包括表示发包与收包的信号 P_s 与 P_r , 其输出为显示帧的起止标记 F_s 与 F_e . 在给出 $m, M, d_0, \theta_0, d_1, e_1, \theta_1$ 等的具体值后, 即可判定该自动机是否无时间动作锁.

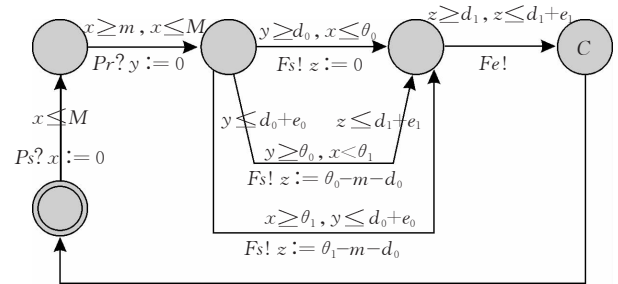


图 3 SPM 的时间自动机模型

7 结论与展望

为解决 TAL-freeness 的检测问题, Behzad 等人提出的方法时间复杂度约为 $O(n^n)$ ^[3,7]. 本文提出

了基于向量空间代数性质的检测时间自动机时间动作锁的测试方法,时间复杂度为 $O(n_c^2 n_t)$,并且不需要引入 RPS 引擎,简化了验证过程.

今后进一步的工作主要从以下两个方面展开:
(1) 在我们的 IDE 中集成各种死锁检测方法,并提高其性能;(2) 将此类方法扩展为针对时间自动机网络的死锁检测算法.

参 考 文 献

[1] Kozo O, Satoshi H et al. Specification of real-time systems using a timed automata model with shared variables and verification of partial-deadlock freeness//Proceedings of the IEEE 1999 International Workshops on Parallel Processing. Aizu-Wakamatsu, Fukushima, Japan, 1999; 576-586

[2] Yeung Chim-Fu, Hung Sheung-Lun. A new deadlock detection algorithms for distributed real-time database systems//Proceedings of the 14th IEEE Symposium on Reliable Distributed Systems. Bad Neuenahr, Germany, 1995; 146-153

[3] Behzad B, Kozo O. Testing deadlock-freeness in real-time systems: A formal approach//Proceedings of the FATES 2004. Linz, Austria, 2004; 95-109

[4] Alur R, Dill D L. A theory for timed automata. Theoretical Computer Science, 1994, 126(2): 183-235

[5] Bowman H. Time and action lock freedom properties for timed automata//Proceedings of the Formal Techniques for Networked and Distributed Systems. Cheju Island, Korea, 2001; 119-134

[6] Tripakis S. Verifying Progress in Timed Systems//Proceedings of the ARTS 1999. Bamberg, Germany, 1999; 299-314

[7] Shibata N, Okano K et al. A decision algorithm for prenex normal form rational Presburger sentences based on combinatorial geometry//Proceedings of the DMTCS'99; Proceedings of the CATS'99. Singapore, 1999; 344-359

[8] Higashino T, Nakata A et al. Generating test cases for a timed I/O automaton model//Proceedings of the IFIP IWTCs'99. Budapest, Hungary, 1999; 197-214

[9] Wang F, Hwang G, Yu F. TCTL inevitability analysis of dense-time systems. Santa Barbara, CA, USA, 2003; 176-187



PENG Rong, born in 1975, Ph. D. . Her research interests include requirement engineering and information security.

CUI Jing-Song, born in 1975, Ph. D. . His research interests include algorithm optimization, information security and network security.

ZENG Xiang-Yong, born in 1973, Ph. D. His research interests include information security and sequence design.

Background

With the rapid development of network technology, network applications are explosion. A successful network application must satisfy the security requirements of each stakeholder. So, acquiring and analyzing the security requirements as early as possible become very critical, which is the core of the authors' NSF project. In order to accomplish the task, the authors develop an MDA-based IDE. One of its most important works is to check whether the acquired security requirements satisfy the safety, liveness property, etc. In order to integrate the TAL-freeness checking into the platform, it is must that find an efficient detection algorithm.

The symbolic model checking is the common method to solve the state space explosion problem in the model checking research field. Many research results on model checking are based on the symbolic model checking including B. Behzad

and O.Kozo's formal approach. The RPS Engine (Rational Presburger Sentences Engine) is involved in their formal approach to detect the TAL, which not only hinders its implementation efficiency, but also makes it difficult to locating the errors and simulating the trace of error occurrence.

In order to verify TAL-freeness directly and find out the exact reason that causes the TAL when it occurs, this paper proposes an algebraic approach for TAL-freeness detection and provides a direct algorithm to detect TAL-freeness with pure algebraic operations. And the proof of correctness of the algorithm is also provided in the paper. By the easily implemented algorithm, the TAL-freeness detection problem in the authors' project has been solved and the algorithm has been integrated successfully into the IDE.