软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2019,30(7):2033−2051 [doi: 10.13328/j.cnki.jos.005757]

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

有关时间自动机重置的若干问题的计算复杂性*

朱 凯[1,2], 毋国庆[1], 吴理华[2], 袁梦霆[1]

[1](武汉大学 计算机学院,湖北 武汉 430072)

[2](华南农业大学 数学与信息学院,广东 广州 510642)

通讯作者: 袁梦霆, E-mail: ymt@whu.edu.cn

摘 要: 自动机的重置序列也称为同步序列,具有以下特性:有限自动机通过运行重置序列 w,可从任意一个未知的或无法观测到的状态 q_0 到达某个特定状态 q_w.这仅依赖于 w,而与开始运行 w 时的状态 q_0 无关.这一特性可用于部分可观察的复杂系统的自动恢复,而无需重启,甚至有时不能重启.基于此,重置问题自出现以来便得到关注和持续研究.最近几年,它被扩展到可以描述诸如分布式、嵌入式实时系统等复杂系统的无限状态模型上,比如时间自动机和寄存器自动机等.以时间自动机的重置问题的计算复杂性为研究对象,发现重置问题与可达性问题有着紧密的联系.主要贡献是:(1) 利用时间自动机可达性问题的最新成果,完善完全的确定的时间自动机重置问题的计算复杂性结论;(2) 对部分规约的确定的时间自动机,研究得出,即使在输入字母表大小减至 2 的情况下,其复杂性仍是 PSPACE-完全的;特别地,在单时钟情况下是 NLOGSPACE-完全的;(3) 对完全的非确定的时间自动机,研究得出其 D_i-可重置问题(i=1,2,3)是不可判定的,其重置问题与非确定的寄存器自动机重置问题在指数时间可以相互归约,通过证明指数时间归约相对高复杂性类具有封闭性,利用非确定的寄存器自动机的结论得出单时钟的时间自动机的重置问题是 Ackermann-完全的、限界的重置问题是 NEXPTIME-完全的.这些复杂性结论,说明关于时间自动机的重置问题大都是难解的,一方面,为时间系统的可重置性的检测和求解奠定坚实的理论基础,另一方面,为以后寻找具有高效算法的特殊结构的时间系统(即具有高效算法的问题子类)给予理论指导.

关键词: 时间自动机;重置序列;归约;计算复杂性

中图法分类号: TP301

中文引用格式: 朱凯,毋国庆,吴理华,袁梦霆.有关时间自动机重置的若干问题的计算复杂性.软件学报,2019,30(7):2033−2051. http://www.jos.org.cn/1000-9825/5757.htm

英文引用格式: Zhu K, Wu GQ, Wu LH, Yuan MT. Computational complexity of several problems for resetting timed automata. Ruan Jian Xue Bao/Journal of Software, 2019,30(7):2033−2051 (in Chinese). http://www.jos.org.cn/1000-9825/5757.htm

Computational Complexity of Several Problems for Resetting Timed Automata

ZHU Kai[1,2], WU Guo-Qing[1], WU Li-Hua[2], YUAN Meng-Ting[1]

[1](School of Computer Science, Wuhan University, Wuhan 430072, China)

[2](College of Mathematics and Informatics, South China Agricultural University, Guangzhou 510642, China)

Abstract: The reset sequences of finite automata, also known as the synchronizing sequences, have a characteristic: a finite automaton can reach a certain state q_w by running a reset sequence from any unknown or unobservable state q_0. This is dependent only on the reset sequence w itself, not on the state q_0 of the finite automaton at the beginning of running the sequence w. It can be used to restore the

* 基金项目: 国家自然科学基金(61472146, 61640221, 61872272); 广东省自然科学基金(2015A030313413)

Foundation item: National Natural Science Foundation of China (61472146, 61640221, 61872272); Natural Science Foundation of Guangdong Province of China (2015A030313413).

本文由“软件形式化验证”专题特约编辑贺飞副教授、张立军研究员推荐.

收稿时间: 2018-07-15; 修改时间: 2018-10-14; 采用时间: 2018-12-16; jos 在线出版时间: 2019-03-28

CNKI 网络优先出版: 2019-03-29 09:16:16, http://kns.cnki.net/kcms/detail/11.2560.TP.20190329.0915.009.html

running partially observable and complex systems automatically that needs no resetting, and sometimes even that cannot reset. Therefore, the reset problem has been paid attention to since it emerged and has been studied continuously. Recently, it has extended to infinite state models that can describe the complex systems, including distributed and embedded real-time systems, such as timed automata, register automata, etc. In this work, the computational complexity of several problems for the resetting timed automata is studied, and the strong connection between resettability problem and reachability problem for timed automata is found. The main contribution includes: (1) the complexity of the problem for resetting the complete and deterministic timed automata is updated more precisely with the recent achievements in reachability problem for timed automata; (2) the complexity of the problem for resetting the partially specified timed automata is studied. Even if the size of input alphabet is decreased to 2, it is still PSPACE-complete, and in the case of single clock, it is NLOGSPACE-complete; (3) for the complete and nondeterministic timed automata, D_i-resetting problems (i=1,2,3) are all undecidable. The resetting problem for nondeterministic register automata and nondeterministic timed automata can be inter-reduced in exponential time, and the reduction in exponential time is closed for relatively high computational complexity classes. Therefore, it concludes that the problem for resetting it in single clock case is Ackermann-complete, and that bounded version is NEXPTIME-complete from the results on corresponding nondeterministic register automata. These conclusions show that most of resetting problems for timed automata are intractable. On the one hand, they make a solid theoretical foundation for checking and solving the resettability of the timed systems, on the other hand, they guide to seek for some subclasses of real time system which have particular structure and effective algorithms for solving it.

Key words: timed auotmata; reset sequences; reduction; computational complexity

1 引 言

有限自动机的重置(或同步)问题[1],最早由Černý在1964年提出,重置的概念从此逐渐受到关注和不间断的研究,形成了许多成果,吸引了来自计算机、数学、控制和生物领域的研究人员,这些成果已应用在离散事件系统控制、软件测试、生物信息计算[2−4]和机器人[5]等领域.重置有限自动机的关键是构造重置字(或序列):有限自动机通过运行重置字 w,将从任意一个未知的或无法观测到的状态到达某个特定状态 q_w.这仅依赖于 w 自身,而与 w 开始运行时有限自动机所处的状态 q_0 无关.比如,Stojanovic 等人[4]2003 年发表在《Nature Biotechnology》上的研究成果提出了一种称为MAYA的分子自动机,可与人对弈TIC-TAC-TOE游戏.这种自动机经过一轮对弈后需要运行重置字,将自动机带到"新一轮游戏"状态.再比如,Huffman 编码作为主流的数据压缩方法之一,当压缩的文本出现错误时,仅仅一个小错误就会毁掉整个编码串.为确保数据的可靠性,可采用特殊的编码方式,向压缩数据中插入重置字,使得不论出现什么错误,解码器都可以通过运行重置字恢复.于是在一定程度上得到了抗错误的数据压缩方法[6].近年来,关于重置这一经典概念又激起了新的研究兴趣,这得益于重置字已被泛化到变迁系统的博弈和无限状态系统[7]上,对诸如分布式数据网络和嵌入式实时系统这一类的复杂系统建模有着更大的意义.它还引发对时序逻辑的扩展,以规约系统的可重置性.比如Chatterjee等人[8]提出一种新的可判定的逻辑,比经典的计算树逻辑(CTL)有更强的表达能力,为针对可重置性的模型检测方法与工具的实现奠定了理论基础.

在理论计算机科学(形式化理论基础)中,对某一问题,比如某种逻辑的可满足性、某种自动机的可达性等,首先考虑它的可判定性(常转化为对应的判定问题),接着考虑它的计算复杂性.在具体实践中,对不可判定的和计算复杂度高的问题,考虑若干可判定的子类或使计算复杂性得以显著降低的子类,或考虑对应有高效近似算法的问题.这是对复杂性研究的真正意义所在.在形式化验证领域中,要为复杂系统开发自动化的验证工具,所涉及问题的计算复杂性是无法回避的重要研究课题之一.具体到本文关注的自动机的可重置性的研究,主要面临如下 5 类问题.(1) 基本问题:给定自动机 $\mathcal{A}$,判定它是否为可重置的,即判断它是否存在重置序列 w.(2) 限界问题:给定自动机 $\mathcal{A}$ 和正整数 k,判断它是否存在满足$|w|\leqslant k$ 的重置序列 w.(3) 最优问题:给定自动机 $\mathcal{A}$ 和正整数 k,判定它是否存在满足$|w|=k$ 的最短重置序列 w.(4) 阈值问题:给定可重置的自动机 $\mathcal{A}$,给出其重置序列的最短长度的上界.其中,著名的 Černý 猜想[1]断言:对于可以重置的含有 n 个状态的自动机 $\mathcal{A}$,其具有长度不超过 $(n-1)^2$ 的重置字.该猜想至今尚未得到证明(或证伪).(5) 近似问题:针对最优问题(3),在给定的常数因子下,判定是否存在多项式时间算法近似重置序列的最小长度.在证明对应问题是可判定的情况下,给出问题(1)~问题(3)

的计算复杂性,给出问题(4)的阈值和问题(5)的存在性证明.目前,针对这 5 类问题,仅在有限自动机上进行了较为全面和系统的研究,其他类型的自动机上的对应研究较少,甚至没有.

时间自动机模型是由 Alur 等人在 20 世纪 90 年代提出的,它在有限自动机的基础上引入了时钟,可测量和约束系统中的时间因素.它自从诞生以来受到广泛关注,文献[9]迄今成为《Theorectical Computer Science》期刊上引用最多的文章(根据 GoogleScholar 数据引用超过 7 700 次).实际上,时间自动机已成为业界有关时间系统的事实标准.20 多年来,在时间自动机上展开了建模、测试、模型检测等深入的研究和实用化工作[10−17],出现了以 UPPAAL[10,11]为代表的在工业界广泛使用的工具.在此期间,我国的林惠民院士[18,19]、王义教授[11,18,19]和赵建华教授[16,17]分别在时间自动机的公理化、UPPAAL 检测工具的研发和模型检测优化方面取得高水平的成果.但对时间自动机重置问题的研究才刚刚起步,目前仅有的工作是 Doyen 及其博士生 Shirmohammadi 在 2014 年给出的结论[20,21].一般情况下,完全的确定的时间自动机的重置问题是 PSPACE-完全的,而对完全的非确定的时间自动机,它是不可判定的.这个结论回答了上述第(1)类问题中的两个子问题,而时间自动机的其他 4 类问题以及第(1)类问题中的其他情况,如非完全的(即部分规约的)时间自动机的重置问题、限界条件下的重置问题等的答案尚处空缺状态.其实,对时间自动机的第(1)类重置问题的深入讨论,可以通过判断自动机的语法或在语法上增加限制条件得到不同子类,通过研究不同子类问题的复杂性,判断它们是否存在高效算法,比如可以限制时钟的个数、字母表的大小,还可以约定变迁函数是否为完全函数(划分完全的时间自动机和部分归约的时间自动机的标准)和变迁函数是否为单值函数(划分为确定的时间自动机和非确定的时间自动机的标准).

目前,对时间自动机的重置问题的研究只是初现端倪,很多问题尚未提出并解决,本文将完全和部分规约的概念从有限自动机延伸到时间自动机,将最早在有限自动机上讨论的有关可重置性的若干问题转移到时间自动机上,得到该问题在时间自动机上的新的复杂性结论.这些复杂性结论指出了时间自动机的重置问题本身的固有难度,它们大都是难解的.通过归约技术说明了该问题与其他经典问题(如可达性问题)的复杂性之间的联系.这些研究工作一方面为时间系统的可重置性的检测和求解奠定了比较坚实的理论基础,另一方面为将来寻找具有高效算法的特殊结构的时间系统(即具有高效算法的问题子类)给予理论指导.

本文研究工作取得的新结论具体是:

(1) 对于完全的确定的时间自动机:当时钟个数为 1 即单时钟时,它的重置问题是 NLOGSPACE-完全的;当时钟个数大于等于 2 时,它是 PSPACE-完全的.另外,在输入字母表的大小减少至 2 时,问题仍是 PSPACE-完全的.

(2) 对于部分规约的确定的时间自动机:通过不同的证明方法,得到与(1)同样的结论.

(3) 对于非确定的时间自动机:证明了 D_i-可重置问题(i=1,2,3)的不可判定性.此外,找到两个可判定的子类:在单时钟情况下,它是 Ackermann-完全的;当重置序列的长度被限制在 k($k\in\mathbb{N}$)时,它是 NEXPTIME-完全的.

本文第 2 节介绍有关时间自动机及其重置序列的概念.第 3 节讨论完全的确定的时间自动机的重置问题,完善已有结论.第 4 节讨论关于部分规约的确定的时间自动机重置的若干问题的复杂性和它们之间的关系,主要是简单的 CAR-RESET 问题和 2-CAR-RESET 问题.第 5 节讨论完全的非确定的时间自动机重置的若干问题,比如 D_i-重置问题(i=1,2,3)、限界的重置问题和单时钟的重置问题.第 6 节主要从有限自动机和其他自动机的重置问题、时间自动机的可达性两方面讨论相关的研究及其进展.最后给出结论以及下一步的研究方向.

2 基础知识

符号约定.设 $\mathbb{N}$ 、 $\mathbb{Q}$ 、 $\mathbb{R}$ 分别是自然数集、有理数集和实数集,$\mathbb{R}_{\geqslant 0}$ 是非负实数集.对有限集 X,$|X|$和 2^X 分别是它的基数和幂集(即 X 所有子集构成的集合).若$|X|$=1,则称 X 是单元集.$X\backslash Y$ 表示由属于 X 同时不属于 Y 的元素构成的集合,即 X–Y 的差.定义在字母表Γ上的变迁系统是二元组$\langle Q,R\rangle$,其中,Q 是状态集,$R\subseteq Q\times\Gamma\times Q$ 是变迁关系.用 $A\leqslant_P B$(或 $A\leqslant_E B$)表示问题 A 可多项式时间(或指数时间)归约到问题 B.

本文中出现的引理、命题和推论将在附录中加以证明,定理直接在正文中证明.

2.1 时间自动机

时间自动机在有限自动机上扩充了时钟,以便对系统记时.时钟是非负实数的变量,它们的初值为 0,均以相

同速率(时间增长的变化率)增加.设 C 是时钟变量的有限集,在 C 上定义时钟约束 ϕ,其语法如下.

$\phi ::= \text{true} \mid x\#c \mid \phi \wedge \phi$

其中, $c \in \mathbb{N} \cup \{0\}$, $x,y \in C$, $\# \in \{<,\leqslant,=,>,\geqslant\}$.用 $\Phi(C)$ 表示定义在 C 上的所有时钟约束的集合.时钟赋值是函数 $v: C \to \mathbb{R}_{\geqslant 0}$,为每个时钟指派一个实数值,有时用 $\vec{v}$ 表示 $(v_1,v_2,...,v_{|C|}) \in \mathbb{R}_{\geqslant 0}^{|C|}$,有时也用 v 表示向量 $\vec{v}$.如果 $g[v(x)/x]$ 的值为真,那么时钟赋值 v 满足时钟约束 g,记为 $v|=g$,这里,$g[v(x)/x]$表示在 g 中用时钟赋值 $v(x)$的值代替时钟变量 x.对于 $t \in \mathbb{R}_{\geqslant 0}$,时钟流逝 $v+t$ 表示一个新赋值,对所有 $v \in C$, $(v+t)(x)=v(x)+t$.对于集合 $r \subseteq C$,时钟重置 $v[r]$表示对所有的 $x \in r$, $v[r](x)=0$,否则, $v[r](x)=v(x)$的赋值.

定义 1(时间自动机,timed automata). 时间自动机 $\mathcal{A}$ 定义为六元组 $\langle L,l_0,C,\Sigma,E,F\rangle$,其中,

- L 是位置的有限集, $l_0 \in L$ 是初始位置.
- C 是时钟(变量)的有限集.
- Σ是动作的有限集(即字母表).
- $E \subseteq L \times Guard \times \Sigma \times 2^C \times L$ 是变迁(边)集, $Guard \in \Phi(C)$.
- $F \subseteq L$ 是最终(接受)位置集.

有时用 $l \xrightarrow{g,a,r} l'$ 表示变迁 $(l,g,a,r,l') \in E$.时间自动机的状态空间是 $Q=L \times C$,由位置的有限集 L 和时钟集 C 中的各时钟赋值组成.状态 (l,v)有时也称为格局.

注意,本文定义的时间自动机未给出位置上的不变式,实际上可将它归入变迁的卫士;也未给出形如 $x-y\#c$ 的对角线约束,实际上可通过增加时钟和约束来度量这类时钟差.因此,不影响时间自动机的表达能力.

定义 2 (时间自动机的语义,semantics of TA). 时间自动机的语义解释在变迁系统 $\langle Q,R\rangle$上.其中,

- $Q=L \times C$.
- $R \subseteq Q \times \Gamma \times Q$,其中, $\Gamma=\Sigma \cup \mathbb{R}_{\geqslant 0}$.

对于 $((l,v),\gamma,(l',v')) \in R$,如果 $\gamma \in \mathbb{R}_{\geqslant 0}$,则 $l'=l, v'=v+\gamma$,这种变迁称为时延,否则, $\gamma \in \Sigma$,存在变迁 $l \xrightarrow{g,\gamma,r} l' \in E$ 使得 $v|=g$, $v'=v[r]$,这种变迁称为 a-变迁 $(a \in \Sigma)$.

给定 $q=(l,v)$, $\gamma \in \Sigma \cup \mathbb{R}_{\geqslant 0}$, $loc(q)=l$ 是状态 q 对应的位置, $post(q,\gamma)=\{q' \mid (q,\gamma,q') \in R\}$ 是应用 γ后状态 q 的后继.对于 $P \subseteq Q$, $loc(P)=\{loc\{q\} \mid q \in P\}$, $post(P,\gamma)=\bigcup_{q \in P} post(q,\gamma)$. 对于序列,可递归定义 $post(q,\gamma w)=post(post(q,\gamma),w)$.

方便起见,本文对变迁函数 $post$ 的使用并不严格规范,出现 $post((l,v),(t,a))$和 $post((l,v),a)$的用法, $post((l,v),(t,a))=post(post(l,v),t),a)$,有时将 $post((l,v),(t,a))$称为 (t,a)-变迁,其中, $t \in \mathbb{R}_{\geqslant 0}$, $a \in \Sigma$.

在时间自动机中,如果 Σ中的每个字母在某个位置 l 上都有定义,且这些变迁不会离开 l,那么,称 l 为 0 位置(Zero)(或吸收位置).形式化定义为:对于 $l \in L$,如果对 $\forall a \in \Sigma. loc(post(l,a))=l$,那么,称 l 为 0 位置.

定义 3(完全规约的时间自动机,completely specified TA). 对于时间自动机 $\mathcal{A}=\langle L,l_0,C,\Sigma,E,F\rangle$, 如果对于可达状态 (l,v)和所有 $a \in \Sigma$, $post((l,v),a)$有定义且 $post((l,v),a) \neq \varnothing$(此时, a-变迁可行),那么,它是完全规约的,简称完全的,否则,是部分规约的(partially specified TA).

定义 4(确定的时间自动机,deterministic TA). 对于时间自动机 $\mathcal{A}=\langle L,l_0,C,\Sigma,E,F\rangle$, 如果对于可达状态 (l,v)和所有的 $a \in \Sigma$, $|loc(post((l,v),a))| \leqslant 1$,那么,它是确定的,否则,它是非确定的时间自动机(nondeterminsitic TA).

注意,与某些文献不同,本文对完全的时间自动机和部分规约的时间自动机的划分要将字母表和时间延迟当作输入考虑进来.这一点在第 3 节和第 4 节通过构造对应自动机进行归约来证明复杂性时有所体现.

时间自动机的(有限的)运行 ρ是指序列对 $(\mathcal{S}(\rho),\mathcal{E}(\rho))$, 使得状态 $((l_i,v_i))_{0 \leqslant i \leqslant n}$ 的序列 $\mathcal{S}(\rho)$ 和变迁 $(l_{i-1},a_i,g_i,r_i,l_i)$的序列 $\mathcal{E}(\rho)$, 满足如下要求:

- 对每个 $i=1,\ldots,n$,状态对 $((l_{i-1},v_{i-1}),(l_i,v_i))$有边 $(l_{i-1},a_i,g_i,r_i,l_i)$是变迁;
- 运行 ρ的长度等于 n,它的时间段用 $d(\rho)$表示,等于 $\sum_{i=0}^{n-1} d_i$, 其中, $d_i \in \mathbb{R}_{\geqslant 0}$ 是两个连续的动作变迁之间的绝对时间差;

- 运行ρ的轨迹是时间字$(d_0,a_0),\ldots,(d_{n-1},a_{n-1})$.当忽略变迁时,运行可用$((l_i,v_i))_{0\leqslant i\leqslant n}$表示;
- 如果$v_0=\vec{0}$和$s_n\in F$,那么,运行$((l_i,v_i))_{0\leqslant i\leqslant n}$是一个接受运行.

对时间字 w,如果它是$\mathcal{A}$的一条接受的运行轨迹,那么,它被自动机$\mathcal{A}$识别(或接受).时间自动机$\mathcal{A}$识别的语言用$\mathcal{L}(\mathcal{A})$表示.用$|w|$表示其长度.设$i,j\in\{1,\ldots,|w|\}$,字的第 i 个字母用 $w[i]$表示,假设 $i<j$,字 $w[i]w[i+1]\ldots w[j]$用$w[i,j]$表示.如果 n 是正整数,Σ是字母表,则用Σ^n和Σ^*分别表示Σ上所有长度为 n 的字的集合和长度大于等于 0 的字的集合.

时间自动机的可达性(reachability)问题(或时间语言的非空(nonemptiness)问题)一般情况下是 PSPACE-完全的;它的普遍性(universality)问题是不可判定的.具体证明细节见文献[9].

2.2 重置序列

定义 5(重置序列,reset sequences). 对于时间自动机$\mathcal{A}$,若存在某个序列$w\in((\mathbb{R}_{\geqslant 0},\Sigma))^*$,使得不论$\mathcal{A}$处于哪个状态,运行 w 上都会达到某个统一的状态,称 w 为重置(或同步(synchronizing))序列,称$\mathcal{A}$是可重置的(或可同步的).

约定重置序列的形式为由$(d_0,a_0),\ldots,(d_{n-1},a_{n-1})$的时延-动作构成的序列,例如图 1 所示引用文献[20]中的例子展示了一个完全的确定的 1-字母单时钟的时间自动机,它具有一条重置序列是$(3,a)(0,a)(0,a)(1,a)(0,a)(0,a)$.有时,将时延为 0 的变迁省略,并将重复动作变迁写成指数形式,得到 $d(3)\cdot a^3\cdot d(1)\cdot a^3$,其中,$d(m),m\in\mathbb{N}$表示时间延迟.

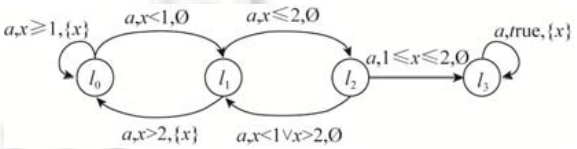

Fig.1 Timed automata $\mathcal{A}$ has a resetting sequence—$(3,a)(0,a)(0,a)(1,a)(0,a)(0,a)$

图 1 时间自动机$\mathcal{A}$含有一个重置序列——$(3,a)(0,a)(0,a)(1,a)(0,a)(0,a)$

时间自动机的重置问题(RESET)可形式化定义为:

问题:RESET

输入:时间自动机$\mathcal{A}$.

询问:$\mathcal{A}$是否是可重置的(即$\mathcal{A}$是否存在一个重置序列 w)?

3 完全的时间自动机的重置问题

本节研究完全的确定的时间自动机的重置问题,考虑时钟个数对复杂性的影响,利用当时钟个数不同时时间自动机的可达性问题的复杂性结论完善 Doyen 等人的结论.

3.1 一般情况

一般情况下的结论是由 Doyen 等人给出的,其复杂性是 PSPACE-完全的,它是这一节的研究基础.具体细节见文献[20,21].主要思想是分两阶段构造重置序列:第 1 阶段利用了完全的确定的时间自动机的结构特点,搜索某个一定存在的序列,将起初的无限状态空间压缩到有限的状态空间.第 2 阶段采用有限自动机的重置序列的求解方法[22],对第 1 阶段得到的状态集,两两成对搜索重置序列以判断它的存在性.这种算法将消耗多项式空间的存储资源,因此属于 PSPACE.对于 PSPACE-难的证明使用从时间自动机的可达性问题到重置问题的归约.

3.2 考虑时钟个数时的复杂性

定理 1. 对于单时钟的完全的确定的时间自动机,它的重置问题是 NLOGSPACE-完全的;对于含 2 个或 2 个以上时钟的完全的确定的时间自动机,它的重置问题是 PSPACE-完全的.

证明思路.根据 Doyen[20]的结论,一般情况的重置问题是 PSPACE-完全的.这里只需要考虑时钟个数的情况.

证明:PSPACE 成员性(即问题属于 PSPACE)的证明与 Doyen 等人的证明相同.PSPACE-难的证明不同于 Doyen 等人的证明,这里,将语言的非空问题归约到重置问题,而 Doyen 等人是从可达性问题归约的.自动机的可达性问题和语言的非空问题本质上等价的,这样归约的目的是为了与第 4 节定理 2 的证明中的归约保持一致.

从 $\mathcal{A}$ 可以构造一个完全的确定的时间自动机 $\mathcal{A}'$,将 $\mathcal{L}(\mathcal{A})$ 的非空问题归约到重置问题.

给定 $\mathcal{A}=\langle L,l_i,C,\Sigma,E,F\rangle$, 可以在多项式时间内构造 $\mathcal{A}'=\langle L',l_0',C',\Sigma',E',F'\rangle$, 其构造过程如下.

- $L'=L\cup\{l_0,l_f\},l_0'=l_0,F'=\{l_f\}$;
- $C'=C$;
- $\Sigma'=\Sigma\cup\{\alpha\}$;
- 对于表示变迁函数定义的边集 E',按以下次序构造,如图 2 所示.

(a) $E'=E$;

(b) $E'=E'-\{(l_p,g,a,r,l)|l_p\in F,a\in\Sigma\}$, 即删除 F 中的每个位置上的出边;

(c) $E'=E'\cup\{(l_p,\text{true},\Sigma,C,l_f)|l_p\in F\}$, 即为 F 中的每个位置增加到 l_f 的变迁,图 2 中用绿色标注的变迁;

(d) $E'=E'\cup\{(l_f,\text{true},\Sigma\cup\{\alpha\},C,l_f)\}$, 即为 l_f 增加到自身的环,图 2 中用蓝色标注的变迁,此时,l_f 成为 $\mathcal{A}'$ 的零位置;

(e) $E'=E'\cup\{(l_0,\text{true},\Sigma\cup\{\alpha\},C,l_i)\}$, 即为 l_0 增加到 l_i 的变迁,图 2 中用蓝色标注的变迁;

(f) $E'=E'\cup\{(l,\text{true},\alpha,C',l_0')|l\in L'\setminus\{l_0,l_f\}\}$, 即为 $\mathcal{A}'$ 中除 l_0 和 l_f 之外的位置增加到 l_0 的变迁,图 2 中用红色标注的变迁.

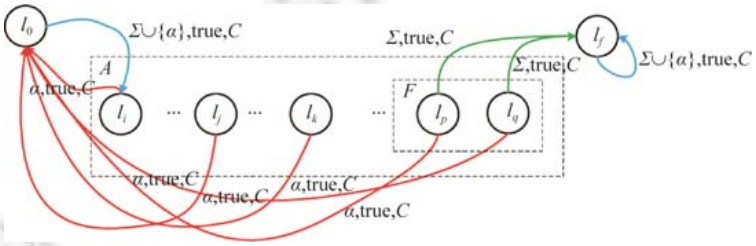

Fig.2 Reduction from NONEMPTINESS of $\mathcal{L}(\mathcal{A})$ to RESET of $\mathcal{A}'$

图 2 从语言 $\mathcal{L}(\mathcal{A})$ 的非空问题归约到 $\mathcal{A}'$ 的重置问题

从图 2 所示的构造方案容易看出: $w\in((\mathbb{R}_{\geqslant 0},\Sigma))^*$ 是 l_p 的接受字,即 $w\in\mathcal{L}(\mathcal{A})$ 当且仅当 $u=(t_1,\alpha)\cdot w\cdot(t_2,c)$ 是 $\mathcal{A}'$ 的重置序列,其中, $t_1,t_2\in\mathbb{R}_{\geqslant 0}$, $c\in\Sigma$.

接着考虑时钟因素:由于 Laroussinie 等人证明了单时钟和双时钟的时间自动机的可达性问题分别是 NLOGSPACE-完全的和 NP-难的(细节见文献[23]中的命题 5.1 和命题 6.1),Fearnley 等人进一步确定了双时钟的自动机的可达性问题是 PSPACE-完全的(细节见文献[24]末尾部分的推论 12).Courcoubetis 等人证明了含 3 个时钟的时间自动机的可达性是 PSPACE-完全的(细节见文献[25]中的定理 2).对含 $k(k>3)$个时钟的时间自动机,Haase 等人[26,27]证明了它与受限的 2-计数器自动机是对数空间内相互归约的,利用受限的 2-计数器自动机的可达性问题是 PSPACE-完全的结论证明了它是 PSPACE-完全的. □

4 部分规约的时间自动机的重置问题

本节研究部分规约的确定的时间自动机在一般情况下和输入字母表大小为 2 时的计算复杂性.将 Martyugin[28,29]在部分规约的有限自动机上的研究扩展到时间自动机,证明的技术路线与其大致相同,由于加入了时钟,因此情况要复杂得多.为方便起见,本节若无特殊说明,均将省略“确定的”字样.

4.1 一般情况

比较常见的是部分归约的时间自动机.对于这类时间自动机,如果存在序列 $w\in((\mathbb{R}_{\geqslant 0},\Sigma))^*$, 使得 $post(L\times C,$

w)有定义且|$post(L\times C,w)$|=1|,那么,称 w 是仔细重置序列(carefully reset sequence),称该时间自动机是可仔细重置的.仔细重置序列没有使用自动机上未定义的变迁(区别有定义但不可行的变迁),是一般重置序列的泛化.

仔细重置问题(CARE-RESET),即部分规约的时间自动机的重置问题,可形式化为:

问题:CARE-RESET(ZERO CARE-RESET)

输入:(含一个零位置的)部分规约的时间自动机 $\mathcal{A}$;

询问: $\mathcal{A}$ 是否是可仔细重置的($\mathcal{A}$ 是否存在一个仔细重置序列)?

定理 2. ZERO CARE-RESET 问题是 PSPACE-完全的.

证明思路.首先证明有非确定的多项式空间的算法判断重置序列的存在性,即证明 PSPACE 的成员性(利用萨维奇定理 NPSPACE=PSPACE 的结论).然后证明是 PSPACE-难的,从 $k(k\in\mathbb{N},k\geqslant 2)$ 个时间自动机交问题归约.

证明:Doyen[20]通过举出反例证实:由于区域图抽象掉了具体时间信息,它的重置序列不一定是原自动机的重置序列.若能在自动机的区域图[9]上找到一条长度不超过区域大小的 3 次方的序列作为候选重置序列(因为可重置的完全的有限自动机的重置序列阈值是$(n^3-n)/6$,其中,n 是区域图的状态个数,可参考文献[28],而可重置的部分规约的有限自动机的重置序列也满足该阈值),则自动机有可能是可重置的,否则,一定是不可重置的.猜测时按需展开(on-the-fly)产生区域图,每次猜一个字母,总次数不超过阈值,如果存在候选序列,就接着对候选序列编码形成线性的实数算术公式,再输入 SMT 求解器检查:(1) 任意两个区域状态;(2) 区域中任意两个状态执行候选序列后时钟赋值(自最后一次重置以来)是否相等(即同步).因为猜测候选重置序列这一步需要多项式空间的存储资源,采用 SMT 求解属于 PSPACE.那么,这个"猜测-检验"算法属于 PSPACE,即算法需要多项式空间的存储资源.

以下给出 PSPACE-难的证明.

对给定的 $k(k\in\mathbb{N},k\geqslant 2)$ 个时间自动机交问题(FINITE TAs INTERSECTION)的一个实例: $\mathcal{A}_1=\langle L_1,l_1,C_1,\Sigma,E_1,F_1\rangle,...,\mathcal{A}_k=\langle L_k,l_k,C_k,\Sigma,E_k,F_k\rangle$, 可以在多项式时间内构造一个包含零位置的部分规约的时间自动机 $\mathcal{B}=\langle L_B,l_B,C_B,\Sigma_B,E_B,F_B\rangle$, 使得 $\mathcal{B}$ 存在一个仔细重置序列当且仅当 $\bigcap_{i=1}^{k}\mathcal{L}(\mathcal{A}_i)\neq\varnothing$.

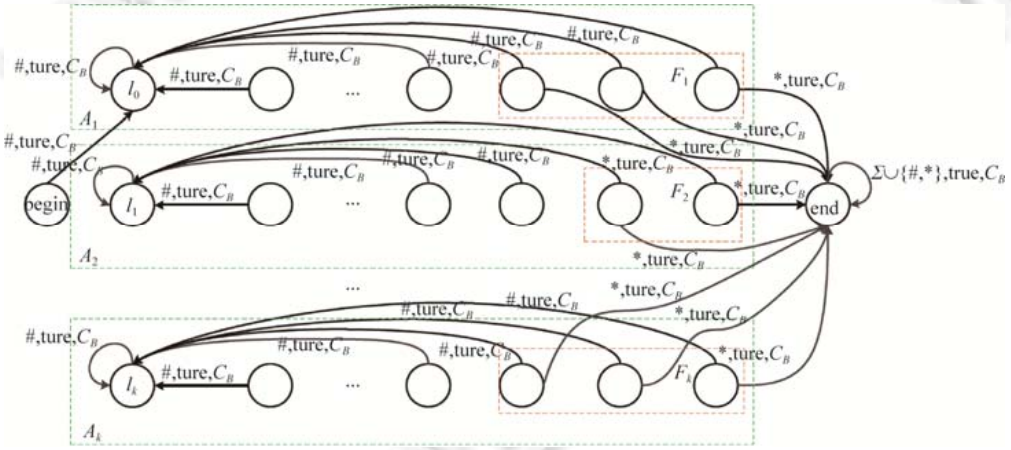

Fig.3 Reduction from NONEMPTINESS of $\mathcal{L}\left(\bigcap_{i=1}^{k}\mathcal{A}_i\right)$ to CARE-RESET of $\mathcal{B}$

图 3 从语言 $\mathcal{L}\left(\bigcap_{i=1}^{k}\mathcal{A}_i\right)$ 的非空问题归约到 $\mathcal{B}$ 的仔细重置问题

$\mathcal{B}$ 的构造过程如下,如图 3 所示.图中绿色和红色的虚线方框分别标识自动机 $\mathcal{A}_i$ 和它的最终位置集 F_i.

• $L_B=\bigcup_{i=1}^{k}L_i\cup\{\text{begin},\text{end}\}$, l_B=begin,F_B={end};

• $C_B=\bigcup_{i=1}^{k}C_i$, 其中,对于 $i\neq j,i,j\in\{1,...,k\}$,$C_i\cap C_j\neq\varnothing$;

• $\Sigma_B=\Sigma\cup\{\#,*\}$;

•对于 $\mathcal{B}$ 的变迁函数,可按以下次序构造 E_B.

(a) $E_B=\bigcup_{i=1}^{k}E_{i=1}$, 保留 k 个时间自动机的原有变迁;

(b) $E_B=E_B\cup\{(l,\text{true},\#,C_B,l_i)|l\in L_i,i\in\{1,...,k\}\}$, 即对于 $\mathcal{A}_i$ 中每个位置,增加到它的初始位置 l_i 的一条#-变迁,卫士为 true 且重置所有时钟;

(c) $E_B=E_B\cup\{(l,\text{true},*,C_B,\text{end})|l\in F_i,i\in\{1,...,k\}\}$, 即对于 $\mathcal{A}_i$ 中的每个终止位置,增加到 end 位置的一条*-变迁,卫士为 true 且重置所有时钟,显然,*-变迁在 end 位置上是没有定义的;

(d) $E_B=E_B\cup\{(\text{end},\text{true},\Sigma\cup\{\#,*\},C_B,\text{end})\}$, 即增加 end 到其自身的变迁,用字母表中的所有字母标记,卫士为 true 且重置所有时钟,显然, $\mathcal{B}$ 具有零位置 end;

(e) $E_B=E_B\cup\{(\text{begin},\text{true},C_B,l_1)\}$.

设 $w\in((\mathbb{R}_{\geqslant 0},\Sigma))^*$ 满足 $w\in\mathcal{L}\left(\bigcap_{i=1}^{k}\mathcal{A}_i\right)$, 则 $(t_1,\#)\cdot w\cdot(t_2,*)$ ($t_1,t_2\in\mathbb{R}_{\geqslant 0}$, 方便起见,可令 t_1=t_2=0)是 $\mathcal{B}$ 的仔细重置序列.由构造 E_B 过程可得,在 $\mathcal{B}$ 的所有位置上,#-变迁都有定义,其卫士条件为 true 且重置 $\mathcal{B}$ 的所有时钟,因此, $post(L_B\times C_B,(0,\#))=\{(l_1,\vec{0}),...,(l_k,\vec{0}),(\text{end},\vec{0})\}$, 显然,经过#-变迁后,状态集削减为大小为|$k$+1|的有限集.因为 $loc(post((l_1,\vec{0}),w))\in F_1,...,loc(post((l_k,\vec{0}),w))\in F_k$, 所以,在 $\mathcal{B}$ 上运行序列 w 将到达状态集 $\{(l_1,\vec{0}),...,(l_k,\vec{0})\}\cup\{(\text{end},\vec{0})\}$, 故 $post(L_B\times C_B,(0,\#)\cdot w)=post(\{(l_1,\vec{0}),...,(l_k,\vec{0}),(\text{end},\vec{0})\},w)\subseteq\bigcup_{i=1}^{k}(F_i\times C_B)\cup\{\text{end},\vec{0}\}$. 于是,再经过序列(0,*)(即不停留直接运行*-变迁)后,得到 $post(L_B\times C_B,(0,\#)\cdot w\cdot(0,*))$={(end,$\vec{0}$)}. 因此, $(t_1,\#)\cdot w\cdot(t_2,*)$ 是 $\mathcal{B}$ 的一个仔细重置序列,其中 $t_1,t_2\in\mathbb{R}_{\geqslant 0}$.

引理 1. 对于部分归约的时间自动机 $\mathcal{B}$,如果它的最短的仔细重置序列 $u\in((\mathbb{R}_{\geqslant 0},\Sigma))^*$ 存在,那么,u 具有 $u=(t_1,\#)\cdot w\cdot(t_2,*)$ 形式,其中, $w\in((\mathbb{R}_{\geqslant 0},\Sigma))^*$,$t_1,t_2\in\mathbb{R}_{\geqslant 0}$.

设 $u\in((\mathbb{R}_{\geqslant 0},\Sigma))^*$ 是 $\mathcal{B}$ 的最短仔细重置序列.由引理 1 可得,对于某个 $w\in((\mathbb{R}_{\geqslant 0},\Sigma))^*$,$u=(t_1,\#)\cdot w\cdot(t_2,*)$, 其中, $t_1,t_2\in\mathbb{R}_{\geqslant 0}$. 简单起见,可令 t_1=t_2=0. 由于 $post(L_B\times C_B,[(0,\#)\cdot w]\cdot(0,*))$ 是有定义的且到达 {end,$\vec{0}$}. 因此, $post(L_B\times C_B,(0,\#)\cdot w)\subseteq(\{F_1\cup...\cup F_k\}\times C_B\cup\{(\text{end},\vec{0})\})$. 那么, $loc(post((l_1,\vec{0}),w))\in F_1,...,loc(post((l_k,\vec{0}),w))\in F_k$, 于是, $w\in\bigcap_{i=1}^{k}\mathcal{L}(\mathcal{A}_i)$, 那么, $\bigcap_{i=1}^{k}\mathcal{L}(\mathcal{A}_i)\neq\varnothing$.

从构造过程可以得到, $|L_B|=\sum_{i=1}^{k}|L_i|+2$ 和 $|E_B|=\sum_{i=1}^{k}|E_i|+\sum_{i=1}^{k}|L_i|+\sum_{i=1}^{k}|F_i|+2$ (对于标记多个字母的变迁只计算 1 次), $\mathcal{B}$ 中的非平凡卫士(不是 true 的卫士)与 $\mathcal{A}_1,...,\mathcal{A}_k$ 总共的非平凡卫士的个数是相等的.于是 FINITE TAs INTERSECTION$\leqslant_p$ZERO CAR-RESET.

命题 1($k(k\in\mathbb{N},k\geqslant 2)$ 个时间自动机的交). 对于 k 个时间自动机 $\mathcal{A}_1\ldots\mathcal{A}_k$ 是否接受共同的时间字的问题,即判定 $\mathcal{L}\left(\bigcap_{i=1}^{k}\mathcal{A}_i\right)\neq\varnothing$ 是 PSPACE-完全的.

由命题 1(k($k\geqslant 2$)个时间自动机交),可得 ZERO CARE-RESET 是 PSPACE-完全的. □

如果采用定理 2 证明中的构造方法,从若干个时间自动机构造出 $\mathcal{B}$,则称 $\mathcal{B}$ 是简单的.

说明:对于完全的确定的时间自动机,构造的新的时间自动机也是完全的确定的,在定理 1 的证明中使用的构造是对 F 中的每个状态删除了它的所有出边,并增加分别指向 l_0 和 l_f 的边,同时将 l_f 改造为零位置,而对于部分规约的自动机,构造思路大致相同但要更加复杂,都增加了零位置.对于完全时间自动机,Doyen 等人给出一种

较暴力的算法[20],对于部分规约的时间自动机,本文没有给出相应的算法细节是基于两点原因:首先,时间自动机的状态空间是无穷的,而基于时间互模拟等价的区域自动机[9]来计算序列是不可靠的,需要进一步的检验.其次,由于变迁函数不是完全函数,所以 Doyen 算法第 1 阶段的构造在部分规约的时间自动机上可能无法实现.但这并不影响问题具有 PSPACE 成员性的结论以及对问题复杂性的讨论.

至于可重置的部分规约的有限自动机的重置序列可以复用完全的有限自动机的重置序列阈值,原因在于,前者的可重置序列一定是对应的扩展(结构(状态和变迁)保持不变,只饱和化变迁上的字母表和时钟卫士)后的完全的有限自动机的重置序列,显然满足其阈值.

选取的简单的 CAR-RESET 问题实际上是 CAR-RESET 问题的子类,复杂性是 PSPACE-完全的,而且 ZERO CAR-RESET$\leqslant_p$ CAR-RESET,那么,对于 CAR-RESET 问题仍是 PSPACE-完全的.

4.2 考虑输入字母表大小时的复杂性

有时还会将问题限制在最多有 $k(k\in\mathbb{N})$ 个输入字母的自动机上,这类问题的命名一般采用前缀 k-的方式,比如 2-CARE-RESET 是指将 CARE-RESET 问题限制到最多含有 2 个输入字母的时间自动机上.

用简单的 CAR-RESET 表示对问题 CAR-RESET 的限制,将在以下定理的证明中加以使用.

定理 3. 2-ZERO CAR-RESET 问题是 PSPACE-完全的.

证明思路.问题的 PSPACE 成员性证明与定理 2 相同.关键是证明它是 PSPACE-难的,可从 ZERO CAR-RESET 问题归约到 2-ZERO CAR-RESET 问题.

证明:这个问题的 PSPACE 成员性的证明与定理 2 相同.以下将简单的 ZERO CAR-RESET 问题归约到 2-ZERO CAR-RESET 问题来证明它是 PSPACE-难的.

设 $\mathcal{B}=\langle L_B,l_1^B,C_B,\Sigma_B,E_B,F_B\rangle$ 是一个简单的部分规约的时间自动机, $L_B=\{l_1^B,...,l_n^B\}$ 和 $\Sigma_B=\{c_1,...,c_m,\#,*\}$. 由定理 2 可知,对 $w\in((\mathbb{R}_{\geqslant 0},\{c_1,...,c_m\}))^*$ 满足 $w\in\mathcal{L}\left(\bigcap_{i=1}^{k}\mathcal{A}_i\right)$, $\mathcal{B}$ 的最短仔细重置序列 $u=(t_1,\#)\cdot w\cdot(t_2,*)$, $t_1,t_2\in\mathbb{R}_{\geqslant 0}$, 并且存在 begin,end$\in L_B$ 使得 $post(L_B\times C_B,(t_1,\#)\cdot w\cdot(t_2,*))=(\text{end},\vec{0})$, 其中,#是在 begin 位置上有动作变迁定义的唯一字母.

构造部分规约的时间自动机 $\mathcal{C}=\langle L_C,l_C,C_C,\Sigma_C,E_C,F_C\rangle$, 含有零位置 z,使得 $\mathcal{C}$ 存在仔细重置序列当且仅当 $\mathcal{B}$ 存在对应的仔细重置序列.

$\mathcal{C}$ 的构造过程如下.

• $L_C=\{l_{k,i}^C\mid k\in\{0,...,m+1\},i\in\{1,...,n-1\},\}\cup\{z\},l_c=l_{0,1}^C$, $F_C=\{z\}$. 其中, $\mathcal{C}$ 的位置个数满足 $|L_C|=(|L_B|-1)\cdot||\Sigma_B|+1=(n-1)\cdot(m+2)+1$;

• $C_C=C_B$;

• $\Sigma_C=\{a,b\}$;

• 对于自动机 $\mathcal{B}$,设 $\text{begin}=l_1^B,\text{end}=l_n^B$, $c_0=*,c_{m+1}=\#$,对于自动机 $\mathcal{C}$, $l_{0,n}^C=...=l_{m+1,n}^C=z$, 变迁函数可以通过构造变迁集 E_C 来定义.

(a) $E_C=E_C\cup\{(l_{k,i}^C,a,\text{true},\varnothing,l_{k+1,i}^C)\mid k\in\{0,...,m\},i\in\{1,...,n-1\}\}$;

(b) $E_C=E_C\cup\{(l_{k,i}^C,a,\text{true},\varnothing,l_{k,i}^C)\mid k=m+1,i\in\{1,...,n-1\}\}$;

(c) 对于 $\forall e=(l_j^B,c_i,g,r,l_k^B)\in E_B$, 其中, $c_i\in\Sigma_B,j,k\in\{1,\ldots,n\}$, $E_C=E_C\cup\{(l_{i,j}^C,b,g,r,l_{0,k}^C)\mid(l_j^B,c_i,g,r,l_k^B)\in E_B\}$;

(d) $E_C=E_C\cup\{(z,\{a,b\},\text{true},C_C,z)\}$, 显然,$z$ 是零位置.

图 4 通过一个简单的例子展示了上述构造方法.首先将给定的时间自动机 $\mathcal{A}_1$ 和 $\mathcal{A}_2$ 转换为简单的部分归约的时间自动机 $\mathcal{B}$, 然后将 $\mathcal{B}$ 转换为一个 2-字母的部分规约的时间自动机 $\mathcal{C}$. $\mathcal{B}$ 中用红色标注的变迁是构造时增加的,它们在 $\mathcal{C}$ 中也对应标注为红色,变迁上的字母变为 b; $\mathcal{B}$ 中用黑色标注的变迁源自 $\mathcal{A}_1$ 和 $\mathcal{A}_2$ 中的变迁,它们在 $\mathcal{C}$ 中保持黑色不变,标注的字母变为 b; $\mathcal{C}$ 中蓝色标注的变迁对应步骤(a)和步骤(b)的构造,标注的字母是 a.另外,$c_0=b,c_1=ab,c_2=a^2b,c_3=a^3b$.

设 $k\in\{0,\ldots,m+1\}$ 和 $i\in\{1,\ldots,n\}$,$Row_k=\{l_{k,1},\ldots,l_{k,n}\}$ 是 L_C 的第 k 行,$Col_i=\{l_{0,i},\ldots,l_{m+1,i}\}$ 是集合 L_C 的第 i 列(在对自动机 C 的引用是清楚的情况下,可省略位置 $l_{i,j}^C$ 的上标),其中,第 n 列退化为 z,显然,z 属于每一行,即 $\forall k\in\{0,\ldots,m+1\}$ 有 $l_{k,n}=z$.

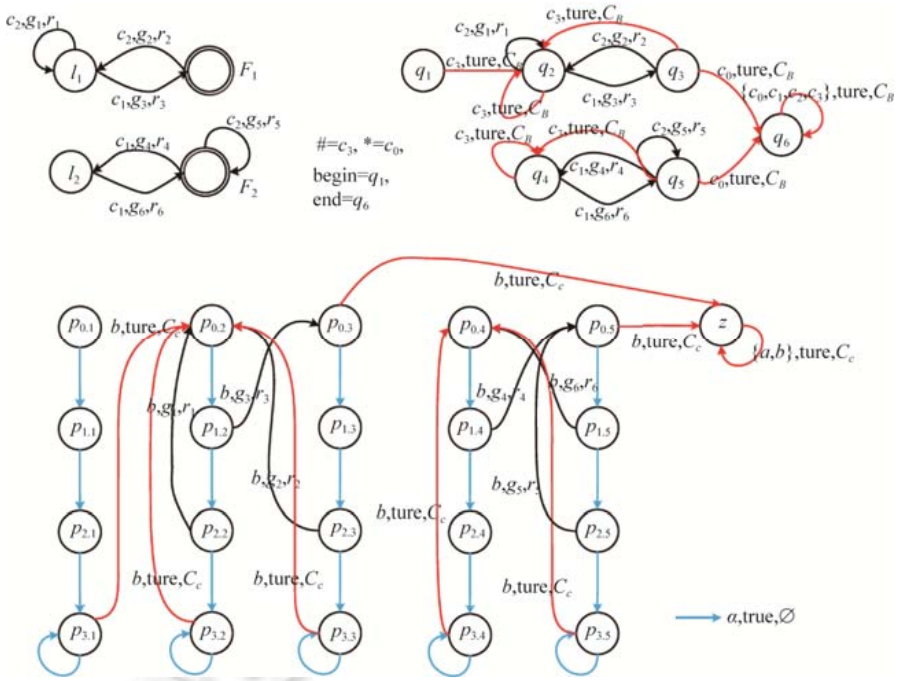

Fig.4 An example of timed automata $\mathcal{A}_1$, $\mathcal{A}_2$, $\mathcal{B}$ and $\mathcal{C}$

图 4 时间自动机 $\mathcal{A}_1$、$\mathcal{A}_2$、$\mathcal{B}$ 和 $\mathcal{C}$ 的例子

引理 2.

(1) 对于 $i\in\{1,\ldots,n\}$,$loc(post(Col_i\times C_C,a))\subseteq Col_i$;

(2) 对于 $k\in\{0,\ldots,m\}$,$loc(post(Row_k\times C_C,a))\subseteq Row_{k+1}$,特别地,$loc(post(Row_{m+1}\times C_C,a))\subseteq Row_{m+1}$;

(3) $loc(post(L_C\times C_C,b))\subseteq Row_0$.

引理 3. 如果 $u\in((\mathbb{R}_{\geqslant 0},\{a,b\}))^*\setminus((\mathbb{R}_{\geqslant 0},\{a\}))^*$ 且 $post(L_C\times C_C,u)$有定义,那么,

(1) 对每个 $i\in\{1,\ldots,n\}$,$|Col_i\cap loc(post(L_C\times C_C,u))|\leqslant 1$;

(2) $|loc(post(L_C\times C_C,u\cdot(t,a)))|=|loc(post(L_C\times C_C,u))|$, 其中, $t\in\mathbb{R}_{\geqslant 0}$;

(3) $loc(post(L_C\times C_C,u\cdot(t,b)))\subseteq Row_0$, 其中, $t\in\mathbb{R}_{\geqslant 0}$.

定义映射 $f:(\mathbb{R}_{\geqslant 0},\{a,b\}^*b)\to(\mathbb{R}_{\geqslant 0},\Sigma)$, 其中, $\Sigma=\{c_0,\ldots,c_{m+1}\}$:

$$f(t,a^kb)=\begin{cases}(t,c_k), & k\in\{1,\ldots,m+1\},t\in\mathbb{R}_{\geqslant 0}\\(t,c_{m+1}), & k\geqslant m+2,t\in\mathbb{R}_{\geqslant 0}\end{cases}\tag{1}$$

由于 $\mathcal{C}$ 上每个字 $u\in((\mathbb{R}_{\geqslant 0},\{a,b\}^*b))^*$, 可被表示为多个形如$(t,a^kb)$的字的积(即连接操作),$k$=0,1,2,…,所以映射 f 的复合映射 g(即 $g=f\cdot\ldots f$,g 可看成是应用 f 后的序列的连接)可以扩展到集合 $((\mathbb{R}_{\geqslant 0},\{a,b\}^*b))^*$. $g:((\mathbb{R}_{\geqslant 0},\{a,b\}^*b))^*\to((\mathbb{R}_{\geqslant 0},\Sigma))^*$.

考虑 f 对应的逆映射 $f^{-1}:(\mathbb{R}_{\geqslant 0},\Sigma)\to(\mathbb{R}_{\geqslant 0},\{a,b\}^*b)$, 定义为

$$f^{-1}(t,c_k)=\begin{cases}(t,a^kb), & k\in\{0,\ldots,m+1\},t\in\mathbb{R}_{\geqslant 0}\\(t,a^{m+1}b), & k\geqslant m+2,t\in\mathbb{R}_{\geqslant 0}\end{cases}\tag{2}$$

注意,在式(1)和式(2)中,如果 $k\leqslant m+1$,那么(t,a^kb)表示 $(t_1,a)\cdot(t_2,a)\cdot\ldots\cdot(t_k,a)\cdot(t_{k+1},b)$, 满足 $t=\sum_{i=1}^{k+1}t_i$, 否则,(t,a^kb)表示 $(t_1,a)\cdot(t_2,a)\cdot\ldots\cdot(t_k,a)\cdot(t_{m+1},b)$, 满足 $t=\sum_{i=1}^{k}t_i+t_{m+1}$. 为简单起见,可令 $t_1=t_2=\ldots=t_k=0$ 且 $t_{k+1}=t$(或 $t_{m+1}=t$). 对于任意 $u\in\Sigma^*$, $f(f^{-1}(u))=u.g^{-1}:((\mathbb{R}_{\geqslant 0},\Sigma))^*\to((\mathbb{R}_{\geqslant 0},\{a,b\}^*b))^*$.

以下证明 $w\in((\mathbb{R}_{\geqslant 0},\Sigma))^*$ 是 $\mathcal{B}$ 的一个仔细重置序列,当且仅当 $g^{-1}(w)$是 $\mathcal{C}$ 的一个仔细重置序列.

假设 $u\in((\mathbb{R}_{\geqslant 0},\{a,b\}^*b))^*$ 且 $post(L_C\times C_C,u)$有定义.由引理 2 的(3)可得 $loc(post(L_C\times C_C,b))\subseteq Row_0$. b-变迁是序列 u 最后的动作变迁,于是, $loc(post(L_C\times C_C,u))\subseteq Row_0$. 设 $I(u)=\{i\mid l_{0,i}^C\in loc(post(L_C\times C_C,u))\}$, 表示在 $\mathcal{C}$ 上运行 u 后到达的位置的列下标.如果 $w\in((\mathbb{R}_{\geqslant 0},\Sigma))^*$ 且 $post(L_B\times C_B,w)$有定义,设 $J(w)=\{i\mid l_i^B\in post(L_B\times C_B,u)\}$, 表示 $\mathcal{B}$ 在运行 w 后到达的位置的下标.

引理 4($\mathcal{B}$ **和**$\mathcal{C}$ **上运行间的互模拟)**. 如果 $u\in((\mathbb{R}_{\geqslant 0},\{a,b\}^*b))^*$ 且 $post(L_C\times C_C,u)$有定义,那么,$I(u)=J(g(u))$.如果 $w\in((\mathbb{R}_{\geqslant 0},\Sigma))^*$ 且 $post(L_B\times C_B,w)$有定义,那么,$J(w)=I(g^{-1}(w))$.

设 w 是 $\mathcal{B}$ 最短的仔细重置序列.由引理 1,对于某个 $\omega\in((\mathbb{R}_{\geqslant 0},\Sigma))^*$, 一定有 $w=(t_1,\#)\cdot\omega\cdot(t_2,*)$, $t_1,t_2\in\mathbb{R}_{\geqslant 0}$, 为方便起见,令 $t_1=t_2=0$.因此, $w=(0,c_{m+1})\cdot\omega\cdot(0,c_0)=f(0,a^{m+1}b)\cdot\omega\cdot f(0,b)$. 于是,对每个 $k\in(1,\ldots,|w|)$,$g^{-1}(w[1,k])$的无时间部分的最后一个字母是 b,且 $g^{-1}(w[1,k])\in(t,\{a,b\}^*b)$, $t\in\mathbb{R}_{\geqslant 0}$. 由引理 4 可得, $J(w[1,k])=I(g^{-1}(w[1,k]))$, 而且 $J(w)=I(g^{-1}(w))$ 和 $|J(w)|=|I(g^{-1}(w))|=1$. 因此,$g^{-1}(w)$是 $\mathcal{C}$ 的一个仔细重置序列.

设 u 是 $\mathcal{C}$ 的一个最短的仔细重置序列.容易发现,任何 $u\in((\mathbb{R}_{\geqslant 0},a))^*$ 都不可能是 $\mathcal{C}$ 的仔细重置序列.因此 $u\in((\mathbb{R}_{\geqslant 0},\{a,b\}))^*\setminus((t,a))^*$,$|loc(post(L_C\times C_C,u[1,|u|-1]\cdot(t,a)))|=|loc(post(P,u[1,|u|-1]))|$(根据引理 3 的(2)得到), 其中, $t\in\mathbb{R}_{\geqslant 0}$. 由于 $I(u)=J(g(u))$,所以 $u[|u|]=(t,b)$, $t\in\mathbb{R}_{\geqslant 0},u\in((\mathbb{R}_{\geqslant 0},\{a,b\}^*b))^*$. 因为 u 是 $\mathcal{C}$ 的一个仔细重置序列,$|I(u)=1|$.因此,$|J(g(u))|=1$.这说明,$g(u)$是 $\mathcal{B}$ 的一个仔细重置序列. □

对每个 $k(k\geqslant 2,k\in\mathbb{N})$, k-字母的重置问题是对应的无约束版本的特例.对于每个无约束版本的问题,其 k-字母的重置问题的实例自动机可以转化为同类问题的(k+1)-字母版本的实例,通过增加一个额外的字母即可得到新的自动机,对原自动机的每个位置增加到其自身的变迁,以这个额外字母为标记.于是,k-RESET$\leqslant_p$(k+1)-RESET.

时钟个数对该问题的影响,在第 3 节已有讨论,有关的结论是一致的.字母表大小对完全的确定的时间自动机的重置问题的复杂性影响也可以用类似定理 3 的归约的方法来加以探讨,于是得到以下两个推论.

推论 1. 对于单时钟的部分规约的确定的时间自动机,它的重置问题是 NLOGSPACE-完全的;对含有 2 个或 2 个以上时钟的部分规约的确定的时间自动机,它的重置问题是 PSPACE-完全的.

推论 2. 对于完全的确定的时间自动机,即使它的字母表大小减少到 2,其重置问题仍是 PSPACE-完全的.

5 非确定的时间自动机的重置问题

本节主要讨论完全的非确定的时间自动机的重置问题的两个可判定子类——限界问题和单时钟问题,直接利用了完全的非确定的寄存器自动机的限界重置问题的结论[30]和结构相同的时间自动机与寄存器自动机之间可指数时间内相互可归约的结论[31].比如,在文献[31]中,从安全的单路交错的单寄存器自动机的非空问题是 EXPSPACE-难的,直接得出对应的交错的安全的单时钟自动机的非空问题是 EXPSPACE-难的.

5.1 一般情况

一般情况下的结论是由 Doyen 等人给出的,它是不可判定的.证明的具体细节见文献[20,21].主要思想是利用时间语言普遍性问题的不可判定性[9],得到非普遍性问题的不可判定性.然后从时间语言的非普遍性问题归约到它的重置问题.

5.2 D_i-可重置问题

类似非确定的有限自动机[28,29,32],考虑对应的非确定的时间自动机的 D_i-可重置问题(i=1,2,3).

对于时间自动机 $\mathcal{A}=\langle L,l_0,C,\Sigma,E,F\rangle$, 序列 $w\in((\mathbb{R}_{\geqslant 0},\Sigma))^*$ 被称为是:

- D_1-可重置的,如果对所有$(l,v)\in L\times C$,满足 $post((l,v),w)$有定义且$|post(L\times C,w)|=1$;
- D_2-可重置的,如果对所有$(l,v)\in L\times C$,满足 $post((l,v),w)$有定义且 $post((l,v),w)=post(L\times C,w)$;
- D_3-可重置的,如果 $\bigcap_{(l,v)\in L\times C} post((l,v),w))\neq\varnothing$.

D_i-可重置问题(Di-RESET)可形式化为:

问题:D1-RESET(D2-RESET,D3-RESET).

输入:非确定的时间自动机 $\mathcal{A}$;

询问:非确定的时间自动机是否是 D_1-可重置的(或 D_2-可重置的,或 D_3-可重置的)?

定理 4. 非确定的时间自动机的 D_i-可重置问题是不可判定的(i=1,2,3).

证明:因为 Doyen 等人证明了完全的非确定的时间自动机的重置问题在一般情况下是不可判定的[20,21],所以,非确定的时间自动机是否是 D_1-可重置的是不可判定的.

引理 5. 每个 D_1-重置序列也是 D_2-重置序列,每个 D_2-重置序列也是 D_3-重置序列.

根据引理 5,D_2-可重置和 D_3-可重置问题均是不可判定的. □

5.3 限界的重置问题的复杂性

定理 5. 完全的非确定时间自动机的限界的非普遍性问题(BOUNDED-NONUNIVERSALITY)是 NEXPTIME-完全的.

证明:首先证明该问题属于 NEXPTIME.可以猜测一个长度小于 k 的序列 w,能在指数时间内检查 w 是不是它接受的字.然后证明它是 NEXPTIME-难的.Babari 和 Quaas 等人[30]证明了完全的非确定的寄存器自动机的限界的普遍性问题(BOUNDED-UNIVERSALITY)是 co-NEXPTIME-完全的,即非确定的寄存器自动机的限界的非普遍性问题是 NEXPTIME-完全的.

命题 2. NEXPTIME 和 ACK 在指数时间归约下封闭.

Figueira 给出的时间自动机和寄存器自动机之间的归约是指数时间的,根据命题 2,指数时间归约对于 NEXPTIME 是封闭的.Figueira 等人[31]证明非确定的寄存器自动机的普遍性问题可以在指数时间内归约到非确定的时间自动机的普遍性问题上.那么,非确定的时间自动机的限界的非普遍性问题是 NEXPTIME-难的. □

定理 6. 完全的非确定时间自动机的限界重置问题(BOUNDED-RESET)是 NEXPTIME-完全的.

证明:首先证明该问题是 NEXPTIME 的成员.可以猜测一个长度小于 k 的序列 w,可以在指数时间内检查 w 是否为重置序列.接下来证明这个问题是 NEXPTIME-难的.Doyen 等人[20,21]证明非确定的时间自动机的非普遍性问题可以在多项式时间内归约到非确定的时间自动机的重置问题,于是非确定的时间自动机的限界的非普遍性问题也可以在多项式时间内归约到非确定的时间自动机的限界重置问题.再根据定理 5 可知,非确定的时间自动机的限界重置问题(BOUNDED-RESET)是 NEXPTIME-难的. □

5.4 单时钟的重置问题的复杂性

定理 7. 完全的非确定的单时钟时间自动机的重置问题是 Ackermann-完全的.

证明:首先证明它属于 ACK.参照文献[30](它的引理 6 和引理 7),可知存在两类完全的非确定的单寄存器自动机,使得在重置时需要读取 Ackermann(n)多的不同数据字和为了到达重置状态需要读取 2^n 次输入数据字,其中,位置个数为 $\mathcal{O}(n)$. 时间自动机重置时,重置序列的长度也存在类似的两种情况.然后再证明它是 Ackermann-难的.Figueira 等人[31]证明完全的非确定的寄存器自动机的普遍性问题可以在指数时间内归约到非确定的时间自动机的普遍性问题,并且保持寄存器的数目与时钟数目相等.Babari 和 Quaas[30]证明单寄存器的非确定的寄存器自动机的重置问题是 Ackermann-完全的.根据命题 2,指数时间归约对于 ACK 是封闭的,所以单时钟的非确定的时间自动机的重置问题是 Ackermann-难的. □

注意,指数时间归约不同于多项式时间归约,相对低复杂性类是不封闭的,对它的使用需具体问题具体分析.

6 相关工作

6.1 有限自动机和其他自动机的重置问题

关于重置问题较早、较全面的研究是在完全的确定的有限自动机上进行的.确定的有限自动机的结论见表1的第2列和第3列.表1的第1行第4列和第1行第5列分别对应本文第3节和第4节的工作.鉴于时间自动机结构上比有限自动机更加复杂,本文考虑了第(1)类问题中的若干子类,比如时钟个数、字母表大小的情况.对于完全的确定的时间自动机,完善了 Doyen 等人的工作,得到若干更精细的结论;对于部分规约的时间自动机,本文对它的重置问题的研究工作是最早的,更具体的结论总结见第 7 节的表 2.对于第(2)类问题,本文的一个后继工作已获知其限界问题的复杂性也是 PSPACE-完全的,限于篇幅,对它的证明将另文给出探讨.表 1 和表 2 中复杂性类后的-C 符号表示该复杂性类是完全的,Open 表示对应位置上的问题还未得到研究.

Table 1 Main results of the problems for resetting deterministic finite automata and timed automata

表 1 关于确定的有限自动机和时间自动机重置问题的主要结论

问题类型	完全的 DFA	部分规约的 DFA	完全的 DTA	部分规约的 DTA
(1) 基本问题	NLOGSPACE-C[20]	PSPACE-C[20]	PSPACE-C[20](定理 1 完善)	PSPACE-C(定理 2)
(2) 限界问题	NP-C[22]	Open	PSPACE-C(另文讨论)	PSPACE-C(另文讨论)
(3) 最优问题	DP-C[33]	Open	Open	Open
(4) 阈值问题	Best upper bound: $k=(n^3-n)/6$[34]	Open	Open	Open
(5) 近似问题	No polynomial algorithms exist[35]	Open	Open	Open

对非确定的有限自动机,传统上研究它的 D_i-可重置问题(i=1,2,3).Imreh 和 Steinby[32]给出它们是可判定的,Martyugin[28,29]证明这3个问题都是PSPACE-完全的.对非确定的有限自动机的重置序列的长度 $d_1(n)$,Gazdag 和 Iván 给出 $d_1(n)=\Theta(2^n)$的结论(见文献[36]中的定理 1).本文在第 5 节针对完全的非确定的时间自动机进行了研究,Doyen 等人指出一般情况下它是不可判定的,在此基础上,本文证明了它的 D_i-可重置问题是不可判定的,还找到了它在限界和单时钟条件下的两个可判定的子类,其复杂度分别是 NEXPTIME-完全的和 Ackermann-完全的.这些结论也是本文首次给出.

从有限自动机的重置问题的研究路线和方法上,大致可以得到时间自动机研究的方向、方法和待解决的问题.因为加入了时钟的要素,问题变得更加复杂,可将在有限自动机上证明的思想和方法扩展到时间自动机上来,如本文第 4 节的工作采用类似的思想但却是不同的归约过程,将 Martyugin[28,29]在部分规约的有限自动机上的工作移植到部分规约的时间自动机上.我们还发现:寻找求解时间自动机重置问题的多项式时间的高效算法不应盲目和乐观,其存在性需要计算复杂性的理论结果作为支撑,首先研究重置问题的计算复杂性意义更重大.

对其他种类自动机的研究有:Doyen 等人还讨论了概率有限自动机[37]和权重自动机的重置问题[20,21].Caucal[38]和 Chistikov 等人[39]还分别研究了下推自动机和嵌入字自动机的重置问题.对于 Babari 和 Quaas 等人[30]在寄存器自动机的数据重置字问题上的工作,本文第 5 节在研究非确定的时间自动机的重置问题时,采用寄存器自动机到时间自动机的指数时间可归约[31],利用他们的结论直接证明了新的结论.

6.2 时间自动机的可达性问题

时间自动机的可达性与它的区域自动机的大小有关,时钟区域个数的界限是 $|C|!\cdot 2^{|C|}\cdot\prod_{x\in C}(2c_x+2)$[9],它主要受到时钟个数和每个时钟在约束中出现的最大常数 c_x 的影响.有关时间自动机的可达性问题的复杂性研究工作,可参考文献[27]的介绍部分.

对时间自动机,它的可达性问题[9,40]与时间自动机机对应的时间正则语言的非空问题是等价的,很多问题都与它们有联系.比如,针对 Fq 和 Gp 等性质的 LTL 模型检测问题和本文讨论的时间自动机的重置问题都可以从它归约,因此这两个问题是基本问题.本文的定理 1 是从完全的确定的时间自动机的可达性问题归约的,定理 2 是从多个部分规约的确定的时间自动机语言的交的非空问题(即多个时间自动机积的可达性问题)归约的.这说明,重置问题至少与可达性问题是一样难的,如果有实用的方法求解可达性,那么,同样也可以求解可重置性.

近年来,对时间自动机的可达性问题的研究进展主要体现在理论和算法优化两个方面:(1) 弄清时间自动机模型与其他自动机模型间的相互模拟关系[26,27,31],比如就时钟这个关键影响因素,得到单时钟和 k-时钟($k\geqslant 2$)的自动机的可达性问题的复杂性[23,24],时间自动机与计数器自动机[26,27]、寄存器自动机之间的关系[31],还弄清时间自动机与某些时序逻辑之间的关系,比如 CLTLoc 逻辑[41]、MTL 和 MITL 逻辑[42,43].这些理论在讨论与之相关的复杂性证明时经常用到,本文第 3 节定理 1 的证明使用的结论就是通过时间自动机与计数器自动机之间的相互模拟得到的复杂性结论.另外,本文第 5 节定理 5~定理 7 的证明也使用了寄存器自动机与时间自动机的相互归约关系.(2) 在可达性分析算法优化方面的进展,主要是在抽象-精化的框架下集成插值[15]、对于 zone 结构引入了保可达性的上近似抽象[13,14].另外,是由 SMT 求解器的进步带来的限界模型检测性能的提升[12],这些算法实现的原型在某些情况下的性能是优于 UPPAAL 的.如果采用定理 1 和定理 2 中证明 PSPACE 成员性时提出的基于“猜测-检验”的非确定性算法计算(或检验)重置序列,那么,可达性分析和 SAT/SMT 求解显然是算法实现的重要组成部分,复杂性的结论还揭示出,只有在时间自动机规模不大时,基于“猜测-检验”思想的算法才是可以实现的.

6.3 涉及的多个问题间的关系

不可判定性和复杂性的证明均使用归约,此外,复杂性证明还需进行算法分析,主要分析对资源(时间和空间)的消耗,算法大多是非确定的,与非确定的计算模型相对应.归约分为高效的多项式时间归约和指数时间归约两类,本文所涉及的各问题间的归约关系在图 5 中给出总结,其中,归约符号右上方所标注的定理旨在说明在定理的证明中使用了这种归约.

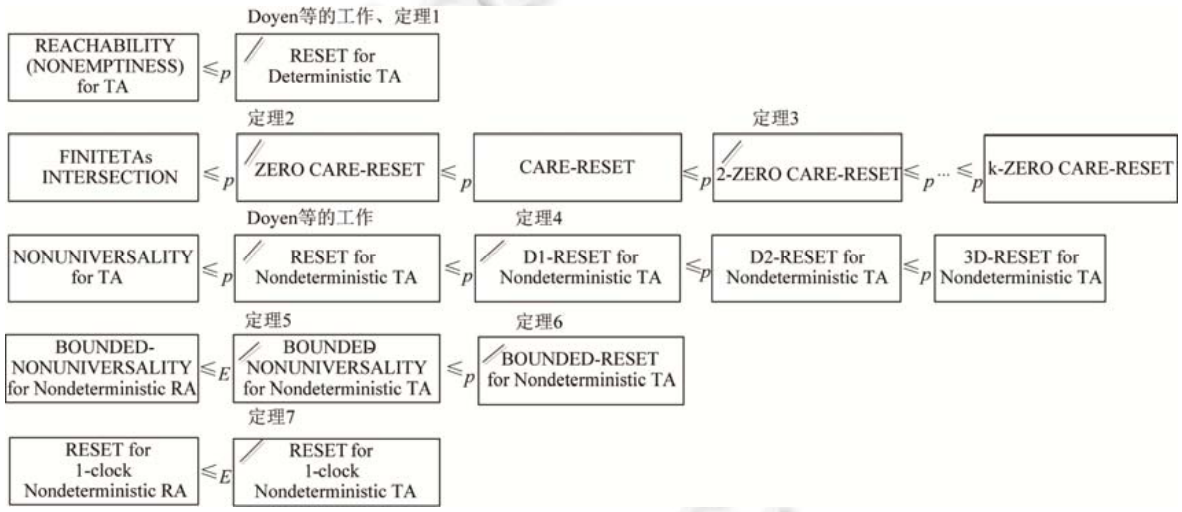

Fig.5 Reductions among these problems involved

图 5 涉及的问题之间的归约

7 结 论

本文最先研究了时间自动机在若干种约束条件下的重置问题的复杂性,具体结论见表 2,表中标注了对应定理和推论的结论都属于本文的新结论,特别是部分规约的确定的时间自动机的重置问题的若干结论.

对于重置问题,可以将时间自动机 $\mathcal{A}=\langle L,l_0,C,\Sigma,E,F\rangle$ 的变迁函数看成是定义在格局上的一个函数 $f_{t,x}$: $L\times C\xrightarrow{g,r}L\times C,\forall t\in\mathbb{R}_{\geqslant 0}\forall x\in\Sigma.\ f_{t,x}(l,v)=(l',v')$ 当且仅当 $v|=g$ 且 $v'=(v+t)[r]$.可以把计算重置序列看作是寻找一个复合函数 g 的过程,这个函数具有 $g(l,v)=f_{t_1,x_1}(f_{t_2,x_2}(...f_{t_k,x_k}(l,v)))$ 的形式,满足 $\forall(l,v)\in L\times C\ \exists(t_1,x_1)...(t_k,x_k)\in(\mathbb{R},\Sigma)^*$, g 是常函数.定理 3 的证明就用到了这一思想.

下一步的研究方向是时间自动机的阈值问题和可描述时间系统的可重置性的逻辑.另一个研究方向是利用已有的开源模型检测框架[15]和 SMT 求解器开发计算重置序列的原型.

Table 2 Results of the problems for resetting timed automata

表 2 关于时间自动机重置问题的结论

问题分类	约束条件	完全的确定的 TA	部分规约的确定的 TA	完全的非确定的 TA
(1) 基本问题	一般情况	PSPACE-C[20,21]	PSPACE-C(定理 2)	Undecidable[20,21]
	单时钟	NLOGSPACE-C(定理 1)	NLOGSPACE-C(推论 1)	Ackermann-C(定理 7)
	时钟个数≥2	PSPACE-C(定理 1)	PSPACE-C(推论 1)	Undecidable
	字母表大小≥2	PSPACE-C(推论 2)	PSPACE-C(定理 3)	Undecidable
(2) 限界问题	$\|w\|\leqslant k$	PSPACE-C(另文讨论)	PSPACE-C(另文讨论)	NEXPTIME-C(定理 6)
其他问题	D_t-可重置	/	/	Undecidable(定理 4)

致谢 感谢华南农业大学数学与信息学院黄琼教授为本文复杂性证明提供了很好的建议.感谢各位匿名评审专家为本文研究工作的进一步完善提出了宝贵意见.

References:

[1] Černý J. Poznámka *k*-homogénnym eksperimentom s konečnými automata mi. Mathematicko-fyzikalny Časopis Slovensk, Akad. Vied, 1964,14(3):208–216 (in Slovak).

[2] Benenson Y, Adar R, Paz-Elizur T, Livneh Z, Shapiro E. DNA molecule provides a computing machine with both data and fuel. Proc. of the National Academy of Sciences, 2003,100(5):2191–2196. [doi: 10.1073/pnas.0535624100]

[3] Benenson Y, Paz-Elizur T, Adar R, Keinan E, Livneh Z, Shapiro E. Programmable and autonomous computing machine made of biomolecules. Nature, 2001,414(6862):430. [doi: 10.1038/35106533]

[4] Stojanovic MN, Stefanovic D. A deoxyribozyme-based molecular automaton. Nature Biotechnology, 2003,21(9):1069–1074. [doi: 10.1038/nbt862]

[5] Natarajan BK. An algorithmic approach to the automated design of parts orienteers. In: Proc. of the 27th Annual Symp. on Foundations of Computer Science. IEEE, 1986. 132–142. [doi: 10.1109/SFCS.1986.5]

[6] Berlinkov MV, Szykula M. Algebraic synchronization criterion and computing reset words. Information Sciences, 2016, 718–730. [doi: 10.1016/j.ins.2016.07.049]

[7] Song F, Wu ZL. Survey on formal models to reason about infinite data values. Ruan Jian Xue Bao/Journal of Software, 2016,27(3): 1–9 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4989.htm [doi: 10.13328/j.cnki.jos.004989]

[8] Chatterjee K, Doyen L. Computation tree logic for synchronization properties. In: Proc. of the LIPIcs-Leibniz Int'l in Informatics. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016. 55. [doi: 10.4230/LIPIcs.ICALP.2016.98]

[9] 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]

[10] Behrmann G, David A, Larsen KG, Håkansson J, Petterson P, Yi W, Hendriks M. UPPAAL 4.0. In: Proc. of the 3rd Int'l Conf. on Quantitative Evaluation of Systems, 2006(QEST 2006). IEEE, 2006. 125–126. [doi: 10.1109/QEST.2006.59]

[11] Bengtsson J, Yi W. Timed automata: Semantics, algorithms and tools. In: Advanced Course on Petri Nets. Berlin, Heidelberg: Springer-Verlag, 2003. 87–124. [doi: 10.1007/978-3-540-27755-2_3]

[12] Bouyer P, Fahrenberg U, Larsen KG, Markey N, Ouaknine J, Worrell J. Model checking real-time systems. In: Handbook of Model Checking. Cham: Springer-Verlag, 2018. 1001–1046. [doi: 10.1007/978-3-319-10575-8_29]

[13] Herbreteau F, Srivathsan B, Walukiewicz I. Lazy abstractions for timed automata. In: Proc. of the 24th Int'l Conf. on Computer Aded Verification (CAV 2013). Springer-Verlag, 2013. 990–1005. [doi: 10.1007/978-3-642-39799-8_71]

[14] Herbreteau F, Srivathsan B, Walukiewicz I. Better abstractions for timed automata. Information and Computation, 2016,251:67–90. [doi: 10.1016/j.ic.2016.07.004]

[15] Tóth T, Hajdu Á, Vörös A, Micskei Z, Majzik I. Theta: A framework for abstraction refinement-based model checking. In: Proc. of the Formal Methods in Computer Aided Design (FMCAD). IEEE, 2017. 176–179. [doi: 10.23919/FMCAD.2017.8102257]

[16] Zhao J, Li X, Zheng T, Zheng G. Removing irrelevant atomic formulas for checking timed automata efficiently. In: Proc. of the Int'l Conf. on Formal Modeling and Analysis of Timed Systems. Berlin, Heidelberg: Springer-Verlag, 2003. 34–45. [doi: 10.1007/978-3-540-40903-8_4]

[17] Zhao J, Li X, Zheng G. A quadratic-time DBM-based successor algorithm for checking timed automata. Information Processing Letters, 2005,96(3):101–105. [doi: 10.1016/j.ipl.2005.05.027]

[18] Lin H, Yi W. A proof system for timed automata. In: Proc. of the Int'l Conf. on Foundations of Software Science and Computation Structures. Berlin, Heidelberg: Springer-Verlag, 2000. 208–222. [doi: 10.1007/3-540-46432-8_14]

[19] Lin H, Yi W. Axiomatising timed automata. Acta informatica, 2002,38(4):277–305. [doi: 10.1007/s236-002-8035-2]

[20] Doyen L, Juhl L, Larsen KG, Markey N, Shirmohammadi M. Synchronizing words for weighted and timed automata. In: Proc. of the Int'l Conf. on Foundation of Software Technology and Theoretical Computer Science. 2014. 121–132. [doi: 10.4230/LIPIcs.FSTTCS.2014.121]

[21] Doyen L, Juhl L, Larsen KG, Markey N, Shirmohammadi M. Synchronizing words for timed and weighted automata. Research Report, LSV-13-15(version 2), Laboratoire Spécification et Vérification, ENS Cachan, France, 2014. 28.

[22] Eppstein D. Reset sequences for monotonic automata. SIAM Journal on Computing, 1990,19(3):500–510. [doi: 10.1137/0219033]

[23] Laroussinie F, Markey N, Schnoebelen P. Model checking timed automata with one or two clocks. In: Proc. of the Int'l Conf. on Concurrency Theory. Berlin, Heidelberg: Springer-Verlag, 2004. 387–401. [doi: 10.1007/978-3-540-28644-8_25]

[24] Fearnley J, Jurdziński M. Reachability in two-clock timed automata is PSPACE-complete. Information and Computation, 2015,243: 26–36. [doi: 10.1016/j.ic.2014.12.004]

[25] Courcoubetis C, Yannakakis M. Minimum and maximum delay problems in real-time systems. Formal Methods in System Design, 1992,1(4):385–415. [doi: 10.1007/BF00709157]

[26] Haase C, Ouaknine J, Worrell J. On the relationship between reachability problems in timed and counter automata. In: Proc. of the Int'l Workshop on Reachability Problems. Berlin, Heidelberg: Springer-Verlag, 2012. 54–65. [doi: 10.1007/978-3-642-33512-9_6]

[27] Haase C, Ouaknine J, Worrell J. Relating reachability problems in timed and counter automata. Fundamenta Informaticae, 2016,143 (3-4):317–338. [doi: 10.3233/FI-2016-1316]

[28] Martyugin P. Complexity of problems concerning carefully synchronizing words for PFA and directing words for NFA. In: Proc. of the Computer Science Symp. in Russia. 2010. 288–302. [doi: 10.1007/978-3-642-13182-0_27]

[29] Martyugin P. Computational complexity of certain problems related to carefully synchronizing words for partial automata and directing words for nondeterministic automata. Theory of Computing Systems, 2014,54(2):293–304. [doi: 10.1007/s00224-013-9516-6]

[30] Babari P, Quaas K, Shirmohammadi M. Synchronizing data words for register automata. In: Proc. of the LIPIcs-Leibniz Int'l in Informatics. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016. 58. [doi: 10.4230/LIPIcs.MFCS.2016.15]

[31] Figueira D, Hofman P, Lasota S. Relating timed and register automata. Mathematical Structures in Computer Science, 2016,26(6): 993–1021. [doi: 10.1017/S0960129514000322]

[32] Imreh B, Steinby M. Directable nondeterministic automata. Acta Cybernetica, 1999,14(1):105–115.

[33] Olschewski J, Ummels M. The complexity of finding reset words in finite automata. Mathematical Foundations of Computer Science, 2010, 568–579. [doi: 10.1007/978-3-642-15155-2_50]

[34] Pin J. On two combinatorial problems arising from automata theory. North-holland Mathematics Studies, 1983, 535–548. [doi: 10.1016/S0304-0208(08)73432-7]

[35] Berlinkov MV. Approximating the minimum length of synchronizing words is hard. In: Proc. of the Computer Science Symp. in Russia. 2010. 37–47. [doi: 10.1007/978-3-642-13182-0_4]

[36] Gazdag Z, Iván S, Nagy-György J. Improved upper bounds on synchronizing nondeterministic automata. Information Processing Letters, 2009,109(17):986–990. [doi: 10.1016/j.ipl.2009.05.007]

[37] Doyen L, Massart T, Shirmohammadi M. Infinite synchronizing words for probabilistic automata. In: Proc. of the Int'l Symp. on Mathematical Foundations of Computer Science. Berlin, Heidelberg: Springer-Verlag, 2011. 278–289. [doi: 10.1007/978-3-642-22993-0_27]

[38] Caucal D. Synchronization of pushdown automata. In: Proc. of the Int'l Conf. on Developments in Language Theory. Berlin, Heidelberg: Springer-Verlag, 2006. 120–132. [doi: 10.1007/11779148_12]

[39] Chistikov D, Martyugin P, Shirmohammadi M. Synchronizing automata over nested words. In: Proc. of the Int'l Conf. on Foundations of Software Science and Computation Structures. Berlin, Heidelberg: Springer-Verlag, 2016. 252–26. [doi: 10.1007/978-3-662-49630-5_15]

[40] Quaas K, Shirmohammadi M, Worrell J. Revisiting reachability in timed automata. In: Proc. of the 32nd Annual ACM/IEEE Symp. on Logic in Computer Science (LICS). IEEE, 2017. 1–12. [doi: 10.1109/LICS.2017.8005098]

[41] Bersani MM, Rossi M, San Pietro P. A logical characterization of timed regular languages. Theoretical Computer Science, 2017, 658:46–59. [doi: 10.1016/j.tcs.2016.07.020]

[42] Maler O, Nickovic D, Pnueli A. From MITL to timed automata. In: Proc. of the Int'l Conf. on Formal Modeling and Analysis of Timed Systems. Berlin, Heidelberg: Springer-Verlag, 2006. 274–289. [doi: 10.1007/11867340_20]

[43] Ničković D, Piterman N. From MTL to deterministic timed automata. In: Proc. of the Int'l Conf. on Formal Modeling and Analysis of Timed Systems. Berlin, Heidelberg: Springer-Verlag, 2010. 152–167. [doi: 10.1007/978-3-642-15297-9_13]

[44] Papadimitriou CH. Computational Complexity. John Wiley and Sons Ltd., 2003.

[45] Schmitz S. Complexity hierarchies beyond elementary. ACM Trans. on Computation Theory (TOCT), 2016,8(1):3:1–3:36. [doi: 10.1145/2858784]

附中文参考文献:

[7] 宋富,吴志林.面向无穷数据的形式模型综述.软件学报,2016,27(3):1–9. http://www.jos.org.cn/1000-9825/4989.htm [doi: 10.13328/j.cnki.jos.004989]

附录

引理 1 的证明:只有经过*-变迁才能将 $\mathcal{A}_1,\ldots,\mathcal{A}_k$ 中的状态最终融合到 end 位置.因此,$(t_2,*)$应出现在 u 中.简单起见,可令 $t_2=0$.显然,如果 $post(L_B\times C_B,(0,*))$有定义,则 $post(L_B\times C_B,(0,*))=(\text{end},\vec{0})$, 它是单元集.如果对于 $n\leqslant|u|$,$u[n]=(t_2,*)$,那么,u 的前缀 $u[1,n]$也是 $\mathcal{B}$ 的一个仔细重置序列.由于假设 u 是最短的仔细重置序列,那么,$n=|u|$且$(t_2,*)$在 u 中仅出现 1 次,它位于 u 的末端.

因为 u 必须从 begin 位置开始,Σ中的字母和符号*在 begin 位置上均没有定义,所以 $u[1]=(t_1,\#)$,简单起见,可令 $t_1=0$,于是对于某个 $w\in((\mathbb{R}_{\geqslant 0},\Sigma\cup\{\#\}))^*$, u 可以被表示为 $u=(0,\#)\cdot w\cdot(t_2,*)$ 的形式.因为对每个 $i\in\{1,\ldots,k\}$,满足 $post(L_B\times C_B,(0,\#))\cap(L_i\times C_B)=\{(l_i,\vec{0})\}$, 所以 $|post(L_B\times C_B,(0,\#))\cap(L_i\times C_B)|=1$. Σ中的字母和符号#都定义在集合 $L_i\times C_B$ 上,并且这些变迁将 $L_i\times C_B$ 映射到 $L_i\times C_B$ 自身.因此,对 $\forall m\in\{1,\ldots,|u|-1\}$, 有 $|loc(post(L_B\times C_B,u[1,m]))\cap L_i|=1$. 于是,对于某个 m,如果 $u[m]=(0,\#)$,那么, $loc(post(L_B\times C_B,u[1,m]))=\{l_1,\ldots,l_k\}=loc(post(L_B\times C_B,(0,\#)))$, 这样, $(0,\#)\cdot u[m+1,|u|]$ 也是 $\mathcal{B}$ 的一个仔细重置序列.由于已假设 u 是最短的仔细重置序列,所以,(0,#)变迁在 u 中只会出现 1 次,因此,(0,#)不会出现在 w 中.

于是, $u=(t_1,\#)\cdot w\cdot(t_2,*)$, 其中, $w\in((\mathbb{R}_{\geqslant 0},\Sigma))^*,t_1,t_2\in\mathbb{R}_{\geqslant 0}$. □

命题 1 的证明:由于时间自动机对于交运算是封闭的(见文献[9]中的定理 3.15),所以 k 个时间自动机 $\mathcal{A}_1\ldots\mathcal{A}_k$ 的交运算形成新的时间自动机 $\mathcal{A}=\bigcap_{i=1}^{k}\mathcal{A}_i$. 由文献[9]定理 3.15 证明中的构造过程得到,该乘积自动机 $\mathcal{A}$ 的状态个数是 $k\cdot\prod_{i=1}^{k}|L_i|$, 时钟个数是 $\sum_{i=1}^{k}|C_i|$, 变迁集的大小是 $k\cdot\prod_{i=1}^{k}|E_i|$. 注意,假设对常量采用二进制编码,$|E|$包括时钟约束的长度.

再由时间正则语言 $\mathcal{L}(\mathcal{A})$ 非空的判定问题是 PSPACE-完全的(具体细节参考文献[9]中的定理 4.17 的证明)可证.

注意,该命题 PSPACE 成员性可以看成是"PSPACE 对交运算封闭"[44]的具体表现形式. □

引理 2 的证明:(1)~(3)均可由变迁函数的定义直接得出. □

注意,(1)和(2)意味着 a-变迁不改变位置所在列号,但会使位置的行号加 1,如果位置已在最后一行,那么该位置形成自环.(3)意味着 b-变迁会使位置进入第 0 行.

引理 3 的证明:

(1) 由于 $u\in((\mathbb{R}_{\geqslant 0},\{a,b\}))^*\setminus((\mathbb{R}_{\geqslant 0},\{a\}))^*$, (t_1,b)-变迁出现在序列 u 中, $t_1\in\mathbb{R}_{\geqslant 0}$. 设 $u[h_1]$是(t_1,b)在 u 中的首次出现.由引理 2 的(3)得到 $loc(post(L_C\times C_C,b))\subseteq Row_0$. 于是, $loc(post(L_C\times C_C,u[1,h_1]))\subseteq Row_0$, 所以, $|Col_i\cap loc(post(L_C\times C_C,u[1,h_1]))|\leqslant 1$. 假设对于某个 $j\in\{1,...,|u|-1\}$, 有 $|Col_i\cap loc(post(L_C\times C_C,u[1,j]))|\leqslant 1$. 如果 $u[j+1]=(t_2,a)$, $t_2\in\mathbb{R}_{\geqslant 0}$, 那么,由引理 2 得到,对每个 $i\in\{1,\ldots,n\}$, $loc(post(Col_i\times C_C,(t_2,a)))\subseteq Col_i$. 因此,

$$|loc(post(L_C\times C_C,u[1,j]\cdot(t_2,a)))\cap Col_i|\subseteq|loc(post(L_C\times C_C,u[1,j]))\cap Col_i|\leqslant 1.$$

如果 $u[j+1]=(t_1,b)$, 那么, $Col_i\cap loc(post(L_C\times C_C,u[j+1]))\subseteq Row_0$. 这意味着 $|loc(post(L_C\times C_C,u[j+1]))\cap Col_i|\leqslant 1$, 故 $|Col_i\cap loc(post(L_C\times C_C,u))|\leqslant 1$.

(2) 设 $i\in\{1,\ldots,n\}$.由(1)得到 $|Col_i\cap loc(post(L_C\times C_C,u))|\leqslant 1$. 由引理 2 的(1)得到:对于 $i\in\{1,\ldots,n\}$, $loc(post(Col_i\times C_C,a))\subseteq Col_i$. a-变迁在 L_C 中的任何位置上都有定义.因此,如果 $|Col_i\cap loc(post(L_C\times C_C,u))|=1$, 那么 $|Col_i\cap loc(post(L_C\times C_C,u\cdot(t,a)))|=1$, $t\in\mathbb{R}_{\geqslant 0}$. 于是,

$$\begin{aligned}|loc(post(L_C\times C_C,u))|&=\sum_{i=1}^{n}|loc(post(L_C\times C_C,u))\cap Col_i|\\&=\sum_{i=1}^{n}|loc(post(L_C\times C_C,u\cdot(t,a)))\cap Col_i|\\&=|loc(post(L_C\times C_C,u\cdot(t,a)))|.\end{aligned}$$

(3) 由引理 2 的(3)得到: $loc(post(L_C\times C_C,b))\subseteq Row_0$. 因此 $loc(post(L_C\times C_C,u\cdot(t,b)))\subseteq loc(post(L_C\times C_C,b))\subseteq Row_0$.

引理 4 的证明:设 $u\in((\mathbb{R}_{\geqslant 0},\{a,b\}^*b))^*$, 那么对于某些 $h_1,...,h_r\in\mathbb{N}$, 有 $u=(t_1,a^{h_1}b)(t_2,a^{h_2}b)...(t_r,a^{h_r}b)$ 成立,其中, $t_1,...,t_r\in\mathbb{R}_{\geqslant 0}$. 对于某个 $k\in\{1,\ldots,r\}$,设 $u_k=(t_1,a^{h_1}b)(t_2,a^{h_2}b)...(t_k,a^{h_k}b)$.

归纳基础.当 k=1 时,如果 $post(L_C\times C_C,u)$ 有定义,那么 $post(L_C\times C_C,(t_1,a^{h_1}b))=post(L_C\times C_C,u_1)$ 也有定义.这只能出现在 $h_1\geqslant m+1$ 的情况下,因为如果 $h_1<m+1$,那么 $l^C_{i,j}\in loc(post(L_C\times C_C),a^{h_1})$, 随后的 b-变迁在位置 $l^C_{i,j}$ 上不一定有定义,矛盾.因此, $g(u_1)=g(t_1,a^{h_1}b)=f(t_1,a^{h_1}b)=(t_1,c_{m+1})=(t_1,\#)$. 可以验证 $I(u_1)=I((t_1,a^{h_1}b))=J((t_1,c_{m+1}))$.

归纳步骤.当 k>1 时,假设 $I(u_{k-1})=J(g(u_{k-1}))$, 证明 $I(u_k)=J(g(u_k))$. 设 $i\in I(u_{k-1})=J(g(u_{k-1}))=J(f_1\circ\ldots\circ f_{k-1})$, 这意味着 $l^C_{0,i}\in loc(post(L_C\times C_C,u_{k-1}))$ 和 $l^B_i\in post(L_B\times C_B,g(u_{k-1}))$, 即 $\mathcal{C}$ 的第 0 行第 i 列上的位置 $l^C_{0,i}$ 对应 $\mathcal{B}$ 的位置 l^B_i. 如果 $post(l^C_{0,i}\times C_C,(t_k,a^kb))$ 有定义,那么对于某个 $m\in I(u_k)$, $loc(post(l^C_{0,i}\times C_C,a^kb))=l^C_{0,m}$, 即 $l^C_{0,i}$ 经过(t_k,a^kb)后可以到达 $l^C_{0,m}$ 位置.因此,从 $\mathcal{C}$ 的变迁函数定义可以得到:对于 $\mathcal{B}$, $post(l^B_i,f(t,a^{h_k}b))$ 有定义,并且 $loc(post(l^B_i,f(t,a^{h_k}b)))=l^B_m$. 因此, $m\in J(g(u_k))$ 和 $I(u_k)\subseteq J(g(u_k))$. 对包含关系 $I(u_k)\supseteq J(g(u_k))$ 的证明类似.于是 $I(u_k)=J(g(u_k))$. 这就意味着 $I(u)=J(g(u))$.

引理的第 2 句可类似地归纳证明. □

推论 1 的证明:由定理 1 的证明中提到的:时钟个数不同时,时间自动机的可达性问题的复杂性结论和定理 2 的证明中提到的:一般情况下部分规约的时间自动机的重置问题的复杂性结论可证. □

推论 2 的证明:采用定理 3 的证明中使用的多字母表到 2-字母表的映射方法可证. □

引理 5 的证明:可由 D_i-重置序列(i=1,2,3)的定义直接得到. □

命题 2 的证明:首先证明,对于问题 A 和问题 B,如果 $A\leqslant_E B$ 且 $A\in$NEXPTIME,那么 $B\in$NEXPTIME.设 M 是判定 B 的算法,f 是从 A 到 B 的指数时间归约.判定 A 的算法 N 可以描述为

N="对于输入 w:

(1) 计算 $f(w)$;

(2) 以 $f(w)$为输入,运行 M,输出 M 的输出."算法 N 判定问题 A 当且仅当算法 M 判定问题 B.如果 w 是 A 的输入,那么 $f(w)$是 B 的输入.因为 f 是从 A 到 B 的指数时间归约,于是只要 M 接受 $f(w)$,N 就接受 w.此外,步骤(1)是在 $O(2^{n^{k_1}})$ 时间内运行的,其中, $k_1\in\mathbb{N}$. 算法 N 的时间消耗是步骤(1)和步骤(2)消耗的总时间,而已知 N 是非确定的指数时间.由于 NEXPTIME $=\bigcup_{k>0}$NTIME(2^{n^k}) 和 EXPTIME $\subseteq$ NEXPTIME[44], 所以步骤(2)(即算法 M)是非确定的指数时间算法.所以 $B\in$NEXPTIME.

然后证明:对于问题 A 和问题 B,如果 $A\leqslant_E B$ 且 $A\in$ACK,则 $B\in$ACK.由于 ACK 是超过 ELEMENTARY 的快速增长的复杂性类[45],显然,它相对指数时间归约是封闭的. □

朱凯(1979－),男,湖北十堰人,讲师,主要研究领域为理论计算机科学,形式化方法.

吴理华(1976－),男,博士,讲师,主要研究领域为自动机理论,离散事件系统.

毋国庆(1954－),男,教授,博士生导师,CCF 高级会员,主要研究领域为软件理论,软件工程,需求工程.

袁梦霆(1976－),男,博士,副教授,CCF 专业会员,主要研究领域为程序语言理论,编译优化,软件工程.