软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2019,30(7):1953−1965 [doi: 10.13328/j.cnki.jos.005752]

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

基于实时自动机的连续时段演算的验证*

安 杰, 张苗苗

(同济大学 软件学院,上海 201804)

通讯作者: 张苗苗, E-mail: miaomiao@tongji.edu.cn

摘 要: 时段演算是描述和推导嵌入式实时系统和混成系统性质的一种区间时态逻辑.扩展线性时段不变式是时段演算的重要子集.针对实时自动机,提出一种连续时间语义下扩展线性时段不变式的有界模型检验方法.该方法将扩展线性时段不变式的有界模型检验问题转化为量词线性算术公式的正确性问题,从而可以采用量词消去技术进行求解.首先,运用符号化的思想,在实时自动机上利用深度优先搜索找到所有满足观测时长约束的符号化路径片段;然后,将每条符号化路径片段转化为一个量词线性算术公式;最后,利用量词消去工具求解.与已有工作相比,基于实时自动机设计了验证算法.另外,降低了验证复杂度,并且加速了验证过程的实际速度.

关键词: 时段演算;扩展线性时段不变式;量词线性算术;量词消去

中图法分类号: TP311

中文引用格式: 安杰,张苗苗.基于实时自动机的连续时段演算的验证.软件学报,2019,30(7):1953−1965. http://www.jos.org.cn/1000-9825/5752.htm

英文引用格式: An J, Zhang MM. Verifying continuous-time duration calculus against real-time automaton. Ruan Jian Xue Bao/Journal of Software, 2019,30(7):1953−1965 (in Chinese). http://www.jos.org.cn/1000-9825/5752.htm

Verifying Continuous-time Duration Calculus against Real-time Automaton

AN Jie, ZHANG Miao-Miao

(School of Software Engineering, Tongji University, Shanghai 201804, China)

Abstract: Duration calculus is a kind of interval temporal logic, which is designed for specifying and reasoning the properties about the embedded real-time systems and hybrid systems. Extend linear duration invariants (ELDI) is an important subset of duration calculus. In this study, a bounded model checking algorithm of ELDI formulas against real-time automaton is proposed. The bounded model checking problem of ELDI is reduced into the validity problem of Quantified linear real arithmetic (QLRA) formulas, which can be solved by Quantifier elimination (QE) technique. Firstly, by using deep first search algorithm, the real-time automaton is searched to find all the symbolic paths segments which satisfy the constraints of observation time length. Secondly, the paths segments are transformed into QLRA formulas. Finally, the QLRA formulas are solved by QE tools. Thus, compared with the related works, a verification algorithm of ELDIs is proposed against real-time automaton with lower complexity. In addition, the practical speed of verifying process is accelerated.

Key words: duration calculus; extended linear duration invariants; quantified linear real arithmetic; quantifier elimination

时段演算[1,2]是周巢尘院士、Hoare 和 Ravn 提出的一种在较高抽象层次上描述和推理嵌入式实时系统及混成系统性质的区间时态逻辑.它对区间时态逻辑[3]进行了扩展,引入了时段(duration)记号,将在观察区间内(从 a 时刻到 b 时刻)系统在某个状态(s)的累积停留时间记为状态表达式积分 $\left(\int_a^b s\right)$.目前时段演算已在实际系统设

* 基金项目: 国家自然科学基金(61472279)
Foundation item: National Natural Science Foundation of China (61472279)
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: 2018-09-28; 采用时间: 2018-12-13; jos 在线出版时间: 2019-03-28
CNKI 网络优先出版: 2019-03-29 09:14:19, http://kns.cnki.net/kcms/detail/11.2560.TP.20190329.0914.004.html

计与验证中有了广泛应用[4,5].时段演算强大的表达能力对系统需求描述有诸多好处,但其较高的抽象程度为自动化验证带来了不便.时段演算的可满足性(satisfiability)、正确性(validity)和模型检验(model checking)问题都是不可判定的(undecidable)[6].

周巢尘等学者提出并分析了时段演算中一个名为线性时段不变式(linear duration invariants,简称 LDI)[7]的子集,其形式为 $b \leqslant \ell \leqslant e \Rightarrow \sum_{s\in S} c_s \int s \leqslant M$; 其中,$\int s$ 表示在观测区间$[t,t']$上,系统在状态 s 的累积停留时间,ℓ 表示观测区间长度,即 $t'-t$,c_s 为常量系数,M 为实数.因此,一个 LDI 公式表示:满足观测区间长度约束时系统状态的时段性质需满足线性约束 $\sum_{s\in S} c_s \int s \leqslant M$. LDI 公式的表达能力较强.例如,对于文献[5]中煤气燃烧器的例子,可以将燃烧器的安全需求"任何时候对于系统观察超过 1min,煤气泄漏的累积时间不得超过观察时长的二十分之一",表示为一个 LDI 公式 $\ell \geqslant 60 \Rightarrow 20\int Leak \leqslant \ell$, 该 LDI 公式可以很容易地改写为上述线性形式 $\ell \geqslant 60 \Rightarrow 19\int Leak - \int Nonleak \leqslant 0$.

文献[7]证明了基于实时自动机的 LDI 性质的模型检验问题是可判定的.实时自动机可以看作是一类特殊的时间自动机,即只有一个时钟变量并且每次迁移都会重置该时钟.基于该工作,学术界研究了在比实时自动机表达能力更强的模型(如时间自动机、混成自动机)下 LDI 公式的模型检验问题[8−12].其中,文献[11,12]提出了一些对于时间自动机下 LDI 的模型检验的高效算法,并将这些方法扩展到时间自动机网络中[13].

那么对于时段演算是否可以找到一个比 LDI 更大的子集,其模型检验问题仍然是可以判定的.一个比较自然的想法是,对 LDI 进行扩展,例如,加入布尔连接符和模态词切变(chop),这样就形成扩展线性时段不变式(extended linear duration invariants,简称 ELDI).根据文献[6]的结论,在离散时间语义和连续时间语义下,ELDI 的可满足性(satisfiability)和正确性(validity)都是不可判定的.Martin 等人证明了在离散时间语义和连续时间语义下,对于有限状态机下 ELDI 的模型检验问题是不可判定的[14].因此,他们提出了 ELDI 的一个近似语义,并且给出了在该近似语义和离散时间条件下的模型检验算法.由于该算法复杂度过高,在实际系统中很难运用,所以 Martin 等人提出了另外一种近似语义,并将 ELDI 的模型检验问题转化为 Presburger 算术[15].受到该工作的启发,我们证明了在标准离散时间语义下基于时间自动机的 ELDI 有界模型检验问题是可以判定的,并给出了一种高效的模型检验算法[16].更进一步地,我们讨论了在连续时间语义下基于时间自动机的 ELDI 的有界模型检验问题,给出了复杂度为三阶指数的判定算法[17].在此指出,文献[16,17]与本文所说的有界模型检验均特指对于 ELDI 公式中观测区间长度 ℓ 有上界,即对于形式 $b\leqslant\ell\leqslant e$,其中的上界 e 有界.

在本文中,我们研究在标准连续时间语义下基于实时自动机的 ELDI 的有界模型检验问题.证明了该问题是可以判定的,并且给出模型检验算法.与文献[7]的区别是,文献[7]讨论的是基于实时自动机的线性时段不变式(LDI)的模型检验问题,而本文讨论的是基于实时自动机的扩展线性时段不变式(ELDI)的模型检验问题,针对的公式更加复杂.与文献[16]比较最主要的一个不同点是,本文关注于连续时间语义,而文献[16]关注的是离散时间语义.与文献[17]比较,本文讨论的模型是实时自动机,可以得到更加简便的模型检验算法,在多个方面降低了时间复杂度,提高了算法实际运行效率.

本文验证方法的基本思路如下.

- 第 1 步,运用符号化的思想,在实时自动机上利用深度优先搜索找到所有满足观测时长约束的符号化路径片段;
- 第 2 步,将每条路径转化为一个量词线性算术表达式(quantified linear real arithmetic,简称 QLRA);
- 第 3 步,将得到的所有量词线性算术表达式运用量词消去工具(例如 REDLOG)求解.

在第 1 步中,我们定义了符号化路径片段,给出了相应的求取方法,并且有效地缩小了搜索空间.在第 2 步中,我们详细定义并构造了组成量词线性算术表达式的不等式约束,并且采用一些方法减少量词的引入个数.由于量词消去的复杂度与量词个数相关,因此第 2 步的处理也加快了第 3 步量词消去的实际运行速度.

本文第 1 节对相关基础知识进行回顾,包括实时自动机和扩展的线性时段不变式,并给出本文涉及的量词线性算术的定义,方便后文叙述.第 2 节对于运用符号化的思想在实时自动机上利用深度优先搜索找到所有满足观测时长约束的符号化路径片段进行描述.第 3 节介绍如何将一条符号化路径片段转化为一个量词线性算

术表达式.第 4 节根据量词线性算术表达式结果给出 ELDI 模型检验的判定结果以及相关定理.第 5 节给出工具实现描述和实验结果.最后对全文进行总结.

1 基础知识及相关定义

本节首先介绍实时自动机,实时自动机作为本文讨论的模型检验问题中的模型语言.之后介绍扩展线性时段不变式(ELDI),ELDI 作为本文模型检验问题中的需求描述逻辑.第 3 节给出量词线性算术的定义.为了叙述简洁,本文固定一组命题集合记为$\mathcal{P}$.

1.1 实时自动机

实时自动机(real-time automaton,简称 RTA)[18]可以看作时间自动机(timed automaton)[19]的一个子集,即只含有一个时钟变量并且每次迁移都重置该时钟变量.实时自动机是对实时系统建模的主流模型.下面给出实时自动机的定义.

定义 1(实时自动机). 一个实时自动机 RTA 是一个 6 元组 $\mathcal{A}=(S,\Sigma,\Delta,s_0,F,\mu)$,其中,

- S 是一个有穷状态(位置)集合;
- Σ是一个有穷字母表;
- $\Delta\subseteq S\times\Sigma\times S$ 是迁移关系;
- $s_0\in S$ 是初始状态;
- $F\subseteq S$ 是接收状态集合;
- $\mu:\Delta\to 2^{\mathbb{R}_{\geqslant 0}}\setminus\{\varnothing\}$ 是一个时间标签函数($\mu(\Delta)$通常是一些区间的集合,这些区间的端点一般落在$\mathbb{N}\cup\{\infty\}$或者 $\mathbb{Q}_{\geqslant 0}\cup\{+\infty\}$).

一条迁移$(s_1,\sigma,s_2)\in\Delta$称为$\sigma$-迁移,迁移的源节点状态和目标节点状态分别表示为 Pre_σ和 $Post_\sigma$.对于实时自动机,它的一个执行(run)要么只是初始状态 s_0,要么为如下形式: $\rho=s_0\xrightarrow{\sigma_1,\lambda_1}s_1\xrightarrow{\sigma_2,\lambda_2}\ldots\xrightarrow{\sigma_n,\lambda_n}s_n$, 其中,$n>0$,对于$1\leqslant i\leqslant n(s_{i-1},\sigma_i,s_i)\in\Delta$ 并且 $\lambda_i\in\mu(s_{i-1},\sigma_i,s_i)$.

例 1:如图 1 所示为一个实时自动机 $\mathcal{A}$,对于其 6 元组,状态(位置)集合为 $S=\{s_0,s_1,s_2\}$,有穷字母表为$\Sigma=\{a,b,c\}$,初始状态为 s_0,接收状态集合为 $F=\{s_2\}$,迁移集合为$\Delta=\{(s_0,a,s_1),(s_1,b,s_1),(s_1,c,s_2)\}$,对应的时间标签函数产生的时间标签为$\mu(\Delta)=\{[1,2],[1,3],[2,4]\}$.

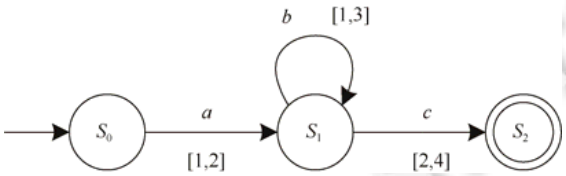

Fig.1 An example for real-time automaton

图 1 一个实时自动机示例

特别地,在不影响理解的情况下,本文之后不对状态和位置做出区分.本文讨论的实时自动机不涉及兹诺行为(non-Zeno)[19],即每一个可控的循环最少消耗时间为$\epsilon\in\mathbb{R}_{>0}$.

1.2 扩展线性时段不变式

扩展线性时段不变式是时段演算的一类重要子集,可以看作是在线性时段不变式(LDI)的基础上引入逻辑连接符($\neg,\wedge,\vee$)以及模态词切变(“;”)得到的.ELDI 与 LDI 相比具有更加复杂的语法结构,相应地,表达能力也更强.同时对其进行验证也变得更加困难,特别是模态词切变的引入,让 ELDI 的模型检验问题与 LDI 的模型检验问题有了根本区别.

扩展线性时段不变式在语法上包括 3 个组成部分:状态表示式 S、线性时段子式 $\mathcal{D}$、ELDI 子式ϕ.扩展线性时段不变式的定义如下.

定义 2(扩展线性时段不变式). 扩展线性时段不变式的结构定义如下.

- $S::=0|P|\neg S|S_1\vee S_2$;
- $\mathcal{D}::=\sum_{i\in\Omega}c_i\int S_i\leqslant c$;
- $\phi::=\mathcal{D}\mid\neg\phi\mid\phi_1\vee\phi_2\mid\phi_1;\phi_2$,

其中, $P\in\mathcal{P}$ 表示状态变量,c_i 和 c 为实数,Ω为一个有穷的下标集合.每一个扩展线性时段不变式都可以表示为如下形式:$b\leqslant\ell\leqslant e\Rightarrow\phi$,其中, $b\in\mathbb{R}_{\geqslant 0}, e\in\mathbb{R}_{\geqslant 0}\cup\{+\infty\}$.

特别地,本文中我们只关注观测时长上界 e 有界的情形,本文所指的有界模型检验指的也是此种情形.

上面给出了 ELDI 的语法介绍,其中包含模态词切变,本文统一记为“;”.对于模态词切变的解释为

$$\phi_1;\phi_2[t_1,t_2]\triangleq\exists m.(t_1\leqslant m\leqslant t_2)\wedge\phi_1[t_1,m]\wedge\phi_2[m,t_2],$$

如图 2 所示.

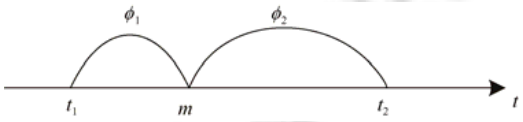

Fig.2 An interpretation for chop

图 2 对于模态词切变的解释

定义 3(ELDI 的语义解释 I_β). 给定一个实时自动机 $\mathcal{A}$ 和它的一个行为β,对于给定的命题集合 $\mathcal{P}$,实时自动机 $\mathcal{A}$ 的状态(位置)可能满足或者不满足集合 $\mathcal{P}$ 中的命题.那么在连续时间语义下,扩展线性时段不变式对于该行为的解释如下.

- 对于状态表达式:给定一个时刻 $t\in\mathbb{R}_{\geqslant 0}$,

  $I_\beta(0)(t)=0$ 并且 $I_\beta(1)(t)=1$;

  $I_\beta(P)(t)=\begin{cases}1,\\0,\end{cases}$ 如果 t 在观察区间内并且系统所处状态满足命题 P,则取 1;否则,取 0;

  $I_\beta(\neg S)(t)=1-I_\beta(S)(t)$;

  $I_\beta(S_1\vee S_2)(t)=\max\{I_\beta(S_1)(t),I_\beta(S_2)(t)\}$.
- 对于时段:给定一个观察区间$[t_1,t_2]$,其中, $t_1,t_2\in\mathbb{R}_{\geqslant 0}$ 并且 $t_1\leqslant t_2$,那么 $\int S$ 的解释为

  $I_\beta(\int S)([t_1,t_2])=\int_{t_1}^{t_2}I_\beta(S)(t)\mathrm{d}t$.
- 对于子式:给定一个观察区间$[t_1,t_2]$,一个 ELDI 子式ϕ的解释为

  $I_\beta,[t_1,t_2]\vDash\sum_{i\in\Omega}c_i\int S_i\leqslant c$ 当且仅当 $\sum_{i\in\Omega}c_iI_\beta(\int S_i)([t_1,t_2])\leqslant c$;

  $I_\beta,[t_1,t_2]\vDash\neg\phi$ 当且仅当 $I_\beta,[t_1,t_2]$ 不满足ϕ;

  $I_\beta,[t_1,t_2]\vDash\phi_1\vee\phi_2$ 当且仅当 $I_\beta,[t_1,t_2]\vDash\phi_1$ 或 $I_\beta,[t_1,t_2]\vDash\phi_2$;

  $I_\beta,[t_1,t_2]\vDash\phi_1;\phi_2$ 当且仅当存在 $t\in[t_1,t_2], I_\beta,[t_1,t]\vDash\phi_1$ 且 $I_\beta,[t,t_2]\vDash\phi_2$.

1.3 量词线性算术

量词线性算术(quantified linear real arithmetic,简称 QLRA)是一种一阶逻辑理论.本文涉及的量词线性算术表达式的语法如下.

定义 4(量词线性算术表达式语法)

$$\psi::=c_0+c_1x_1+\ldots+c_nx_n\lhd 0\mid\neg\psi\mid\psi_1\vee\psi_2\mid\forall x.\psi,$$

其中, $c_0,c_1,...,c_n \in \mathbb{R}, \lhd \in \{=,<\}$, 其他记号的解释与一阶逻辑和实数算术相同.

2 寻找满足观测时长约束的路径片段.

本节介绍给定一个实时自动机 $\mathcal{A}$ 和观测时长约束[*b*,*e*],求取所有满足时长约束的路径片段.由于本文讨论的时间语义是连续语义,对于一个时间区间[*t*,*t*′],其包含无穷个时刻点,那么对于从同一个位置出发具有相同时间长度的执行路径可能有无数条.因此,本文的基本思路是采用符号化的思想,求取满足观测时长约束的符号化路径片段.

文献[17]采用基于区域图(zone graph)的方法,需要先将自动机模型转化为区域图.求取区域图后,在开始搜索时需要引入一个额外时钟变量 t 并对区域图中的区域(zone)进行隐式变换,该时钟变量用来隐式记录路径片段的时间长度.但是求取区域图在最坏情形下会使得搜索空间呈单指数扩大.本文基于实时自动机模型,所采用的方法不需要像文献[17]那样求取区域图并进行处理,而是显式地刻画符号化路径片段的时间长度,这样大大降低了算法时间复杂度.

2.1 符号化路径片段的定义

下面对符号化路径片段及其时间长度给出显性刻画.

定义 5(符号化路径片段). 给定一个实时自动机 $\mathcal{A}$ 和它的某个执行 $\rho = s_0 \overset{\sigma_1,\lambda_1}{\to} s_1 \overset{\sigma_2,\lambda_2}{\to} ... \overset{\sigma_n,\lambda_n}{\to} s_n$, 那么对于这个执行的某个执行片段 $s_i \overset{\sigma_{i+1},\lambda_{i+1}}{\to} ... \overset{\sigma_j,\lambda_j}{\to} s_j \overset{\sigma_{j+1},\lambda_{j+1}}{\to}$, 其中, $0 \leqslant i < j+1 \leqslant n$, 其对应的符号化路径片段为 $\omega:(s_i,\mu(s_i,\sigma_{i+1},s_{i+1}))...(s_j,\mu(s_j,\sigma_{j+1},s_{j+1}))$.

可见,符号化路径片段记录了某个执行片段经过的状态(位置)以及系统在该位置可以停留的时限约束.特别地,对于执行片段的最后一个位置,在实时自动机 $\mathcal{A}$ 中没有迁移从该位置发生,那么可以认为系统可以在该位置停留的时限约束为[0,+∞].

定义 6(符号化路径片段的长度和时间长度). 给定一个符号化执行片段 $\omega:(s_i,\mu(s_i,\sigma_{i+1},s_{i+1}))...(s_j,\mu(s_j,\sigma_{j+1},s_{j+1}))$, 它的长度|$\omega$|是指其中离散位置的个数,|$\omega$|=*j*–*i*+1.它的时间长度||$\omega$||是一个区间,指的是系统在每个位置停留的时间长度区间加和 $\|\omega\| = \sum_i^j \mu(s_i,\sigma_{i+1},s_{i+1})$.

2.2 符号化路径片段的搜索算法

根据符号化路径片段的定义,给定一个实时自动机 $\mathcal{A}$ 和观测时长约束[*b*,*e*],找到所有满足观测时长约束的符号化路径片段即为找到所有 $\omega:(s_i,\mu(s_i,\sigma_{i+1},s_{i+1}))...(s_j,\mu(s_j,\sigma_{j+1},s_{j+1}))$, 其中, $\|\omega\| \cap [b,e] \neq \varnothing$.

寻找的基本思路是从实时自动机的每一个位置出发,利用深度优先搜索算法寻找.每到一个位置就计算一次当前这条符号化执行片段的时间长度||ω||,并判断 $\|\omega\| \cap [b,e] \neq \varnothing$ 是否成立.如果成立,则说明此条符号化执行路径包含某些满足时长约束的执行片段;如果不成立,则说明目前符号化执行片段的长度不满足观测时长约束,需要更深地搜索或者开始回溯其他分支.深度优先搜索过程依靠一个栈 *STK* 实现,具体过程参考表 1 中算法.

特别地,对于一条符号化执行片段的第 1 个位置,由于不知道开始观测时系统已在该位置停留了多长时间,因此需要对于其时间长度进行特殊处理,即修改在该位置上停留的时间长度的下界为 0,上界不改变.

该算法的输入是实时自动机 $\mathcal{A}$ 和观测时长约束[*b*,*e*],输出为所有满足观测时长约束的符号化路径片段的集合Θ.一开始,Θ为空,分别从实时自动机 $\mathcal{A}$ 中每一个状态(位置)开始寻找.起初,当前符号化执行片段ω为空(ϵ),将栈 *STK* 中栈顶的元素作为ω的下一个位置,判断当前ω的时间长度与约束[*b*,*e*]之间的关系(行 9~行 10,行 19).第 1 种情况,如果当前ω的时间长度区间与约束区间有交集,说明当前ω是一条满足观测时长约束的符号化路径片段,那么将当前ω复制到集合Θ(行 11).然后继续向前寻找,查看当前状态是否有后继状态,如果有,则将所有后继状态加入到栈中(行 12~行 14),如果没有,则说明需要回溯(行 15~行 18).第 2 种情况(第 1 种情况不满足时进入),如果当前ω的时间长度区间的最大值小于约束区间的下界 *b*,则不将ω放入集合Θ中,直接继续向前寻找(行

12~行 18),继续寻找过程与第 1 种情况处理相同.第 3 种情况(前两种情况都不满足时进入),如果当前ω的时间长度区间的最小值第 1 次超过了时长约束的上界 e(行 19),由于是第 1 次从没有超过上界变为超过上界,这时同样需要将该符号化执行片段加入集合Θ中;然后直接开始回溯,不再向前寻找.直到栈 *STK* 为空,说明以 s 为开始位置的满足观测时长约束的符号化执行片段已搜索完毕.由于以实时自动机 $\mathcal{A}$ 中所有的状态为起点寻找,因此,该算法可以找到所有满足时长约束的符号化执行片段.

Table 1 The algorithm for finding all symbolic path segments whose lengths are in $[b,e]$

表 1 寻找所有满足观测时长约束$[b,e]$的符号化路径片段的算法

算法 1. *Search*($\mathcal{A}$,$[b,e]$).
输入:$\mathcal{A}$; $[b,e]$;
输出:Θ.

```
1  begin
2      Θ := ϵ;
3      foreach s ∈ A.S do
4          ω := ϵ;
5          STK.push(s);
6          while STK ≠ ∅ do
7              current_state := STK.pop;
8              ω := ω ∘ current_state;
9              if ‖ω‖ ∧ [b,e) ≠ ∅ or ‖ω‖ ∧ (0,b) ≠ ∅ then
10                 if ‖ω‖ ∧ [b,e) ≠ ∅ then
11                     Θ := Θ ∪ {ω};
12                 if Post^σ(current_state) ≠ ∅ then
13                     foreach s' ∈ Post^σ(current_state) do
14                         STK.push(s');
15                 else
16                     ω := remove(ω, current_state);
17                     while Post^σ(laststate(ω)) have been popped out do
18                         ω := remove(ω, laststate(ω));
19             else if ‖ω‖ ∧ [e,+∞) ≠ ∅ then
20                 Θ := Θ ∪ {ω};
21                 ω := remove(ω, current_state);
22                 while Post^σ(laststate(ω)) have been popped out do
23                     ω := remove(ω, current_state);
24     return Θ;
```

2.3 搜索算法的正确性证明

对于算法 1 的正确性证明,即需要证明:(1) 算法会终止;(2) 通过其找到的集合Θ中任意一条符号化路径片段ω一定是一条真实的路径片段,并且时间长度与区间$[b,e]$相交;(3) 任意一条 $\mathcal{A}$ 能够产生的一条符号化路径片段ω,如果其时间长度与区间$[b,e]$相交,那么,它一定在集合Θ中.因此,我们证明如下定理.

定理 1(算法 1 正确性). 给定一个实时自动机 $\mathcal{A}$ 和观测时长约束$[b,e]$,算法 1 是正确的,即:

- 终止性(termination):算法会终止;
- 可靠性(soundness):如果ω是集合Θ中的一条符号化执行路径,那么,ω一定是 $\mathcal{A}$ 能够产生的一条符号化路径的一个片段,并且该符号化执行片段的时间长度与区间$[b,e]$相交;
- 完备性(completeness):如果ω是 $\mathcal{A}$ 能够产生的一条符号化路径的一个片段,并且片段的长度与区间$[b,e]$相交,那么,ω一定在集合Θ中.

证明:对于终止性来说,我们只需要证明第 6 行的 while 循环会终止,因为比较明显的是最外层的 foreach 和最里层的 foreach 及 while 循环均能够终止.给定搜索起始位置 s,从 s 开始的一个满足观测时长约束$[b,e]$的符号

化路径片段ω的长度为$|\omega|=N\times(1+\lceil e/\epsilon\rceil)$,其中,$N$ 为 $\mathcal{A}$ 中状态(位置)个数($|S|$),ϵ为假设的一个可控循环消耗的最小时间(不考虑兹诺行为).根据鸽笼原理,某个状态最多在ω中出现$1+\lceil e/\epsilon\rceil$;也就意味着$\omega$中最多有$\lceil e/\epsilon\rceil$个可控循环.因此,从 s 开始的所有满足观测时长约束$[b,e]$的符号化路径片段长度最大为$N\times(1+\lceil e/\epsilon\rceil)$.那么,找到所有满足观测时长约束的符号化路径片段的循环次数最多为$N^{N\times(1+\lceil e/\epsilon\rceil)}$.加入到栈中的状态数有上限,而且每次循环都会去掉栈中一个元素,因此栈一定会为空,即循环一定终止.

对于可靠性,假设ω是集合Θ中的一条符号化执行路径,显然,根据算法 1 找到的执行片段一定是自动机 $\mathcal{A}$ 可以产生的,并且根据第 9 行、第 10 行、第 19 行的判断条件,该符号化执行片段的长度与区间$[b,e]$相交.

对于完备性,假设 $\omega:(s_i,\mu(s_i,\sigma_{i+1},s_{i+1}))...(s_j,\mu(s_j,\sigma_{j+1},s_{j+1}))$ 是自动机 $\mathcal{A}$ 能够产生的一条符号化路径的一个片段,并且长度区间与约束区间$[b,e]$相交.那么,可以构造一个$\omega'\in\Theta$.首先,我们让ω'从 $(s_i,\mu(s_i,\sigma_{i+1},s_{i+1}))$ 开始,显然,算法 1 中第 3 行可以办到.由于ω是自动机 $\mathcal{A}$ 能够产生的一条符号化路径的一个片段,那么,ω中的第 2 个位置 $(s_{i+1},\mu(s_{i+1},\sigma_{i+2},s_{i+2}))$ 一定是 $(s_i,\mu(s_i,\sigma_{i+1},s_{i+1}))$ 的一个后继,并且 $(s_i,\mu(s_i,\sigma_{i+1},s_{i+1}))(s_{i+1},\mu(s_{i+1},\sigma_{i+2},s_{i+2}))$ 的时间长度的最大值一定小于 e,那么,根据第 9 行~第 18 行,ω'会等于 $(s_i,\mu(s_i,\sigma_{i+1},s_{i+1}))(s_{i+1},\mu(s_{i+1},\sigma_{i+2},s_{i+2}))$.以此推导,直到 $(s_j,\mu(s_j,\sigma_{j+1},s_{j+1}))$ 被加到ω'的最后.那么,ω'会在执行第 10 行、第 11 行或者执行第 19 行、第 20 行后被加入到Θ中,即$\omega'\in\Theta$. □

3 构造量词线性算术表达式

本节我们介绍如何将一条满足观测时长约束的路径片段转化为一个量词线性算术表达式.给定一个观测区间$[t,t']$,那么它一定被自动机的状态(位置)所占据,如图 3 所示,在该段时间区间上,系统依次处于状态(位置)$l_1,l_2,\ldots,l_n$(可以有重复状态,这里为了描述简洁,重新给出下标号).如果区间长度 $t'-t$ 在观测区间长度约束$[b,e]$内,即 $b\leqslant t'-t\leqslant e$,那么,其对应的路径片段即为一条满足观测时长约束的路径片段.

对于每一个位置 $l_i(1\leqslant i\leqslant n)$,我们引入一个变量 $\delta_i\in\mathbb{R}_{\geqslant 0}$,表示系统在该位置停留的时长.对于 ELDI 子式,如果含有模态词切变,例如式子$\phi_1;\phi_2;\ldots;\phi_r$,那么,每一个切变点必然位于某个位置 l_j 内.对于某个切变点 $chop_i$,我们引入一个变量 $\tau_i\in\mathbb{R}_{\geqslant 0}$.如图 3 所示,如果切变点 $chop_i$ 位于位置 l_j 内,那么,τ_i 表示从系统进入位置 l_j 到切变点 $chop_i$ 的时间长度.运用引入的这些变量,我们可以将一条满足观测时长约束的符号化路径片段表达为$(l_1,\delta_1)(l_2,\delta_2)\ldots(l_n,\delta_n)$.下面我们给出将其转化为一种量词线性算术表达式的方法.

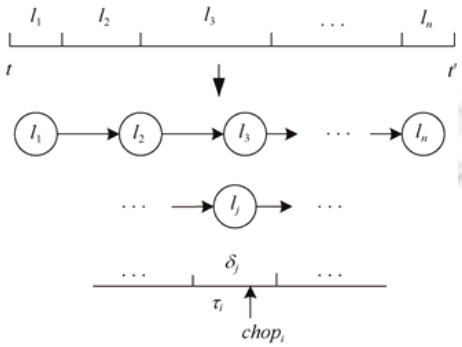

Fig.3 Interpretation for introduced variables δ,τ

图 3 对于引入变量δ,τ的解释

3.1 构造符号化路径片段的时限约束

首先,我们需要将这条符号化路径片段的时限约束表示出来,即刻画实时自动机产生这条符号化路径片段的所有时限约束.这些时限约束包括 3 种类型:非负约束、加和有界约束和δ约束.

定义 7(非负约束). 给定一个满足观测时长约束$[b,e]$的路径片段$(l_1,\delta_1)(l_2,\delta_2)\ldots(l_n,\delta_n)$,那么,系统在每个位置

的停留时长δ_i是一个非负实数:

$$\bigcap_{i=1}^{n}(\delta_i \geqslant 0).$$

定义 8(加和有界约束). 给定一个满足观测时长约束$[b,e]$的路径片段$(l_1,\delta_1)(l_2,\delta_2)...(l_n,\delta_n)$,那么,系统在所有位置停留的总时长满足观测时长约束:

$$b \leqslant \sum_{i=1}^{n}\delta_i \leqslant e.$$

加和有界约束的意义在于,第 2 节中求取的每个符号化路径片段的时间长度均是一个区间,只是与时长约束$[b,e]$有交集.那么,加和有界约束则进一步限定,无论每个δ_i如何取值,加和一定需要满足时长约束$[b,e]$,去除掉路径片段的时间长度在$[b,e]$约束之外时的δ_i取值情况.

定义 9(δ约束). 给定一个满足观测时长约束$[b,e]$的路径片段$(l_1,\delta_1)(l_2,\delta_2)\ldots(l_n,\delta_n)$,那么,系统在每个位置停留的时长$\delta_i$应满足实时自动机在相应位置停留的时限约束,即保证该路径片段确实为一条实时自动机能够产生的路径的一个片段.

对于实时自动机,δ约束求取的基本思路如下,分两种情况加以讨论.

第 1 种情况,如果路径片段只含有一个位置(l_1,δ_1),那么,获取实时自动机中所有以位置 l_1 作为源节点的迁移,可以用迁移上的约束求取系统在该位置停留的时限约束.由于不能够知晓开始观测时系统已经在该位置停留了多长时间,因此需要对迁移上的约束做出修改,即将约束下界修改为 0 即可.例如,如图 1 中,假设有一个满足时长约束的路径片段为(s_1,δ_1),有两条迁移$s_1 \xrightarrow{b,[1,3]} s_1$和$s_1 \xrightarrow{c,[2,4]} s_2$以 s_1 作为源节点,那么,对于某条执行路径,系统在 s_1 上停留的时限约束为[1,3]或[2,4].由于不知道开始观察时系统已在 s_1 上停留了多长时间,因此,观察到系统在 s_1 上停留的时限约束为[0,3]或[0,4].那么,对于这条路径片段来说,δ约束为 $0\leqslant\delta_1\leqslant 3$ 或 $0\leqslant\delta_1\leqslant 4$.

第 2 种情况,如果执行片段长度大于 1,假设路径片段为$(l_1,\delta_1)(l_2,\delta_2)\ldots(l_n,\delta_n)$,那么,对于路径片段中相邻的两个位置$(l_i,\delta_i)$和$(l_{i+1},\delta_{i+1})$,先找到 l_i 到 l_{i+1} 的所有迁移,将迁移上的时限约束作为对于(l_i,δ_i)的δ约束.同样地,对于(l_1,δ_1)的δ约束需要处理下界.

由于第 2 节查找满足观测时限的符号化执行路径时已采用该思想,因此具体实现时只需将第 2 节中对于该条符号化执行路径片段所求的每个位置时间长度区间变为相应的δ约束.表 2 中算法为生成 3 种时限约束的算法.该算法输入为某个符号化路径片段ω:$(l_1,\delta_1)(l_2,\delta_2)\ldots(l_n,\delta_n)$,输出为这个符号化路径片段所满足的时限约束$\Gamma$.该算法比较简单,行 2 到行 3 是对每个位置生成其需要满足的δ约束,运用函数$\delta_constraint()$求取,具体方法如前所述.第 4 行将 3 种类型的约束取交集,即为时限约束Γ.

Table 2 The algorithm for generating the timing constraints

表 2 生成时限约束的算法

算法 2. $Constarints(\omega)$.
输入:$\omega:(l_1,\delta_1),...,(l_n,\delta_n)$;
输出:Γ.
1 **begin**
2 **foreach** $i\in\{1,...,n\}$ **do**
3 $\Gamma_i := \delta\_constraint(l_i,\delta_i)$;
4 $\Gamma := \wedge_{i=1}^{n}(\Gamma_i \wedge \delta_i \geqslant 0) \wedge b \leqslant \sum_{i=1}^{n}\delta_i \leqslant e$;
5 **return** Γ;

3.2 构造语义不等式

为了能够用量词线性算术表达式表达每一个线性时段子式$\mathcal{D}$的语义信息,我们需要构造 3 种不等式:线性时段子式不等式、τ~δ不等式以及τ附加不等式.

定义 10(线性时段子式不等式). 给定一个 ELDI 子式ϕ,对于其中每一个线性时段子式$\mathcal{D}$,将$\mathcal{D}$中每一个状

态(位置)积分$\int l_i$用引入的变量δ和τ表示,得到的不等式$\mathcal{D}'$即为一个线性时段子式不等式.

定义 11(τ~δ不等式). 如果一个切变点$chop_i$落在位置l_j内,那么对应的变量τ_i和δ_j应满足的不等关系为$0\leqslant\tau_i\leqslant\delta_j$,称为一个$\tau$~$\delta$不等式.

定义 12(τ附加不等式). 如果存在两个相邻的切变点$chop_i$和$chop_{i+1}$位于同一个位置l_j内,如果$chop_{i+1}$在$chop_i$左边,那么对应的变量τ_i和τ_{i+1}应满足的不等关系为$0\leqslant\tau_{i+1}\leqslant\tau_i$,称为一个$\tau$附加不等式.

给定一个符号化路径片段$\omega:(l_1,\delta_1)(l_2,\delta_2)\ldots(l_n,\delta_n)$和 EDLI 子式$\phi$,将其翻译为一组不等式的思路是,根据EDLI 子式ϕ的逻辑结构,分解到每个线性时段子式,将其翻译为线性时段子式不等式、τ~δ不等式、τ附加不等式;然后再用相应的逻辑结构组合起来.

- 当ϕ是一个线性时段子式,即$\phi=\mathcal{D}=\sum_{i\in\Omega}c_i\int S_i\leqslant c$时,假设其含有$k$个状态积分式子$\int S_1,\ldots,\int S_k$,对于$\omega:(l_1,\delta_1)(l_2,\delta_2)\ldots(l_n,\delta_n)$,$\int S_i=\sum_{S_i=l_j}\delta_j$,即将$\omega$中所有与$S_i$相同的位置$l_j$对应的停留时长$\delta_j$求和,用该加和表达式代替$\mathcal{D}$中$\int S_i$;如果没有与$S_i$相同的$l_j$,则用 0 代替$\int S_i$.因此得到的线性时段子式不等式为$\mathcal{D}'=\mathcal{D}\left[\sum_{S_i=l_j}\delta_j/\int S_i\right]$,表示将线性时段子式中的状态积分用相应的变量$\delta$的加和表达式替换.
- 当$\phi=\neg\phi,\phi_1\vee\phi_2,\phi_1\wedge\phi_2$时,我们将它们递归分解为线性时段子式.
- 当$\phi=\phi_1;\phi_2$时,将ω分为两个子片段ω_1和ω_2,分别对应到公式ϕ_1和ϕ_2上,从而将其转化为与关系($\wedge$)上的两个子问题.

3.3 构造量词线性算术表达式

与文献[17]相比,本文构造的时限约束和不等式更加明确,同时,构造算法中不需要引入额外的δ变量,即在遇到切变时,通过已有δ变量和τ变量的算术运算形式来表示片段中被切割的位置分属两个子片段的时间长度$(\tau,\delta_i-\tau)$.由于求解量词线性算术表达式的复杂度与变量个数相关,因此,本文采用的方法减少了变量个数,有效地缩短了求解所需时间.具体算法参考表 3 中算法.

Table 3 The algorithm for generating a QLRA formula

表 3 生成一个 QLRA 公式的算法

算法 3. $QLRA(\phi,\omega)$.

输入:$\omega=(l_1,\delta_1),\ldots,(l_n,\delta_n);\phi$;

输出:ξ.

```
1  begin
2      case φ := 𝒟 do
3          ξ := ( 𝒟[ Σ_{S_i=l_j} δ_j / S_1, ..., Σ_{S_k=l_j} δ_j / ∫S_k ] );
4      case φ := ¬φ_1 do
5          ξ := ¬QLRA(φ_1, ω);
6      case φ := φ_1 ∧ φ_2 do
7          ξ := QLRA(φ_1, ω) ∧ QLRA(φ_2, ω);
8      case φ := φ_1 ∨ φ_2 do
9          ξ := QLRA(φ_1, ω) ∨ QLRA(φ_2, ω);
10     case φ := φ_1; φ_2 do
11         ξ := ∨_{i=0}^{n+1} (QLRA(φ_1, ((l_1, δ_1), ..., (l_i, τ))) ∧ QLRA(φ_2, ((l_i, δ_i − τ), ..., (l_n, δ_n))) ∧ 0 ≤ τ ≤ δ_i);
12         Q := Q, ∃τ;
13     return ξ;
```

表 3 中生成一种量词线性算术表达式的算法,其输入为一个扩展线性时段表达式ϕ和一个符号化路径片段$\omega:(l_1,\delta_1)(l_2,\delta_2)\ldots(l_n,\delta_n)$;输出为一个 ELDI 子式表达式$\xi$.整个算法是一种递归算法,根据扩展线性时段表达式$\phi$的逻辑结构进行递归.当递归到某个线性时段子式时,运用引入的变量的加和形式替换掉式子中的状态积分,得到

线性时段子式不等式(行 3).当碰到逻辑连接词时,根据逻辑连接词的语义,将问题分解为子问题,递归地调用本算法.每当碰到一个切变时,引入一个变量τ,将ω分为两部分(行 11),同时需要添加相应的$\tau\sim\delta$不等式(行 11 中最后一个不等式).当遇到两个切变点落在同一个状态时,$\tau\sim\delta$不等式会变成一个τ附加不等式.例如,当ϕ_1仍然含有切变,且切变点落在ω_1:$(l_1,\delta_1)\ldots(l_i,\tau)$的最后一个位置时,新引入的变量$\tau'$需要满足的$\tau\sim\delta$不等式为 $0\leqslant\tau'\leqslant\tau$,那么,实际上是一个$\tau$附加不等式.当片段有 n 个位置时,当前切变点的落位会有 n+2 种情况,即在这 n 个位置之前、在这 n 个位置中的某个、在这 n 个位置之后.集合 Q 记录引入的(多个)τ变量.最后,我们将引入的δ变量、算法 2 求出的时限约束表达式Γ和得到的表达式ξ构造一个 QLRA 表达式 $\pi=\forall\delta_1,\ldots,\delta_n,Q.\Gamma\Rightarrow\xi$.

3.4 构造算法的正确性证明

同样地,我们需要证明构造算法的正确性,正确性证明主要依赖递归正确性.

定理 2(算法 3 的正确性). 给定一个满足观测时长约束$[b,e]$的符号化路径片段ω和 ELDI 公式ϕ,算法 3 是正确的,即:

- 终止性:算法会终止;
- 可靠性:如果 $\omega\vDash\phi$, 那么,$QLRA(\phi,\omega)$得到的 QLRA 表达式是满足的(satisfiable);
- 完备性:如果 $QLRA(\phi,\omega)$得到的 QLRA 表达式是满足的,则 $\omega\vDash\phi$.

证明:对于终止性来说,由于算法是根据 ELDI 公式ϕ的结构进行递归的,因此可以看出算法是终止的.对于可靠性和完备性,下面分析每种递归结构.

- 当ϕ是一个线性时段子式,即 $\phi=\mathcal{D}=\sum_{i\in\Omega}c_i\int S_i\leqslant c$ 时, $I_\omega,[t_1,t_2]\vDash b\leqslant\ell\leqslant e\Rightarrow\sum_{i\in\Omega}c_i\int S_i\leqslant c$ 当且仅当 $b\leqslant t_2-t_1\leqslant e\Rightarrow\sum_{i\in\Omega}c_iI_\omega(\int S_i)([t_1,t_2])\leqslant c$, 其中,$t_1,t_2$分别为路径片段的起始时刻和终止时刻.显然,当按照语义替换后,仍然有这一关系.因此,如果 $\omega\vDash\phi$, 那么,$QLRA(\phi,\omega)$得到的 QLRA 表达式是满足的;如果 $QLRA(\phi,\omega)$得到的 QLRA 表达式是满足的,则 $\omega\vDash\phi$.
- 当$\phi=\neg\phi_1$ 时,对于可靠性,假设 $QLRA(\neg\phi,\omega)$不满足,那么,根据算法 3 有$\neg QLRA(\phi_1,\omega)$不满足,也就意味着 $QLRA(\phi_1,\omega)$正确.根据归纳假设条件,我们有 $\omega\vDash\phi$, 可以得到ω不满足$\neg\phi_1$.对于完备性,假设 $QLRA(\neg\phi_1,\omega)$正确,根据算法 3 有 $QLRA(\phi_1,\omega)$不满足.根据归纳假设,我们有ω不满足$\neg\phi_1$,因此得到 $\omega\vDash\phi$.
- 当$\phi=\phi_1\vee\phi_2$时,对于可靠性,假设 $QLRA(\phi_1\vee\phi_2,\omega)$不满足,根据算法 3 有 $QLRA(\phi_1,\omega)\vee QLRA(\phi_2,\omega)$不满足.根据归纳假设,我们有$\omega$不满足$\phi_1$并且$\omega$不满足$\phi_2$,因此,$\omega$不满足$\phi_1\vee\phi_2$.对于完备性,假设$QLRA(\phi_1\vee\phi_2,\omega)$正确,根据算法 3 有$\neg QLRA(\phi_1,\omega)$或者$\neg QLRA(\phi_2,\omega)$不满足.根据归纳假设,我们有$\omega$不满足$\phi_1$ 或者ω不满足ϕ_2,因此得到 $\omega\vDash\phi_1\vee\phi_2$.
- 当$\phi=\phi_1\wedge\phi_2$时,证明与$\phi=\phi_1\vee\phi_2$类似.
- 当$\phi=\phi_1;\phi_2$时,显然 $\omega\vDash\phi_1;\phi_2$ 当且仅当ω被分为两个部分ω_1和ω_2且 $\omega_1\vDash\phi_1\wedge\omega_2\vDash\phi_2$. 根据算法 3,我们选取了所有可能的切变点位置,并按语义分解为两个子问题 $\omega_1\vDash\phi_1$ 和 $\omega_2\vDash\phi_2$. 因此,根据归纳假设,我们得出:如果 $\omega\vDash\phi_1;\phi_2$, 那么,$QLRA(\phi_1;\phi_2,\omega)$得到的 QLRA 表达式是满足的;如果 $QLRA(\phi_1;\phi_2,\omega)$得到的 QLRA 表达式是满足的,则 $\omega\vDash\phi_1;\phi_2$. □

3.5 构建量词线性算术表达式举例

下面举例说明如何构建一个量词线性算术表达式.

例 2:我们考虑图 1 所示的实时自动机以及 ELDI 公式: $3\leqslant\ell\leqslant4\Rightarrow\int s_0+\int s_1\leqslant1;2\int s_1-\int s_2\leqslant2$, 令 $\mathcal{D}_1=\int s_0+\int s_1\leqslant1,\mathcal{D}_2=2\int s_1-\int s_2\leqslant2$. 现有一条满足观测时长约束的符号化路径片段$\omega$:$(s_0,\delta_0)(s_1,\delta_1)(s_2,\delta_2)$,可见我们引入了 3 个$\delta$变量.由于有一个切变,因此我们引入一个$\tau$变量$\tau_1$.

首先,我们生成符号化路径片段的所有时限约束.对于δ约束,由于(s_0,δ_0)是第 1 个位置,后继位置为(s_1,δ_1),因此,δ_0应先取迁移(s_0,a,s_1)的时限约束,即 $1\leqslant\delta_0\leqslant2$.由于$(s_0,\delta_0)$是第 1 个位置,开始观测时并不知道系统在 s_0 已停留多长时间,因此,将δ_0 的下界改为 0,得到δ_0 的最后约束为 $0\leqslant\delta_0\leqslant2$.用同样的方法得到$(s_1,\delta_1)$的时限约束为 $2\leqslant\delta_1\leqslant4$,$(s_2,\delta_2)$的时限约束为$\delta_2\geqslant0$.非负约束为 $(\delta_0\geqslant0\wedge\delta_1\geqslant0\wedge\delta_2\geqslant0)$, 加和有界约束为 $3\leqslant\delta_0+\delta_1+\delta_2\leqslant4$. 所有时限约束为 $3\leqslant\delta_0+\delta_1+\delta_2\leqslant4\wedge(0\leqslant\delta_0\leqslant2\wedge2\leqslant\delta_1\leqslant4\wedge\delta_2\geqslant0)\wedge(\delta_0\geqslant0\wedge\delta_1\geqslant0\wedge\delta_2\geqslant0)$.

然后,我们根据算法 3 来构造不等式和 QLRA 表达式.由于当前 ELDI 公式中是一个切变形式,那么会匹配到第 11 行,并有 5 种可能情况:(1) 当切变点落在 s_0 左侧时,ω_1 为空,ω_2 为$(s_0,\delta_0)(s_1,\delta_1)(s_2,\delta_2)$,那么,$\mathcal{D}_1=0+0\leqslant 1$, $\mathcal{D}_2=2\delta_1-\delta_2\leqslant 2$,即得到线性时段子式的不等式 $0+0\leqslant 1\wedge 2\delta_1-\delta_2\leqslant 2$.(2) 当切变点落在 s_0 时,ω_1 为(s_0,τ_1),ω_2 为$(s_0,\delta_0-\tau_1)(s_1,\delta_1)(s_2,\delta_2)$,那么,$\mathcal{D}_1=\tau_1+0\leqslant 1$,$\mathcal{D}_2=2\delta_1-\delta_2\leqslant 2$,即得到线性时段子式的不等式 $\tau_1+0\leqslant 1\wedge 2\delta_1-\delta_2\leqslant 2\wedge 0\leqslant\tau_1\leqslant\delta_0$.(3) 当切变点落在 s_1 时,ω_1 为$(s_0,\delta_0)(s_1-\tau_1)$,$\omega_2$ 为$(s_1,\delta_1-\tau_1)(s_2,\delta_2)$,那么,$\mathcal{D}_1=\delta_0+\tau_1\leqslant 1$,$\mathcal{D}_2=2(\delta_1-\tau_1)-\delta_2\leqslant 2$,即得到线性时段子式的不等式 $\delta_0+\tau_1\leqslant 1\wedge 2(\delta_1-\tau_1)-\delta_2\leqslant 2\wedge 0\leqslant\tau_1\leqslant\delta_1$.(4) 当切变点落在 s_2 时,用同样的方法得到线性时段子式的不等式为 $\delta_0+\delta_1\leqslant 1\wedge 0-(\delta_2-\tau_1)\leqslant 2\wedge 0\leqslant\tau_1\leqslant\delta_2$.(5) 当切变点落在 s_2 的右侧时,用上述方法得到线性时段子式的不等式为 $\delta_0+\delta_1\leqslant 1\wedge 0-0\leqslant 2$.因此,根据算法 3 构造的 QLRA 表达式为

$$\pi=\forall\delta_0,\forall\delta_1,\forall\delta_2,\exists\tau_1.3\leqslant\delta_0+\delta_1+\delta_2\leqslant 4\wedge(0\leqslant\delta_0\leqslant 2\wedge 2\leqslant\delta_1\leqslant 4\wedge\delta_2\geqslant 0)\wedge(\delta_0\geqslant 0\wedge\delta_1\geqslant 0\wedge\delta_2\geqslant 0)$$
$$\Rightarrow(0+0\leqslant 1\wedge 2\delta_1-\delta_2\leqslant 2)\vee(\tau_1+0\leqslant 1\wedge 2\delta_1-\delta_2\leqslant 2\wedge 0\leqslant\tau_1\leqslant\delta_0)\vee(\delta_0+\tau_1\leqslant 1\wedge 2(\delta_1-\tau_1)-\delta_2\leqslant 2\wedge 0\leqslant\tau_1\leqslant\delta_1)$$
$$\vee(\delta_0+\delta_1\leqslant 1\wedge 0-(\delta_2-\tau_1)\leqslant 2\wedge 0\leqslant\tau_1\leqslant\delta_2)\vee(\delta_0+\delta_1\leqslant 1\wedge 0-0\leqslant 2).$$

4 求解 QLRA 表达式及判定定理

根据定理 1 和定理 2,给定一个实时自动机 $\mathcal{A}$ 和一个观测时长约束有上界的 ELDI 公式Φ,我们可以将模型检验问题 $\mathcal{A}\vDash\Phi$ 转化为 QLRA 表达式的正确性(validity)问题.

定理 3. 给定一个实时自动机 $\mathcal{A}$ 和一个观测时长约束有上界的 ELDI 公式Φ, $\mathcal{A}\vDash\Phi$ 当且仅当 $\bigcap_{\omega\in Search(\mathcal{A},[b,e])}QLRA(\Phi,\omega)$ 是正确的(valid).

定理 3 可以从定理 1 和定理 2 得出,这里不再证明.根据 Tarski 理论,量词线性算术表达式的可满足性和正确性问题是可以判定的[20],因此有下面的结论.

定理 4. 给定一个实时自动机 $\mathcal{A}$ 和一个观测时长约束有上界的 ELDI 公式Φ, $\mathcal{A},[b,e]\vDash\Phi$ 是可判定的.

QLRA 表示式可以运用量词消去技术求解,量词消去技术基于柱形代数分解理论(cylindrical algebraic decomposition,简称 CAD)[21],该方法的最坏复杂度与变量规模呈二阶指数关系,但是,由于 QLRA 表达式中均为线性不等式,因此,求解比较迅速.目前,量词消去技术在许多求解器中都有实现,例如,REDLOG、QEPCAD[22]、Z3 等.

5 工具实现与实验

目前,我们在 Linux 平台上实现了一个原型工具,如图 4 所示为工具的架构图.工具接收两个输入:一是,实时自动机($\mathcal{A}$)模型文件,格式参考 UPPAAL 模型文件格式;二是,扩展线性时段不变式(ϕ),存放在一个文本文件中.工具会接收这两个输入,首先寻找所有满足观测时长约束 ℓ 的符号化路径片段,并得到其集合Θ,然后构造 QLRA 表达式,生成所有 QLRA 表达式的集合Π,并保存在符合 REDLOG 输入格式的文本文件中;然后将该文件传递给量词消去工具 REDLOG,REDLOG 会依次求解 QLRA 表达式π_i,并将每个表达式的结果(true or false)记录在一个文本文件中.

在实验部分,我们讨论文献[14,16]中给出的一个实验.一个实时自动机 $\mathcal{A}$ 由 N 个简单的实时自动机 M 迭代组成.M_i 如图 5 所示,由 4 个状态组成 A_i,B_i,C_i,D_i,系统在每个状态上停留一个时间单位.特别地,对于 $1\leqslant i\leqslant N$,D_i 有一条迁移前往 A_{i+1};对于 $1\leqslant j\leqslant i\leqslant N$,$D_i$ 有一条迁移前往 A_j.文献[14,16]在离散时间语义下进行了实验,本文则在标准连续时间语义下进行了实验,迭代次数 N 从 3 到 6.表 4 是我们的实验结果,t_{qlra} 表示生成所有 QLRA 表达式所用时间,t_{qe} 表示运用量词消去工具求解所有 QLRA 表达式所用时间,t_{total} 表示验证的总时间,而 t'_{total} 表示采用文献[17]中算法的总时间,时间单位均为 s.我们同样对含有不同逻辑运算符和切变的 ELDI 式子进行了实验,工具和实验可以参考 https://github.com/Leslieaj/VCELDI.

由于本文采用标准连续时间语义,而文献[16]采用离散时间语义,所以模型检验的总时间比文献[16]中相同问题所用时间要长.这是由于采用方法、基于的模型以及讨论的时间语义不同所致,我们的优势是可以解决连

续时间语义下的模型检验问题.与文献[17]中算法的实验结果相比,我们的验证方法所用时间明显低于文献[17]验证所需时间,可见本文方法有效地降低了验证算法的复杂度,提高了实际验证速度.

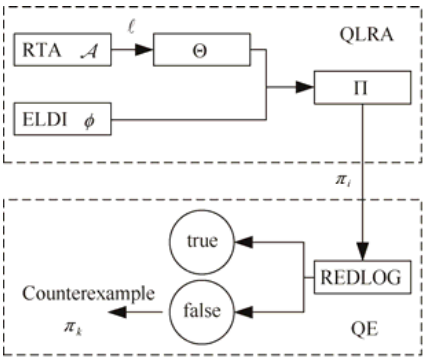

Fig.4 The tool structure

图 4 工具架构图

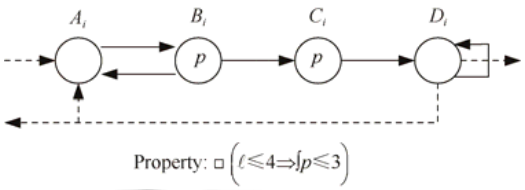

Fig.5 The model and property of the experiment

图 5 实验的模型与性质

Table 4 The results of the experiment

表 4 实验结果

N	t_{qlar}	t_{qe}	t_{total}	t'_{total}
3	0.7	0.7	1.4	1.8
4	1.1	1.0	2.1	3.1
5	1.7	1.5	3.2	4.5
6	2.4	1.9	4.3	6.1

6 总 结

本文讨论了在连续时间语义下,对于一个观测时长约束有上界的扩展线性时段不变式(ELDI)在实时自动机上的模型检验问题.本文证明了该问题是可以判定的,并且给出了判定方法.首先,我们采用符号化的思想,运用深度优先搜索得到所有满足观测时长约束的符号化路径片段;然后,将每一个符号化路径片段转化为一个量词线性算术表达式;最后,运用量词消去工具 REDLOG 求解这些量词线性算术表达式.

时间复杂度方面,对于第 1 步,我们可以看到算法 1 与自动机中状态(位置)数目呈一阶指数关系.第 2 步中,生成 QLRA 表达式的算法复杂度是多项式复杂度.最后一步运用量词消去工具求解,量词消去工具的最坏时间复杂度与变量个数呈二阶指数关系.但是,根据前文分析,QLRA 表示式中均为线性不等式,所以求解比较快,近似为多项式和一阶指数,这一点可以从实验 t_{qlar} 和 t_{qe} 的比较中看出.所以,对于时间复杂度来说,与实时自动机的规模和扩展线性时段不变式的切变个数呈一阶指数关系,最坏是二阶指数关系.而文献[17]中的验证算法复杂度一般为二阶指数,最坏情况为三阶指数.与文献[17]相比,复杂度和运行速度方面的优化有以下两点:第一,缩短了搜索空间,降低了搜索算法的复杂度,并减少了所产生的 QLRA 公式数量;第二,对于单个 QLRA 表达式,减少了量词的引入个数.从而,整体上降低了验证算法的复杂度,并加快了量词消去工具求解时的实际运行速度.

References:

[1] Zhou CC, Hoare CAR, Ravn AP. A calculus of durations. Information Processing Letters, 1991,40(5):269–276.

[2] Li XS, Zhou CC. Duration calculi: An overview. Chinese Journal of Computers, 1994,(11):842–851 (in Chinese with English abstract).

[3] Halpern JY, Manna Z, Moszkowski BC. A hardware semantics based on temporal intervals. In: Proc. of the 10th Int'l Colloquium on Automata, Languages, and Programming (ICALP). Berlin: Springer-Verlag, 1983. 278–291. [doi: 10.1007/BFb0036915]

[4] Meyer R, Faber J, Hoenicke J, Rybalchenko A. Model checking duration calculus: A practical approach. Formal Aspects of Computing, 2008,20(4):481–505. [doi: 10.1007/s00165-008-0082-7]

[5] Zhou CC, Hansen MR. Duration Calculus: A Formal Approach to Real-Time Systems. Berlin, Heidelberg: Springer-Verlag, 2004. [doi: 10.1007/978-3- 662-06784-0]

[6] Zhou CC, Hansen MR, Sestoft P. Decidability and undecidability results for duration calculus. In: Proc. of the 10th Annual Symp. on Theoretical Aspects of Computer Science (STACS). Berlin, Heidelberg: Springer-Verlag, 1993. 58–68. [doi: 10.1007/3-540-56503-5_8]

[7] Zhou CC, Zhang J, Yang L, Li XS. Linear duration invariants. In: Proc. of the 3rd Int'l Symp. on Formal Techniques in Real-Time and Fault-Tolerant Systems (FTRTFT). Berlin, Heidelberg: Springer-Verlag, 1994. 86–109. [doi: 10.1007/3-540-58468-4_161]

[8] Braberman VA, Huang DV. On checking timed automata for linear duration invariants. In: Proc. of the 19th IEEE Real-Time Systems Symp. (RTSS). Washington: IEEE Computer Society, 1998. 264–273. [doi: 10.1109/REAL.1998.739752]

[9] Li XD, Huang DV. Checking linear duration invariants by linear programming. In: Proc. of the Concurrency and Parallelism, Programming, Networking, and Security: The 2nd Asian Computing Science Conf. (ASIAN). Berlin, Heidelberg: Springer-Verlag, 1996. 321–332. [doi: 10.1007/BFb0027804]

[10] Li XD, Huang DV, Zheng T. Checking hybrid automata for linear duration invariants. In: Proc. of the Advances in Computing Science: The 3rd Asian Computing Science Conf. (ASIAN). Berlin, Heidelberg: Springer-Verlag, 1997. 166–180. [doi: 10.1007/3-540-63875-X_51]

[11] Thai P, Huang DV. Verifying linear duration constraints of timed automata. In: Proc. of the 1st Int'l Colloquium on Theoretical Aspects of Computing (ICTAC). Berlin, Heidelberg: Springer-Verlag, 2004. 295–309. [doi: 10.1007/978-3-540-31862-0_22]

[12] Zhang MM, Huang DV, Liu ZM. Verification of linear duration invariants by model checking CTL properties. In: Proc. of the 5th Int'l Colloquium on Theoretical Aspects of Computing (ICTAC). Berlin, Heidelberg: Springer-Verlag, 2008. 395–409. [doi: 10.1007/978-3-540-85762-4_27]

[13] Zhang MM, Liu ZM, Zhan NJ. Model checking linear duration invariants of networks of automata. In: Proc. of the 3rd Int'l IPM Conf. on Fundamentals of Software Engineering (FSEN). Berlin, Heidelberg: Springer-Verlag, 2009. 244–259. [doi: 10.1007/978-3-642-11623-0_14]

[14] Fränzle M, Hansen MR. Efficient model checking for duration calculus based on branching-time approximations. In: Proc. of the 6th IEEE Int'l Conf. on Software Engineering and Formal Methods (SEFM). Washington: IEEE Computer Society, 2008. 63–72. [doi: 10.1109/SEFM.2008.26]

[15] Fränzle M, Hansen MR. Efficient model checking for duration calculus. Int'l Journal of Software and Informatics, 2009,3(2-3): 171–196.

[16] Zu Q, Zhang MM, Zhu JQ, Zhan NJ. Bounded model-checking of discrete duration calculus. In: Proc. of the 16th Int'l Conf. on Hybrid Systems: Computation and Control (HSCC). New York: ACM, 2013. 213–222. [doi: 10.1145/2461328.2461362]

[17] An J, Zhan NJ, Li XS, Zhang MM, Wang Y. Model checking bounded continuous-time extended linear duration invariants. In: Proc. of the 21st Int'l Conf. on Hybrid Systems: Computation and Control (HSCC). New York: ACM, 2018. 81–90. [doi: 10.1145/3178126.3178147]

[18] Dima C. Real-time automata. Journal of Automata, Languages and Combinatorics, 2001,6(1):3–24. [doi: 10.25596/jalc-2001-003]

[19] Alur R, Dill DL. A theory of timed automata. Theoretical Computer Science, 1994,126(2):183–235.

[20] Tarski A. A Decision Method for Elementary Algebra and Geometry. Berkeley: University of California Press, 1951.

[21] Collins G. Hauptvortrag: Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In: Proc. of the 2nd GI Conf. on Automata Theory and Formal Languages. Berlin, Heidelberg: Springer-Verlag, 1975. 134–183. [doi: 10.1007/3-540-07407-4_17]

[22] Brown CW. QEPCAD B: A program for computing with semi-algebraic sets using CADs. ACM SIGSAM Bulletin, 2003,37(4): 97–108. [doi: 10.1145/968708.968710]

附中文参考文献:

[2] 李晓山,周巢尘.时段演算综述.计算机学报,1994,(11):842–851.

安杰(1990－),男,湖北京山人,博士,主要研究领域为形式化方法,混成系统,模型检测.

张苗苗(1971－),女,博士,研究员,博士生导师,CCF 专业会员,主要研究领域为形式化方法.