软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2011,22(4):640−658 [doi: 10.3724/SP.J.1001.2011.03766]

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

BACH:线性混成系统有界可达性模型检验工具*

卜 磊[1,2+], 李 游[1,2], 王林章[1,2], 李宣东[1,2]

[1](南京大学 计算机软件新技术国家重点实验室,江苏 南京 210093)

[2](南京大学 计算机科学与技术系,江苏 南京 210093)

BACH: A Toolset for Bounded Reachability Analysis of Linear Hybrid Systems

BU Lei[1,2+], LI You[1,2], WANG Lin-Zhang[1,2], LI Xuan-Dong[1,2]

[1](State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China)

[2](Department of Computer Science and Technology, Nanjing University, Nanjing 210093, China)

+ Corresponding author: E-mail: bulei@nju.edu.cn

Bu L, Li Y, Wang LZ, Li XD. BACH: A toolset for bounded reachability analysis of linear hybrid systems. *Journal of Software*, 2011,22(4):640−658. http://www.jos.org.cn/1000-9825/3766.htm

Abstract: The model-checking problem for hybrid systems is very difficult to resolve. Even for a relatively simple class of hybrid systems, the class of linear hybrid automata, the most common problem of reachability is unsolvable. Existing techniques for the reachability analysis of linear hybrid automata do not scale well to problem sizes of practical interest. Instead of developing a tool to perform a reachability checking of the complete state space of linear hybrid automata, a prototype toolset BACH (bounded reachability checker) is presented to perform path-oriented reachability checking and bounded reachability checking of the linear hybrid automata and the compositional linear hybrid systems, where the length of the path being checked can be made very large, and the size of the system can be made large enough to handle problems of practical interest. The experiment data shows that BACH has good performance and scalability and supports the fact that BACH can become a powerful assistant for design engineers in the reachability analysis of linear hybrid automata.

Key words: linear hybrid system; linear hybrid automata; bounded reachability analysis; linear programming

摘 要: 混成自动机的模型检验问题非常困难,即使是其中相对简单的一个子类——线性混成自动机,它的可达性问题仍然是不可判定的.现有的相关工具大都使用多面体计算来判定线性混成自动机状态空间的可达集,复杂度高、效率低,无法解决实际应用规模的问题.描述了一个面向线性混成系统有界可达性模型检验工具——BACH(bounded reachability checker),该工具能够沿指定路径(组)对单个线性混成自动机、多个线性混成自动机的组合进行可达性检验,并且在此基础上结合路径遍历技术完成对所有路径的有界可达性检验.实验数据显示,BACH 不仅在面向路径可达性检验方面性能优异,可以适用于足够长度的路径,而且在针对所有路径的有界可达性检验时,BACH 可以解决的问题规模也远远超过同类工具,已接近工业界应用的要求.

关键词: 线性混成系统;线性混成自动机;有界可达性检验;线性规划

* 基金项目: 国家自然科学基金(90818022, 61021062); 国家重点基础研究发展计划(973)(2009CB320702); 国家高技术研究发展计划(863)(2009AA01Z148, 2007AA010302)

收稿时间: 2009-05-12; 修改时间: 2009-08-12; 定稿时间: 2009-10-10

中图法分类号: TP311 文献标识码: A

混成系统(hybrid system)是一类同时具有离散和连续行为特征的复杂系统.在现实生活中,特别是航天、军工、机械制造等嵌入式相关领域,混成系统均以核心控制器的形式大量存在,并发挥着至关重要的作用.因此,该系统的正确性验证就有着特别重要的现实意义.当前,相关科研工作者主要采用混成自动机(hybrid automata)来为混成系统建模.一个混成自动机的运行既包含状态的离散变化,又包含状态的连续变化,因此,相应的模型检验问题十分困难.比如,即使是混成自动机的一个相对简单的子类——线性混成自动机(linear hybrid automata)[1],它的可达性问题也已被证明是不可判定的[1−4].

现有的混成自动机模型检验相关技术与工具,如 HyTech[5],PHAVer[6]等,大都使用多面体计算来判定系统相应的可达状态集,但该方法的指数级复杂度大大限制了其所能解决问题的规模,只能适用于小规模系统.其他研究中,文献[7]提出了一种可以使 HyTech[5]所运行的符号化进程在初始矩阵自动机上正常结束的符号化方法;文献[3]也提出了一种自动化构造混成自动机可达性区域的方法,但该方法却并不能保证终止.另外,文献[8]提出了一种利用线性规划技术对实时系统进行检查的方法,在文中,对实数时间自动机线性时段性质的可满足性进行了分析,并且扩展此方法应用于对线性混成系统[9,10]进行检查,但这一类方法仅适用于满足一定条件的线性混成自动机.上述工作均为研究工作者对线性混成自动机完整状态空间验证的一些尝试,但是效果均颇为有限,可以解决的问题规模很小或者可适用的自动机受到相当大的限制,因而与实际应用需求尚有很大的距离.

近年来,作为基于 BDD(binary decision diagram)的符号化模型检验[11]的一种补充方法,有界模型检验(bounded model checking,简称 BMC)技术被提出并得到了广泛的应用.其基本思想是,将模型行为步数通过整数 k 来限制,将系统 k 步内行为采用布尔约束编码,然后利用 SAT(Boolean satisfiability)[12]方法来寻找此布尔约束集的可行解,从而判定系统在 k 步内行为是否有不满足规约的情况.在文献[13]中发现,虽然 BMC 方法因为只能验证有限步数内系统性质而缺失了一定的完整性,但是通过限制步数,它在发现错误上的能力超越了传统模型检验方法.即当模型规模超过经典模型检验方法可检验范围的情况下,通过限定被检验模型状态空间范围的方法来使用有界模型检验技术高效地发现限定范围内系统的问题,而这也正是 BMC 方法被广泛认可的亮点所在.

BMC 思想同样被扩展到线性混成自动机的可达性检验工作中,特别是在 SMT(satisfiability modulo theories)思想被提出之后,作为一种将 SAT 技术从纯布尔值域向其他多种混合值域的扩展,SMT 相关的研究者对时间自动机与线性混成自动机的有界可达性问题进行了大量的实验与分析,如文献[14–16]等.这些研究者们通过 SMT 约束求解器及其可调用的底层域相关约束求解器来对相应约束集求解,从而寻找给定步长内可达相应目标的路径.但是,由于该方法需要在检验前将系统 k 步内所有行为编码成一个约束集,当问题规模,如给定步长大小、系统变量数目、自动机组合内成员数目等增长后,约束集大小将快速增长,从而导致相应内存需求急剧上升,进而限制了可解决问题的规模.除此以外,不可不提的是:虽然线性混成系统的可达性问题可以通过一定的编码方式使用 SMT 方法加以解决,但是编码本身是一项非常复杂的工作,目前没有任何相应的支撑工具来完成此项工作.对于混成系统的设计和建模工程师而言,如果没有进行相关长期培训的话,将相应线性混成自动机模型转换成 SMT 问题将会是一项非常困难且极易出错的工作.

为了规避 SAT 及 SMT 方法在线性混成自动机有界可达性检验上面临的问题,基于文献[17,18],我们针对线性混成系统的有界可达性检验问题提出了一套实用而有效的解决方案:

(1) 针对单个线性混成自动机提出了面向单条路径的可达性检验方法,将线性混成自动机中一条路径编码成一组线性约束,进而利用线性规划技术来高效地判定此路径的可达性;

(2) 针对多个线性混成自动机组合形成的复杂混成系统模型,提出了面向路径组的可达性检验方法,对各自动机路径单独编码并添加一组特定的约束进行同步控制,从而可以大幅减小自动机组合所形成的问题规模;

(3) 基于单条路径的可达性检验方法,通过遍历 k 步内的所有路径完成单个自动机的有界可达性判定.根据混成自动机的图形结构,在 k 步内进行深度优先搜索,寻找出相应路径后,利用单条路径的可达性算法验证其可达性;

(4) 基于面向路径组的可达性检验方法,通过同步分别遍历各自动机 k 步内的所有路径完成自动机组合的有界可达性判定.采用由共享事件制导的深度优先搜索算法,高效而同步地遍历各自动机路径所形成的路径组,实现有界可达性检验.

基于以上解决方案,本文设计并实现了相应线性混成系统有界可达性检验工具 BACH(bounded reachability checker).该工具提供的主要功能包括一个便捷的图形化线性混成自动机编辑、预检及文本化生成器,以及一组分别面向单个自动机和自动机组合系统的面向路径及有界可达性检验工具集.

在该工具的基础上,我们对多个经典系统进行了可达性验证实验并与相关工具进行了比较,通过实验结果可以发现:

(1) 在面向路径可达性检验过程中,本工具可检查路径的长度及相应自动机的规模大小可以满足实际问题的需要;

(2) 在有界可达性检验中,本工具可检查的问题规模远大于同类工具;并且在同样问题上,本工具的效率优于同类工具.

本文首先回顾线性混成自动机的相关内容,并对其面向路径(组)可达性约束的检验方法进行介绍.之后详细阐述 BACH 的设计与实现,包括 BACH 的概述、系统结构、关键实现技术等.然后通过实验数据以及相关工作比较,对 BACH 各个模块的性能给出一个直观的描述.最后给出全文总结以及对后续工作的展望.

1 BACH 的理论背景和原理

1.1 线性混成自动机及可达性约束定义

本节将给出本文所研究的线性混成自动机及其面向路径可达性约束的定义.

1.1.1 线性混成自动机

定义 1. 线性混成自动机(linear hybrid automata)是如下形式的八元组:$H=(X,\Sigma,V,E,V^0,\alpha,\beta,\gamma)$.其中:

- X 是实数值系统变量的有限集合,X 中变量的个数也被称为自动机的维度;
- Σ是事件名的有限集合;
- V 是位置节点的有限集合;
- E 是转化关系的集合,E 中的元素具有形式$(v,\sigma,\varphi,\psi,v')$.其中,$v,v'$是 V 中的元素,$\sigma\in\Sigma$是转换上的事件名,φ是形为 $a\leqslant\sum_{i=0}^{l}c_ix_i\leqslant b$ 的转换卫式集合,ψ是形为 $x:=c$ 的重置动作集合.以上 $x_i\in X(0\leqslant i\leqslant l),x\in X,a,b,c_i(0\leqslant i\leqslant l)\in R,a$ 可以是$-\infty,b$ 可以是∞;
- $V^0\subseteq V$ 是初始位置的集合;
- α是一个标注函数,它将每个位置映射到一个节点不变式,节点不变式是形为 $a\leqslant\sum_{i=0}^{l}c_ix_i\leqslant b$ 的变量约束的集合.以上 $x_i\in X(0\leqslant i\leqslant l),a,b,c_i(0\leqslant i\leqslant l)\in R,a$ 可以是$-\infty,b$ 可以是∞;
- β是一个标注函数,它将每个位置映射到一个变化率的集合,变化率是形如 $\dot{x}=[a,b]\,(x\in X,a,b\in R,a\leqslant b)$的式子.对任意位置 v,对任意 $x\in X$,有且仅有一个 $\dot{x}=[a,b]\in\beta(v)$;
- γ是一个标注函数,它将初始位置 V^0 中每个位置映射到一组初始条件,初始条件具有形式 $x:=a(x\in X,a\in R)$.对任意位置 $v\in V^0$,对任意 $x\in X$,有且仅有一个 $x:=a\in\gamma(v)$.

给定线性混成自动机 $H=(X,\Sigma,V,E,V^0,\alpha,\beta,\gamma)$,我们将形如 $\rho=\langle v_0\rangle\xrightarrow[\sigma_0]{(\varphi_0,\psi_0)}\langle v_1\rangle\xrightarrow[\sigma_1]{(\varphi_1,\psi_1)}\ldots\xrightarrow[\sigma_{n-1}]{(\varphi_{n-1},\psi_{n-1})}\langle v_n\rangle$ 的一个位置序列称为路径片段.其中,对任意 $i(0\leqslant i<n)$,此序列满足$(v_i,\sigma_i,\varphi_i,\psi_i,v_{i+1})\in E$.而从初始节点 V^0 开始的一条路径片段,我们称其为一条路径.通过给路径ρ中每个节点赋予一个非负实数$\delta_i(0\leqslant i\leqslant n)$,我们可以生成此路径的时间序列: $\omega=\begin{pmatrix}v_0\\ \delta_0\end{pmatrix}\xrightarrow[\sigma_0]{(\varphi_0,\psi_0)}\begin{pmatrix}v_1\\ \delta_1\end{pmatrix}\xrightarrow[\sigma_1]{(\varphi_1,\psi_1)}\ldots\xrightarrow[\sigma_{n-1}]{(\varphi_{n-1},\psi_{n-1})}\begin{pmatrix}v_n\\ \delta_n\end{pmatrix}$,该时间序列可以表示 H 基于路径ρ的一个行为:H 从位置 v_0 开始出发,在此位置停留δ_0 时间单位后,跳转到位置 v_1,在此位置停留δ_1 时间单位,然后继续往后跳转.我们用变量$\varsigma_i(x)$表示自动机沿着时间序列ω运行,在位置 v_i 停留δ_i 时间单位后变量 x 的值$(0\leqslant i\leqslant n,x\in X)$.$\varsigma_i(x)$取

值的最大值θ_i及最小值θ_i'分别如下计算:

$$\theta_i = d + u_k\delta_k + u_{k+1}\delta_{k+1} + ... + u_i\delta_i,\ \theta_i' = d + u_k'\delta_k + u_{k+1}'\delta_{k+1} + ... + u_i'\delta_i,$$

其中,k 为 x 在节点 v_i 前最后一次重置点,即 $x:=d\in\psi_{k-1}(0<k\leqslant i)$,或者 $x:=d\in\gamma(v_0)(k=0)$,并且对任意 $p(k\leqslant p<i)$,$x:=d'\notin\psi_p$;同时,对任意 $q(k\leqslant q\leqslant i)$,$\dot{x}=[u_q,u_q']\in\beta(v_q)$.

定义 2. 给定线性混成自动机 $H=(X,\Sigma,V,E,V^0,\alpha,\beta,\gamma)$,时间序列:

$$\left\langle\begin{matrix}v_0\\ \delta_0\end{matrix}\right\rangle\xrightarrow[\sigma_0]{(\varphi_0,\psi_0)}\left\langle\begin{matrix}v_1\\ \delta_1\end{matrix}\right\rangle\xrightarrow[\sigma_1]{(\varphi_1,\psi_1)}...\xrightarrow[\sigma_{n-1}]{(\varphi_{n-1},\psi_{n-1})}\left\langle\begin{matrix}v_n\\ \delta_n\end{matrix}\right\rangle,$$

表示 H 的一个行为,当且仅当其满足以下条件:

- H 中存在以下形式的一条路径:$\langle v_0\rangle\xrightarrow[\sigma_0]{(\varphi_0,\psi_0)}\langle v_1\rangle\xrightarrow[\sigma_1]{(\varphi_1,\psi_1)}...\xrightarrow[\sigma_{n-1}]{(\varphi_{n-1},\psi_{n-1})}\langle v_n\rangle$;
- $\delta_1,\delta_2,\ldots,\delta_n$ 满足转换卫式$\varphi_i(0\leqslant i\leqslant n-1)$中的所有变量约束.例如:对$\varphi_i$中任意一个变量约束$a\leqslant c_0x_0+c_1x_1+\ldots+c_lx_l\leqslant b$,满足 $a\leqslant c_0\varsigma_i(x_0)+c_1\varsigma_i(x_1)+\ldots+c_l\varsigma_i(x_l)\leqslant b$,对于任意 $k(0\leqslant k\leqslant m)$,$\theta_k\leqslant\varsigma_i(x_k)\leqslant\theta_k'$;其中,$\varsigma_i(x_k)$表示自动机在位置 v_i 停留δ_i 时间单位后变量 x_k 的值$(0\leqslant k\leqslant m)$,$\theta_k$ 与 θ_k' 表示$\varsigma_i(x_k)$的最大值与最小值(下同);
- $\delta_1,\delta_2,\ldots,\delta_n$ 满足任意位置 $v_i(0\leqslant i\leqslant n)$中的所有节点不变式.对$\alpha(v_i)$中任意一个变量约束 $a\leqslant c_0x_0+c_1x_1+\ldots+c_lx_l\leqslant b(0\leqslant i\leqslant n)$,满足 $a\leqslant c_0\varsigma_i(x_0)+c_1\varsigma_i(x_1)+\ldots+c_l\varsigma_i(x_l)\leqslant b$;同时,对$\alpha(v_{i+1})$中任意一个变量约束$a\leqslant c_0x_0+c_1x_1+\ldots+c_lx_l\leqslant b(0\leqslant i<n)$,满足 $a\leqslant c_0\lambda_i(x_0)+c_1\lambda_i(x_1)+\ldots+c_l\lambda_i(x_l)\leqslant b$,若 $x_k:=e_{i_k}\in\psi_i$ $(0\leqslant k<m)$,则有 $\lambda_i(x_k)=e_{i_k}$;否则,$\lambda_i(x_k)=\varsigma_i(x_k)$.

1.1.2 可达性约束定义

对线性混成自动机 H,其可达性规约由 H 中的位置 v 和一个变量约束集ϕ组成,我们通过符号 $R(v,\phi)$来表示.其中:v 是H中的位置;ϕ是形为 $a\leqslant\sum_{i=0}^{l}c_ix_i\leqslant b$ 的变量约束集合,其中,$x_i\in X(0\leqslant i\leqslant l)$,$a,b,c_i(0\leqslant i\leqslant l)\in R$.对于$H$中的一条路径是否满足给定可达性规约的问题,我们定义如下:

定义 3. 给定线性混成自动机 H 中的路径 $\rho=\langle v_0\rangle\xrightarrow[\sigma_0]{(\varphi_0,\psi_0)}\langle v_1\rangle\xrightarrow[\sigma_1]{(\varphi_1,\psi_1)}...\xrightarrow[\sigma_{n-1}]{(\varphi_{n-1},\psi_{n-1})}\langle v_n\rangle$ 及可达性规约 $R(v,\phi)$,H 基于ρ的行为 $\left\langle\begin{matrix}v_0\\ \delta_0\end{matrix}\right\rangle\xrightarrow[\sigma_0]{(\varphi_0,\psi_0)}\left\langle\begin{matrix}v_1\\ \delta_1\end{matrix}\right\rangle\xrightarrow[\sigma_1]{(\varphi_1,\psi_1)}...\xrightarrow[\sigma_{n-1}]{(\varphi_{n-1},\psi_{n-1})}\left\langle\begin{matrix}v_n\\ \delta_n\end{matrix}\right\rangle$ 满足 $R(v,\phi)$,当且仅当 $v_n=v$ 且该行为满足ϕ中的所有变量约束.

例如:对ϕ中的任意一个变量约束 $a\leqslant c_0x_0+c_1x_1+\ldots+c_lx_l\leqslant b$,该行为满足如下约束:$a\leqslant c_0\varsigma_i(x_0)+c_1\varsigma_i(x_1)+\ldots+c_l\varsigma_i(x_l)\leqslant b$,并且对于任意 $k(0\leqslant k\leqslant m)$,$\theta_k\leqslant\varsigma_i(x_k)\leqslant\theta_k'$.

1.2 面向路径(组)可达性检验方法

在本节中,我们将分别面向单线性混成自动机及自动机组合系统描述如何将其对应的面向路径(组)可达性约束编码成一组线性约束,从而利用线性规划技术进行有效判定.

1.2.1 单自动机面向路径可达性检验方法

基于定义2和定义3,给定线性混成自动机H中的路径 $\rho=\langle v_0\rangle\xrightarrow[\sigma_0]{(\varphi_0,\psi_0)}\langle v_1\rangle\xrightarrow[\sigma_1]{(\varphi_1,\psi_1)}...\xrightarrow[\sigma_{n-1}]{(\varphi_{n-1},\psi_{n-1})}\langle v_n\rangle$ 及可达性规约 $R(v,\phi)$,如果ρ满足 $R(v,\phi)$,则必然存在行为 $\left\langle\begin{matrix}v_0\\ \delta_0\end{matrix}\right\rangle\xrightarrow[\sigma_0]{(\varphi_0,\psi_0)}\left\langle\begin{matrix}v_1\\ \delta_1\end{matrix}\right\rangle\xrightarrow[\sigma_1]{(\varphi_1,\psi_1)}...\xrightarrow[\sigma_{n-1}]{(\varphi_{n-1},\psi_{n-1})}\left\langle\begin{matrix}v_n\\ \delta_n\end{matrix}\right\rangle$ 满足 $R(v,\phi)$.那么,$\delta_1,\delta_2,\ldots,\delta_n$ 必须同时满足$\varphi_i,\alpha(v_i)(0\leqslant i\leqslant n)$以及$\phi$中的所有变量约束.以上条件构成了关于变量$\delta_1,\delta_2,\ldots,\delta_n$的一组线性不等式,我们用符号$\theta(\rho,R(v,\phi))$来表示这组不等式.如果$\theta(\rho,R(v,\phi))$有解,那么就存在一组$\delta_1,\delta_2,\ldots,\delta_n$的取值来构成满足 $R(v,\phi)$的自动机行为.也就是说,我们可以通过检查$\theta(\rho,R(v,\phi))$是否有解来判断ρ是否满足$R(v,\phi)$,而这一问题可以通过线性规划技术高效地加以解决.在上述方法的基础上,我们在文献[17]中进一步提出了两种优化算法:路径分割法(decomposing)与路径缩减法(shortening)来提高该方法的效率.

1.2.2 自动机组合系统面向路径组可达性检验方法

在传统的模型检验方法情况下,多个线性混成自动机组合而成的系统的可达性检验需要将所有成员自动机进行笛卡尔乘积,得到一个新的自动机,继而在此新生成的自动机的基础上进行相应的验证.然而该方法存在一个与生俱来的问题:如果成员自动机的数目增加,那么经过笛卡尔乘积后生成的自动机规模将呈几何级数增长,从而产生所谓的状态空间爆炸问题,使得无法有效地对相应组合系统开展需要的检验工作.因此,在我们的工作中没有采用笛卡尔乘积来构建完整的组合系统,而是针对每个成员自动机的相应路径分别加以处理,再通过增加一组特定的约束来保证成员间同步相关的需求,从而大幅减小了自动机组合所形成的问题规模,规避了状态空间爆炸问题.

我们采用火车道口系统[1]作为自动机组合系统的示范案例,对自动机组合系统面向路径可达性检验问题进行一个示范说明.该系统(如图 1 所示)包含 3 个成员自动机,分别是火车自动机(train,*T*)、道口自动机(gate,*G*)、控制器自动机(controller,*C*).其中,自动机 *T* 与 *C* 通过共享事件 *approach* 和 *exit* 交互,而自动机 *G* 与 *C* 通过共享事件 *lower* 以及 *raise* 相互交互.

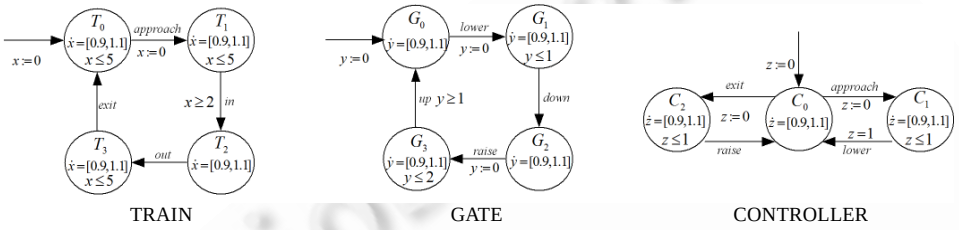


Fig.1 Linear hybrid system for train gate controller (TGC) model

图 1 火车道口组合混成自动机系统

下面,我们将采用图 2(a)所示的路径组来描述面向路径组可达性检验算法.验证的可达性问题为,组合系统能否按照此路径组进行运行.对这一问题,我们按照以下步骤进行线性编码:

(1) 对图 2(a)中的路径组生成对应的时间序列组来表示该系统的行为,如图 2(b)所示.同时生成相应的变量 δ_Y^X 与约束 $\delta_Y^X \geqslant 0$,其中,δ_Y^X 表示系统在 *X* 自动机上的第 *Y* 个节点上停留的时间;

$$\langle T_0\rangle \underset{approach}{\rightarrow} \langle T_1\rangle \underset{in}{\rightarrow} \langle T_2\rangle \underset{out}{\rightarrow} \langle T_3\rangle \underset{exit}{\rightarrow} \langle T_0\rangle \qquad \left\langle \begin{matrix} T_0 \\ \delta_1^T \end{matrix} \right\rangle \underset{approach}{\rightarrow} \left\langle \begin{matrix} T_1 \\ \delta_2^T \end{matrix} \right\rangle \underset{in}{\rightarrow} \left\langle \begin{matrix} T_2 \\ \delta_3^T \end{matrix} \right\rangle \underset{out}{\rightarrow} \left\langle \begin{matrix} T_3 \\ \delta_4^T \end{matrix} \right\rangle \underset{exit}{\rightarrow} \left\langle \begin{matrix} T_0 \\ \delta_5^T \end{matrix} \right\rangle$$

$$\langle G_0\rangle \underset{lower}{\rightarrow} \langle G_1\rangle \underset{down}{\rightarrow} \langle G_2\rangle \underset{raise}{\rightarrow} \langle G_3\rangle \underset{up}{\rightarrow} \langle G_0\rangle \qquad \left\langle \begin{matrix} G_0 \\ \delta_1^G \end{matrix} \right\rangle \underset{lower}{\rightarrow} \left\langle \begin{matrix} G_1 \\ \delta_2^G \end{matrix} \right\rangle \underset{down}{\rightarrow} \left\langle \begin{matrix} G_2 \\ \delta_3^G \end{matrix} \right\rangle \underset{raise}{\rightarrow} \left\langle \begin{matrix} G_3 \\ \delta_4^G \end{matrix} \right\rangle \underset{up}{\rightarrow} \left\langle \begin{matrix} G_0 \\ \delta_5^G \end{matrix} \right\rangle$$

$$\langle C_0\rangle \underset{approach}{\rightarrow} \langle C_1\rangle \underset{lower}{\rightarrow} \langle C_0\rangle \underset{exit}{\rightarrow} \langle C_2\rangle \underset{raise}{\rightarrow} \langle C_0\rangle \qquad \left\langle \begin{matrix} C_0 \\ \delta_1^C \end{matrix} \right\rangle \underset{approach}{\rightarrow} \left\langle \begin{matrix} C_1 \\ \delta_2^C \end{matrix} \right\rangle \underset{lower}{\rightarrow} \left\langle \begin{matrix} C_0 \\ \delta_3^C \end{matrix} \right\rangle \underset{exit}{\rightarrow} \left\langle \begin{matrix} C_2 \\ \delta_4^C \end{matrix} \right\rangle \underset{raise}{\rightarrow} \left\langle \begin{matrix} C_0 \\ \delta_5^C \end{matrix} \right\rangle$$

(a) Path set (b) Behavior set

(a) 路径组 (b) 行为组

Fig.2 Example of path-set and behavior-set selected for TGC model

图 2 火车道口系统路径及行为组示例

(2) 与单自动机面向路径可达性检验一致,对每个成员自动机中的节点不变式、转换卫式等生成相应的约束.例如:对自动机 *T* 中转换 *in* 上的约束 *x*>2,生成约束 $x_{T_{1_{out}}} > 2$,根据节点 T_1 上 *x* 的变化率为[0.9,1.1],生成约束 $x_{T_{1_{in}}} + 0.9\delta_2^T \leqslant x_{T_{1_{out}}} \leqslant x_{T_{1_{in}}} + 1.1\delta_2^T$;根据转换 *approach* 上的重置动作 *x*:=0,生成相应的约束 $x_{T_{1_{in}}} = 0$,其中,$x_{T_{1_{in}}}$ 表示变量 *x* 在系统进入节点 T_1 时的取值,$x_{T_{1_{out}}}$ 表示变量 *x* 在系统离开节点 T_1 时的取值;

(3) 每个成员自动机在行为上的总时间一致.例如,针对自动机 *G* 与 *C*,生成约束:

$$\delta_1^G+\delta_2^G+\delta_3^G+\delta_4^G+\delta_5^G=\delta_1^C+\delta_2^C+\delta_3^C+\delta_4^C+\delta_5^C.$$

(4) 根据共享事件的需求,自动机组合系统中与给定共享事件相关的各成员需在同一时间触发此事件.例如,针对自动机 T 与自动机 G 的共享事件 *exit*,生成约束 $\delta_1^T+\delta_2^T+\delta_3^T+\delta_4^T=\delta_1^C+\delta_2^C+\delta_3^C$.

根据以上步骤,可以将面向此路径组的可达性问题转换成一组包含 22 个变量、68 个约束的线性不等式的可满足性问题,这一问题同样可以通过线性规划技术在毫秒级时间内解决.上述方法的理论基础参见文献[19].

近 10 年以来,线性规划问题已经得到了大量充分的研究,可以用具有多项式时间复杂度的算法加以解决.并且,大量性能优异的科研或者商用性质的线性规划软件包也已相继面世,因此,我们可以利用这个成熟的技术来验证具有相当长度及规模的面向路径可达性问题,从而满足实际生产和生活的需要.

2 BACH 的设计和实现

本节将对 BACH 的设计和实现进行全面阐述,包括 BACH 的概述、系统结构和关键实现技术.

2.1 BACH概述

BACH 是针对线性混成自动机的有界可达性检验工具,其主要功能包括:

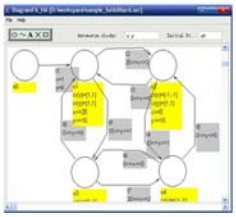
- 线性混成自动机图形编辑器与文本生成器,其中包括线性混成自动机图形编辑器、语法预检器及文本化生成器等;
- 线性混成自动机可达性检验工具集,其中包括单自动机可达性检验工具与自动机组合系统可达性检验工具,每个工具又分别对面向路径可达性和有界可达性两种可达性验证提供支持.

BACH 所包含的工具均使用 Java 语言实现,能够在多个平台上直接运行.BACH 所有版本均已在网上发布,目前最新版为 BACH 2.0,其发布网址为 http://seg.nju.edu.cn/BACH/.自从第 1 版工具发布以来,BACH 已在相关研究领域具有一定的知名度,目前已有来自美国、加拿大、德国、匈牙利等多国的科研工作者下载使用,其中不乏美国加州大学伯克利分校、加拿大英属哥伦比亚大学等名校的相关科研人员.

当前版本 BACH 的各主要部件界面如图 3 所示.

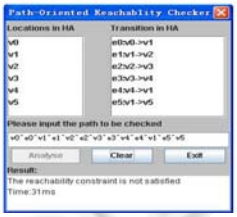
(a) Main GUI

(a) 主界面



(b) LHA editor

(b) LHA 图形编辑器



(c) Path-Oriented reachability checker for single LHA

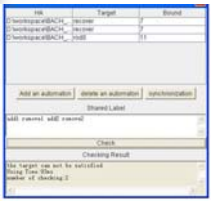
(c) LHA 面向路径可达性检验器

(d) Bounded reachability checker for single LHA

(d) LHA 有界可达性检验器

(e) Path-Oriented reachability checker for compositional LHA system

(e) LHA 组合系统面向路径可达性检验器



(f) Bounded reachability checker for compositional LHA system

(f) LHA 组合系统有界

Fig.3 GUI snapshots of main components of BACH 2.0

图 3 BACH 2.0 各主要部件用户界面图例

2.2 BACH的系统结构

BACH 的系统结构如图 4 所示,主要由两个相互松散耦合的模块构成:可视化自动机编辑模块与可达性检验模块.编辑模块提供可视化编辑工具支持以图形的方式编辑自动机,并可以基于图形化模型生成经过语法预检查的线性混成自动机文本模型.可达性检验模块使用编辑工具生成的自动机模型作为输入文件,该模型也可以由用户根据工具规定的格式自行生成.根据用户选择,系统会分别调用面向单自动机的可达性检验模块与面向自动机组合的检验模块.在每一个模块中,用户同样可以进一步选择具体进行面向路径可达性检验还是有界可达性检验.

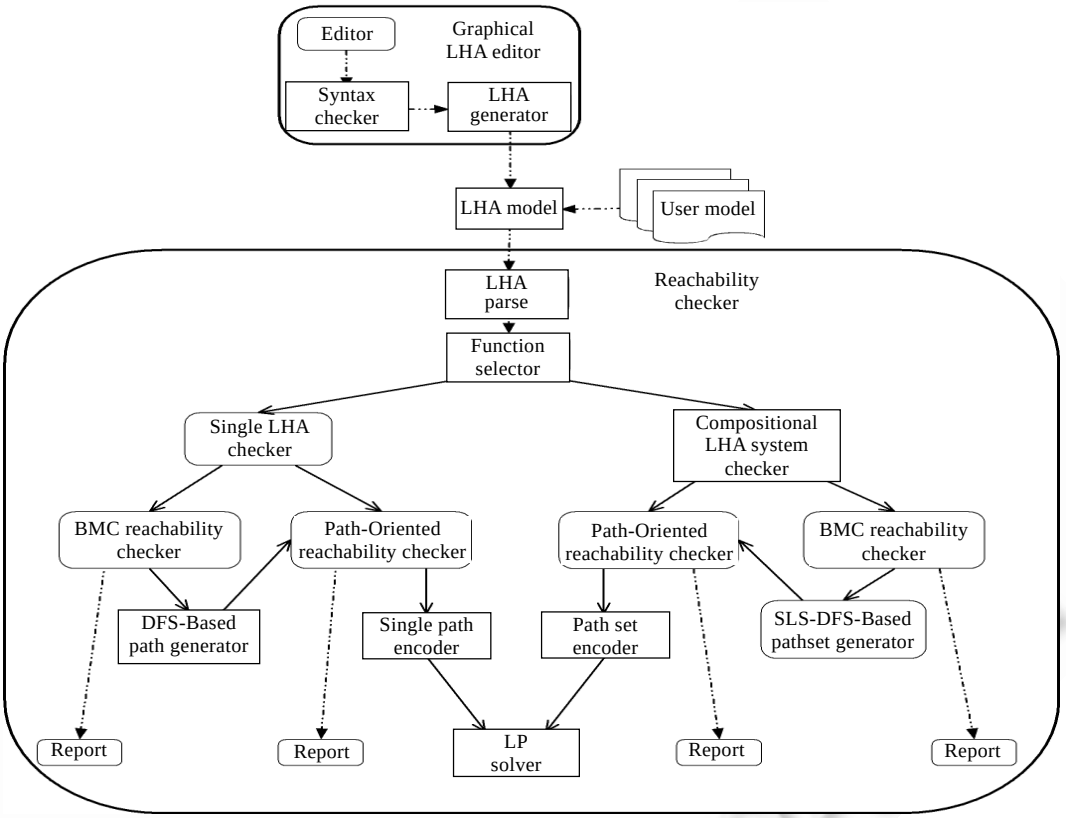


Fig.4 Structure of BACH 2.0

图 4 BACH 2.0 系统结构图

BACH 的典型检验过程如图 5 所示.用户首先为需要检验的系统建模,在通过语法检验后,BACH 会生成标准格式的混成自动机文本化模型.该模型可用于进行检验工作,工具有特定的输入框提示用户输入待检验的自动机模型,并选择相应的验证类型.

(1) 单自动机面向路径可达性验证

BACH 首先预判用户输入路径在图形结构上是否合法,不合法直接报错.如果该路径在图形结构上正确,那么将采用第 1.2.1 节中的思想对其进行编码,生成一组线性不等式,并调用线性规划包对不等式组进行求解,如果有解,则路径可达,反之,则不可达.

(2) 单自动机有界可达性验证

BACH 基于用户选择在给定阈值内进行深度优先搜索,对每一条路径的可达性进行检验:如果该路径终点为目标节点并且可达,则结束并返回可达信息,如果不可达或者路径长度达到阈值限定,则进行回溯.如果模型

给定阈值内所有路径均被检验过,则宣布目标不可达.

(3) 自动机组合系统面向路径组可达性验证

与单自动机面向路径可达性验证类似,BACH 首先预判用户输入路径集在图形结构上是否合法,若不合法,则直接报错.工具将接着对给定路径进行分析,判定这一组路径针对共享事件集是否仍然合法.如果两项预检都正确,那么将采用第 1.2.2 节中的思想对其进行编码,生成一组线性不等式,并调用线性规划包对不等式组进行求解,如果有解,则路径可达,反之,则不可达.

(4) 自动机组合系统有界可达性验证

用户需要为每个自动机分别选定目标集与给定步长,BACH 从第 1 个自动机开始进行深度优先搜索,当发现在步长限制内且能抵达目标的路径后,基于该路径中共享事件的顺序,在下一个自动机中进行深度优先搜索,直到为每个自动机都找到一条符合共享事件序列并且能够抵达目标节点的路径.当得到该路径组后,对其进行检验,若可达,则返回可达信息,反之,则回溯继续搜索.如果模型给定阈值内所有路径均被检验过,则宣布目标不可达.

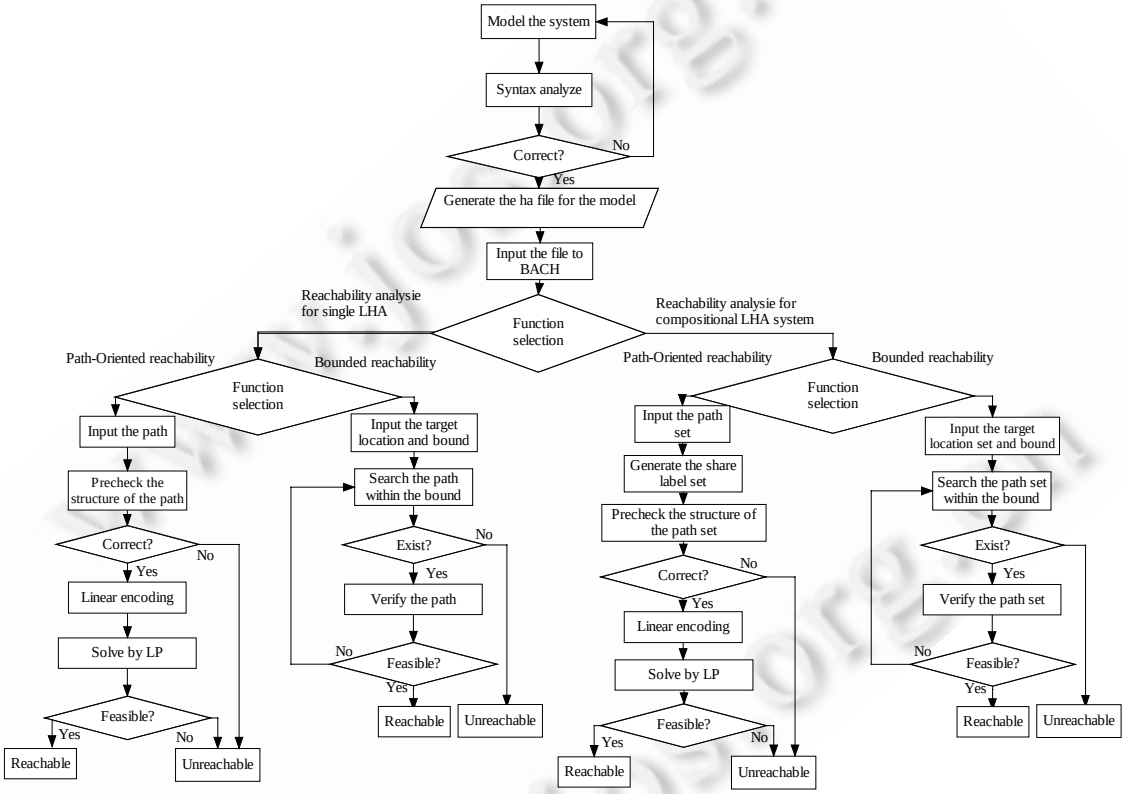


Fig.5 Flowchart of the verification in BACH 2.0

图 5 BACH 2.0 模型检验流程图

整个系统的主要包/类图结构如图 6 所示.

- ModelChecking.diagram_ha 包对应了文本输入模块与图形输入模块,在 Frame 类中实现图形化编辑的主界面,Analyzer 类对用户输入的图形化自动机进行一些预检判断是否有语法错,如果通过语法检测才能够以文本格式保存,以供后期路径验证使用;
- ModelChecking.modelchecking 包为主要的算法实现包,其中:ModelChecking 类实现整个工具的主界面与主算法体系;BoundedCheck,PathCheck 等 4 个类分别对本工具的 4 种可达性检验功能提供接口支持,

并在相应的对话框内返回检验结果;

- ModelChecking.modelchecking.hybridautomata 包主要为程序定义时间自动机各元素的数据结构与操作,其下包含 State,Variable,Transition,Constraint,ChangeRate,Reset 等各个具体类来处理其各元素;
- ModelChecking.drasys 包为整个系统提供线性规划技术支持,目前,我们的这个工具中集成的线性规划软件包来自 OpsResearch 的 OR_Objects(http://opsresearch.com/OR-Objects/).

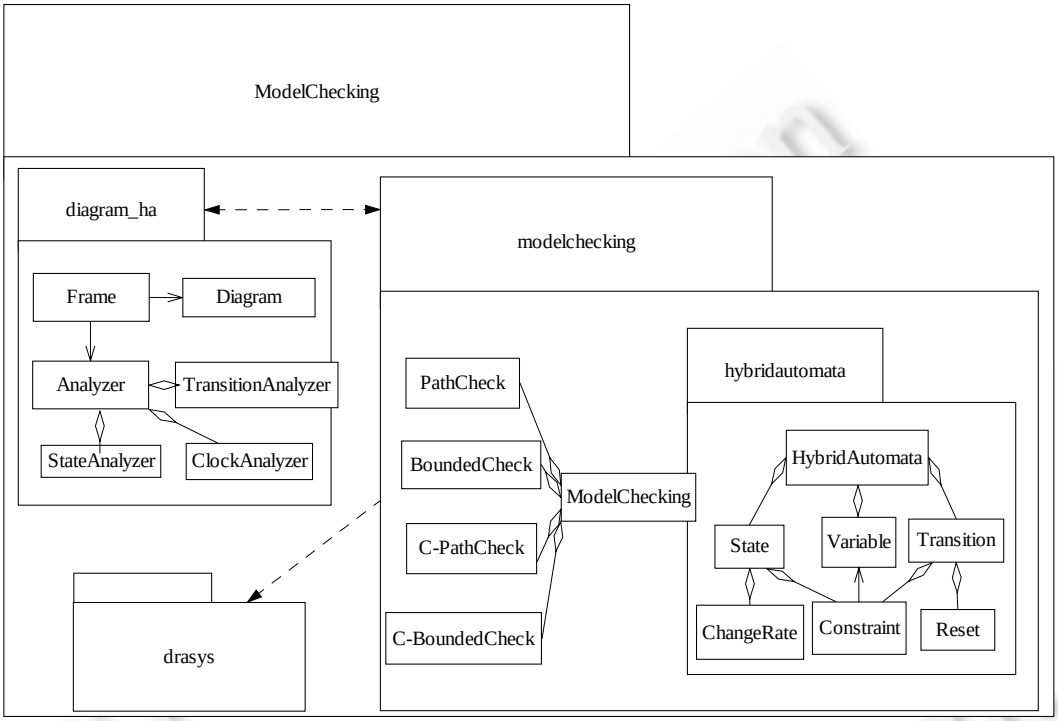


Fig.6 Package/Class diagram of BACH 2.0

图 6 BACH 2.0 包/类结构图

2.3 BACH的关键实现技术——路径遍历的有效实现

BACH 主要技术思想是,将线性混成自动机中的一条路径或者自动机组合中的一个路径组的可达性问题通过线性编码转换成一组线性约束的可满足性问题,从而利用成熟的线性规划技术高效地加以判定;然后遍历自动机或自动机组合的所有路径,实现有界可达性检验.如何有效地遍历自动机或自动机组合的所有路径,是 BACH 实现的关键技术,它是影响 BACH 整体性能的重要因素.下面将介绍如何通过深度优先算法来遍历模型给定阈值下的所有路径(组),以实现有界可达性检验.

2.3.1 单自动机深度优先遍历算法

针对线性混成自动机中单条给定路径的可达性问题,我们已经给出了一种基于线性规划技术的高效编码与解决方法.基于这一方法,我们可以通过遍历阈值内的所有路径来完成单个自动机的有界可达性判定.具体算法大意为:从给定线性混成自动机 H 的起始位置出发,基于 H 的图形结构,采用深度优先搜索方式遍历在给定步数 k 范围内的所有路径,每遍历一条未访问路径均采用在上节中给出的面向单条路径的可达性检验方法来判定其是否满足可达性规约,如果满足,则继续向下遍历,不满足,则向上回溯,直到所有路径均被访问过或者发现抵达目标节点并且满足可达性规约的路径,从而完成给定步数内所有路径的枚举与检查.需要特别注明的是,在本文中,我们称一条路径的步数为 k,表示该路径中含有 k 个节点.

具体算法如图 7 所示:函数 *VERIFY*(*H*,*R*(*v*,*φ*),*bound*)中的输入分别为:待检验的线性混成自动机 *H*,给定的可达性规约 *R*(*v*,*φ*),给定阈值 *bound*;函数 *TRAVERSE*(*loc*,*stack*,*step*)中的输入分别为:当前访问节点 *loc*,当前保存栈 *stack*,当前路径步数 *step*.

```
VERIFY(H,R(v,φ),bound)
for each location v_I∈V^0
begin
        new stack s;
        int step=1;
        s.push(v_I);
        boolean res=TRAVERSE(v_I,s,step);
       if (res==true) return true;
        s.pop(v_I);
 end
return false;
```

```
TRAVERSE(loc,stack,step)
Get the ongoing path ρ from stack;
if (loc==v)
        check ρ according to R(v,φ);
        if (reachable) return true else return false;
else
        check the feasibility of ρ;
        if (unfeasible) return false;
       if ((step==bound)||(loc doesn't have any succeed location)) return false;
       for each succeed location sloc of loc
        begin
             s.push(sloc);
              boolean res=TRAVERSE(sloc,s,step+1);
              if (res==true) return true;
              s.pop(sloc);
        end
        return false
```

Fig.7 Algorithm for the DFS-based bounded reachability verification of LHA

图 7 基于深度优先搜索的线性混成自动机有界可达性验证算法

图 8 为一个包含 5 个位置节点、两个系统变量的线性混成自动机.该自动机描述了一个水位监视器[3]的工作流程.监视器监视水箱里的水位,并相应地打开或关闭阀门.自动机的 5 个位置 v_1,v_2,v_3,v_4,v_5,分别对应着系统的 5 种状态.在位置 v_1,v_2 处,阀门打开;在位置 v_3,v_4 处,阀门关闭;在位置 v_5 处,系统水位降为 0,系统停止工作.系统中,变量 y 描述当前水位,x 反映当前信号延时.我们以该自动机来示例上述深度优先遍历算法,所检验的有界可达性约束为系统在运行 k 步内能否抵达节点 v_5(为简明起见,将 k 设定为 6).算法的具体步骤如下:

(1) 从初始节点 v_1 开始,按照深度优先遍历得到路径 $v_1\rightarrow v_2$,调用面向路径可达性检验方法对其检验,可知该路径可达,继续遍历;

(2) 同上,依次遍历 $v_1\rightarrow v_2\rightarrow v_3$,$v_1\rightarrow v_2\rightarrow v_3\rightarrow v_4$,$v_1\rightarrow v_2\rightarrow v_3\rightarrow v_4\rightarrow v_1$,$v_1\rightarrow v_2\rightarrow v_3\rightarrow v_4\rightarrow v_1\rightarrow v_2$,均可达;

(3) 此时,路径长度为 6,达到阈值,因此,回溯至当前路径中最后一个有分支节点 v_1,访问下一分支 $v_1\rightarrow v_2\rightarrow v_3\rightarrow v_4\rightarrow v_1\rightarrow v_5$.经检验得知不可达,于是继续回溯,检验路径 $v_1\rightarrow v_5$ 同样不可达;

(4) 至此,阈值 6 步之内所有路径皆检验完毕,得出结论:在给定阈值内,系统运行不会达到节点 v_5.

以上算法可以进一步优化,比如 $v_1\rightarrow v_2\rightarrow v_3\rightarrow v_4$ 等路径中并不包含目标节点 v_5,因此并不需要对这些路径进行判定,也即,当且仅当被访问路径最后一个节点为目标节点 v_5 时,我们才需要调用面向路径的检验方法来对当前访问路径进行检验.因此,在阈值 6 步内只需判定路径 $v_1\rightarrow v_2\rightarrow v_3\rightarrow v_4\rightarrow v_1\rightarrow v_5$ 和 $v_1\rightarrow v_5$ 的可达性,即可回答目标节点的有界可达性问题.

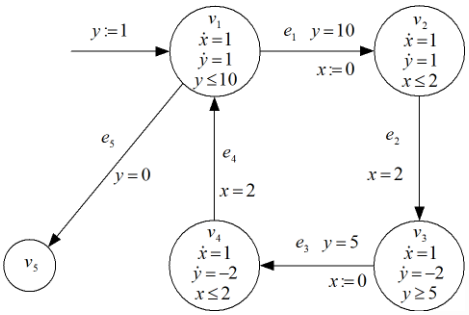


Fig.8 LHA model of the water-level monitor (WLM) system

图 8 水箱水位监视器混成系统模型

2.3.2 自动机组合共享事件序列制导深度优先遍历算法

与单自动机可达性检验工作类似,如果我们能够有效枚举出自动机组合系统在用户给定阈值内所有的路径组,并用自动机组合系统面向路径可达性检验方法对所有路径组一一检验,那么我们就可以回答自动机组合系统有界可达性问题.基于以上指导思想,我们提出了一种基于深度优先遍历给定阈值内所有路径组并检验的算法.由于各成员自动机间通过共享事件来进行同步,任意两个存在共享事件的路径,它们在相应共享事件上的映射必须是一致的,我们称此映射为共享事件序列(share label sequence,简称 SLS).例如:一个含有两个成员 H_1 和 H_2 的自动机组合系统,H_1 的事件集 Σ_1 为 $\{a,b,c\}$,H_2 的事件集 Σ_2 为 $\{a,b,d\}$,$\Sigma_1\cap\Sigma_2=\{a,b\}$,那么,$H_1$ 中路径 $\rho_1=b\to c\to a\to c\to b$ 与 H_2 中路径 $\rho_2=b\to d\to a\to b\to a$ 必然不能满足任何可达性规约.因为,ρ_1 在共享事件集上的映射为 $b\to a\to b$,而 ρ_2 在共享事件集上的映射为 $b\to a\to b\to a$,两者不相吻合,从而不能构成任何合法行为.基于以上思想,我们给出共享事件序列制导下基于深度优先搜索的线性混成自动机组合系统有界可达性验证算法(SLS-DFS).

该算法与面向单自动机系统的有界可达性算法类似却又不尽相同:在面向单自动机时,BACH 严格按照深度优先思想对当前节点的所有后续节点进行逐个遍历;而面向自动机组合时,我们只有自动机组的第 1 个成员是采用简单深度优先算法进行遍历,而所有后续成员自动机在深度优先遍历时,均要参考已遍历自动机所维护的路径,保证当前遍历的路径与其他成员自动机路径在共享事件序列上保持一致,若不一致,则需要及时回溯.当遍历到最后一个成员自动机,寻找到一条在阈值内抵达目标节点的路径,并且每个自动机所访问的路径均在共享事件序列上保持一致时,就调用面向路径组的可达性规约检验方法来检验该路径组是否满足相应规约,若不满足,则从最后一个成员自动机开始回溯,直到所有自动机的可能路径均已访问过为止.具体算法如图 9 所示,其中:函数 $VERIFY(N,R(v,\phi),bound)$ 中的输入分别为:待检验的自动机组合系统 N,$N=H_1\|H_2\|\ldots\|H_m$,给定的可达性规约 $R(v,\phi)$,给定阈值 $bound$;函数 $SLS\text{-}DFS(H_i,\vartheta^*,\rho^*)$ 中的输入分别为:当前遍历自动机 H_i,当前维护共享事件序列集 ϑ^*,当前维护遍历路径组 ρ^*.

我们采用火车道口系统来示例上述算法.首先给出需要检验的可达性目标为 $\{T_0,G_0,C_0\}$ 以及步长阈值 {[2,5],[2,5],[2,5]},然后按照上述算法步骤开始检验该目标的有界可达性:

(1) 对成员自动机中所有事件进行分析,生成共享事件集 $\{approach,exit,lower,raise\}$;

(2) 对每个成员自动机均生成一条初始路径用来维护该自动机当前访问路径 ρ_T,ρ_G,ρ_C,并且将这 3 条路径加入路径组 $\rho^*=\{\rho_T,\rho_G,\rho_C\}$,同时生成共享事件序列集 ϑ^*;

(3) 使用深度优先搜索,寻找自动机 T 中能够在给定阈值[2,5]内抵达目标节点 T_0 的路径:

$$\rho_T=\langle T_0\rangle \underset{approach}{\to} \langle T_1\rangle \underset{in}{\to}\langle T_2\rangle \underset{out}{\to}\langle T_3\rangle \underset{exit}{\to}\langle T_0\rangle;$$

(4) 根据 ρ_T 生成相应的共享事件序列 $\vartheta_T=approach\to exit$,将 ϑ_T 添加至 ϑ^*;

(5) 类似地,在自动机 G 中找到路径 $\rho_G=\langle G_0\rangle \underset{lower}{\to}\langle G_1\rangle \underset{down}{\to}\langle G_2\rangle \underset{raise}{\to}\langle G_3\rangle \underset{up}{\to}\langle G_0\rangle$;

(6) 根据 ρ_G 生成相应的共享事件序列 $\vartheta_G=lower\to raise$,将 ϑ_G 添加至 ϑ^*;

(7) 根据ϑ^*和目标 C_0 的指导,在自动机 C 中寻找到相应路径ρ_C形为$\langle C_0\rangle \xrightarrow[approach]{} \langle C_1\rangle \xrightarrow[lower]{} \langle C_0\rangle \xrightarrow[exit]{} \langle C_2\rangle \xrightarrow[raise]{} \langle C_0\rangle$;

(8) 因为自动机 C 为组合系统中最后一个自动机,利用面向路径组可达性检验方法检验ρ^*的可达性;

(9) 如果ρ^*可达,则结束搜索并给出可达报告;

(10) 如果ρ^*不可达,则在自动机 C 上继续深度优先搜索.在本示例中,自动机 C 上不存在另一条既满足共享事件序列集又满足给定阈值且可以抵达目标节点的路径,因此,回溯至上层自动机 G;同样无法寻找到这样的路径,于是继续回溯到自动机 T;仍然找不到这样的路径,因此程序结束,宣布目标$\{T_0,G_0,C_0\}$在给定阈值{[2,5],[2,5],[2,5]}内不可达.

值得一提的是,在上述示例中,无论目标是可达或者不可达,我们的算法均只对一个路径组进行了检验,这充分显示了这一算法在裁剪路径组上的能力.因此,无需将所有成员自动机进行笛卡尔乘积,构筑一个完整的状态空间,我们的算法就可以在给定阈值内对组合系统的状态空间进行快捷而有效的遍历.

```
VERIFY(N,R(v,φ),bound)
generate path set ρ*={ρ1,ρ2,…,ρm} and SLS set ϑ*={ϑ1,ϑ2,…,ϑm}, ∀ρi and ∀ϑi are empty;
SLS-DFS(H1,ϑ*,ρ*);
SLS-DFS(Hi,ϑ*,ρ*)
WHILE true
begin
    using DFS to find the next potential path ρ of Hi after ρi in bound;
    if (no such path exists)
        set ρi and ϑi to empty;
        return false;
    else
        update ρi, ρ*, ϑi and ϑ* by ρ,
        if (i==m)
            check reachability of ρ* according to R(v,φ).
            if (ρ* is reachable) return true else continue
        else
            boolean result=SLS-DFS(Hi+1,ϑ*,ρ*)
            if (result==true) return true else continue
end
```

Fig.9 Algorithm for the SLS-guided-DFS-based bounded reachability verification of CLHA

图 9 共享事件序列制导下基于深度优先搜索的组合线性混成系统有界可达性验证算法

3 实例研究与相关工作比较

在 HP 工作站(Intel Xeon CPU 1.8GHz/3.78GB)和 Dell 工作站(Intel Core2 Quad CPU 2.4GHz/ 4GB RAM)上,我们对 BACH 的性能进行了一系列的实验评估,并与相关工具作了比较,下面详细介绍相关情况.

3.1 单自动机可达性检验的实验评估

我们应用本工具对如图 8 所示的水箱系统及如图 10 所示的温度控制器分别进行了研究,这两个系统均选自文献[3],并且是混成系统验证的典型实例.对于水箱自动机模型,需要验证的可达性问题为:节点 v_5(水位为 0)是否可达.温度控制箱自动机模型描述的是这样一个反应箱装置:它通过两根控制杆来维持箱内温度在 3~15 之间,当温度(用符号θ描述)达到最高值 15,反应箱必须立刻使用其中一根控制棒来进行降温.同一根控制棒两次使用间隔之间不能少于 6 个时间单位.如果当温度达到最高点,而没有控制棒可用时,那么系统就抵达 *shutdown* 状态,其中,时钟变量 x_1 与 x_2 分别描述距离上次控制棒 1 或者控制棒 2 被使用多长时间.在该系统上,我们所需要检验的可达性问题即为不安全节点 v_4(*shutdown*)在系统运行中是否可达.

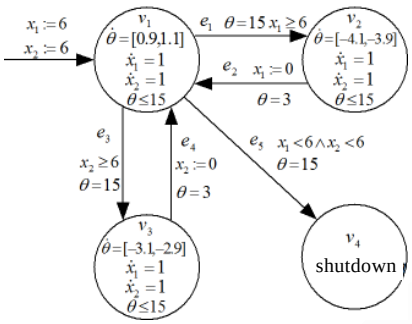


Fig.10 LHA model of the temperature control system (TCS)

图 10 温度控制系统自动机模型

3.1.1 面向路径可达性检验模块实验数据

我们分别选取水箱系统中的路径 $v_1 \to (v_2 \to v_3 \to v_4 \to v_1)^k \to v_5$ 与温度控制系统中的路径$(v_1 \to v_2 \to v_1 \to v_3)^k \to v_1 \to v_4$作为实验所用路径,其中,$k$ 表示相应路径片段重复次数.相应的实验数据分别列于表 1 和表 2 中.由数据中可见,在不使用优化技术的情况下,我们单次可以检查的路径长度超过 1 600 个节点,足以满足任何规模问题的需要.而当循环数 k 达到 500 时,如果不采用任何优化算法,程序将会报出内存分配异常“Java.lang.out of memory”.这一现象的原因在于,目标路径转化成的线性约束集过大,超过了分配给 Java 虚拟机的内存配额(在 32 位机器上,此配额上限一般不超过 2GB,具体数值与 JDK 版本相关).但是我们可以发现,如果采用路径分解或者路径缩减优化技术,工具表现均会得到极大的提高.如表 1 所示,在循环次数达到 10 000 的情况下,如果使用路径缩减技术,只需 0.031s 即可完成判定.此时,相应路径中离散节点数目超过 40 000 个.

Table 1 Path-Oriented reachability data of WLM system

表 1 水位控制器系统面向路径可达性实验数据表

Path: $v_1 \to (v_2 \to v_3 \to v_4 \to v_1)^k \to v_5$						
k	Original technique (without optimization)				Path decomposing	Path shortening
	Constraint	Variable	Memory	Time (s)	Time (s)	Time (s)
200	7 207	2 002	512M	503.140	1.562	0.031
400	14 407	4 002	1 470M	4202.421	3.187	0.031
500	Java.lang.out of memory error				4.109	0.031
10 000	Java.lang.out of memory error				38.047	0.031

Table 2 Path-Oriented reachability data of TCS system

表 2 温度控制箱系统面向路径可达性实验数据表

Path: $(v_1 \to v_2 \to v_1 \to v_3)^k \to v_1 \to v_4$						
k	Original technique (without optimization)				Path decomposing	Path shortening
	Constraint	Variable	Memory	Time (s)	Time (s)	Time (s)
200	8 815	2 004	512M	686.938	686.938	686.938
400	17 591	3 998	1 470M	5 574.312	5 574.312	5 574.312
450	Java.lang.out of memory error					

3.1.2 有界可达性检验模块实验数据

与上述实验类似,在基于水箱系统和温控系统的有界可达性实验中,我们选择的目标节点分别为水箱系统中的 v_5 节点与温控系统中的 v_4 节点.同时,因为线性混成系统的有界可达性检验可以通过 SAT 技术完成,所以在对这两个模型分别进行编码之后,采用工具 HySAT[14]进行了实验比较,HySAT 是当前唯一面向混成自动机开发的、基于 SAT 思想的有界模型检验工具.具体实验数据图示如下,其中,横坐标为给定步长,纵坐标为解决相应问题所需时间.与第 2 节所述相同,实验中步长定义为路径中离散节点的数目:

如图 11 和图 12 所示中数据分别为水箱系统和温度控制系统的实验室数据.其中,下方折线为我们的工具 BACH 的实验数据,上方折线为工具 HySAT 的实验数据.从中可以看出:

(1) BACH 所处理的问题规模大于 HySAT.由于 SAT 方法需要在检验前先行将系统 k 步内所有行为编码成一个约束集,当问题规模,如给定步长、自动机维度等增长后,约束集大小将快速增长,从而限制了可解决问题的规模.而反观 BACH,在运行过程中,BACH 只需在内存中保存当前所运行的路径以及相应路径转换成的线性不等式集即可,对内存需求极小.因此,BACH 可以解决问题的规模大于基于 SAT 技术的 HySAT;

(2) 对于相同问题规模,BACH 所需时间小于 HySAT.首先,BACH 通过深度优先遍历直接在模型图形结构上寻路,相应路径为离散的节点序列,不会引入在相应节点空转的 stuttering 转换.而 HySAT 则通过使用 SAT 技术在将 k 步内所有行为编码成的约束集上寻找能够满足目标的一组谓词来完成类似工作,由于 stuttering 转换的引入,导致其面对的图形空间更加复杂;其次,寻找到相应路径后,BACH 通过线性规划来验证每条路径的可达性,而 HySAT 用区间运算(interval analysis)的方法来判定相应谓词组所推出的线性约束是否有解.众所周知,在线性约束集的求解方面,线性规划技术的性能要远远超越区间运算的性能.

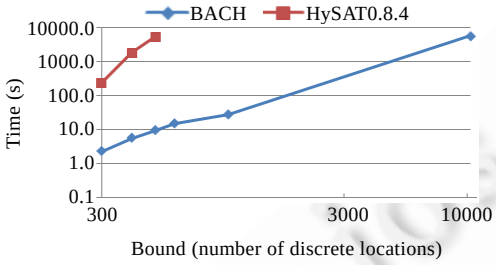


Fig.11 Bounded reachability data of WLM system

图 11 水箱系统有界可达性实验数据

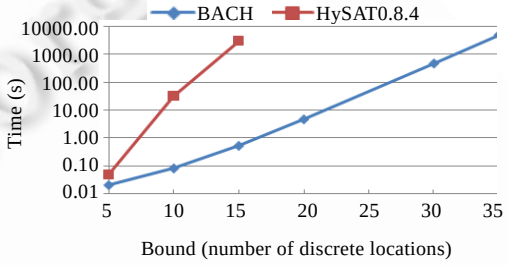


Fig.12 Bounded reachability data of TCS system

图 12 温度控制系统有界可达性实验数据

3.2 自动机组合可达性检验的实验评估

在本节中,我们使用 Fischer 互斥协议与核反应堆控制棒系统来对自动机组合系统可达性验证模块进行评估.这两个系统的原型分别来源与文献[3]和文献[20],其系统结构如图 13 和图 14 所示.

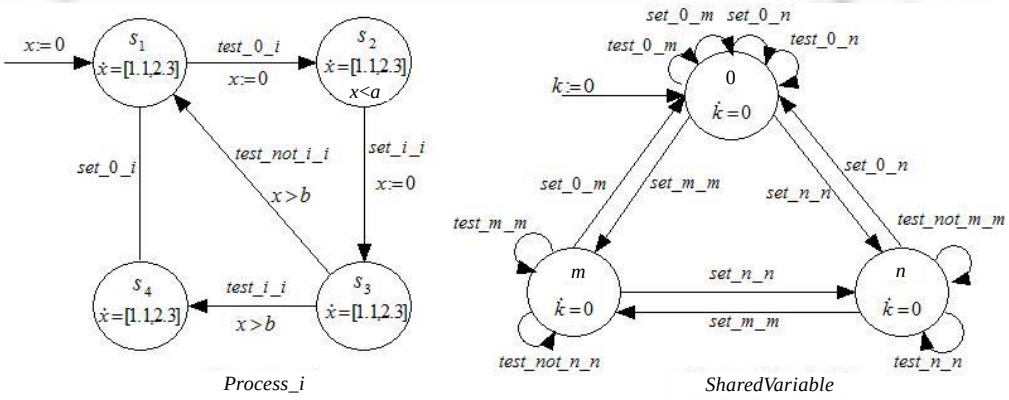


Fig.13 Hybrid automata for Fischer mutual exclusion system (FME)

图 13 Fischer 互斥系统混成自动机模型

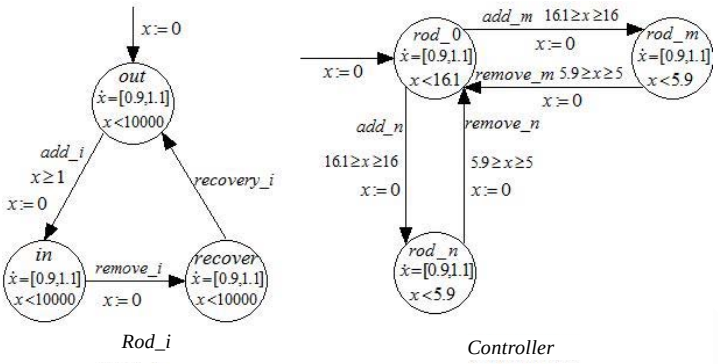


Fig.14 Hybrid automata for Nuclear reactor system (NRS)

图 14 核反应堆控制棒系统混成自动机模型

Fischer 互斥系统由一组相互竞争进入临界区的进程组成,如图 13 所示,系统与一般 Fischer 系统的区别在于,为了实验需要,我们将其从使用共享变量来进行同步控制的自动机系统转换成了基于共享事件来进行同步的自动机系统.因此,我们为此系统添加了一个描述共享变量的自动机 SharedVariable(SV),如图 13 中右图所示.如图 14 所示,核反应堆控制棒系统描述的场景为一组用来吸收中子的控制棒.系统通过控制器来调度控制棒按序进入重水并吸取其中的中子,从而控制系统的温度在安全范围内.每根刚从重水中取出的控制棒必须在外面停留一段时间才可以继续使用.这两个自动机系统的规模都可以通过增加进程自动机或者控制棒自动机的数目来扩大,因此非常适合用来评估 BACH 在处理大规模自动机组合系统时的性能.

3.2.1 面向路径可达性检验模块实验数据

在面向路径可达性实验评估中,为了简明起见,我们选取的系统分别为包含了 40 个进程的 Fischer 互斥系统和拥有 40 根控制棒的核反应堆控制棒系统.我们为 Fischer 互斥系统混成自动机选取的路径组为对每个进程自动机从初始节点 s_1 按照 $(s_1 \xrightarrow{test\_0\_i} s_2 \xrightarrow{set\_i\_i} s_3 \xrightarrow{test\_i\_i} s_4)^k \xrightarrow{set\_0\_i} s_1$ 的方式经过临界区节点 s_4 再回到 s_1,相应的所描述的可达性问题为整个系统能否按照此序列正常运行 k 次.对核反应堆控制棒系统选取路径组为每个控制棒自动机皆依次从 *out* 节点开始按照 $(out \xrightarrow{add\_i} in \xrightarrow{remove\_i} recover)^k \xrightarrow{recovery\_i} out$ 的方式运行,所描述的可达性问题同样为整个系统能否按照此序列正常运行 k 次,相应的实验结果见表 3 和表 4.

由表 3 和表 4 的数据可知,实验中成功解决的最大问题是一个由 40 个互斥进程组成的 Fischer 互斥协议系统中每个进程重复进入临界区达 15 次之多的路径组.因为我们的算法所生成的线性规划问题的规模与被检验自动机组合系统中路径组的规模线性相关,所以,如果缩短每个自动机的路径长度,那么就可以检验含有更多成员的自动机组合系统.事实上,通过减少路径循环次数,我们成功地验证了一个由 320 个进程组成的 Fischer 互斥协议系统以及一个由 320 根控制棒组成的核反应堆系统,具体数据将在下一节加以展示.

Table 3 Path-Oriented reachability data of FME system

表 3 Fischer 互斥协议系统面向路径可达性实验数据表

Path set	Pro_1	$(s_1 \xrightarrow{test\_0\_1} s_2 \xrightarrow{set\_1\_1} s_3 \xrightarrow{test\_1\_1} s_4)^k \xrightarrow{set\_0\_1} s_1$		
	…	…		
	Pro_{40}	$(s_1 \xrightarrow{test\_0\_40} s_2 \xrightarrow{set\_40\_40} s_3 \xrightarrow{test\_40\_40} s_4)^k \xrightarrow{set\_0\_40} s_1$		
	SV	$(0 \xrightarrow{test\_0\_1} 0 \xrightarrow{test\_0\_2} \ldots \xrightarrow{set\_0\_2} 0)^k \xrightarrow{set\_0\_1} 0$		
k	Constraint	Variable	Memory	Time (s)
5	4 962	2 041	256M	160.661
10	9 762	4 041	1 024M	1 233.208
15	14 562	6 041	2 048M	4 099.463

Table 4 Path-Oriented reachability data of NRS system

表 4 核反应堆控制棒系统面向路径可达性实验数据表

Path set	Rod_1 … Rod_{40} Controller	$(out \xrightarrow{add\_1} in \xrightarrow{remove\_1} recover)^k \xrightarrow{recovery\_1} out$ … $(out \xrightarrow{add\_40} in \xrightarrow{remove\_40} recover)^k \xrightarrow{recovery\_40} out$ $(rod\_0 \xrightarrow{add\_1} rod\_1 \xrightarrow{add\_2} ... \xrightarrow{add\_40} rod\_40)^k \xrightarrow{remove\_40} rod\_0$			
k	Constraint	Variable	Memory	Time (s)	
4	6 246	1 682	256M	235.066	
8	12 166	3 282	1 024M	1 602.527	
12	18 086	4 882	2 048M	5 334.934	

3.2.2 有界可达性检验模块实验数据

在有界可达性检验模块中,我们为 Fischer 互斥协议系统设定的可达目标为每个进程均达到节点 s_4,为核反应堆控制棒系统设定的可达目标为每个控制棒均达到节点 *recover*,其相应的所反应的实际模型行为分别是:Fischer 互斥系统中所有进程同时抵达临界区的不安全状态与核反应堆系统中所有控制棒均被使用过一次从而导致的没有待命控制棒的潜在不安全状态.在实验中,我们给相应模型设定的步长分别为各个自动机抵达相应目标节点的最小步数.例如,对 Fischer 互斥系统,每个进程步长为 4,共享变量自动机步长为 $3n$+1,其中, n 为系统内进程数目.

与面向单自动机的有界可达性检验模块实验类似,在本实验中,我们同样与 HySAT 进行了比较.同时,鉴于处理大规模组合系统带来的状态空间爆炸问题长久以来一直是模型检验的一项重要课题,因此,在此模块的实验评估上花费了更多精力.我们将相应的自动机模型通过主流编码方式编码成 SMT 文件,在此基础上,使用所有可以在 Linux 平台上运行的、与线性实数求解相关的 SMT 处理器对相应问题进行求解,从而能够更加客观地评估我们的这一工具的性能.线性实数求解问题在 SMT 社区中被统称为 QF-LRA 问题.在比较中用到的 SMT 处理器包括了参加 2008 年 SMT 大赛(SMT-COMP 2008)中 QF-LRA 分组竞赛的除 Z3[21](仅提供 Windows 版)之外的所有 SMT 工具,其中包括 MathSAT[22],Yices[23],CVC3[24]以及 Barcelogic[25].实验中所有使用到的版本皆为当前公开发布的最新版.

为了更清晰地反映我们的工具 BACH 与其他工作相比在各方面的优、缺点,我们通过调整参数的方式分别为两个系统各构造了两个版本:“不可达版”和“可达版”.“不可达版”表示为系统中不存在可达的路径组,反之,“可达版”表示为系统中任意一个满足语法结构的路径组都是可达的.通过将每次实验的时间上限设为 24 小时,分别得出了 4 组实验数据,其中,基于不可达模型的两组实验数据分别如图 15 和图 16 所示,基于可达模型的两组实验数据分别如图 17 和图 18 所示.

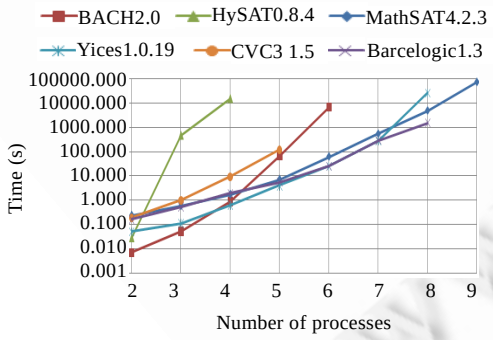


Fig.15 Bounded data of unreachable FME system

图 15 Fischer 互斥系统不可达模型实验数据

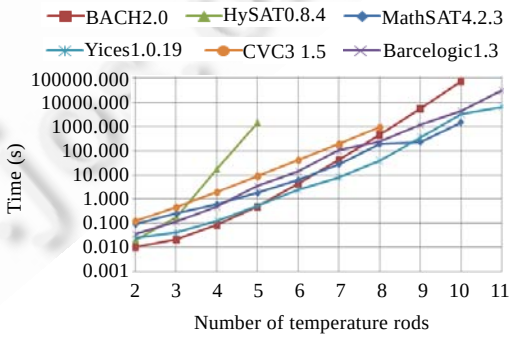


Fig.16 Bounded data of unreachable NRS system

图 16 核反应堆控制棒系统不可达模型实验数据

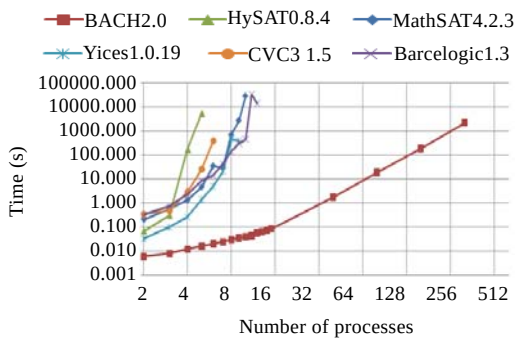


Fig.17 Bounded data of reachable FME system

图 17 Fischer 互斥系统可达模型实验数据

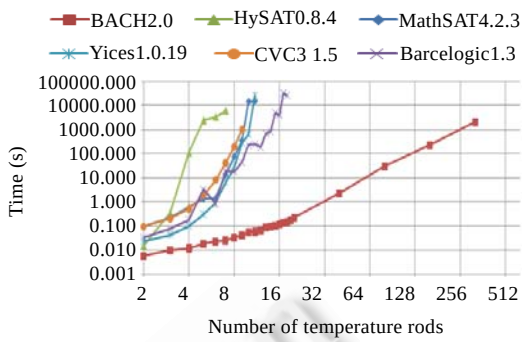


Fig.18 Bounded data of reachable NRS system

图 18 核反应堆控制棒系统可达模型实验数据

基于以上数据,我们可以看出:

(1) 在不可达模型上,由于 BACH 需要枚举并检验阈值内所有路径组,在给出的两个系统中,相应路径组数目非常大,因此,这个过程会导致 BACH 在验证相关问题时消耗非常多的时间.而反观 SMT 处理器,得益于其内部的冲突子句学习(conflict clause learning)技术,SMT 处理器可以从已经被否证的路径组上学习到导致其不可达的路径片段,从而在后期寻路中发现类似的路径片段时可以先期判定其不可达而直接回退,节省了计算时间,因此,部分 SMT 处理器在不可达模型上的实验性能略优于 BACH.比如在 Fischer 互斥系统上,BACH 验证的最大问题含有 6 个进程,而在 SMT 处理器中,MathSAT 用一整天时间成功验证了一个含有 9 个进程的系统.在核反应堆系统中,BACH 在一天时间内成功验证的最大问题含有 10 个控制棒,而 Yices 与 Barcelogic 成功验证了一个含有 11 个控制棒的系统;

(2) 而在可达模型中,只需寻找到一组语法上正确的路径就可以证明目标的可达性为真.这时,BACH 就充分体现了其寻路能力.由于 BACH 运行中基于每个成员的图形结构分别寻路,与 SMT 将给定阈值内所有行为均编码成相应约束,再在此约束集上进行寻路相比,寻路所基于的问题规模要大为精简.如图 17、图 18 所示,BACH 所能解决的系统中成员数目达到了 320 个.BACH 验证含有 320 个进程的 Fischer 系统只花费了 2 258.5s,验证含有 320 个控制棒的核反应堆系统也仅花费了 2 068.29s,均不超过 1 小时.而 SMT 处理器所解决的最大规模问题为一个仅仅含有 18 个控制棒的核反应堆系统,该问题由工具 Barcelogic 花费 25 648.9s 解决,接近 7 小时,而相同问题仅花费 BACH 0.153s.由此可以看出,BACH 在处理大规模组合系统时的巨大潜力.另外一点值得说明的是,绝大多数 SMT 处理器在被处理系统中当其成员数目超过 10 个时,会使用超过 1GB 内存乃至报出内存分配错误,而这种规模的系统在 BACH 运行时耗费内存不到 64MB.因此,从各个角度而言,在可达模型的实验数据上,BACH 均远远优于 SMT 处理器.另外,需特别提出的是,不可达模型上的实验数据只能说明系统行为在一定范围内不能达到相应状态,并不能说明系统整个行为均无法达到相应状态.相比之下,可达模型上的数据更能体现有界模型检验的优势所在.即当系统行为存在错误状态,并且常规方法无法对系统进行检验时,通过限制行为步数能够有效发现该错误的存在.

4 总结与展望

本文介绍了线性混成系统有界可达性检验工具 BACH 的原理、设计与实现,并针对所提供的各个功能模块的具体应用给出了应用实例.通过与现有相关工具的实验比较可以发现,无论是解决单个自动机还是自动机组合系统,以及无论是解决面向路径可达性问题或是有界可达性问题,BACH 都显示出良好的性能和可扩展性.在当前欠缺一个面向线性混成自动机全状态空间的有效工具的现实情况下,我们相信,BACH 可以凭借其在有界模型检验范畴下,其良好的性能成为相关研发人员的有力助手,帮助他们在设计过程中检验过去无法判定的大规模系统的可达性问题.另一方面,与相关工具提供的难以掌握的纯文本模型描述方法相比,BACH 中整合的

图形化线性混成自动机编辑与生成工具更将极大地方便用户轻松地创建线性混成自动机模型,并加以使用以解决更多问题.

在后续工作中,我们将研究如何对已经检验并且判定为不可达的路径或路径组进行学习,从而指导下一条路径的选择,特别是如何根据已积累的错误信息在前期对当前访问路径进行预判,从而节省深度优先中所需检验的路径数,从而进一步节省系统运行时间.另一方面,我们当前的工作主要基于可达性(reachability)验证,这属于安全性(safety)验证的一个子集.在下一步工作中,我们将对如何把线性规划技术引入到线性混成系统的活性(liveness)检验上进行研究与尝试,以对线性混成系统模型检验提供更全面的支持.在工具进一步完善方面,我们正在开发全新版本的图形编辑器以更加方便用户使用.同时,为了更好地推广 BACH,基于 Eclipse 插件的一个 BACH 特别版本也即将发布.

References:

[1] Henzinger T. The theory of hybrid automata. In: Proc. of the 11th Annual IEEE Symp. on Logic in Computer Science (LICS'96). IEEE Computer Society Press, 1996. 278−292.

[2] Kesten Y, Pnueli A, Sifakis J, Yovine S. Integration graphs: A class of decideable hybrid systems. In: Grossman R, Nerode A, Rvan A, Rischel H, eds. Proc. of the Hybrid System. LNCS 736, Springer-Verlag, 1993. 179−208. [doi: 10.1007/3-540-57318-6_29]

[3] Alur R, Courcoubetis C, Halbwachs N, Henzinger TA, Ho PH, Nicollin X, Olivero A, Sifakis J, Yovine S. The algorithmic analysis of hybrid systems. Theoretical Computer Science, 1995,138:3−34. [doi: 10.1016/0304-3975(94)00202-T]

[4] Henzinger T, Kopke PW, Puri A, Varaiya P. What's decidable about hybrid automata? Journal of Computer and System Sciences, 1998,57(1):94−124.

[5] Henzinger TA, Ho PH, Wong-Toi H. HYTECH: A model checker for hybrid systems. Software Tools for Technology Transfer, 1997,1:110−122. [doi: 10.1007/s100090050008]

[6] Frehse G. PHAVer: Algorithmic verification of hybrid systems past HyTech. In: Morari M, Thiele L, eds. Proc. of the Int'l Conf. on Hybrid Systems: Computation and Control 2005. LNCS 2289, Springer-Verlag, 2005. 258−273. [doi: 10.1007/s10009-007-0062-x]

[7] Henzinger T, Majumdar R. Symbolic model checking for rectangular hybrid systems. In: Graf S, Schwartzbach M, eds. Proc. of the 6th Workshop on Tools and Algorithms for the Construction and Analysis of Systems. LNCS 1785, Springer-Verlag, 2000. 142−156. [doi: 10.1007/3-540-46419-0_11]

[8] Zhou C, Zhang J, Yang L, Li X. Linear duration invariants. In: Langmaack H, Roever W, Vytopil J, eds. Proc. of the Formal Techniques in Real-Time and Fault-Tolerant Systems. LNCS 863, Berlin: Springer-Verlag, 1994. 86−109.

[9] Li X, Dang V, Zheng T. Checking hybrid automata for linear duration invariants. In: Shyamasundar R, Ueda K, eds. Advances in Computing Science (ASIAN'97). LNCS 1345, Springer-Verlag, 1997. 166−180. [doi: 10.1007/3-540-63875-X_51]

[10] Li X, Zhao J, Pei Y, Li Y, Zheng T, Zheng G. Positive loop-closed automata: A decidable class of hybrid systems. Journal of Logic and Algebraic Programming, 2002,52-53:79−108. [doi: 10.1016/S1567-8326(02)00023-1]

[11] Bryant R. Graph-Based algorithms for boolean function manipulation. IEEE Trans. on Computers, 1986,35(8):677−691. [doi: 10.1109/TC.1986.1676819]

[12] Zhang L, Malik S. The quest for efficient Boolean satisfiability solvers. In: Brinksma D, Larsen G, eds. Proc. of the Int'l Conf. on Computer Aided Verification 2002. LNCS 2404, Springer-Verlag, 2002. 17−36. [doi: 10.1007/3-540-45620-1_26]

[13] Biere A, Cimatti A, Clarke EM, Strichman O, Zhu Y. Bounded model checking. Advance in Computers, 2003,58:118−149.

[14] Franzle M, Herde C, Ratschan S, Schubert T, Teige T. Efficient solving of large non-linear arithmetic constraint systems with complex Boolean structure. Journal on Satisfiability, Boolean Modeling and Computation, 2007,1:209−236.

[15] Audemard G, Bozzano M, Cimatti A, Sebastiani R. Verifying industrial hybrid systems with MathSAT. Electronic Notes in Theoretical Computer Science, 2005,119(2):17−32. [doi: 10.1016/j.entcs.2004.12.022]

[16] Audemerd G, Cimatti A, Sebastiani R. Bounded model checking for timed systems. In: Peled D, Vardi M, eds. Proc. of the Int'l Conf. on Formal Techniques Networked and Distributed Systems. LNCS 2529, Springer-Verlag, 2002. 243−259.

[17] Li X, Sumit J, Bu L. Towards an efficient path-oriented tool for bounded reachability analysis of linear hybrid systems using linear programming. Electronic Notes in Theoretical Computer Science, 2007,174:57−70. [doi: 10.1016/j.entcs.2006.12.023]

[18] Bu L, Li Y, Wang L, Li X. BACH: Bounded reachability checker for linear hybrid automata. In: Cimatti A, Jones R, eds. Proc. of the 8th Int'l Conf. on Formal Methods in Computer Aided Design. IEEE Computer Society Press, 2008. 65−68. [doi: 10.1109/FMCAD.2008.ECP.13]

[19] Bu L, Li X. Path-Oriented bounded reachability analysis of composed linear hybrid systems. Int'l Journal on Software Tools for Technology Transfer. [doi: 10.1007/s10009-010-0163-9]

[20] Wang F. Symbolic parametric safety analysis of linear hybrid systems with bdd-like data structures. IEEE Trans. on Software Engineering, 2005,31(1):38−51. [doi: 10.1109/TSE.2005.13]

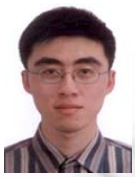
[21] Moura L, Bjørner N. Z3: An efficient SMT solver. In: Ramakrishnan C, Rehof J, eds. Proc. of the Conf. on Tools and Algorithms for the Construction and Analysis of Systems 2008. LNCS 4963, Springer-Verlag, 2008. 337−340.

[22] Bruttomesso R, Cimatti A, Franzen A, Griggio A, Sebastiani R. The MathSAT 4 SMT solver. In: Gupta A, Malik S, eds. Proc. of the Int'l Conf. on Computer Aided Verification 2008. LNCS 5123, Springer-Verlag, 2008. 299−303. [doi: 10.1007/978-3-540-70545-1_28]

[23] Dutertre B, de Moura L. The Yices SMT solver. 2006. http://yices.csl.sri.com/tool-paper.pdf

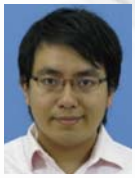
[24] Barrett C, Tinelli C. CVC3. In: Damm W, Hermanns H, eds. Proc. of the Int'l Conf. on Computer Aided Verification 2007. LNCS 4590, Springer-Verlag, 2007. 298−302.

[25] Bofill M, Nieuwenhuis R, Oliveras A, Rodriguez-Carbonell E, Rubio A. The Barcelogic SMT solver. In: Gupta A, Malik S, eds. Proc. of the Int'l Conf. on Computer Aided Verification 2008. LNCS 5123, Springer-Verlag, 2008. 294−298. [doi: 10.1007/978-3-540-70545-1_27]



卜磊(1983－),男,江苏东台人,博士,讲师,主要研究领域为软件工程,形式化方法,软件验证,实时与混成系统.

王林章(1973－),男,博士,副教授,CCF 高级会员,主要研究领域为软件工程,软件测试.



李游(1986－),男,博士生,主要研究领域为软件工程.

李宣东(1963－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为软件工程,软件建模与分析,软件测试与验证.