软件学报 ISSN 1000-9825, CODEN RUXUEW
2026,37(9):3577−3597 [doi: 10.13328/j.cnki.jos.007607] [CSTR: 32375.14.jos.007607]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

观察树驱动的确定性时间自动机主动学习*

滕 宇, 张苗苗

(同济大学 计算机科学与技术学院, 上海 200092)

通信作者: 张苗苗, E-mail: miaomiao@tongji.edu.cn

摘 要: 时间自动机的主动学习是一个重要的研究话题. 多时钟时间自动机的主动学习是其中一个重要的研究方向. 然而, 已有的多时钟时间自动机的学习算法的学习速度较慢. 基于时间观察树提出一种改进的主动学习算法. 定义一种名为时间观察树的数据结构, 用于存储学习过程中获取的信息. 基于时间观察树的特殊结构, 可以利用二分搜索技术来分析反例. 通过使用该反例分析技术, 可以减少成员查询的数量和重置信息查询的数量, 从而提高算法的效率. 实验结果证明了该方法的有效性.

关键词: 主动学习; 多时钟时间自动机; 时间观察树; 反例分析; 二分搜索

中图法分类号: TP311

中文引用格式: 滕宇, 张苗苗. 观察树驱动的确定性时间自动机主动学习. 软件学报, 2026, 37(9): 3577–3597. http://www.jos.org.cn/1000-9825/7607.htm

英文引用格式: Teng Y, Zhang MM. Observation-tree-driven Active Learning of Deterministic Timed Automata. Ruan Jian Xue Bao/Journal of Software, 2026, 37(9): 3577–3597 (in Chinese). http://www.jos.org.cn/1000-9825/7607.htm

Observation-tree-driven Active Learning of Deterministic Timed Automata

TENG Yu, ZHANG Miao-Miao

(School of Computer Science and Technology, Tongji University, Shanghai 200092, China)

Abstract: Active learning of timed automata is an important research topic. Active learning of multi-clock timed automata is one of the important research directions. However, the existing learning algorithms for multi-clock timed automata have relatively slow learning speed. This study proposes an improved active learning algorithm based on a timed observation tree. A data structure called the timed observation tree is defined to store the information obtained during the learning process. Based on the special structure of the timed observation tree, binary search techniques can be used to analyze counterexamples. By using this counterexample analysis technique, the number of membership queries and reset information queries can be reduced, thereby improving the efficiency of the proposed algorithm. Experimental results demonstrate the effectiveness of the method.

Key words: active learning; multi-clock timed automata; timed observation tree; counterexample analysis; binary search

模型学习是一种自动化建模的方法[1]. 它可以根据系统的输入输出行为生成系统的形式模型. 模型学习从学习方式上分为被动学习和主动学习. 被动学习能从一组给定的样本中推断出一个模型[2], 已经被用于生成各种模型[3–13]. 然而, 被动学习不能保证学习出一个完全正确的形式模型. 与之相反, 主动学习所学模型能够准确描述实际系统的行为. 1987 年, Angluin[14]基于主动学习框架 MAT 提出了著名的 L^* 算法. 该算法构造了包括老师和学生以及其交互的主动学习框架, 其中老师掌握目标系统的接收语言. 为了学习该语言, 学生向老师提出成员查询来询问一个给定的字是否属于目标系统的接收语言, 并将查询结果存入观察表中. 当观察表满足一定的性质时, 学生根据观察表构造一个假设自动机, 然后发起等价查询来判断该假设自动机的语言与目标语言是否相等. 如果不相等,

* 基金项目: 国家重点研发计划 (2022ZD0120303); 国家自然科学基金 (62472316); 上海市 2023 年度“科技创新行动计划”区块链关键技术攻关专项 (23511100800)
本文由“形式化方法与应用”专题特约编辑安杰副研究员、顾斌研究员、李建文教授推荐.
收稿时间: 2025-09-08; 修改时间: 2025-10-28; 采用时间: 2025-12-08; jos 在线出版时间: 2025-12-24
CNKI 网络首发时间: 2026-03-19

则老师返回一个反例. 学生根据该反例来更新观察表, 从而生成新的假设自动机, 然后针对新的假设自动机再一次发起等价查询. 该过程直至等价查询结果为真, 即假设自动机的语言与目标语言相等. 目前, 国内外学者已将主动学习应用于各种非时间模型, 如 Mealy 机[15–17]、非确定有限自动机[18]、寄存器自动机[19–22]、输入输出自动机[23]、符号自动机[24,25], 以及各种时间模型, 如具有计时器的 Mealy 机[26,27]、实时自动机[28,29]、单时钟时间自动机[30,31].

值得注意的是, 上述主动学习算法尚不能学习多时钟时间自动机. 针对此问题, Waga[32]在 2023 年提出一种能够学习确定性多时钟时间自动机的主动学习算法. 然而, 该算法学习出的自动机中的位置数量庞大. 为解决此问题, Teng 等人[33]提出了一种新的学习算法. 该算法能够大大减少所学自动机中的位置数量. 他们分别考虑向一个智慧的老师和一个普通的老师学习确定性多时钟时间自动机的情况. 智慧的老师和普通的老师都掌握目标确定性多时钟时间自动机的接收语言. 不同的是, 在学习过程中, 智慧的老师除了回答学生发出的成员查询和等价查询, 还可以告诉学生目标确定性多时钟时间自动机在发生迁移时每个时钟的重置情况. 具体而言, 他们首先考虑向一个智慧的老师学习确定性多时钟时间自动机的情况, 智慧的老师可以告诉学生目标系统中时钟的重置信息. 接下来, 他们考虑向一个普通的老师学习自动机的情况. 普通的老师无法告诉学生系统中时钟的重置信息. 学生可以通过猜测时钟的重置信息来将第 2 种情况转换为第 1 种情况. 虽然文献 [33] 提出的算法能够大大减少所学自动机中的位置数量, 却只能学习规模非常小的系统, 这限制了其在复杂系统中的应用. 针对此问题, Teng 等人[34]提出基于多时钟时间分类树的确定性多时钟时间自动机的主动学习算法. 该算法解决了向智慧老师学习的情况中的若干问题, 避免了对树的检查和对反例中无用信息的处理, 从而减少了成员查询和等价查询的数量, 加快了学习速度.

在基于多时钟时间分类树的主动学习算法中, 对反例进行分析是一个重要环节. 分析反例的目的是寻找反例中第 1 个导致假设自动机和目标自动机之间发生冲突的错误位置. 为此, 需要从前向后依次分析反例中的每一个位置, 直至找到错误位置, 但该方式存在两大弊端.

(1) 成员查询次数多. 每当分析反例中的一个位置时, 都需要进行一次成员查询. 因此, 为寻找反例中第 1 个导致冲突的错误位置, 所需的成员查询次数为线性级别.

(2) 成员查询所需的额外开销高. 在分析反例时, 为进行成员查询, 需要进行如下操作: 寻找重置时钟字对于区域字的合法后继. 寻找合法后继的过程需要额外开销, 且该开销取决于区域字的长度. 在反例分析的过程中, 区域字普遍较长. 因此, 寻找合法后继需要大量开销, 导致进行成员查询所需的额外开销过高.

针对上述问题, 借鉴文献 [17] 中提出的观察树结构, 设计向智慧老师学习情况下的新学习算法. Vaandrager 等人[17]基于分离的思想设计了观察树, 从而提出了基于观察树的 Mealy 机的主动学习算法. 然而, 该算法不能学习多时钟时间自动机. 为此, 本文定义了一个新的数据结构——时间观察树, 并提出了基于时间观察树的确定性多时钟时间自动机的主动学习算法. 基于时间观察树的特殊结构, 可以利用二分搜索技术来分析反例. 具体而言, 采用二分搜索技术根据反例来不断地构造更短的导致目标自动机和假设自动机之间发生冲突的字. 该过程直至构造出的导致冲突的字足够短, 以至于根据该字修改后的观察树能用来对假设自动机进行实质性的修改. 通过该方式, 可以得到如下效果.

(1) 成员查询次数少. 由于导致冲突的字最初为反例且采用二分搜索的方式来缩短字的长度, 因此所需的缩短操作的次数为对数级别. 此外, 每当构造更短的字时, 都需要进行一次成员查询. 因此, 在缩短导致冲突的字的过程中, 所需的成员查询次数为对数级别.

(2) 成员查询所需的额外开销低. 在构造更短的导致冲突的字时, 为进行成员查询, 依然要进行如下操作: 寻找重置时钟字对于区域字的合法后继. 由于采用了二分搜索技术来分析反例, 使得在新的反例分析方法中, 区域字普遍较短. 因此, 寻找合法后继所需开销减少, 从而减少为进行成员查询所需的额外开销.

本文的贡献包含如下 3 点.

- 定义时间观察树代替多时钟时间分类树来存储学习过程中获得的信息. 具体而言, 首先定义时间观察树的结构. 接下来, 定义树中节点之间的分离关系, 并根据该关系将树中的节点分为 3 大类, 进而根据该分类定义时间观察树所要满足的性质和对树的操作.
- 提出新的分析反例的方法. 具体而言, 在时间观察树中利用二分搜索技术来分析反例, 从而降低在分析反例

时所需的成员查询次数和单次成员查询所需的额外开销.

● 实现本文提出的确定性时间自动机的主动学习算法, 并将其与基于多时钟时间分类树的主动学习算法进行比较. 实验结果表明, 本文算法可以减少成员查询次数并显著减少重置信息查询次数, 从而提高算法效率.

本文第 1 节介绍时间自动机的相关概念以及基于多时钟时间分类树的主动学习算法. 第 2 节介绍时间观察树的结构、分离关系以及树的性质和操作. 第 3 节介绍改进后的学习算法流程, 并对流程中的一些细节进行详细说明. 第 4 节展示一个详细的学习示例, 以帮助读者更好地理解改进后的学习算法. 第 5 节通过对比实验证明本文所提改进算法的有效性. 第 6 节对本文工作进行总结并提出进一步的研究方向.

1 基础知识

在本文中, 采用 $\mathbb{R}_{\geqslant 0}$ 表示非负实数的集合, $\mathbb{N}$ 表示自然数的集合, $\mathbb{B}=\{\top,\bot\}$ 表示布尔集合, 其中 $\top$ 表示 true, $\bot$ 表示 false. 字母表 Σ 表示动作的有穷集合.

1.1 时间自动机、时间语言和重置时钟语言

时钟变量是时间自动机描述时间变化的变量, 简称时钟. 使用 $\mathcal{C}$ 表示时钟的集合. 在介绍时间自动机之前, 首先介绍以下与时钟有关的概念.

● 时钟约束: 时钟约束是时钟所需满足的条件. 一个时钟约束 ϕ 是形式为 $c\sim n$ 的原子约束的连接公式, 其中 $c\in\mathcal{C}$, $n\in\mathbb{N}$, $\sim\in\{\leqslant,<,\geqslant,>,=\}$. 使用 $\Phi(\mathcal{C})$ 来表示时钟约束的集合.

● 时钟解释: 一个时钟解释是一个函数 $\nu\colon \mathcal{C}\to\mathbb{R}_{\geqslant 0}$. 该函数为 $\mathcal{C}$ 中每一个时钟赋值一个非负实数. 对于一个非负实数 d, $\nu+d$ 表示一个时钟解释. 该时钟解释将 $\mathcal{C}$ 中每一个时钟 c 赋值为 $\nu(c)+d$. 对于一个集合 $\mathcal{B}\subseteq\mathcal{C}$, $[\mathcal{B}\to 0]\nu$ 是一个时钟解释, 它将 $\mathcal{B}$ 中的每一个时钟 c 重置为 0, 并且使其余时钟的赋值与 ν 保持一致.

定义 1. 时间自动机[35]. 一个时间自动机是一个六元组 $\mathcal{A}=(\Sigma,L,l_0,F,\mathcal{C},\Delta)$, 其中 Σ 表示字母表, L 表示位置的有穷集合, l_0 表示初始位置, $F\subseteq L$ 表示接收位置的集合, $\mathcal{C}$ 表示时钟变量的有穷集合, $\Delta\subseteq L\times\Sigma\times\Phi(\mathcal{C})\times 2^{\mathcal{C}}\times L$ 表示迁移的有穷集合.

一条迁移 $\delta=(l,\sigma,\phi,\mathcal{B},l')\in\Delta$ 表示当 $\mathcal{C}$ 的目前的时钟解释满足时钟约束 ϕ 时, 从源位置 l 出发, 执行动作 σ 后, 会到达目标位置 l'. $\mathcal{B}\subseteq\mathcal{C}$ 是时钟变量集合的一个子集. 在迁移 δ 发生之后, 需要将 $\mathcal{B}$ 中的时钟变量重置为 0.

时间自动机 $\mathcal{A}$ 的一个状态是一个二元组 (l,ν). 其中, $l\in L$ 且 ν 是一个时钟解释. 一个时间字是一个有限序列 $\omega=(\sigma_1,t_1)(\sigma_2,t_2)\ldots(\sigma_n,t_n)\in(\Sigma,\mathbb{R}_{\geqslant 0})^*$. 其中, 对于任意 $1\leqslant i\leqslant n$, t_i 表示在执行动作 σ_i 之前的延迟时间. 对于一个时间字 ω 而言, $\mathcal{A}$ 中对应于 ω 的一条执行 ρ 是一个包含一系列状态的序列 $(l_0,\nu_0)\xrightarrow{t_1,\sigma_1}(l_1,\nu_1)\xrightarrow{t_2,\sigma_2}\ldots\xrightarrow{t_n,\sigma_n}(l_n,\nu_n)$. 称 ω 是 ρ 的路径, 记为 $\omega=trace(\rho)$. 在执行 ρ 中, 对于任意 $1\leqslant i\leqslant n$, 存在迁移 $(l_{i-1},\sigma_i,\phi_i,\mathcal{B}_i,l_i)$, 该迁移满足 $(\nu_{i-1}+t_i)\in\phi_i$ 且 $\nu_i=[\mathcal{B}_i\to 0](\nu_{i-1}+t_i)$. 若 $l_n\in F$, 则 ρ 是 $\mathcal{A}$ 的一条可接收的执行. 对于一个时间字 ω, 通过记录与之对应的执行 ρ 中的重置信息, 得到与 ω 对应的重置时间字 $\omega_r=trace_r(\rho)=(\sigma_1,t_1,\mathbf{b}_1)(\sigma_2,t_2,\mathbf{b}_2)\ldots(\sigma_n,t_n,\mathbf{b}_n)$. 其中, $\mathbf{b}_i\in\mathbb{B}^{|\mathcal{C}|}$ 是一个 $|\mathcal{C}|$ 元组. 对于任意 $c_j\in\mathcal{C}$ 和 $1\leqslant j\leqslant|\mathcal{C}|$, $\mathbf{b}_{i,j}$ 记录当执行 (σ_i,t_i) 时, 第 j 个时钟是否重置. 基于执行和时间字以及重置时间字之间的关系, 可定义时间自动机 $\mathcal{A}$ 的时间语言和重置时间语言. 具体地, $\mathcal{A}$ 的时间语言 $\mathcal{L}(\mathcal{A})=\{trace(\rho)\,|\,\rho$是$\mathcal{A}$的一条可接收的执行$\}$. 两个时间自动机 $\mathcal{A}_1$ 和 $\mathcal{A}_2$ 是等价的, 当且仅当 $\mathcal{L}(\mathcal{A}_1)=\mathcal{L}(\mathcal{A}_2)$. 与时间语言的定义类似, $\mathcal{A}$ 的重置时间语言 $\mathcal{L}_r(\mathcal{A})=\{trace_r(\rho)\,|\,\rho$是$\mathcal{A}$的一条可接收的执行$\}$.

时间字从全局时钟的角度描述系统行为, 而全局时钟不同于系统内的逻辑时钟 $\mathcal{C}$. 为从逻辑时钟的角度描述系统行为, 基于执行 $\rho=(l_0,\nu_0)\xrightarrow{t_1,\sigma_1}(l_1,\nu_1)\xrightarrow{t_2,\sigma_2}\ldots\xrightarrow{t_n,\sigma_n}(l_n,\nu_n)$ 定义时钟字 $\gamma=(\sigma_1,\mathbf{v}_1)(\sigma_2,\mathbf{v}_2)\ldots(\sigma_n,\mathbf{v}_n)$. 其中, $\mathbf{v}_i\in\mathbb{R}_{\geqslant 0}^{|\mathcal{C}|}$ 记录当执行动作 σ_i 时, 每个时钟的值. 用 $\boldsymbol{\Sigma}=\Sigma\times\mathbb{R}_{\geqslant 0}^{|\mathcal{C}|}$ 表示时钟动作的无穷集合, $\boldsymbol{\Sigma}^*$ 表示时钟字的无穷集合. 通过记录 ρ 中的重置信息, 可以得到与时钟字 γ 对应的重置时钟字 $\gamma_r=(\sigma_1,\mathbf{v}_1,\mathbf{b}_1)(\sigma_2,\mathbf{v}_2,\mathbf{b}_2)\ldots(\sigma_n,\mathbf{v}_n,\mathbf{b}_n)$. 定义函数 $vw(\gamma_r)=\gamma=(\sigma_1,\mathbf{v}_1)(\sigma_2,\mathbf{v}_2)\ldots(\sigma_n,\mathbf{v}_n)$, 函数 $resets(\gamma_r)=\{\mathbf{b}_1,\mathbf{b}_2,\ldots,\mathbf{b}_n\}$. 类似地, 采用 $\boldsymbol{\Sigma}_r=\Sigma\times\mathbb{R}_{\geqslant 0}^{|\mathcal{C}|}\times\mathbb{B}^{|\mathcal{C}|}$ 表示重置时钟动作的无穷集合, $\boldsymbol{\Sigma}_r^*$ 表示重置时钟字的无穷集合. 此外, 对于一个给定的重置时间字 $\omega_r=trace_r(\rho)=(\sigma_1,t_1,\mathbf{b}_1)(\sigma_2,t_2,\mathbf{b}_2)\ldots(\sigma_n,t_n,\mathbf{b}_n)$, 可以通过计算得到唯一与之对应的重置时钟字 $\gamma_r=(\sigma_1,\mathbf{v}_1,\mathbf{b}_1)(\sigma_2,\mathbf{v}_2,\mathbf{b}_2)\ldots(\sigma_n,\mathbf{v}_n,\mathbf{b}_n)$. 其中, 对于任意的

$1 \leqslant i \leqslant n$ 和 $1 \leqslant j \leqslant |\mathcal{C}|$, 若 $i=1$ 或者 $\mathbf{b}_{i-1,j}=\top$, 则 $\mathbf{v}_{i,j}=t_i$, 否则, $\mathbf{v}_{i,j}=\mathbf{v}_{i-1,j}+t_i$. 根据唯一与 ω_r 对应的重置时钟字 γ_r, 可得到唯一与 ω_r 对应的时钟字 $\gamma=vw(\gamma_r)=(\sigma_1,\mathbf{v}_1)(\sigma_2,\mathbf{v}_2)\ldots(\sigma_n,\mathbf{v}_n)$.

定义 2. 确定性时间自动机.一个时间自动机是一个确定性时间自动机 (deterministic timed automaton, DTA), 当且仅当对于一个给定的时间字, $\mathcal{A}$ 中最多只有一个执行.

也就是说, 在确定性时间自动机中, 对于任意一个位置 $l\in L$ 和任意一个动作 $\sigma\in\Sigma$, 从 l 出发执行 σ 的任意两条迁移 $(l,\sigma,\phi_1,-,-)$ 和 $(l,\sigma,\phi_2,-,-)$ 中的两个时钟约束 ϕ_1 和 ϕ_2 必定是互斥的.

定义 3. 完备确定性时间自动机. 一个时间自动机是一个完备确定性时间自动机 (complete deterministic timed automaton, CTA), 当且仅当对于一个给定的时间字, $\mathcal{A}$ 中有且只有一个执行.

对于任意的确定性时间自动机, 都可以利用文献 [33] 中的操作将其转换为完备确定性时间自动机. 图 1 展示了一个转换示例, 初始位置由“start”标识且接收位置由双圆圈标识. 为了清楚地展示所提出的学习方法, 在本文中, 假设所要学习的目标 DTA 均为 CTA.

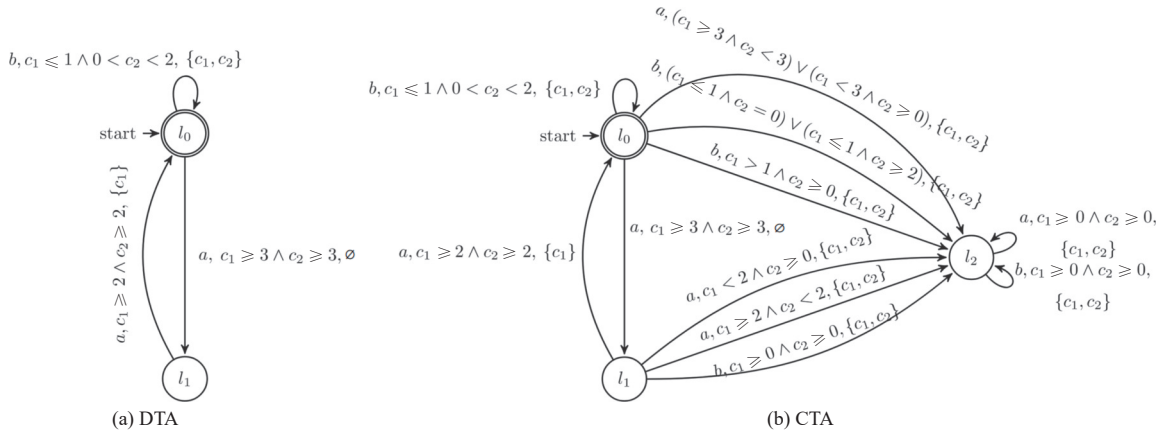


图 1　一个 DTA 及其对应的 CTA

在 CTA $\mathcal{A}$ 中, 对于一个 (重置) 时间字, 函数 Γ 根据其在 $\mathcal{A}$ 中唯一对应的执行来得到唯一与其对应的 (重置) 时钟字. 对于一个时钟字, 函数 π 根据其在 $\mathcal{A}$ 中唯一对应的执行来得到唯一与其对应的重置时钟字. 一个 CTA $\mathcal{A}$ 的时钟语言 $L(\mathcal{A})=\{\Gamma(trace(\rho))\,|\,\rho是\mathcal{A}的一条可接收的执行\}$. $\mathcal{A}$ 的重置时钟语言 $L_r(\mathcal{A})=\{\Gamma(trace_r(\rho))\,|\,\rho是\mathcal{A}的一条可接收的执行\}$. 在一个 CTA $\mathcal{A}$ 中, 对于一个时钟字 γ, 如果存在一个满足 $\Gamma(trace(\rho))=\gamma$ 的执行 ρ, 则该时钟字 γ 是 $\mathcal{A}$ 的一个合法时钟字. 类似地, 对于一个重置时钟字 γ_r, 如果存在一个满足 $\Gamma(trace_r(\rho))=\gamma_r$ 的执行 ρ, 则该重置时钟字 γ_r 就是 $\mathcal{A}$ 的一个合法重置时钟字.

由于时钟解释的数量是无穷的, 所以一个时间自动机的状态数也是无穷的. 为了对状态空间进行有限的划分, region 的概念被提出.

定义 4. Region 等价关系. 给定一个时间自动机 $\mathcal{A}$, 对于每一个时钟 c, 用 $\kappa(c)$ 来表示 $\mathcal{A}$ 中所有对于 c 的约束中出现的最大整数. 对于任意的实数 a, 用 $frac(a)$ 来表示 a 的小数部分, 用 $\lfloor a\rfloor$ 来表示 a 的整数部分. 基于这些概念, 两个时钟解释 ν 和 ν' 是 region 等价的, 即 $\nu\sim\nu'$, 当且仅当:

- 对于所有的时钟 $c\in\mathcal{C}$, 要么 $\lfloor\nu(c)\rfloor=\lfloor\nu'(c)\rfloor$, 要么 $\nu(c)>\kappa(c)$ 且 $\nu'(c)>\kappa(c)$.
- 对于所有的时钟 $c\in\mathcal{C}$, 如果 $\nu(c)\leqslant\kappa(c)$, 则 $frac(\nu(c))=0$ 当且仅当 $frac(\nu'(c))=0$ 且
- 对于任意两个时钟变量 $c_i,c_j\in\mathcal{C}$, 如果 $\nu(c_i)\leqslant\kappa(c_i)$ 且 $\nu(c_j)\leqslant\kappa(c_j)$, 那么 $frac(\nu(c_i))\leqslant frac(\nu(c_j))$ 当且仅当 $frac(\nu'(c_i))\leqslant frac(\nu'(c_j))$.

在一个时间自动机 $\mathcal{A}$ 中, 一个 region 是一个基于等价关系 $\sim$ 来定义的等价类. 设 $\mathcal{R}$ 是 $\mathcal{A}$ 的 region 的集合. $\mathcal{R}$ 的数量 $|\mathcal{R}|$ 最大为 $\Lambda=|\mathcal{C}|!\cdot 2^{|\mathcal{C}|}\cdot\Pi_{c\in\mathcal{C}}(2\kappa(c)+2)$. 对于一个时钟解释 ν, 用 $[\![\nu]\!]$ 来表示包含它的 region.

定义 5. 符号化状态. 给定时间自动机的一个符号化状态是一个二元组 $(l,[[\mathrm{v}]])$. 其中, l 是时间自动机中的一个位置, $[[\mathrm{v}]]$ 是时间自动机中的一个 region.

1.2 基于多时钟时间分类树的 DTA 的学习

在本节中, 将简要介绍基于多时钟时间分类树的 DTA 的主动学习方法, 其基本思想如下. 首先, 通过文献 [33] 中的证明可知, 如果两个 DTA 的重置时钟语言是等价的 (即 $L_r(\mathcal{A}_1)=L_r(\mathcal{A}_2)$), 那么它们的时间语言也是等价的 (即 $\mathcal{L}(\mathcal{A}_1)=\mathcal{L}(\mathcal{A}_2)$). 因此, 为了学习 $\mathcal{A}$ 的时间语言 $\mathcal{L}(\mathcal{A})$, 只需要学习 $\mathcal{A}$ 的重置时钟语言 $L_r(\mathcal{A})$. 随后, 定义多时钟时间分类树, 再基于分类树设计 $L_r(\mathcal{A})$ 的主动学习算法. 该学习算法依赖于在文献 [33] 中提出的重置时钟语言 $L_r(\mathcal{A})$ 的等价关系, 该关系能够保证 $L_r(\mathcal{A})$ 的等价类的数量是有限的, 从而保证学习算法是可终止的.

下面将首先对重置时钟语言 $L_r(\mathcal{A})$ 的等价关系进行详细说明, 再介绍依据该关系提出的基于分类树的 $L_r(\mathcal{A})$ 主动学习算法.

在介绍 $L_r(\mathcal{A})$ 的等价关系之前, 首先介绍两个概念: 区域字和合法后继.

定义 6. 区域字. 给定一个时间自动机 $\mathcal{A}$, 一个区域字 ξ 是一个由二元组 $(\sigma,[[\mathrm{v}]])$ 组成的有穷序列. 其中, σ 是 $\mathcal{A}$ 的一个动作且 $[[\mathrm{v}]]$ 是 $\mathcal{A}$ 的一个 region.

用 $\mathbf{\Sigma}_G=\Sigma\times\mathcal{R}$ 来表示 region 动作的有穷集合. 对于 $\mathcal{A}$ 的每一个时钟字 γ, 有且仅有一个区域字 ξ 与之对应, 记为 $\xi=[[\gamma]]$.

定义 7. 合法后继. 给定一个 DTA $\mathcal{A}$, 一个重置时钟字 γ_r 和一个区域字 ξ, 若一个重置时钟字 γ'_r 满足 $[[vw(\gamma'_r)]]=\xi$ 且 $\gamma_r\gamma'_r$ 是 $\mathcal{A}$ 的一个合法重置时钟字, 则 γ'_r 是 γ_r 对应于 ξ 的合法后继.

用 $vs_{\mathcal{A}}(\gamma_r,\xi)$ 来表示 γ_r 对应于 ξ 的所有的合法后继的集合. 寻找合法后继时需要进行一系列重置信息查询, 具体方法请见文献 [33]. 基于合法后继的定义, 将进一步介绍重置时钟语言 $L_r(\mathcal{A})$ 的等价关系.

定义 8. 重置时钟字的等价关系. 假设 $L_r(\mathcal{A})$ 是一个 DTA $\mathcal{A}$ 的重置时钟语言. 若 $L_r(\mathcal{A})$ 无法对两个重置时钟字 γ_{r1} 和 γ_{r2} 进行区分, 即 $\gamma_{r1}\sim_{L_r(\mathcal{A})}\gamma_{r2}$, 则对于任意的区域字 ξ, 任意的 $\gamma'_{r1}\in vs_{\mathcal{A}}(\gamma_{r1},\xi)$ 和任意的 $\gamma'_{r2}\in vs_{\mathcal{A}}(\gamma_{r2},\xi)$ 满足以下两个条件:

- $\gamma_{r1}\gamma'_{r1}\in L_r(\mathcal{A})$ 当且仅当 $\gamma_{r2}\gamma'_{r2}\in L_r(\mathcal{A})$.
- $resets(\gamma'_{r1})=resets(\gamma'_{r2})$.

引理 1. 若 $\mathcal{A}$ 的两个合法的重置时钟字 γ_{r1} 和 γ_{r2} 到达 $\mathcal{A}$ 的相同的符号化状态, 则 $\gamma_{r1}\sim_{L_r(\mathcal{A})}\gamma_{r2}$.

根据引理 1, 可以证明一个 DTA $\mathcal{A}$ 的重置时钟语言的等价类数量是有限的. 这保证了 $L_r(\mathcal{A})$ 的主动学习算法是可终止的.

下面介绍基于多时钟时间分类树的主动学习算法. 在该算法中, 学生可以向老师提出 3 种查询: 重置信息查询 ($\mathrm{RQ}_{\mathcal{A}}$)、成员查询 ($\mathrm{MQ}_{\mathcal{A}}$) 和等价查询 (EQ). 在重置信息查询中, 学生询问一个合法的时钟字 γ 在 $\mathcal{A}$ 中对应的重置时钟字 $\pi(\gamma)$ 的重置信息. 在成员查询中, 老师收到一个重置时钟字 γ_r 并回答 γ_r 是否属于 $L_r(\mathcal{A})$. 在等价查询中, 学生询问一个假设 DTA $\mathcal{H}$ 的时间语言 $\mathcal{L}(\mathcal{H})$ 是否等于 $\mathcal{A}$ 的时间语言 $\mathcal{L}(\mathcal{A})$.

为了存储学习过程中产生的查询结果, 多时钟时间分类树被定义. 一个多时钟时间分类树是一个七元组 $\mathcal{CTR}=(\Sigma,V,Q_{tr},E_{tr},f^E_{tr},g^E_{tr},T_{Q_{tr}})$. 其中, Σ 是字母表, V 是中间节点的有穷集合, 每一个中间节点记录一个区域字, Q_{tr} 是叶节点的有穷集合, 每一个叶节点记录一个重置时钟字, $E_{tr}\subseteq V\times(V\cup Q_{tr})$ 是边的有穷集合, $T_{Q_{tr}}\subseteq Q_{tr}\times\mathbf{\Sigma}_r\times Q_{tr}$ 是描述叶节点之间关系的有穷集合, 分类函数 f^E_{tr} 和标记函数 g^E_{tr} 则用于在边中记录成员查询结果和重置信息查询结果. 具体而言, 对于一条边 $e=(u_1,u_2)\in E_{tr}$, 为了计算 $f^E_{tr}((u_1,u_2))$ 和 $g^E_{tr}((u_1,u_2))$, 记 q 为 u_2 的子树中的任意一个叶节点, γ_r 为 q 中记录的重置时钟字, ξ 为 u_1 中记录的区域字, 然后寻找一个合法后继 $e_r\in vs_{\mathcal{A}}(\gamma_r,\xi)$. $f^E_{tr}((u_1,u_2))$ 记录 $\gamma_r\cdot e_r$ 的成员查询结果, $g^E_{tr}((u_1,u_2))$ 则记录 e_r 的重置信息. 直观地说, 一个叶节点用于构造假设自动机 $\mathcal{H}$ 中的一个位置, 中间节点则利用函数 f^E_{tr} 和 g^E_{tr} 来区分叶节点. 叶节点之间关系的集合 $T_{Q_{tr}}$ 则保存了用于构造假设自动机的迁移的信息.

基于多时钟时间分类树设计的学习算法的大致流程如下. 首先, 学生初始化分类树 $\mathcal{CTR}=(\Sigma,V,Q_{tr},E_{tr},f^E_{tr},g^E_{tr},T_{Q_{tr}})$. 接下来, 学生根据 $\mathcal{CTR}$ 构建一个 DFA M. 基于文献 [33] 中提出的划分函数, 学生将 M 转化为一个假设 CTA $\mathcal{H}$.

然后学生提出等价查询来确定是否 $\mathcal{L}(\mathcal{A})=\mathcal{L}(\mathcal{H})$. 如果答案为“否”, 老师返回一个重置时间字 ω_r 作为反例. 随后, 学生根据 ω_r 进行反例分析和处理以修改 $\mathcal{CTR}$, 从而生成新的 $\mathcal{H}$, 然后针对新的 $\mathcal{H}$ 再一次发起等价查询. 学生重复整个过程, 直到等价查询的结果为“是”.

下面详细介绍反例分析的过程. 在进行反例分析时, 学生首先将反例 ω_r 转换为与其对应的重置时钟字 $ctx=\Gamma(\omega_r)$, 再寻找 ctx 中第 1 个导致冲突的错误位置. 为寻找错误位置, 需要从头到尾依次检查 ctx 中的每个位置. 对于每一个位置 i, 学生都需要通过前缀分析和后缀分析来检查它是否为错误位置. 用 $ctx[:i]$ 表示 ctx 的第 i 个位置之前的前缀, 用 $ctx[i:]$ 表示 ctx 的第 i 个位置之后的后缀. 在进行前缀分析时, 需要判断 $ctx[:i]$ 对应的时钟字 $vw(ctx[:i])$ 在 $\mathcal{H}$ 和 $\mathcal{A}$ 中对应的重置时钟字是否相同, 从而保证 $ctx[:i]$ 在 $\mathcal{H}$ 中是可执行的. 在进行后缀分析时, 用 l_q 来表示在 $\mathcal{H}$ 中执行 $ctx[:i]$ 所到达的位置, 用 q 表示 l_q 在分类树中对应的叶节点, 记 q 中存储的重置时钟字为 γ_r. 然后, 利用 $ctx[i+1:]$ 来构造区域字 $\xi=[\![vw(ctx[i+1:])]\!]$, 再寻找一个合法后继 $e_r\in vs_{\mathcal{A}}(\gamma_r,\xi)$. 接下来, 对 $\gamma_r\cdot e_r$ 进行成员查询, 判断 $\gamma_r\cdot e_r$ 的成员查询结果是否与 ctx 的成员查询结果相同. 若不相同, 则说明 γ_r 和 $ctx[:i]$ 在 $\mathcal{H}$ 中应该对应不同的位置, 因此 i 为错误位置.

由上述反例分析方法可知, 若反例中的错误位置为第 i 个位置, 则在反例分析过程中需要发起 i 次成员查询. 此外, 为进行成员查询需要寻找合法后继. 寻找合法后继需要进行一系列重置信息查询, 重置信息查询的次数取决于区域字的长度 $|\xi|$, $|\xi|$ 起初为 $|ctx|-1$.

2 时间观察树和分离关系

在本节中, 首先定义时间观察树来代替分类树作为学习算法的存储结构, 再定义分离关系, 并根据该关系将时间观察树中的节点分为 3 类, 进而根据该分类定义时间观察树所要满足的性质和对树的操作.

在文献 [33] 中, 定义了重置时钟字对应于区域字的合法后继. 有些情况下, 重置时钟字对应区域字的第 i 个位置之前存在合法后继, 在第 $i+1$ 个位置不存在合法后继. 针对这种情况, 定义重置时钟字对应区域字的半合法后继和半合法后继拼接字. 基于此概念, 再定义两个函数 f_{tr} 和 g_{tr}. 可以利用这两个函数在观察树中记录成员查询结果和时钟的重置信息.

定义 9. 半合法后继. 对于一个重置时钟字 $\gamma_r\in\mathbf{\Sigma}_r^*$ 和一个区域字 $\xi\in\mathbf{\Sigma}_G^*$, 记 $\xi[:i]$ 为 ξ 的第 i 个位置 (包括第 i 个位置) 之前的前缀, 其中 $1\leqslant i<|\xi|$. 若存在满足 $vs_{\mathcal{A}}(\gamma_r,\xi[:i])\neq\varnothing$ 且 $vs_{\mathcal{A}}(\gamma_r,\xi[:i+1])=\varnothing$ 的 i, 则 $e_r\in vs_{\mathcal{A}}(\gamma_r,\xi[:i])$ 为 γ_r 对应于 ξ 的一个半合法后继.

定义 10. 半合法后继拼接字. 对于一个重置时钟字 $\gamma_r\in\mathbf{\Sigma}_r^*$ 和一个区域字 $\xi\in\mathbf{\Sigma}_G^*$, 若 $vs_{\mathcal{A}}(\gamma_r,\xi)=\varnothing$, 则可以根据是否存在 γ_r 对应于 ξ 的半合法后继来构建 γ_r 对应于 ξ 的半合法后继拼接字. 具体而言, 若存在 γ_r 对应于 ξ 的半合法后继, 则可以利用一个半合法后继 e_r 来构建一个半合法后继拼接字 $\boldsymbol{\sigma}_r^*=e_r\cdot\boldsymbol{\sigma}_{r1}\boldsymbol{\sigma}_{r2}\ldots\boldsymbol{\sigma}_{r(|\xi|-|e_r|)}$. $\boldsymbol{\sigma}_r^*$ 满足 $[\![vw(\boldsymbol{\sigma}_r^*)]\!]=\xi$ 且 $resets(\boldsymbol{\sigma}_r^*)=\left\{resets(e_r),\{\top\}^{(|\xi|-|e_r|)\times|C|}\right\}$. 若不存在 γ_r 对应于 ξ 的半合法后继, 则 γ_r 对应于 ξ 的一个半合法后继拼接字 $\boldsymbol{\sigma}_r^*=\boldsymbol{\sigma}_{r1}\boldsymbol{\sigma}_{r2}\ldots\boldsymbol{\sigma}_{r|\xi|}$. $\boldsymbol{\sigma}_r^*$ 满足 $[\![vw(\boldsymbol{\sigma}_r^*)]\!]=\xi$ 且 $resets(\boldsymbol{\sigma}_r^*)=\{\top\}^{|\xi|\times|C|}$.

- 函数 f_{tr} 将 $\mathbf{\Sigma}_r^*\times\mathbf{\Sigma}_G^*$ 映射到集合 $\{+,-\}$ 中. f_{tr} 的计算方法如下: 对于一个重置时钟字 $\gamma_r\in\mathbf{\Sigma}_r^*$ 和一个区域字 $\xi\in\mathbf{\Sigma}_G^*$, 在 $\xi\neq\epsilon$ 的情况下, 需要寻找一个合法后继 $e_r\in vs_{\mathcal{A}}(\gamma_r,\xi)$. 若 $vs_{\mathcal{A}}(\gamma_r,\xi)\neq\varnothing$ 且 $\mathrm{MQ}_{\mathcal{A}}(\gamma_r\cdot e_r)=+$, 则 $f_{tr}(\gamma_r,\xi)=+$. 否则, $f_{tr}(\gamma_r,\xi)=-$. 在 $\xi=\epsilon$ 的情况下, $f_{tr}(\gamma_r,\xi)=\mathrm{MQ}_{\mathcal{A}}(\gamma_r)$.
- 函数 g_{tr} 将 $\mathbf{\Sigma}_r^*\times\mathbf{\Sigma}_G^*$ 映射到集合 $\{\top,\bot\}^{|\xi\in\mathbf{\Sigma}_G^*|\times|C|}$ 中. g_{tr} 的计算方法如下: 对于一个重置时钟字 $\gamma_r\in\mathbf{\Sigma}_r^*$ 和一个区域字 $\xi\in\mathbf{\Sigma}_G^*$, 在 $\xi=\epsilon$ 的情况下, $g_{tr}(\gamma_r,\xi)=\varnothing$. 在 $\xi\neq\epsilon$ 的情况下, 需要寻找一个合法后继 $e_r\in vs_{\mathcal{A}}(\gamma_r,\xi)$. 若 $vs_{\mathcal{A}}(\gamma_r,\xi)\neq\varnothing$, 则 $g_{tr}(\gamma_r,\xi)=resets(e_r)$. 否则, 构建 γ_r 对应于 ξ 的一个半合法后继拼接字 $\boldsymbol{\sigma}_r^*$, 并令 $g_{tr}(\gamma_r,\xi)=resets(\boldsymbol{\sigma}_r^*)$.

2.1 时间观察树

定义 11. 时间观察树. 一个时间观察树 $\mathcal{TR}=(\Sigma,Q,E,access,T_Q,f_{\mathcal{TR}})$ 是一个多叉树, 其中,

- Σ 是字母表;
- Q 是节点的有穷集合.

● $E \subseteq Q \times \boldsymbol{\Sigma}_r \times Q$ 是边的有穷集合. 一条边 $e=(q,\boldsymbol{\sigma}_r,q') \in E$ 表示从节点 q 出发, 执行重置时钟动作 $\boldsymbol{\sigma}_r$ 后会到达节点 q';

● $access\colon Q \to \boldsymbol{\Sigma}_r^*$ 是将节点映射到重置时钟字的函数. 对于任意一个节点 $q \in Q$, 从根节点出发, 到达 q 需要经过一系列边. 这一系列边中的重置时钟动作所组成的序列是一个重置时钟字. 将该重置时钟字记为 $access(q)$, 即 q 的访问序列.

● $T_Q \subseteq \boldsymbol{\Sigma}_r^*$ 用于标记一部分边;

● $f_{\mathcal{TR}}\colon Q \to \{+,-\}$ 是一个分类函数. 对于一个节点 $q \in Q$, $f_{\mathcal{TR}}(q)=f_{tr}(access(q),\epsilon)=\mathrm{MQ}_{\mathcal{A}}(access(q))$.

在下文对树的操作中, 将始终保证对于树中的任意一条边 $(q,\boldsymbol{\sigma}_r,q')$, 若 $vs_{\mathcal{A}}(access(q),[\![vw(\boldsymbol{\sigma}_r)]\!]) \neq \varnothing$, 则 $\boldsymbol{\sigma}_r \in vs_{\mathcal{A}}(access(q),[\![vw(\boldsymbol{\sigma}_r)]\!])$. 否则, $resets(\boldsymbol{\sigma}_r)=\{\top\}^{|C|}$. 这进而保证了在观察树中, 若从 q 出发, 经过一系列边到达节点 q', 则这一系列边中的重置时钟动作所组成的重置时钟字 $\boldsymbol{\sigma}_r^*$ 满足以下条件: 若 $vs_{\mathcal{A}}(access(q),[\![vw(\boldsymbol{\sigma}_r^*)]\!]) \neq \varnothing$, 则 $\boldsymbol{\sigma}_r^* \in vs_{\mathcal{A}}(access(q),[\![vw(\boldsymbol{\sigma}_r^*)]\!])$. 否则, $\boldsymbol{\sigma}_r^*$ 为 $access(q)$ 对应于 $[\![vw(\boldsymbol{\sigma}_r^*)]\!]$ 的半合法后继拼接字. 因此, $resets(\boldsymbol{\sigma}_r^*)=g_{tr}(access(q),[\![vw(\boldsymbol{\sigma}_r^*)]\!])$ 且 $f_{\mathcal{TR}}(q')=f_{tr}(access(q'),\epsilon)=f_{tr}(access(q)\cdot\boldsymbol{\sigma}_r^*,\epsilon)=f_{tr}(access(q),[\![vw(\boldsymbol{\sigma}_r^*)]\!])$.

相较于多时钟时间分类树 $\mathcal{CTR}=(\Sigma,V,Q_{tr},E_{tr},f_{tr}^E,g_{tr}^E,T_{Q_{tr}})$, 时间观察树是一个六元组, 而非七元组. 虽然在观察树中, 若从 q 出发, 经过一系列重置时钟动作所组成的重置时钟字 $\boldsymbol{\sigma}_r^*$ 到达节点 q', 则 $resets(\boldsymbol{\sigma}_r^*)=g_{tr}(access(q),[\![vw(\boldsymbol{\sigma}_r^*)]\!])$, 然而并没有在观察树中明确记录 $g_{tr}(access(q),[\![vw(\boldsymbol{\sigma}_r^*)]\!])$, 而是通过提取 $resets(\boldsymbol{\sigma}_r^*)$ 来获取 $g_{tr}(access(q),[\![vw(\boldsymbol{\sigma}_r^*)]\!])$.

2.2 分离关系

基于时间观察树的结构, 定义树中节点之间的分离关系如下.

定义 12. 分离关系. 对于时间观察树中任意的两个节点 $q_1,q_2 \in Q$, 若满足下列任意一个条件, 则 q_1,q_2 是相互分离的, 即 $q_1 \# q_2$:

● $\mathrm{MQ}_{\mathcal{A}}(access(q_1)) \neq \mathrm{MQ}_{\mathcal{A}}(access(q_2))$.

● 若树中存在访问序列为 $access(q_1)\gamma_{r1}$ 的节点 q_1' 和访问序列为 $access(q_2)\gamma_{r2}$ 的节点 q_2', 满足 $[\![vw(\gamma_{r1})]\!]=[\![vw(\gamma_{r2})]\!]$ 且树中记录了 $f_{\mathcal{TR}}(q_1')$ 和 $f_{\mathcal{TR}}(q_2')$, 则 $f_{\mathcal{TR}}(q_1') \neq f_{\mathcal{TR}}(q_2')$ 或 $resets(\gamma_{r1}) \neq resets(\gamma_{r2})$. 在这种情况下, 记区域字 $\xi=[\![vw(\gamma_{r1})]\!]$ 为 q_1 和 q_2 的可区分序列.

对于树中访问序列为 $access(q_1)\gamma_{r1}$ 的节点 q_1', 根据第 2.1 节中的描述可知, $resets(\gamma_{r1})=g_{tr}(access(q_1),[\![vw(\gamma_{r1})]\!])$ 且 $f_{\mathcal{TR}}(q_1')=f_{tr}(access(q_1),[\![vw(\gamma_{r1})]\!])$. 类似地, 若树中存在访问序列为 $access(q_2)\gamma_{r2}$ 的节点 q_2', 则 $resets(\gamma_{r2})=g_{tr}(access(q_2),[\![vw(\gamma_{r2})]\!])$ 且 $f_{\mathcal{TR}}(q_2')=f_{tr}(access(q_2),[\![vw(\gamma_{r2})]\!])$. 若 $[\![vw(\gamma_{r1})]\!]=[\![vw(\gamma_{r2})]\!]$, 则 $f_{\mathcal{TR}}(q_1') \neq f_{\mathcal{TR}}(q_2')$ 表明 $f_{tr}(access(q_1),[\![vw(\gamma_{r1})]\!]) \neq f_{tr}(access(q_2),[\![vw(\gamma_{r1})]\!])$, $resets(\gamma_{r1}) \neq resets(\gamma_{r2})$ 表明 $g_{tr}(access(q_1),[\![vw(\gamma_{r1})]\!]) \neq g_{tr}(access(q_2),[\![vw(\gamma_{r1})]\!])$. 在这种情况下, q_1,q_2 相互分离, $[\![vw(\gamma_{r1})]\!]$ 可用于区分 q_1 和 q_2.

定理 1. 对于两个节点 $q_1,q_2 \in Q$, 若在 $\mathcal{A}$ 中分别执行重置时钟字 $access(q_1)$ 和 $access(q_2)$ 后, 到达 $\mathcal{A}$ 中相同的符号化状态, 则 q_1 和 q_2 一定互不分离.

证明: 根据符号化状态的定义可知, 在 $\mathcal{A}$ 中分别执行 $access(q_1)$ 和 $access(q_2)$ 后, 会到达同一位置. 因此, $\mathrm{MQ}_{\mathcal{A}}(access(q_1))=\mathrm{MQ}_{\mathcal{A}}(access(q_2))$, 即分离关系中的第 1 个条件一定是不满足的. 接下来, 考虑分离关系中的第 2 个条件. 根据符号化状态的定义还可知, 在 $\mathcal{A}$ 中分别执行 $access(q_1)$ 和 $access(q_2)$ 之后, 时钟解释必定属于同一 region. 根据文献 [36] 可知, 对于属于同一 region 中的任意两个时钟解释 ν_1 和 ν_2, 随着时间流逝, ν_1+d 和 ν_2+d 依然属于同一个 region. 因此, 对于任意一个区域字 ξ, $access(q_1)$ 和 $access(q_2)$ 对应 ξ 的合法后继只有两种情况: 第 1 种情况是 $vs_{\mathcal{A}}(access(q_1),\xi) \neq \varnothing$ 且 $vs_{\mathcal{A}}(access(q_2),\xi) \neq \varnothing$. 第 2 种情况是 $vs_{\mathcal{A}}(access(q_1),\xi)=vs_{\mathcal{A}}(access(q_2),\xi)=\varnothing$.

首先考虑第 1 种情况. 若树中存在访问序列为 $access(q_1)\gamma_{r1}$ 的节点 q_1' 且 $[\![vw(\gamma_{r1})]\!]=\xi$, 则根据第 2.1 节中的描述可知, $\gamma_{r1} \in vs_{\mathcal{A}}(access(q_1),\xi)$. 类似地, 若树中存在访问序列为 $access(q_2)\gamma_{r2}$ 的节点 q_2' 且 $[\![vw(\gamma_{r2})]\!]=\xi$, 则 $\gamma_{r2} \in vs_{\mathcal{A}}(access(q_2),\xi)$. 由于 $vs_{\mathcal{A}}(access(q_1),\xi) \neq \varnothing$ 且 $vs_{\mathcal{A}}(access(q_2),\xi) \neq \varnothing$, 所以 $access(q_1)$ 和 $access(q_2)$ 都是合法的. 根据文献 [33] 中的引理 3.6 可知, $resets(\gamma_{r1})=resets(\gamma_{r2})$ 且 $access(q_1)\gamma_{r1}$ 和 $access(q_2)\gamma_{r2}$ 到达 $\mathcal{A}$ 中另一个相同的

符号化状态. 进而可知, $f_{tr}(access(q_1)\gamma_{r1},\epsilon)=f_{tr}(access(q_2)\gamma_{r2},\epsilon)$, 即 $f_{\mathcal{TR}}(q_1')=f_{tr}(access(q_1')\epsilon)=f_{tr}(access(q_1)\gamma_{r1},\epsilon)=f_{tr}(access(q_2)\gamma_{r2},\epsilon)=f_{tr}(access(q_2')\epsilon)=f_{\mathcal{TR}}(q_2')$. 因此, q_1 和 q_2 互不分离.

考虑第 2 种情况, 即 $vs_{\mathcal{A}}(access(q_1),\xi)=vs_{\mathcal{A}}(access(q_2),\xi)=\varnothing$. 若树中存在访问序列为 $access(q_1)\gamma_{r1}$ 的节点 q_1' 且 $[\![vw(\gamma_{r1})]\!]=\xi$, 则根据第 2.1 节的描述可知, γ_{r1} 为 $access(q_1)$ 对应于 ξ 的半合法后继拼接字. 类似地, 若树中存在访问序列为 $access(q_2)\gamma_{r2}$ 的节点 q_2' 且 $[\![vw(\gamma_{r2})]\!]=\xi$, 则 γ_{r2} 为 $access(q_2)$ 对应于 ξ 的半合法后继拼接字. 因此, $f_{tr}(access(q_1)\gamma_{r1},\epsilon)=f_{tr}(access(q_2)\gamma_{r2},\epsilon)=-$, 即 $f_{\mathcal{TR}}(q_1')=f_{\mathcal{TR}}(q_2')=-$. 接下来, 考虑重置信息. 根据文献 [36] 可知, 对于属于同一 region 中的任意两个时钟解释 ν_1 和 ν_2, 随着时间流逝, ν_1+d 和 ν_2+d 依然属于同一个 region. 因此, 如果在 ξ 的第 i 个位置找到 $access(q_1)$ 对应于 ξ 的半合法后继 $\gamma_{r1}'\in vs_{\mathcal{A}}(access(q_1),\xi[:i])$, 则一定可以在 ξ 的第 i 个位置找到 $access(q_2)$ 对应于 ξ 的半合法后继 $\gamma_{r2}'\in vs_{\mathcal{A}}(access(q_2),\xi[:i])$. 根据文献 [33] 中的引理 3.6 可知, $resets(\gamma_{r1}')=resets(\gamma_{r2}')$. 因此, $resets(\gamma_{r1})=\left\{resets(\gamma_{r1}'),\{\top\}^{(|\xi|-i)\times|C|}\right\}=resets(\gamma_{r2})$. 综上, q_1 和 q_2 互不分离.

基于分离关系, 将时间观察树中的节点分入 3 个集合中:

- 基础节点集合 S: 基础节点集合中的节点两两相互分离, 即对于任意两个节点 $q_1,q_2\in S$, $q_1\#q_2$.
- 边界节点集合 F: 边界节点集合是基础节点的直接后继节点的集合 (不包含已在基础节点集合中的节点), 即 $F=\{q'\in Q\backslash S\mid\exists q\in S,\boldsymbol{\sigma}_r\in\boldsymbol{\Sigma}_r:(q,\boldsymbol{\sigma}_r,q')\in E\}$. 若一个边界节点和所有的基础节点相互分离, 则称该边界节点是孤立的. 若一个边界节点只与基础节点集合中的一个节点不分离, 与其他所有的基础节点相互分离, 则称该边界节点是可识别的. 对于一个可识别的边界节点 $q\in F$, 将基础节点集合中唯一与 q 不分离的基础节点记为 $noapart(q)$.
- 其余节点集合 $Q\backslash(S\cup F)$.

在时间观察树中, 基础节点将对应于假设自动机 $\mathcal{H}$ 中的位置, 从基础节点出发的一部分边将对应于假设自动机中的迁移. 具体的由树构造 $\mathcal{H}$ 的方法将在下文详细介绍. 对于从一个基础节点 q 出发的边, 有一部分只用于将 q 和其他节点分离, 另一部分则真正用于构造从与 q 对应的 $\mathcal{H}$ 中的位置 l_q 出发的迁移. 利用集合 T_Q 来标记用于构造迁移的边, 以便在构造假设自动机中的迁移时使用. 具体而言, 对于一条用于构造迁移的边 $(q,\boldsymbol{\sigma}_r,q')\in E$, 将 $access(q')$ 存储在 T_Q 中.

假设图 1 中的 DTA $\mathcal{A}$ 为目标 DTA, 图 2 为学习过程中生成的一个时间观察树 $\mathcal{TR}$. 它包含 3 个由红色标记的基础节点, 9 个由蓝色标记的边界节点和 6 个由黑色标记的其余节点. 每个节点 q 中记录了 $f_{\mathcal{TR}}(q)$, 以节点 q_8 为例, $f_{\mathcal{TR}}(q_8)=f_{tr}(access(q_8),\epsilon)=\mathrm{MQ}_{\mathcal{A}}(access(q_8))=\mathrm{MQ}_{\mathcal{A}}((a,\{3,3\},\{\bot,\bot\})(a,\{3,3\},\{\top,\bot\}))=+$. 下面以节点 q_1 和节点 q_7 为例, 介绍如何判断这两个节点是相互分离的. 首先, $\mathrm{MQ}_{\mathcal{A}}(access(q_1))=\mathrm{MQ}_{\mathcal{A}}(access(q_7))=-$. 因此需要做进一步判断. 树中存在访问序列为 $access(q_1)\cdot(a,\{3,3\},\{\top,\top\})$ 的 q_9 和访问序列为 $access(q_7)\cdot(a,\{3,3\},\{\top,\bot\})$ 的节点 q_8, 满足 $[\![vw((a,\{3,3\},\{\top,\top\}))]\!]=[\![vw((a,\{3,3\},\{\top,\bot\}))]\!]$ 且树中记录了 $f_{\mathcal{TR}}(q_9)$ 和 $f_{\mathcal{TR}}(q_8)$. 然而, $f_{\mathcal{TR}}(q_9)\neq f_{\mathcal{TR}}(q_8)$ 且 $resets((a,\{3,3\},\{\top,\top\}))\neq resets((a,\{3,3\},\{\top,\bot\}))$. 由此可知, 节点 q_1 和节点 q_7 是相互分离的, 区域字 $\xi=[\![(a,\{3,3\})]\!]$ 为 q_1 和 q_7 的可区分序列.

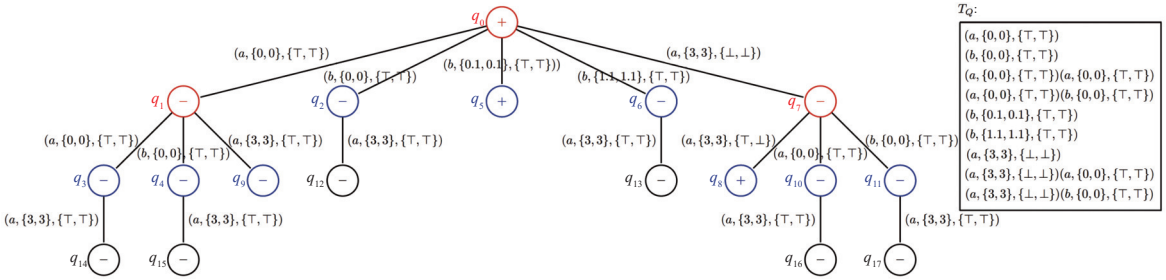


图 2 时间观察树示例 $\mathcal{TR}$

2.3 时间观察树的性质和操作

在构造假设自动机之前, 时间观察树需要满足以下两个性质.

● 基础节点集合 S 是完备的. 对于任意一个基础节点 $q\in S$ 和任意一个动作 $\sigma\in\Sigma$, 存在边 $e=(q,\boldsymbol{\sigma}_r,q')\in E$. 其中, $vw(\boldsymbol{\sigma}_r)=\left(\sigma,\{0\}^{|C|}\right)$ 且 $access(q')\in T_Q$.

● 对于任意的边界节点 $q\in F$, 若 $access(q)\in T_Q$, 则 q 为非孤立的且可识别的.

下面定义以下对树的操作. 首先, 定义对树的基本操作: 构建边操作.

● 构建边操作. 对于一个节点 q_0 和一个重置时钟字 $\boldsymbol{\sigma}_r^*=\boldsymbol{\sigma}_{r1}\boldsymbol{\sigma}_{r2}\ldots\boldsymbol{\sigma}_{rn}$, 可构建从 q_0 出发, 执行 $\boldsymbol{\sigma}_r^*$ 到达节点 q_n 的一系列边. 其中, $access(q_n)=access(q_0)\cdot\boldsymbol{\sigma}_r^*$. 为此需要执行以下步骤: 从重置时钟字 $\boldsymbol{\sigma}_r^*$ 的第 1 个重置时钟动作 $\boldsymbol{\sigma}_{r1}$ 开始, 对于每一个重置时钟动作 $\boldsymbol{\sigma}_{ri}$, 构建一个节点 q_i 以及一条相应的边 $(q_{i-1},\boldsymbol{\sigma}_{ri},q_i)$. 通过此操作, $access(q_i)=access(q_{i-1})\cdot\boldsymbol{\sigma}_{ri}$.

接下来, 分别介绍 3 种操作: 使树完备化、使边界节点非孤立且可识别、将可识别的边界节点转换为孤立的.

● 使树完备化. 每当有新的节点 s 加入基础节点集合 S 时, 对于任意一个动作 $\sigma\in\Sigma$, 执行如下操作.

(1) 若 $vw(access(s))\cdot\left(\sigma,\{0\}^{|C|}\right)$ 是合法的, 则学生通过重置信息查询 $\mathrm{RQ}_{\mathcal{A}}\left(vw(access(s))\cdot\left(\sigma,\{0\}^{|C|}\right)\right)$ 得到一个重置时钟字 $access(s)\cdot\left(\sigma,\{0\}^{|C|},\mathbf{b}\right)$, 并记重置时钟动作 $\boldsymbol{\sigma}_r=\left(\sigma,\{0\}^{|C|},\mathbf{b}\right)$. 否则, 学生构造一个重置时钟动作 $\boldsymbol{\sigma}_r=\left(\sigma,\{0\}^{|C|},\mathbf{b}\right)$, 其中 $\mathbf{b}=\{\top\}^{|C|}$.

(2) 构建从 q 出发, 执行重置时钟动作 $\boldsymbol{\sigma}_r$ 到达 q' 的边 $e=(q,\boldsymbol{\sigma}_r,q')$, 更新边界节点集合 F 并将 $access(q')$ 加入 T_Q, 再寻找 $noapart(q')$.

● 使边界节点非孤立且可识别. 每当有新的节点 f 加入边界节点集合 F 且 $access(f)\in T_Q$ 时, 需要在基础节点集合 S 中寻找唯一与之不分离的基础节点 s, 即 $noapart(f)$. 若不存在与 f 不分离的基础节点, 即 f 是孤立的, 则将 f 从边界节点集合 F 中移除并加入基础节点集合 S 中. 若 f 是不可识别的, 则在基础节点集合 S 中, 至少存在两个节点 s_1,s_2 与 f 不分离. 在这种情况下, 利用 s_1 和 s_2 的可区分序列 ξ 来进一步判断 f 与 s_1 和 s_2 是否分离. 为此, 需要先寻找 $access(f)$ 对应于 ξ 的合法后继 (若存在合法后继) 或半合法后继拼接字 (若不存在合法后继), 并将其记为 $\boldsymbol{\sigma}_r^*$, 再构建从 f 出发, 执行 $\boldsymbol{\sigma}_r^*$ 到达节点 q 的一系列边, 然后根据 $access(q)$ 计算 $f_{\mathcal{TR}}(q)$ 并将 $f_{\mathcal{TR}}(q)$ 记录在树中.

● 将可识别的边界节点转换为孤立的. 假设时间观察树中存在一个可识别的边界节点 f, 且基础节点 $s=noapart(f)$. 此时, 若存在一个区域字 ξ 满足 $f_{tr}(access(s),\xi)\neq f_{tr}(access(f),\xi)$ 或 $g_{tr}(access(s),\xi)\neq g_{tr}(access(f),\xi)$, 则可以利用区域字 ξ 对 s 和 f 进行区分. 为此, 执行如下操作.

(1) 寻找 $access(s)$ 对应于 ξ 的合法后继 (若存在合法后继) 或半合法后继拼接字 (若不存在合法后继), 并将其记为 $\boldsymbol{\sigma}_{r1}^*$. 构建从 s 出发, 执行 $\boldsymbol{\sigma}_{r1}^*$ 到达节点 q 的一系列边, 再计算 $f_{\mathcal{TR}}(q)$ 并将 $f_{\mathcal{TR}}(q)$ 记录在树中.

(2) 类似地, 寻找 $access(f)$ 对应于 ξ 的合法后继 (若存在合法后继) 或半合法后继拼接字 (若不存在合法后继), 并将其记为 $\boldsymbol{\sigma}_{r2}^*$. 构建从 f 出发, 执行 $\boldsymbol{\sigma}_{r2}^*$ 到达节点 q' 的一系列边, 再计算 $f_{\mathcal{TR}}(q')$ 并将 $f_{\mathcal{TR}}(q')$ 记录在树中.

(3) 经过操作 (1) 和 (2), f 是孤立的, 将 f 从边界节点集合 F 中移除并加入基础节点集合 S 中.

(4) 对于观察树中所有满足 $noapart(f')=s$ 的边界节点 f', 若 $access(f')\in T_Q$, 则需要利用区域字 ξ 来重新计算 $noapart(f')$.

定理 2. 对于从一个基础节点 q 出发的任意两条边 $(q,\boldsymbol{\sigma}_{r1},q')$ 和 $(q,\boldsymbol{\sigma}_{r2},q'')$, 若 $[\![vw(\boldsymbol{\sigma}_{r1})]\!]=[\![vw(\boldsymbol{\sigma}_{r2})]\!]$, 则 $resets(\boldsymbol{\sigma}_{r1})=resets(\boldsymbol{\sigma}_{r2})$ 且 $noapart(q')=noapart(q'')$.

证明: 首先考虑 $vs_{\mathcal{A}}(access(q),[\![vw(\boldsymbol{\sigma}_{r1})]\!])\neq\varnothing$ 的情况. 由于对树的操作中, 始终保证对于树中任意一条边 $(q,\boldsymbol{\sigma}_r,q')$, 若 $vs_{\mathcal{A}}(access(q),[\![vw(\boldsymbol{\sigma}_r)]\!])\neq\varnothing$, 则 $\boldsymbol{\sigma}_r\in vs_{\mathcal{A}}(access(q),[\![vw(\boldsymbol{\sigma}_r)]\!])$, $\boldsymbol{\sigma}_{r1}$、$\boldsymbol{\sigma}_{r2}$ 均属于 $vs_{\mathcal{A}}(access(q),[\![vw(\boldsymbol{\sigma}_{r1})]\!])$. 由于 $vs_{\mathcal{A}}(access(q),[\![vw(\boldsymbol{\sigma}_{r1})]\!])\neq\varnothing$, $access(q)$ 是合法的, 根据文献 [33] 的引理 3.5 可知, $access(q)\cdot\boldsymbol{\sigma}_{r1}$ (即 $access(q')$) 和 $access(q)\cdot\boldsymbol{\sigma}_{r2}$ (即 $access(q'')$) 在 $\mathcal{A}$ 中执行相同迁移序列并到达相同符号化状态. 因此, $resets(\boldsymbol{\sigma}_{r1})=resets(\boldsymbol{\sigma}_{r2})$ 且通过定理 1 可知, q' 和 q'' 互不分离. 进而可知, $noapart(q')=noapart(q'')$.

再考虑 $vs_{\mathcal{A}}(access(q),[\![vw(\boldsymbol{\sigma}_{r1})]\!])=\varnothing$ 的情况. 在对树的操作中, 对于这种情况, 将始终保证 $resets(\boldsymbol{\sigma}_{r1})=resets(\boldsymbol{\sigma}_{r2})=\{\top\}^{|C|}$. 下面判断 $noapart(q')$ 和 $noapart(q'')$ 是否相同. 由于 $access(q)\cdot\boldsymbol{\sigma}_{r1}$ (即 $access(q')$) 和 $access(q)\cdot\boldsymbol{\sigma}_{r2}$ (即 $access(q'')$) 均为不合法的, 因此 $\mathrm{MQ}_{\mathcal{A}}(access(q'))=\mathrm{MQ}_{\mathcal{A}}(access(q''))=-$, 即分离关系的第 1 个条件不满足. 此

外, 由于 $access(q')$ 和 $access(q'')$ 均为不合法的, 若树中存在访问序列为 $access(q')\gamma_{r1}$ 的节点 q'_1 和访问序列为 $access(q'')\gamma_{r2}$ 的节点 q'_2, 满足 $[\![vw(\gamma_{r1})]\!]=[\![vw(\gamma_{r2})]\!]$, 则 $resets(\gamma_{r1})=resets(\gamma_{r2})=\{\top\}^{|C|\times|\gamma_{r1}|}$ 且 $f_{\mathcal{TR}}(q'_1)=f_{\mathcal{TR}}(q'_2)=-$. 因此, 分离关系的第 2 个条件不满足. 综上, q' 和 q'' 互不分离.

3 改进的学习算法

基于第 2 节提出的时间观察树, 设计改进的 DTA 学习算法, 算法整体流程如下.

(1) 学生首先初始化一个时间观察树 $\mathcal{TR}=(\Sigma,Q,E,access,T_Q,f_{\mathcal{TR}})$. 初始化步骤如下: 学生首先创建一个根节点 q, q 的访问序列 $access(q)=\epsilon$. 再根据访问序列计算出 $f_{\mathcal{TR}}(q)$ 并将 $f_{\mathcal{TR}}(q)$ 记录在树中. 接下来, 将节点 q 加入基础节点集合中, 并使树完备化, 在进行完备化操作时需要进行重置信息查询并填充集合 E 和 T_Q.

(2) 学生先将时间观察树转换为一个 DFA M, 再利用文献 [33] 所提出的划分函数将 DFA M 转换成一个 DTA $\mathcal{H}$ 来作为一个假设自动机.

(3) 学生发起等价查询来判断 $\mathcal{L}(\mathcal{A})$ 是否等于 $\mathcal{L}(\mathcal{H})$. 文献 [33] 中提出了等价查询的实现方式, 其基本步骤如下: ① 为了判断 $\mathcal{L}(\mathcal{A})=\mathcal{L}(\mathcal{H})$ 是否成立, 需要判断 $\mathcal{L}(\mathcal{H})\subseteq\mathcal{L}(\mathcal{A})$ 和 $\mathcal{L}(\mathcal{A})\subseteq\mathcal{L}(\mathcal{H})$ 是否成立; ② 由于确定性时间自动机在求补操作下是封闭的[35], $\mathcal{L}(\mathcal{H})\subseteq\mathcal{L}(\mathcal{A})$ 当且仅当 $\mathcal{L}(\mathcal{H})\cap\overline{\mathcal{L}(\mathcal{A})}=\varnothing$, $\mathcal{L}(\mathcal{A})\subseteq\mathcal{L}(\mathcal{H})$ 当且仅当 $\mathcal{L}(\mathcal{A})\cap\overline{\mathcal{L}(\mathcal{H})}=\varnothing$; ③ 若 $\mathcal{L}(\mathcal{H})\subseteq\mathcal{L}(\mathcal{A})$ 不成立, 即 $\mathcal{L}(\mathcal{H})\cap\overline{\mathcal{L}(\mathcal{A})}\neq\varnothing$, 则存在反例能被 $\mathcal{H}$ 接收但不能被 $\mathcal{A}$ 接收; 若 $\mathcal{L}(\mathcal{A})\subseteq\mathcal{L}(\mathcal{H})$ 不成立, 则存在反例能被 $\mathcal{A}$ 接收但不能被 $\mathcal{H}$ 接收. 如果等价查询答案为“否”, 则老师返回一个重置时间字 ω_r 作为反例.

(4) 学生对 ω_r 进行分析以修改时间观察树 $\mathcal{TR}$. 在对反例进行分析时, 学生发起一系列重置信息查询和成员查询来寻找冲突. 在修改时间观察树的过程中, 学生也需要发起一系列重置信息查询和成员查询来对树进行操作. 分析反例和修改树的方法将在下面详细介绍. 最后, 根据新的树构造新的假设自动机并发起等价查询. 学生重复步骤 (1)–(4), 直到等价查询的结果为“是”.

3.1 构建假设自动机

给定一个时间观察树 $\mathcal{TR}=(\Sigma,Q,E,access,T_Q,f_{\mathcal{TR}})$, 可以根据 $\mathcal{TR}$ 构建一个假设自动机 $\mathcal{H}$. 该过程需要如下两步: ① 将时间观察树转换为一个 DFA M; ② 将 DFA M 转换成一个 DTA $\mathcal{H}$.

对于一个给定的时间观察树 $\mathcal{TR}=(\Sigma,Q,E,access,T_Q,f_{\mathcal{TR}})$, 首先将其转换为对应的 DFA $M=(L_M,l_M^0,F_M,\Sigma_M,\Delta_M)$, 其中,

- 位置的有穷集合 $L_M=\{l_q\mid q\in S\}$;
- 初始位置 $l_M^0=l_q$, 其中 $q\in S$ 且 $access(q)=\epsilon$;
- 接收位置的有穷集合 $F_M=\{l_q\mid q\in S\wedge access(q)\in L_r(\mathcal{A})\}$;
- 有穷的抽象字母表 $\Sigma_M=\{\boldsymbol{\sigma}_r\mid q\in S\wedge(q,\boldsymbol{\sigma}_r,q')\in E\wedge access(q')\in T_Q\}$;
- 迁移的有穷集合 $\Delta_M=\{(l_q,\boldsymbol{\sigma}_r,l_{q'})\mid q\in S\wedge q'\in S\wedge(q,\boldsymbol{\sigma}_r,q')\in E\wedge access(q')\in T_Q\}\cup\{(l_q,\boldsymbol{\sigma}_r,l_{noapart(q')})\mid q\in S\wedge q'\in F\wedge(q,\boldsymbol{\sigma}_r,q')\in E\wedge access(q')\in T_Q\}$.

通过以上转换, 时间观察树 $\mathcal{TR}$ 中的基础节点和 DFA M 中的位置是一一对应的, M 的初始位置对应于 $\mathcal{TR}$ 的根节点. 对于基础节点之间的每一条边 $(q,\boldsymbol{\sigma}_r,q')$, 若 $access(q')\in T_Q$, 则 M 中都有一条迁移与之对应. 对于每一条从基础节点 q 出发到达边界节点 q' 的边 $(q,\boldsymbol{\sigma}_r,q')$, 若 $access(q')\in T_Q$, 则 M 中也有一条迁移与之对应, 且该迁移的目标位置对应于 $\mathcal{TR}$ 中与边界节点 q' 不分离的基础节点 $noapart(q')$.

在得到 DFA M 之后, 利用文献 [33] 所提出的划分函数, 将 DFA M 转换成 DTA $\mathcal{H}$.

定义 13. 导致冲突的重置时钟字. 对于树中的一个访问序列 γ_r, 若满足以下两个条件之一, 则称重置时钟字 γ_r 导致了冲突.

- γ_r 在 $\mathcal{H}$ 中是不可执行的. 具体而言, 记时钟字 $vw(\gamma_r)$ 在 $\mathcal{H}$ 中对应的重置时钟字为 γ'_r, $resets(\gamma_r)\neq resets(\gamma'_r)$.
- 记 tr 为在树中执行 γ_r 所到达的节点, hy 为在 $\mathcal{H}$ 中执行 γ_r 所到达的位置在树中对应的基础节点. 存在一个区域字 ξ 满足 $f_{tr}(access(tr),\xi)\neq f_{tr}(access(hy),\xi)$ 或 $g_{tr}(access(tr),\xi)\neq g_{tr}(access(hy),\xi)$, 即可以利用 ξ 对 tr 和 hy

进行区分.

定理 3. 对于一个重置时钟字 γ_r, 若在树中执行 γ_r 所到达的节点为基础节点, 则 γ_r 必定不会导致冲突.

证明: 根据由树构建 DFA M 的过程可知, 树中的基础节点与 M 中的位置一一对应, 且对于任意两个基础节点 q 和 q' 之间的边 $(q,\boldsymbol{\sigma}_r,q')$, 若 $access(q')\in T_Q$, 则 M 中都有一条唯一与之对应的迁移 $(l_q,\boldsymbol{\sigma}_r,l_{q'})$. 根据全文中对树的操作可知, 对于任意两个基础节点 q 和 q' 之间的边 $(q,\boldsymbol{\sigma}_r,q')$, $access(q')$ 均属于 T_Q. 因此, 对于基础节点之间任意的边 $(q,\boldsymbol{\sigma}_r,q')$, M 中总有一条唯一与之对应的迁移 $(l_q,\boldsymbol{\sigma}_r,l_{q'})$. 由此可知, 从 γ_r 的第 1 个重置时钟动作 $\boldsymbol{\sigma}_{r1}$ 开始, 依次在树中和 M 中执行每个重置时钟动作 $\boldsymbol{\sigma}_{ri}$, 若在树中执行重置时钟动作 $\boldsymbol{\sigma}_{ri}$ 后到达节点 q_i, 则在 M 中执行 $\boldsymbol{\sigma}_{ri}$ 会触发到达位置 l_{q_i} 的迁移. 因此, 若在树中执行 γ_r 到达基础节点 tr, 则在 $\mathcal{H}$ 中执行 γ_r 所到达的位置必定对应于基础节点 tr, 即 $hy=tr$. 进而可知, γ_r 必定不会导致树和 $\mathcal{H}$ 之间发生冲突.

3.2 反例分析与处理

当等价查询结果为“否”时, 老师返回一个重置时间字 ω_r 作为反例. 学生需要对该反例进行分析和处理以修改时间观察树. 为此, 学生首先将 ω_r 转换为相应的重置时钟字 $ctx=\Gamma(\omega_r)$, 再利用 ctx 对观察树进行扩展. 具体步骤如下: 构建从根节点出发, 执行 ctx 到达节点 q 的一系列边, 并根据 $access(q)$ 计算 $f_{\mathcal{TR}}(q)$ 并将 $f_{\mathcal{TR}}(q)$ 记录在树中, 同时更新边界节点集合和其他节点集合. 由于 ctx 本身会导致冲突, 记 ctx 为 $conflict$. 然后根据 $conflict$ 进行如下分析和处理.

反例分析: 若在树中执行 $conflict$ 所到达的节点 $tr\notin S\cup F$, 则采用二分搜索方式不断寻找比 $conflict$ 短的导致冲突的重置时钟字 γ_r, 并更新 $conflict$ 为 γ_r. 该过程直至在树中执行 $conflict$ 所到达的节点 $tr\in S\cup F$. 具体操作如下.

对于一个重置时钟字 γ_r, 用 $\gamma_r[:i]$ 来表示 γ_r 的第 i 个位置 (包括第 i 个位置) 之前的前缀, 用 $\gamma_r[i:]$ 来表示 γ_r 的第 i 个位置 (包括第 i 个位置) 之后的后缀, 用 $\gamma_r[i]$ 来表示 γ_r 的第 i 个位置的重置时钟动作. 学生寻找 $conflict$ 中达到边界节点集合 F 的前缀 ρ, 并记 ρ 的长度为 x. 再将 $conflict$ 拆分为 $\gamma_{r1}\cdot\gamma_{r2}$. 其中, $|\gamma_{r1}|=\left\lfloor\dfrac{x+|conflict|}{2}\right\rfloor$. 记在时间观察树中执行当前的 $conflict$ 所达到的节点为 tr, 在假设自动机中执行当前的 $conflict$ 所到达的位置在树中对应的基础节点为 hy. 类似地, 记在树中执行 γ_{r1} 所达到的节点为 tr', 在 $\mathcal{H}$ 中执行 γ_{r1} 所到达的位置在树中对应的基础节点为 hy'. 然后进行如下分析.

- 首先判断 γ_{r1} 在 $\mathcal{H}$ 中是否是可执行的. 为此, 需要寻找时钟字 $vw(\gamma_{r1})$ 在 $\mathcal{H}$ 中对应的重置时钟字 γ'_{r1}, 再判断 $resets(\gamma_{r1})=resets(\gamma'_{r1})$ 是否成立. 若成立, 学生根据 γ_{r2} 以及 tr 和 hy 的可区分序列 η 构造一个区域字 $\xi=[\![vw(\gamma_{r2})\cdot\eta]\!]$. 注意, 当 $conflict=ctx$ 时, $\eta=\epsilon$. 然后计算 $f_{tr}(access(hy'),\xi)$ 和 $g_{tr}(access(hy'),\xi)$, 以判断 $\xi=[\![vw(\gamma_{r2})\cdot\eta]\!]$ 能否对 hy' 和 tr' 进行区分.

(1) 若可以利用 $\xi=[\![vw(\gamma_{r2}\cdot\eta)]\!]$ 对 hy' 和 tr' 进行区分, 则 γ_{r1} 会导致冲突. 学生将 $conflict$ 更新为 γ_{r1}.

(2) 若无法利用 $\xi=[\![vw(\gamma_{r2}\cdot\eta)]\!]$ 对 hy' 和 tr' 进行区分, 则可以利用 hy' 和 tr' 的等价性来缩短 $conflict$. 具体而言, 寻找 $access(hy')$ 对应于区域字 $[\![vw(\gamma_{r2})]\!]$ 的合法后继 (若存在合法后继) 或半合法后继拼接字 (若不存在合法后继), 并将其记为 γ'_{r2}. 然后, 利用 $access(hy')\cdot\gamma'_{r2}$ 对树进行扩展, 并将 $conflict$ 由 $\gamma_{r1}\cdot\gamma_{r2}$ 更新为 $access(hy')\cdot\gamma'_{r2}$. 此时, 导致冲突的可区分序列仍为 η.

- 若 γ_{r1} 在 $\mathcal{H}$ 中是不可执行的, 则存在满足以下条件的 γ_{r1} 中的位置 i: $resets(\gamma_{r1}[i])\neq resets(\gamma'_{r1}[i])$ 且对于任意 $1\leqslant j<i$, $resets(\gamma_{r1}[j])=resets(\gamma'_{r1}[j])$. 然后, 记在 $\mathcal{H}$ 中执行 $\gamma_{r1}[:(i-1)]$ 所到达的位置在树中对应的基础节点为 s. 构造一个区域字 $[\![vw(\gamma_{r1}[i])]\!]$, 并采用如下方式构建一个重置时钟动作 $\boldsymbol{\sigma}_r$: 若存在 $e_r\in vs_{\mathcal{A}}(access(s),[\![vw(\gamma_{r1}[i])]\!])$, 则记 e_r 为 $\boldsymbol{\sigma}_r$. 否则, 令 $\boldsymbol{\sigma}_r=\left(vw(\gamma_{r1}[i]),\{\top\}^{|\mathcal{C}|}\right)$. 在得到 $\boldsymbol{\sigma}_r$ 后, 判断 $resets(\boldsymbol{\sigma}_r)=resets(\gamma_{r1}[i])$ 是否成立.

(3) 若 $resets(\boldsymbol{\sigma}_r)\neq resets(\gamma_{r1}[i])$, 则说明区域字 $[\![vw(\gamma_{r1}[i])]\!]$ 可以将 s 和在树中执行 $\gamma_{r1}[:(i-1)]$ 所到达的节点区分开. 因此, 学生将 $conflict$ 更新为 $\gamma_{r1}[:(i-1)]$.

(4) 若 $resets(\boldsymbol{\sigma}_r)=resets(\gamma_{r1}[i])$, 则 $resets(\boldsymbol{\sigma}_r)\neq resets(\gamma'_{r1}[i])$, 从而说明在 $\mathcal{H}$ 中, 从 s 对应的位置 l_s 出发的迁移上的时间区间划分粒度较粗, 导致执行 $vw(\gamma_{r1}[i])$ 时, 时钟的重置信息发生了错误. 因此, 利用 $s\cdot\boldsymbol{\sigma}_r$ 对树进行扩展,

并将 *conflict* 更新为 $access(s)\cdot\sigma_r$. 注意, 此时在树中执行新的 *conflict* 所到达的节点 $tr\in S\cup F$.

反例处理: 在根据当前的 *conflict* 更新 *tr* 和 *hy* 后, 若 $tr\in S\cup F$, 则反例分析结束. 接下来, 利用如下操作来处理反例, 以修改时间观察树.

● 若最终的 *conflict* 是在上述情况 (1)–(3) 中得到的, 则将 *conflict* 加入 T_Q 中, 并记 $noapart(tr)=n$. 若 $n=hy$, 则在树中 *tr* 与 *hy* 原本是不分离的. 此时, 利用已找到的可区分序列将 *tr* 与 *hy* 分离. 具体方法参见第 2.3 节中将可识别的边界节点转换为孤立的操作. 至此, 反例处理结束. 若 $n\neq hy$, 则在树中 *tr* 与 *hy* 是分离的. 在这种情况下, 记在树中执行 $conflict[:|conflict|-1]$ 所到达的基础节点为 *s*, 则在 $\mathcal{H}$ 中从位置 l_s 出发, 执行 $conflict\left[\left|conflict\right|\right]$ 到达位置 l_{hy} 的迁移是一条错误的迁移, 该迁移本该到达 l_n. 原因是该迁移上时间区间划分的粒度较粗. 通过将 *conflict* (即 $access(tr)$) 加入 T_Q 并计算 $noapart(tr)$, 使得在新的树构成的 DFA 中, 存在目标位置为 l_n 的迁移 $\left(l_s,conflict\left[\left|conflict\right|\right],l_n\right)$, 从而使得在新的 $\mathcal{H}$ 中, 从位置 l_s 出发, 执行 $conflict\left[\left|conflict\right|\right]$ 到达位置 l_n. 至此, 反例处理结束.

● 若起初在树中执行 *conflict* (即 *ctx*) 所到达的节点 $tr\in S\cup F$, 或最终的 *conflict* 是在上述情况 (4) 中得到的, 则直接将 *conflict* 加入 T_Q 中, 并计算 $noapart(tr)$. 至此, 反例处理结束.

下面通过 3 个示例来展示不同情况下的反例分析和处理过程.

假设图 1 中的 DTA $\mathcal{A}$ 为目标 DTA, 图 3 中 $\mathcal{TR}_1$ 为学习过程中生成的一个时间观察树. 根据 $\mathcal{TR}_1$, 学生构建 M_1 和 $\mathcal{H}_1$, 然后发起等价查询从而得到 $\omega_r=(a,3,\{\bot,\bot\})(a,0,\{\top,\bot\})$. 学生利用 $\Gamma(\omega_r)$ 对树进行扩展并得到 $ctx=(a,\{3,3\},\{\bot,\bot\})(a,\{3,3\},\{\top,\bot\})=conflict$. 由于在树中执行 *conflict* 所到达的节点 $q_8\notin S\cup F$, 且在执行 *conflict* 时, 达到集合 *F* 的前缀 $\rho=(a,\{3,3\},\{\bot,\bot\})$, 因此将 *conflict* 拆分为 $\gamma_{r1}\cdot\gamma_{r2}$. 其中, $\gamma_{r1}=(a,\{3,3\},\{\bot,\bot\})$, $\gamma_{r2}=(a,\{3,3\},\{\top,\bot\})$. 由于时钟字 $vw(\gamma_{r1})$ 在 $\mathcal{H}_1$ 中对应的重置时钟字 $\gamma'_{r1}=(a,\{3,3\},\{\top,\top\})\neq\gamma_{r1}$, 即 $resets(\gamma'_{r1}[1])\neq resets(\gamma_{r1}[1])$, 学生找到了在 $\mathcal{H}_1$ 中执行 $\gamma_{r1}[:(1-1)]$ 所到达的位置 l_0, l_0 在树中所对应的基础节点为 q_0. 然后构造一个区域字 $[\![vw(\gamma_{r1}[1])]\!]=[\![(a,\{3,3\})]\!]$, 并找到 $e_r=(a,\{3,3\},\{\bot,\bot\})\in vs_{\mathcal{A}}(access(q_0),[\![(a,\{3,3\})]\!])$. 由于 $resets(e_r)=resets(\gamma_{r1}[1])$, 学生在树中构建从 q_0 出发, 执行 $(a,\{3,3\},\{\bot,\bot\})$ 到达 q_7 的边 (该边已存在于树中), 并将 $access(q_7)$ 加入 T_Q 中, 再计算出 $noapart(q_7)=q_1$, 进而得到了 $\mathcal{TR}_2$.

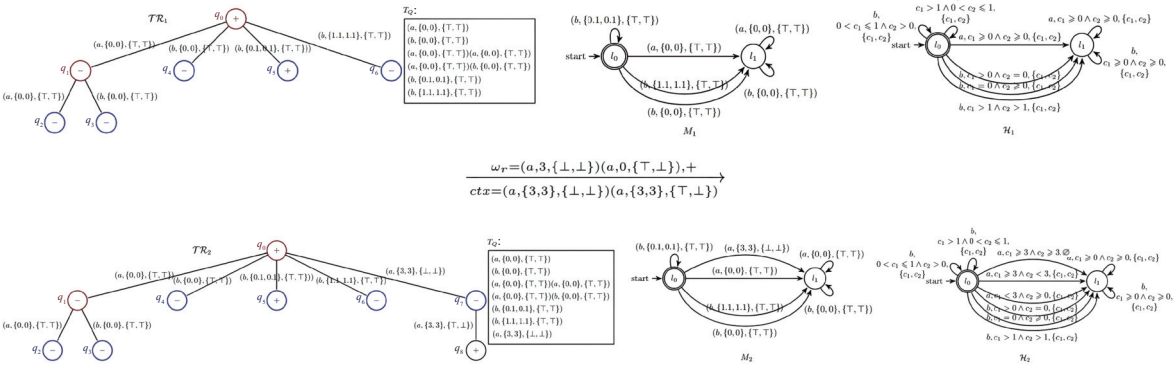


图 3 反例分析示例 1

假设图 1 中的 DTA $\mathcal{A}$ 为目标 DTA, 图 4 中 $\mathcal{TR}_1$ 为学习过程中生成的一个时间观察树. 学生构建 M_1 和 $\mathcal{H}_1$, 发起等价查询从而得到 $\omega_r=(a,3,\{\bot,\bot\})(a,0,\{\top,\bot\})$. 学生利用 $\Gamma(\omega_r)$ 对树进行扩展并得到 $ctx=(a,\{3,3\},\{\bot,\bot\})(a,\{3,3\},\{\top,\bot\})=conflict$. 由于在树中执行 *conflict* 所到达的节点 $q_8\notin S\cup F$, 学生将 *conflict* 拆分为 $\gamma_{r1}\cdot\gamma_{r2}$. 其中, $\gamma_{r1}=(a,\{3,3\},\{\bot,\bot\})$, $\gamma_{r2}=(a,\{3,3\},\{\top,\bot\})$. 由于 γ_{r1} 在 $\mathcal{H}_1$ 中是可执行的, 学生进一步发现: 在树中执行 *conflict* 所达到的节点 $tr=q_8$, 在 $\mathcal{H}_1$ 中执行 *conflict* 所到达的位置在树中对应的基础节点 $hy=q_1$, 在树中执行 $\gamma_{r1}=(a,\{3,3\},\{\bot,\bot\})$ 所达到的节点 $tr'=q_7$, 在 $\mathcal{H}_1$ 中执行 γ_{r1} 所到达的位置对应的基础节点 $hy'=q_1$. 学生根据 γ_{r2} 以及 q_8 和 q_1 的可区分序列 η, 即 ϵ, 构造一个区域字 $\xi=[\![vw(\gamma_{r2}\cdot\eta)]\!]=[\![(a,\{3,3\})]\!]$. 由于 $f_{tr}(access(q_1),\xi)=-\neq f_{tr}(access(q_7),\xi)$

且 $g_{tr}(access(q_1),\xi)=\{\top,\top\}\neq g_{tr}(access(q_7),\xi)$, 可知 ξ 可以对 q_1 和 q_7 进行区分, 即 γ_{r1} 会导致冲突. 学生将 *conflict* 更新为 $\gamma_{r1}=(a,\{3,3\},\{\bot,\bot\})$, 并根据新的 *conflict* 更新 tr 和 hy, 得到 $tr=q_7$, $hy=q_1$. 由于 $q_7\in F$, 学生将新的 *conflict* 加入 T_Q 中, 并寻找 $noapart(q_7)$. 由于 $noapart(q_7)=q_1$, 需要利用 $[\![(a,\{3,3\})]\!]$ 来区分 q_1 和 q_7. 具体方法参见第 2.3 节中将可识别的边界节点转换为孤立的操作. 在此过程中, 得到 $noapart(q_7)=\varnothing$, 需要将 q_7 移动到集合 S 中, 并构建从 q_7 出发, 分别执行 $(a,\{0,0\},\{\top,\top\})$ 和 $(b,\{0,0\},\{\top,\top\})$ 的边. 此外, 对于所有满足 $noapart(f')=q_1$ 的边界节点 f', 若 $f'\in T_Q$, 则需要利用 $[\![(a,\{3,3\})]\!]$ 来重新计算 $noapart(f')$. 在完成这些操作后, 得到 $\mathcal{TR}_2$.

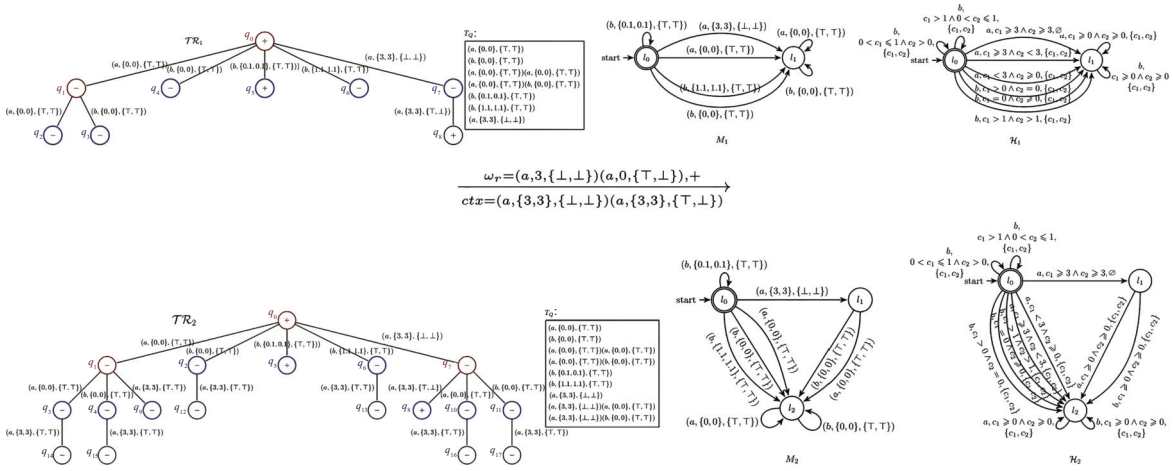


图 4　反例分析示例 2

假设图 1 中的 DTA $\mathcal{A}$ 为目标 DTA, 图 5 中的 $\mathcal{TR}_1$ 为学习过程中生成的一个时间观察树. 学生构建 DFA M_1 和假设 DTA $\mathcal{H}_1$. 经过等价查询, 学生得到了一个新的反例 $\omega_r=(a,3,\{\bot,\bot\})(a,0,\{\top,\bot\})(b,0.5,\{\top,\top\})$. 学生利用 $\Gamma(\omega_r)$ 对树进行扩展并得到 $ctx=(a,\{3,3\},\{\bot,\bot\})(a,\{3,3\},\{\top,\bot\})(b,\{0.5,3.5\},\{\top,\top\})=conflict$, 再将 *conflict* 拆分为 $\gamma_{r1}\cdot\gamma_{r2}$. 其中, $\gamma_{r1}=(a,\{3,3\},\{\bot,\bot\})(a,\{3,3\},\{\top,\bot\})$, $\gamma_{r2}=(b,\{0.5,3.5\},\{\top,\top\})$. 由于 γ_{r1} 在 $\mathcal{H}_1$ 中是可执行的, 学生进一步发现: 在树中执行 *conflict* 所达到的节点 $tr=q_{18}$, 在 $\mathcal{H}_1$ 中执行 *conflict* 所到达的位置在树中对应的基础节点 $hy=q_0$, 在树中执行 $\gamma_{r1}=(a,\{3,3\},\{\bot,\bot\})(a,\{3,3\},\{\top,\bot\})$ 所达到的节点 $tr'=q_8$, 在 $\mathcal{H}_1$ 中执行 γ_{r1} 所到达的位置对应的基础节点 $hy'=q_0$. 学生根据 γ_{r2} 以及 q_{18} 和 q_0 的可区分序列 η 构造一个区域字 $\xi=[\![vw(\gamma_{r2}\cdot\eta)]\!]=[\![(b,\{0.5,3.5\})]\!]$. 由于 $f_{tr}(access(q_0),\xi)=--=f_{tr}(access(q_8),\xi)$ 且 $g_{tr}(access(q_0),\xi)=g_{tr}(access(q_8),\xi)=\{\top,\top\}$, 学生用 γ_{r2} 构造一个区域字 $\xi'=[\![vw(\gamma_{r2})]\!]=[\![(b,\{0.5,3.5\})]\!]$, 并寻找合法后继 $e_r\in vs_{\mathcal{A}}(access(q_0),\xi')$. 由于不存在合法后继, 学生构造 $access(q_0)$ 对应于 ξ' 的半合法后继拼接字 $\boldsymbol{\sigma}_r^*=(b,\{0.5,3.5\},\{\top,\top\})$, 再利用 $access(q_0)\cdot\boldsymbol{\sigma}_r^*$ 对树进行扩展, 从而构建出节点 q_{19}. 学生将 *conflict* 由 $\gamma_{r1}\cdot\gamma_{r2}$ 变为 $access(q_{19})=(b,\{0.5,3.5\},\{\top,\top\})$. 然后, 根据新的 *conflict* 更新 tr 和 hy, 得到 $tr=q_{19}$, $hy=q_0$. 由于 $q_{19}\in F$, 学生将 *conflict* 加入 T_Q 中, 并寻找 $noapart(q_{19})$. 由于 q_{19} 与 q_1 和 q_7 均不分离, 学生利用 q_1 和 q_7 的可区分序列 $[\![(a,\{3,3\})]\!]$ 来做进一步判断. 具体方法参见第 2.3 节中使边界节点非孤立且可识别的操作. 然后得到 $noapart(q_{19})=q_1\neq hy=q_0$. 至此, 反例处理结束, 得到 $\mathcal{TR}_2$.

3.3　算法分析

假设 DTA $\mathcal{A}$ 为目标自动机. 用 $n=|L|$ 来表示 $\mathcal{A}$ 中的状态数量, 用 $m=|\Sigma|$ 来表示 $\mathcal{A}$ 中的动作数量, 用 h 来表示学习过程中获得的最长反例的长度, 用 Λ 来表示 $\mathcal{A}$ 中的 region 数量的上限, 有以下结论.

定理 4. 本文提出的基于时间观察树的学习算法可以终止并返回一个识别目标时间语言 $\mathcal{L}$ 的 CTA $\mathcal{H}$.

证明: 首先, 等价查询保证了算法的正确性. 现在证明算法是可终止的. 根据定理 1 可知, 对于两个节点 q_1, $q_2\in Q$, 若在 $\mathcal{A}$ 中分别执行重置时钟字 $access(q_1)$ 和 $access(q_2)$ 后, 到达 $\mathcal{A}$ 中相同的符号化状态, 则 q_1 和 q_2 一定互

不分离. 又由于基础节点集合中的节点两两相互分离. 因此, 基础节点集合中的节点数量最多为符号化状态的数量, 即 $n\cdot\Lambda$. 由于假设自动机 $\mathcal{H}$ 中的位置是由基础节点映射而来, 因此 $\mathcal{H}$ 中的位置数量最多为 $n\cdot\Lambda$.

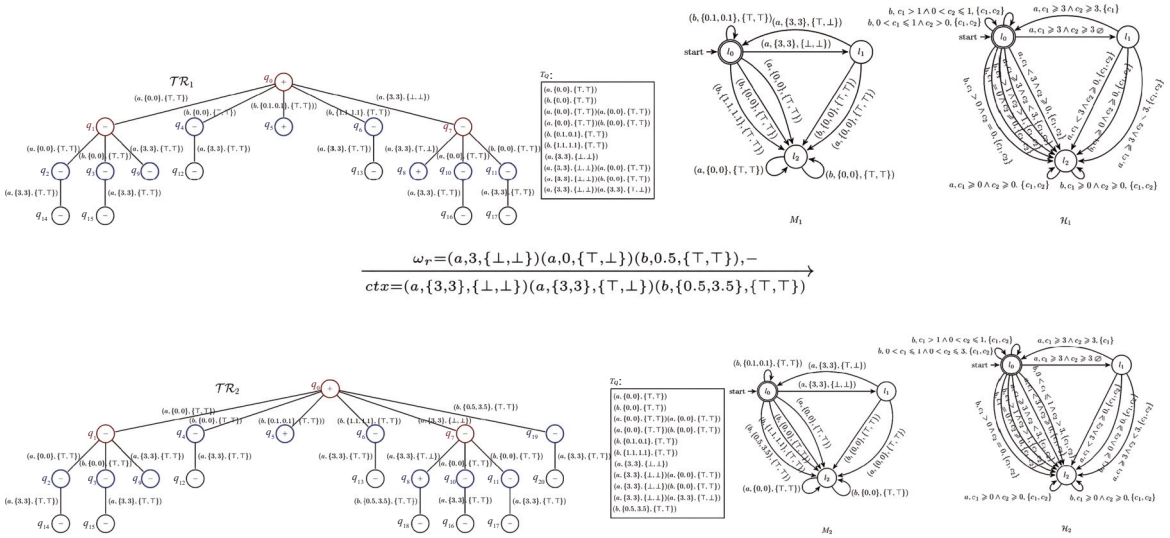


图 5　反例分析示例 3

接下来, 计算迁移的数量. 由定理 2 可知, 对于从一个基础节点 q 出发的任意两条边 $(q,\boldsymbol{\sigma}_{r1},q')$ 和 $(q,\boldsymbol{\sigma}_{r2},q'')$, 若 $[\![vw(\boldsymbol{\sigma}_{r1})]\!]=[\![vw(\boldsymbol{\sigma}_{r2})]\!]$, 则 $resets(\boldsymbol{\sigma}_{r1})=resets(\boldsymbol{\sigma}_{r2})$ 且 $noapart(q')=noapart(q'')$. 因此, 根据由树构建 DFA M 的方法可知, 对于从位置 l_M 出发执行 σ_M 的两条迁移 $(l,(\sigma,\mathbf{v},\mathbf{b}),l')$ 和 $(l,(\sigma,\mathbf{v}',\mathbf{b}'),l'')$, 若 $[\![(\sigma,\mathbf{v})]\!]=[\![(\sigma,\mathbf{v}')]\!]$, 则 $l'=l''$ 且 $\mathbf{b}=\mathbf{b}'$. 再根据将 M 转换为假设自动机 $\mathcal{H}$ 的方法可知, 在 $\mathcal{H}$ 中, 从一个位置出发, 执行一个动作的迁移数量最多为 Λ. 由于 $\mathcal{H}$ 中的位置数量最多为 $n\cdot\Lambda$ 且动作数量为 m, 因此 $\mathcal{H}$ 中的迁移的数量最多为 $m\cdot n\cdot\Lambda^2$. 由于每次对反例进行处理时, 都会至少增加一条 $\mathcal{H}$ 中的迁移 (增加 $\mathcal{H}$ 中的位置时也会增加迁移), 因此算法是可终止的.

定理 5. 基于时间观察树的学习算法的等价查询的最坏复杂度为 $\mathrm{O}(m\cdot n\cdot\Lambda^2)$, 成员查询的最坏复杂度为 $\mathrm{O}(m\cdot n^2\cdot\Lambda^3)$, 重置信息查询的最坏复杂度为 $\mathrm{O}(h\cdot(m\cdot n^2\cdot\Lambda^3))$.

证明: 根据定理 4 的证明可知, $\mathcal{H}$ 中的迁移数量至多为 $m\cdot n\cdot\Lambda^2$. 由于每次对反例进行处理时, 都会至少增加一条 $\mathcal{H}$ 中的迁移, 因此学习过程中产生的反例的数量和等价查询的数量都至多为 $m\cdot n\cdot\Lambda^2$.

下面考虑成员查询的复杂度. 对于每一个反例, 学生首先将其转换为一个重置时钟字 *ctx* 并记 *ctx* 为 *conflict*. 若在树中执行 *conflict* 所到达的节点不属于 $S\cup F$, 则利用二分搜索的方式不断寻找更短的导致冲突的重置时钟字 γ_r, 并更新 *conflict* 为 γ_r. 该过程直至在树中执行 *conflict* 所到达的节点 $tr\in S\cup F$. 在缩短 *conflict* 的过程中, 最多需要进行 $\log_2^{|ctx|}$ 次成员查询. 接下来, 根据新的 *conflict* 修改观察树. 在此过程中, 若 *tr* 从一个边界节点转换为一个基础节点, 则需要对部分边界节点 f 发起一次成员查询以重新计算 $noapart(f)$. 由于最多转换得到 $n\cdot\Lambda$ 个基础节点且 f 的数量至多为 $m\cdot n\cdot\Lambda^2$, 因此此过程至多需要 $m\cdot n^2\cdot\Lambda^3$ 次成员查询. 若在修改观察树时, 只是添加了一条迁移而没有产生新的基础节点, 则由于迁移的数量至多为 $m\cdot n\cdot\Lambda^2$ 且为构建一条迁移所需的成员查询数量至多为 $n\cdot\Lambda$, 该过程所需的成员查询的数量也至多为 $m\cdot n^2\cdot\Lambda^3$.

下面考虑重置信息查询的复杂度. 根据文献 [33] 中的方法可知, 为了寻找一个重置时钟字 γ_r 对于一个区域字 ξ 的合法后继, 至多需要进行 $|\xi|$ 次重置信息查询. 由于每次进行成员查询之前, 都需要寻找一个重置时钟字 γ_r 对于一个可区分序列 ξ 的合法后继. 因此, 在每次进行成员查询之前, 所需的重置信息查询数量取决于可区分序列的长度. 根据第 3.2 节中的描述可知, 可区分序列总是在反例分析过程中被构造, 该过程中产生的可区分序列的长度不会超过反例的最大长度 h. 因此, 重置信息查询的最坏复杂度为 $\mathrm{O}(h\cdot(m\cdot n^2\cdot\Lambda^3))$.

4 学习示例

在本节中, 以图 1 中的 DTA $\mathcal{A}$ 为例, 展示如何利用本文中提出的算法来对它进行学习. 在 DTA $\mathcal{A}$ 中, 初始位置为 l_0, 接收位置也为 l_0, 字母表 $\Sigma=\{a,b\}$. 图 6 展示了学习过程中时间观察树 $\mathcal{TR}$ 的变化. 图 7 展示了 DFA M 和假设 DTA $\mathcal{H}$ 的变化. 首先, 学生通过初始化操作得到 $\mathcal{TR}_1$. 然后, 学生根据 $\mathcal{TR}_1$ 构建 DFA M_1 和假设 $\mathcal{H}_1$, 再发起等价查询从而得到 $\omega_{r1}=(b,0.1,\{\top,\top\})$. 学生利用 $\Gamma(\omega_{r1})$ 对树进行扩展并得到 $ctx_1=(b,\{0.1,0.1\},\{\top,\top\})$. 由于在树中执行 ctx_1 所到达的节点 $q_5\in F$, 学生将 ctx_1 加入 T_Q 中, 然后寻找 $noapart(q_5)$, 得到 $noapart(q_5)=q_0$. 在完成这些操作之后, 得到 $\mathcal{TR}_2$. 学生根据 $\mathcal{TR}_2$ 来构建 DFA M_2 和假设 DTA $\mathcal{H}_2$. 类似地, 学生通过发起等价查询得到 $\omega_{r2}=(b,1.1,\{\top,\top\})$, 进而构造 $\mathcal{TR}_3$.

根据 $\mathcal{TR}_3$, 学生构建 M_3 和 $\mathcal{H}_3$, 然后发起等价查询得到 $\omega_{r3}=(a,3,\{\bot,\bot\})(a,0,\{\top,\bot\})$. 学生利用 $\Gamma(\omega_{r3})$ 对树进行扩展并得到 $ctx_3=(a,\{3,3\},\{\bot,\bot\})(a,\{3,3\},\{\top,\bot\})=conflict$, 再将 $conflict$ 拆分为 $\gamma_{r1}\cdot\gamma_{r2}$. 其中, $\gamma_{r1}=(a,\{3,3\},\{\bot,\bot\})$, $\gamma_{r2}=(a,\{3,3\},\{\top,\bot\})$. 由于时钟字 $vw(\gamma_{r1})$ 在 $\mathcal{H}_3$ 中对应的重置时钟字 $\gamma'_{r1}=(a,\{3,3\},\{\top,\top\})\neq\gamma_{r1}$, 即 $resets(\gamma'_{r1}[1])\neq resets(\gamma_{r1}[1])$, 学生找到了在 $\mathcal{H}_3$ 中执行 $\gamma_{r1}[:(1-1)]$ 所到达的位置 l_0, l_0 在树中所对应的基础节点为 q_0. 然后, 学生构造区域字 $[\![vw(\gamma_{r1}[1])]\!]=[\![(a,\{3,3\})]\!]$, 并找到 $e_r=(a,\{3,3\},\{\bot,\bot\})\in vs_{\mathcal{A}}(access(q_0),[\![(a,\{3,3\})]\!])$. 由于 $resets(e_r)=resets(\gamma_{r1}[1])$, 学生在树中构建从 q_0 出发, 执行 $(a,\{3,3\},\{\bot,\bot\})$ 到达 q_7 的边 (该边已存在于树中), 并将 $access(q_7)$ 加入 T_Q 中, 再计算出 $noapart(q_7)=q_1$, 进而得到了 $\mathcal{TR}_4$.

学生构建 M_4 和 $\mathcal{H}_4$, 然后发起等价查询得到 $ctx_4=(a,\{3,3\},\{\bot,\bot\})(a,\{3,3\},\{\top,\bot\})=conflict$. 可以发现, $ctx_4=ctx_3$. 而此次 ctx_4 导致冲突的原因有所不同. 具体分析过程如下: 由于在树中执行 $conflict$ 所到达的节点 $q_8\notin S\cup F$, 学生将 $conflict$ 拆分为 $\gamma_{r1}\cdot\gamma_{r2}$. 其中, $\gamma_{r1}=(a,\{3,3\},\{\bot,\bot\})$, $\gamma_{r2}=(a,\{3,3\},\{\top,\bot\})$. 由于 γ_{r1} 在 $\mathcal{H}_4$ 中是可执行的, 学生进一步发现: 在树中执行 $conflict$ 所达到的节点 $tr=q_8$, 在 $\mathcal{H}_4$ 中执行 $conflict$ 所到达位置在树中对应的基础节点 $hy=q_1$, 在树中执行 $\gamma_{r1}=(a,\{3,3\},\{\bot,\bot\})$ 所达到的节点 $tr'=q_7$, 在 $\mathcal{H}_4$ 中执行 γ_{r1} 所到达位置对应的基础节点 $hy'=q_1$. 学生根据 γ_{r2} 以及 q_8 和 q_1 的可区分序列 η 构造一个区域字 $\xi=[\![vw(\gamma_{r2}\cdot\eta)]\!]=[\![(a,\{3,3\})]\!]$. 由于 $f_{tr}(access(q_1),\xi)=-\neq f_{tr}(access(q_7),\xi)$ 且 $g_{tr}(access(q_1),\xi)=\{\top,\top\}\neq g_{tr}(access(q_7),\xi)=\{\top,\bot\}$, 可知 ξ 可以对 q_1 和 q_7 进行区分, 即 γ_{r1} 会导致冲突. 学生将 $conflict$ 更新为 $\gamma_{r1}=(a,\{3,3\},\{\bot,\bot\})$. 然后, 根据新的 $conflict$ 更新 tr 和 hy, 得到 $tr=q_7$, $hy=q_1$. 由于 $q_7\in F$, 学生将新的 $conflict$ 加入 T_Q 中, 并寻找 $noapart(q_7)$. 由于 $noapart(q_7)=q_1$, 则利用 $[\![(a,\{3,3\})]\!]$ 来区分 q_1 和 q_7. 具体方法参见第 2.3 节中将可识别的边界节点转换为孤立的操作. 在完成这些操作之后, 得到 $\mathcal{TR}_5$.

根据 $\mathcal{TR}_5$, 学生构建 M_5 和 $\mathcal{H}_5$, 然后发起等价查询进而得到 $ctx_5=(a,\{3,3\},\{\bot,\bot\})(a,\{3,3\},\{\top,\bot\})$. 由于在树中执行 ctx_5 所到达的节点 $q_8\in F$, 学生将 $access(q_8)$ 加入 T_Q 中, 并在基础节点集合中寻找 $noapart(q_8)$, 得到 $noapart(q_8)=q_0$. 在完成这些操作后, 得到 $\mathcal{TR}_6$. 学生根据 $\mathcal{TR}_6$ 构建 DFA M_6 和假设 DTA $\mathcal{H}_6$.

经过又一次等价查询, 学生得到 $ctx_6=(a,\{3,3\},\{\bot,\bot\})(a,\{3,3\},\{\top,\bot\})(b,\{0.5,3.5\},\{\top,\top\})=conflict$, 将 $conflict$ 拆分为 $\gamma_{r1}\cdot\gamma_{r2}$, 其中, $\gamma_{r1}=(a,\{3,3\},\{\bot,\bot\})(a,\{3,3\},\{\top,\bot\})$, $\gamma_{r2}=(b,\{0.5,3.5\},\{\top,\top\})$. 由于 γ_{r1} 在 $\mathcal{H}_6$ 中是可执行的, 学生进一步发现: 在树中执行 $conflict$ 所达到的节点 $tr=q_{18}$, 在 $\mathcal{H}_6$ 中执行 $conflict$ 所到达的位置在树中对应的基础节点 $hy=q_0$, 在树中执行 $\gamma_{r1}=(a,\{3,3\},\{\bot,\bot\})(a,\{3,3\},\{\top,\bot\})$ 所达到的节点 $tr'=q_8$, 在 $\mathcal{H}_6$ 中执行 γ_{r1} 所到达的位置对应的基础节点 $hy'=q_0$. 学生根据 γ_{r2} 以及 q_{18} 和 q_0 的可区分序列 η 构造 $\xi=[\![vw(\gamma_{r2}\cdot\eta)]\!]=[\![(b,\{0.5,3.5\})]\!]$. 由于 $f_{tr}(access(q_0),\xi)=-=f_{tr}(access(q_8),\xi)$ 且 $g_{tr}(access(q_0),\xi)=g_{tr}(access(q_8),\xi)=\{\top,\top\}$, 学生用 γ_{r2} 构造一个区域字 $\xi'=[\![vw(\gamma_{r2})]\!]=[\![(b,\{0.5,3.5\})]\!]$, 并寻找合法后继 $e_r\in vs_{\mathcal{A}}(access(q_0),\xi')$. 由于不存在合法后继, 学生构造 $access(q_0)$ 对应于 ξ' 的半合法后继拼接字 $\boldsymbol{\sigma}_r^*=(b,\{0.5,3.5\},\{\top,\top\})$. 接下来, 利用 $access(q_0)\cdot\boldsymbol{\sigma}_r^*$ 对树进行扩展, 从而构建出节点 q_{19}. 学生将 $conflict$ 由 $\gamma_{r1}\cdot\gamma_{r2}$ 变为 $access(q_{19})=(b,\{0.5,3.5\},\{\top,\top\})$. 然后, 根据新的 $conflict$ 更新 tr 和 hy, 得到 $tr=q_{19}$, $hy=q_0$. 由于 $q_{19}\in F$, 学生将 $conflict$ 加入 T_Q 中, 并寻找 $noapart(q_{19})$. 由于 q_{19} 与 q_1 和 q_7 均不分离, 学生利用 q_1 和 q_7 的可区分序列 $[\![(a,\{3,3\})]\!]$ 来做进一步判断, 得到 $noapart(q_{19})=q_1\neq q_0=hy$. 至此, 反例处理结束, 得到 $\mathcal{TR}_7$, 进而构造 M_7 和 $\mathcal{H}_7$. 学生再一次发起等价查询. 由于等价查询结果为“是”, 学习算法终止, $\mathcal{H}_7$ 即为所学出的时间自动机.

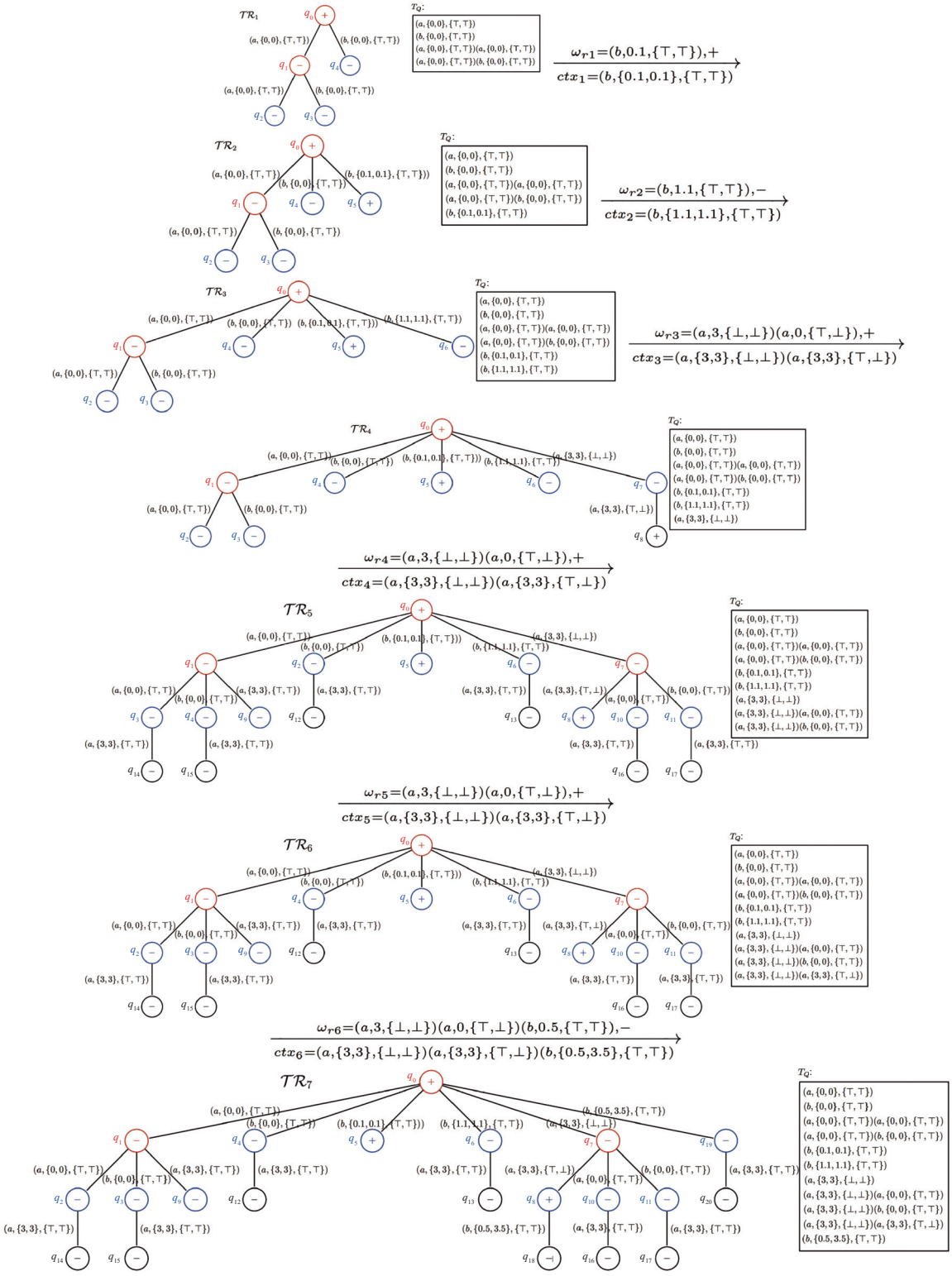
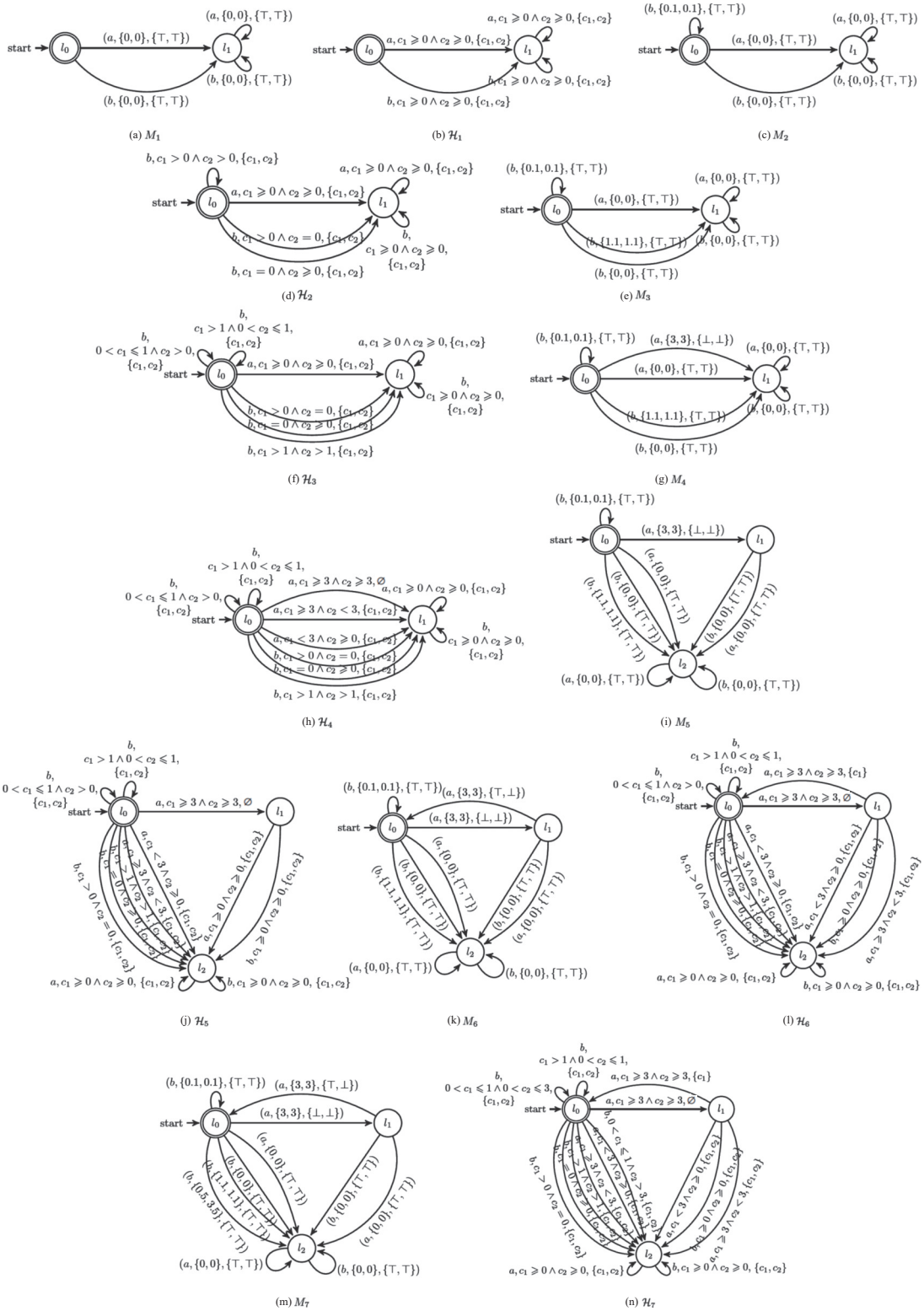


图 6　学习示例产生的时间观察树



(a) M_1 (b) $\mathcal{H}_1$ (c) M_2

(d) $\mathcal{H}_2$ (e) M_3

(f) $\mathcal{H}_3$ (g) M_4

(h) $\mathcal{H}_4$ (i) M_5

(j) $\mathcal{H}_5$ (k) M_6 (l) $\mathcal{H}_6$

(m) M_7 (n) $\mathcal{H}_7$

图 7　学习示例产生的 DFA 和假设 DTA

5 实验与结果分析

首先实现本文提出的改进的 DTA 学习算法 (详见 https://github.com/Tongji-lab/ObservationTree), 并将其与基于多时钟时间分类树的 DTA 学习算法进行比较. 为比较上述两种算法, 首先随机生成一系列包含双时钟的 DTA, 再分别利用两种算法学习这些 DTA, 记录学习过程中的各项参数. 所有实验均在 32 GB 的 i7 处理器上进行.

根据 DTA 中的位置数量、字母表大小以及约束中出现的最大整数的不同, 随机生成 6 组 DTA. 每一组分别包含 10 个 DTA.表 1 展示了实验结果. 表中每一行代表一组 DTA. 其中, $|L|$ 是位置的数量; $|\Sigma|$ 表示字母表的大小; $\kappa(\mathcal{C})$ 是时间约束中的最大常数; DTAL_Observation 表示本文提出的基于观察树的学习算法; DTAL_Classify 表示基于分类树的学习算法; #Membership、#Equivalence 和#Reset 分别表示学习过程中所需的成员查询数量、等价查询数量和重置信息查询数量; n 表示学习出的自动机中的位置数量; time 表示学习一个 DTA 所需的平均时间, 以 s 为单位.

表 1 学习包含 2 个时钟的 DTA 的实验结果

$\|L\|$	$\|\Sigma\|$	$\kappa(\mathcal{C})$	Algorithm	#Membership	#Equivalence	#Reset	n	time (s)
10	3	8	DTAL_Observation	152.2	38.6	222.2	9.1	5.3
			DTAL_Classify	235.8	38.7	356.0	9.2	5.9
12	2	8	DTAL_Observation	130.0	42.3	167.8	10.4	7.7
			DTAL_Classify	212.8	40.2	379.8	10.8	8.5
12	2	16	DTAL_Observation	210.3	53.0	311.4	11.3	19.6
			DTAL_Classify	264.7	51.8	419.8	11.2	22.7
12	3	8	DTAL_Observation	141.7	49.3	174.6	10.7	9.3
			DTAL_Classify	231.7	52.3	411.2	11.2	13.0
14	3	8	DTAL_Observation	222.9	58.6	404.6	12.0	17.5
			DTAL_Classify	347.6	59.8	632.3	12.1	19.6
16	3	8	DTAL_Observation	213.7	57.5	345.6	13.2	19.2
			DTAL_Classify	397.0	64.5	733.9	14.6	28.1

通过分析表 1 可知, 本文提出的方法需要更少的成员查询. 这是因为, 在使用算法 DTAL_Observation 的反例分析技术时, 导致冲突的字最初即为反例, 且通过二分搜索方式缩短字的长度, 因此所需的缩短操作次数为对数级别; 又因为每当构造更短的字时都需要进行一次成员查询, 所以为缩短操作而执行的成员查询数量也为对数级别. 而在使用算法 DTAL_Classify 的反例分析技术时, 为寻找反例中导致冲突的错误位置所进行的成员查询数量为线性级别. 此外, 本文提出的方法所需的重置信息查询的数量也有减少. 一方面是由于成员查询数量减少, 另一方面是由于在使用算法 DTAL_Observation 中的反例分析技术时, 采用了二分搜索技术来分析反例, 使得在新的反例分析方法中, 区域字普遍较短. 因此, 寻找合法后继所需开销减少, 从而减少为进行成员查询所需的重置信息查询. 通过减少学习过程所需的成员查询次数和重置信息查询次数, 学习过程所需的时间也随之缩短. 随着目标 DTA 的状态数量的增加, 两种算法所需的成员查询次数和重置信息查询次数的差异逐渐增大. 两种学习算法所需的等价查询次数相似, 获得的模型的规模也近似. 注意, 由于学习过程中等价查询需要消耗时间, 因此即使本文提出的方法在学习时所需的成员查询次数和重置信息查询次数明显减少, 但整个学习过程中所需的时间缩短幅度有限.

接下来, 随机生成了包含更多时钟的 DTA, 并用 DTAL_Observation 和 DTAL_Classify 分别进行学习. 随着时钟数量的增多, 学习所需等价查询的时间会相应增加. 因此, 相较于表 1 中的目标 DTA, 表 2 中展示了对一些规模较小但包含更多时钟的 DTA 进行学习的实验结果. 与表 1 所得结论类似, 本文提出的方法能够减少学习所需的成员查询次数和重置信息查询次数, 受等价查询所需时间的限制, 本文提出的方法在整个学习过程中所需的时间缩短幅度有限.

表 2 学习包含更多时钟的 DTA 的实验结果

$\|L\|$	$\|\Sigma\|$	$\kappa(\mathcal{C})$	$\|\mathcal{C}\|$	Algorithm	#Membership	#Equivalence	#Reset	n	time (s)
8	2	8	3	DTAL_Observation	17.0	7.0	31.0	10.0	1.1
				DTAL_Classify	67.0	11.0	297.0	12.0	1.2
10	1	8	3	DTAL_Observation	16.0	10.0	16.0	9.0	0.8
				DTAL_Classify	39.0	10.0	102.0	8.0	0.8
10	2	8	3	DTAL_Observation	42.0	10.0	48.0	9.0	0.7
				DTAL_Classify	64.0	10.0	117.0	9.0	0.8
10	3	8	3	DTAL_Observation	33.0	14.0	44.0	11.0	1.4
				DTAL_Classify	91.0	18.0	151.0	13.0	2.5
10	4	8	3	DTAL_Observation	81.0	11.0	103.0	9.0	1.0
				DTAL_Classify	82.0	11.0	124.0	9.0	1.1
10	2	16	3	DTAL_Observation	68.0	12.0	126.0	11.0	0.8
				DTAL_Classify	92.0	12.0	139.0	11.0	0.9
10	2	8	4	DTAL_Observation	33.0	12.0	32.0	9.0	2.4
				DTAL_Classify	51.0	12.0	77.0	9.0	3.2
10	2	8	5	DTAL_Observation	25.0	9.0	77.0	8.0	2.1
				DTAL_Classify	39.0	9.0	84.0	8.0	2.3
12	3	8	3	DTAL_Observation	122.0	20.0	154.0	12.0	1.5
				DTAL_Classify	155.0	20.0	226.0	12.0	1.8

6 总结和展望

本文定义了时间观察树并设计了基于时间观察树的确定性多时钟时间自动机的主动学习算法. 通过相关实验, 说明了该算法能够减少成员查询数量和重置信息查询数量, 进而提升算法效率. 在未来的工作中, 将探讨非确定性多时钟时间自动机的主动学习算法.

References

[1] Vaandrager F. Model learning. Communications of the ACM, 2017, 60(2): 86–95. [doi: 10.1145/2967606]

[2] Gold EM. Complexity of automaton identification from given data. Information and Control, 1978, 37(3): 302–320. [doi: 10.1016/S0019-9958(78)90562-4]

[3] Oliveira AL, Silva JPM. Efficient algorithms for the inference of minimum size DFAs. Machine Learning, 2001, 44(1–2): 93–119. [doi: 10.1023/A:1010828029885]

[4] Avellaneda F, Petrenko A. Learning minimal DFA: Taking inspiration from RPNI to improve SAT approach. In: Proc. of the 17th Int'l Conf. on Software Engineering and Formal Methods. Oslo: Springer, 2019. 243–256. [doi: 10.1007/978-3-030-30446-1_13]

[5] Bohn L, Löding C. Passive learning of deterministic Büchi automata by combinations of DFAs. In: Proc. of the 49th Int'l Colloquium on Automata, Languages, and Programming (ICALP 2022). Paris: Schloss Dagstuhl—Leibniz-Zentrum für Informatik, 2022. 114: 1–114: 20.

[6] Schmidt J, Ghorbani A, Hapfelmeier A, Kramer S. Learning probabilistic real-time automata from multi-attribute event logs. Intelligent Data Analysis, 2013, 17(1): 93–123. [doi: 10.3233/ida-120569]

[7] Cornanguer L, Largouët C, Rozé L, Termier A. TAG: Learning timed automata from logs. In: Proc. of the 36th AAAI Conf. on Artificial Intelligence. AAAI, 2022. 3949–3958. [doi: 10.1609/aaai.v36i4.20311]

[8] Verwer S, de Weerdt M, Witteveen C. One-clock deterministic timed automata are efficiently identifiable in the limit. In: Proc. of the 3rd Int'l Conf. on Language and Automata Theory and Applications. Tarragona: Springer, 2009. 740–751. [doi: 10.1007/978-3-642-00982-2_63]

[9] Verwer S, de Weerdt M, Witteveen C. The efficiency of identifying timed automata and the power of clocks. Information and Computation, 2011, 209(3): 606–625. [doi: 10.1016/j.ic.2010.11.023]

[10] Verwer S, de Weerdt M, Witteveen C. Efficiently identifying deterministic real-time automata from labeled data. Machine Learning, 2012, 86(3): 295–333. [doi: 10.1007/s10994-011-5265-4]

[11] Dierl S, Howar FM, Kauffman S, Kristjansen M, Larsen KG, Lorber F, Mauritz M. Learning symbolic timed models from concrete timed

data. In: Proc. of the 15th Int'l Symp. on NASA Formal Methods. Houston: Springer, 2023. 104–121. [doi: 10.1007/978-3-031-33170-1_7]

[12] Tappler M, Aichernig BK, Larsen KG, Lorber F. Time to learn—Learning timed automata from tests. In: Proc. of the 17th Int'l Conf. on Formal Modeling and Analysis of Timed Systems. Amsterdam: Springer, 2019. 216–235. [doi: 10.1007/978-3-030-29662-9_13]

[13] Tappler M, Aichernig BK, Lorber F. Timed automata learning via SMT solving. In: Proc. of the 14th Int'l Symp. on NASA Formal Methods. Pasadena: Springer, 2022. 489–507. [doi: 10.1007/978-3-031-06773-0_26]

[14] Angluin D. Learning regular sets from queries and counterexamples. Information and Computation, 1987, 75(2): 87–106. [doi: 10.1016/0890-5401(87)90052-6]

[15] Aarts F, Kuppens H, Tretmans J, Vaandrager F, Verwer S. Improving active mealy machine learning for protocol conformance testing. Machine Learning, 2014, 96(1-2): 189–224. [doi: 10.1007/s10994-013-5405-0]

[16] Shahbaz M, Groz R. Inferring mealy machines. In: Proc. of the 2nd World Congress on FM 2009: Formal Methods. Eindhoven: Springer, 2009. 207–222. [doi: 10.1007/978-3-642-05089-3_14]

[17] Vaandrager F, Garhewal B, Rot J, Wißmann T. A new approach for active automata learning based on apartness. In: Proc. of the 28th Int'l Conf. on Tools and Algorithms for the Construction and Analysis of Systems. Munich: Springer, 2022. 223–243. [doi: 10.1007/978-3-030-99524-9_12]

[18] Bollig B, Habermehl P, Kern C, Leucker M. Angluin-style learning of NFA. In: Proc. of the 21st Int'l Joint Conf. on Artificial Intelligence. Pasadena: Morgan Kaufmann Publishers Inc., 2009. 1004–1009.

[19] Howar F, Steffen B, Jonsson B, Cassel S. Inferring canonical register automata. In: Proc. of the 13th Int'l Conf. on Verification, Model Checking, and Abstract Interpretation. Philadelphia: Springer, 2012. 251–266. [doi: 10.1007/978-3-642-27940-9_17]

[20] Cassel S, Howar F, Jonsson B, Steffen B. Active learning for extended finite state machines. Formal Aspects of Computing, 2016, 28(2): 233–263. [doi: 10.1007/s00165-016-0355-5]

[21] Aarts F, Fiterau-Brostean P, Kuppens H, Vaandrager F. Learning register automata with fresh value generation. In: Proc. of the 12th Int'l Colloquium on Theoretical Aspects of Computing. Cali: Springer, 2015. 165–183. [doi: 10.1007/978-3-319-25150-9_11]

[22] Dierl S, Fiterau-Brostean P, Howar F, Jonsson B, Sagonas K, Tåquist F. Scalable tree-based register automata learning. In: Proc. of the 30th Int'l Conf. on Tools and Algorithms for the Construction and Analysis of Systems. Luxembourg City: Springer, 2024. 87–108. [doi: 10.1007/978-3-031-57249-4_5]

[23] Aarts F, Vaandrager F. Learning I/O automata. In: Proc. of the 21th Int'l Conf. on CONCUR 2010—Concurrency Theory. Paris: Springer, 2010. 71–85. [doi: 10.1007/978-3-642-15375-4_6]

[24] Maler O, Mens IE. Learning regular languages over large alphabets. In: Proc. of the 20th Int'l Conf. on Tools and Algorithms for the Construction and Analysis of Systems. Grenoble: Springer, 2014. 485–499. [doi: 10.1007/978-3-642-54862-8_41]

[25] Argyros G, D'Antoni L. The learnability of symbolic automata. In: Proc. of the 30th Int'l Conf. on Computer Aided Verification. Oxford: Springer, 2018. 427–445. [doi: 10.1007/978-3-319-96145-3_23]

[26] Vaandrager F, Bloem R, Ebrahimi M. Learning mealy machines with one timer. In: Proc. of the 15th Int'l Conf. on Language and Automata Theory and Applications. Milan: Springer, 2021. 157–170. [doi: 10.1007/978-3-030-68195-1_13]

[27] Bruyère V, Garhewal B, Pérez GA, Staquet G, Vaandrager FW. Active learning of mealy machines with timers. arXiv: 2403.02019, 2025.

[28] An J, Wang LT, Zhan BH, Zhan NJ, Zhang MM. Learning real-time automata. Science China Information Sciences, 2021, 64(9): 192103. [doi: 10.1007/s11432-019-2767-4]

[29] An J, Zhan BH, Zhan NJ, Zhang MM. Learning nondeterministic real-time automata. ACM Trans. on Embedded Computing Systems, 2021, 20(5s): 99. [doi: 10.1145/3477030]

[30] An J, Chen MS, Zhan BH, Zhan NJ, Zhang MM. Learning one-clock timed automata. In: Proc. of the 26th Int'l Conf. on Tools and Algorithms for the Construction and Analysis of Systems. Dublin: Springer, 2020. 444–462. [doi: 10.1007/978-3-030-45190-5_25]

[31] Mi JR, Zhang MM, An J, Du BW. Learning deterministic one-clock timed automata based on timed classification tree. Ruan Jian Xue Bao/Journal of Software, 2022, 33(8): 2797–2814 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6599.htm [doi: 10.13328/j.cnki.jos.006599]

[32] Waga M. Active learning of deterministic timed automata with myhill-nerode style characterization. In: Proc. of the 35th Int'l Conf. on Computer Aided Verification. Paris: Springer, 2023. 3–26. [doi: 10.1007/978-3-031-37706-8_1]

[33] Teng Y, Zhang MM, An J. Learning deterministic multi-clock timed automata. In: Proc. of the 27th ACM Int'l Conf. on Hybrid Systems: Computation and Control. Hong Kong, China: ACM, 2024. 6. [doi: 10.1145/3641513.3650124]

[34] Teng Y, Chen HY, Mi JR, Zhang MM, An J, Zhan NJ. Active learning of deterministic timed automata via timed classification tree.

Science China Information Sciences, 2025, 68(12): 222101. [doi: 10.1007/s11432-024-4393-8]
[35] Alur R, Dill DL. A theory of timed automata. Theoretical Computer Science, 1994, 126(2): 183–235. [doi: 10.1016/0304-3975(94)90010-8]
[36] Maler O, Pnueli A. On recognizable timed languages. In: Proc. of the 7th Int'l Conf. Foundations of Software Science and Computation Structures. Barcelona: Springer, 2004. 348–362. [doi: 10.1007/978-3-540-24727-2_25]

附中文参考文献

[31] 米钧日, 张苗苗, 安杰, 杜博闻. 运用时间分类树的确定单时钟时间自动机学习. 软件学报, 2022, 33(8): 2797–2814. http://www.jos.org.cn/1000-9825/6599.htm [doi: 10.13328/j.cnki.jos.006599]

作者简介

滕宇, 博士生, 主要研究领域为形式化建模与验证.
张苗苗, 博士, 研究员, 博士生导师, CCF 专业会员, 主要研究领域为模型学习, 智能系统建模和验证.