中国科学: 信息科学 2017 年 第 47 卷 第 3 期: 288–309
SCIENTIA SINICA Informationis
《中国科学》杂志社
SCIENCE CHINA PRESS
论文

基于组合 IIS 路径抽取的组合线性混成系统有界可达性分析优化

解定宝[1], 周岳翔[1], 卜磊[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, 南京 210023
2. 江苏省软件新技术与产业化协同创新中心, 南京 210023
* 通信作者. E-mail: bulei@nju.edu.cn

收稿日期: 2016–03–02; 接受日期: 2016–05–31; 网络出版日期: 2016–09–14
国家重点基础研究发展计划 (973) (批准号: 2014CB340703) 和国家自然科学基金 (批准号: 61561146394, 61572249, 61321491) 资助项目

摘要 混成系统是一类同时具有离散和连续行为的复杂系统, 被广泛应用于控制系统建模. 针对其安全性需求, 对不安全状态进行有界可达性验证, 是保障系统安全的重要手段. 然而, 当前技术所能处理的问题规模和现实生活里的实际需要尚有一定的距离. 特别是组合混成系统由于涉及到各个组件间的协作与同步, 组合状态空间快速爆炸, 对其进行验证具有极高的复杂性. 为控制问题的复杂度, 一种面向路径的可达性分析方法在前期工作中被提出用来对组合线性混成系统进行有界可达性分析. 该方法通过依次枚举潜在路径并进行验证的方式, 有效地提升了所能处理的问题规模. 当面对复杂系统时, 上述面向路径的检测方法将会因为待检测路径数量的急剧上升而使得验证效率大幅降低, 这也是模型检验状态空间爆炸问题的一种体现. 为解决此问题, 本文提出了一种状态空间约减技术以加速验证过程. 当一组路径被判定为不可行时, 定位出导致其不可行的原因, 得到一个组合不可行路径片段. 由于包含同样片段的组合路径一定不可行, 因此在后续的路径遍历里只需要枚举与检验不包含组合不可行路径片段的路径, 从而大幅减少需要检验的路径数量. 此外, 为了有效地规避此类组合路径片段, 我们设计了一种全新的基于 SMT 编码的有界图结构遍历方法. 实验表明, 该优化技术大幅地提升了面向路径有界可达性分析方法的性能, 整体性能也超越了当前最先进的同类工具.

关键词 混成系统, 有界模型检验, 可达性分析, 组合线性混成自动机, 可满足性, 不可约不可解子集

1 引言

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, 被广泛应用于工业控制系统的建模. 近年来, 随着软件失效带来的损失越来越大, 人们对于系统的安全性要求越来越高,

引用格式: 解定宝, 周岳翔, 卜磊, 等. 基于组合 IIS 路径抽取的组合线性混成系统有界可达性分析优化. 中国科学: 信息科学, 2017, 47: 288–309, doi: 10.1360/N112016-00042
Xie D B, Zhou Y X, Bu L, et al. Composed IIS path locating based optimization for bounded reachability analysis of compositional linear hybrid systems (in Chinese). Sci Sin Inform, 2017, 47: 288–309, doi: 10.1360/N112016-00042

尤其是安全攸关领域, 几乎已经到了零容忍的地步. 模型检验 (model checking) [1] 是一种可以自动化发现系统错误的高效方法, 其基本思想是遍历系统模型的状态空间, 找出所有可能的错误, 是保障系统质量的重要手段. 当前, 混成自动机 (hybrid automata) [2] 是一种广泛使用的混成系统建模语言. 因此, 对混成自动机进行模型检验是相关领域的一个重要问题. 由于混成自动机里离散与连续行为的混杂交织, 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), 它的可达性问题已经被证明是不可判定的 [3].

对于单个线性混成自动机, 经典模型检验技术的主要思想是基于多面体计算来判定系统的完整可达状态集. 其基本步骤是从初始状态集开始, 通过后像操作不断地计算系统的下一时刻可达状态集, 直至系统可达状态集不再变化, 算法终止. 当拥有了系统的完整可达状态集后, 即可检查系统行为是否满足安全性规约. 在单自动机的基础上, 为了对具有多个组件的系统进行建模, 出现了组合线性混成系统. 该类系统由多个线性混成自动机组成, 系统内的成员自动机通过共享事件来完成协作与同步. 现有的验证技术大多是通过笛卡尔乘积来将组合线性混成自动机转换成一个单线性混成自动机, 然后再使用上述可达性检验方法进行分析. 然而, 单纯的笛卡尔乘积会使得系统的状态随着子系统数目的增加成指数级增长, 极大地限制了相关方法所能处理的问题规模. 现有的基于此类计算的模型检验工具 HyTech [4], PHAVer [5], SpaceEx [6] 等, 能处理的问题规模离实际需要尚有一定的距离. 另一方面, 由于混成自动机可达性问题的不可判定性, 基于这种技术的工具不能保证终止.

近年来, 有界模型检验 (bounded model checking, 简称 BMC) [7] 作为经典模型检验的一种补充方法, 越来越受到学者的关注. 其主要思想是通过一个整数 k 来限制模型行为步数, 将系统 k 步内行为编码成一组约束, 通过约束求解来高效地找出 k 步内的所有错误. 虽然 BMC 的方法因为只能验证阈值内的系统性质而缺失了模型检验的完备性, 但是通过限定模型行为的步数, 其在发现错误的能力上超越了传统模型检验方法. 此外, 由于 BMC 只需要遍历阈值内的系统状态空间, 使得其能处理的系统规模得到了提升, 这也是 BMC 得到广泛认可的优势所在.

BMC 的思想同样被运用到线性混成自动机的有界可达性检验上. 特别是在 SMT (satisdiability modulo theories) [8] 技术取得巨大进步后, 目前主流的方法是将线性混成自动机 k 步内的行为编码成一组 SMT 约束, 然后调用高效的 SMT 求解器来对相应约束集进行求解. 由于近十年来 SMT 求解器的高速发展, 这类方法已经具有不错的性能 [9,10]. 但是, 由于该方法需要在检验前将系统 k 步内的行为整体一次性编码, 当模型规模、给定阈值等增长后, 相应的约束集大小将快速增长, 特别是面向组合系统的时候, 由于成员自动机同步带来的笛卡尔乘积状态空间爆炸, 其对应的约束集规模更是急剧增长, 进而超出当前 SMT 求解器的处理能力, 严重制约了可处理问题的规模.

为解决此问题, 一种面向路径的有界可达性检验方法被学者提出以控制单次验证的问题规模 [11]. 其主要思想是将线性混成自动机的连续和离散行为分层考虑处理, 在离散的自动机图结构上使用深度优先搜索 (depth first search, 简称 DFS) 遍历路径, 然后在连续层予以编码与验证 [12]. 该方法有效地控制了单次求解的问题复杂度, 进而提升了能处理的系统规模. 然而, 在这种面向路径的有界可达性分析方法里, 当系统较为复杂或者阈值较大, 候选路径的数量将会急剧增加, 逐个遍历并验证的方法将会消耗大量的计算时间, 严重制约了相关方法的性能.

在单个线性混成自动机面向路径有界可达性检验里, 一种 SAT-LP-IIS 联合反馈制导的分析方法被提出, 以解决上述路径爆炸问题 [13]. 其基本思想是当一条路径被证明为不可行时, 利用线性规划里的不可约不可解子集 (irreducible infeasible subset, 简称 IIS) 技术从中定位出不可行的原因 [14], 并学习出经验教训, 从而约减掉同一类不可行的路径, 减少需验证的路径数量. 实验表明该方法有效地约减了候选路径数量, 极大地提升了面向路径有界可达性分析的性能.

在上述工作基础上, 本文提出一种方法在组合系统层面进行不可行子路径组抽取, 依据同步语义定位此路径组关联的组件及路径片段, 并设计了新的路径遍历与枚举算法, 从而在后续验证中规避含有相关不可行组合路径片段的路径组, 从而提升组合线性混成系统可达性分析的规模和性能. 具体而言, 本文的主要贡献如下所述.

• 在文献 [14] 中, 当我们判定一条路径为不可行时, 借助 IIS 技术可以从中直接截取不可行子路径片段. 然而, 在组合线性混成系统可达性分析里, 由于涉及到多组件间路径同步等问题, 相关的直接截取方法不再适用. 因此依据组合自动机的同步语义, 本文提出了一种面向组合线性混成系统的不可行组合路径片段抽取方法, 用以从一条不可行的路径组里定位出一条不可行组合路径片段. 我们称这样的不可行组合路径片段为 "组合 IIS 路径". 然后, 相关组合 IIS 路径将会被反馈到后续路径遍历过程, 从而约减同一类的不可行路径组, 减少需要验证的路径组数目. 更进一步, 由于一组不可满足的约束里可能存在多个 IIS, 本文基于 MARCO [15] 算法实现了一个可以从一组不可满足的线性约束里抽取出多个 IIS 的算法, 用以从一条不可行的路径组里定位出多个组合 IIS 路径, 从而更进一步约减待遍历的状态空间, 提高验证效率.

• 由于任何包含组合 IIS 路径的潜在路径组在编码过程中都将会生成同样的 IIS, 从而导致该路径组不可行. 为了提高路径枚举的效率, 在后续的路径遍历里需要规避相关组合 IIS 路径. 由于传统的 DFS 搜索是一种局部搜索算法, 每次只看下一个即将要遍历的节点, 无法提前预知是否将会遍历一个已知的 IIS 路径, 这将会导致路径遍历算法有可能会将大量时间浪费在遍历包含组合 IIS 路径的路径组上. 为此, 我们提出了一种全新的基于 SMT 编码的有界图结构全局遍历算法, 可以提前规避已发现的组合 IIS 路径, 提升路径枚举的效率.

• 为了评估本文提出方法的有效性, 我们在组合线性混成系统有界可达性检验工具 BACH [16] 上集成了上述可达性分析方法, 并在一组被广泛使用的组合混成系统模型上评估了该工具的性能. 实验表明, 该方法可有效提升组合线性混成系统面向路径有界可达性检验的性能. 此外, 和当前最先进 (state-of-the-art) 的同类验证工具 HyCOMP [17] 相比, BACH 的性能也整体领先.

本文的组织如下: 第 2 节回顾线性混成自动机的定义及在此基础上的面向路径可达性检验技术, 接着将相关定义和检验技术拓展到组合线性混成系统上, 并简要介绍组合线性混成系统的面向路径有界可达性检验方法. 之后第 3 节将介绍本文的主要贡献: 一种组合线性混成系统面向路径有界可达性检验的状态空间约减技术. 其主要思想是从路径验证历史中学习出错误信息, 从而可以约减同一类的不可行路径, 提升可达性检验效率. 第 4 节将通过实验数据评估比较本方法和同类工具的性能. 第 5 节对现有的相关工作给予简要介绍. 最后, 第 6 节给出全文总结.

2 理论背景和原理

2.1 线性混成自动机及路径、行为定义

首先, 给出本文工作所基于的线性混成自动机的形式化定义.

定义1 一个线性混成自动机为多元组 $H = (X, \Sigma, V, E, V^0, \alpha, \beta, \gamma)$, 其中

- X 为实数变量的有限集合.
- Σ 为转换名的有限集合.
- V 为控制节点的有限集合.
- E 为形如 $(v, \sigma, \phi, \psi, v')$ 的转换集合, 其中

- $v, v' \in V$, $\delta \in \Sigma$,

- ϕ 是形如 $a \leqslant \sum_{i=0}^{l} c_i x_i \leqslant b$ 的转换卫式集合, 其中 $x_i \in X$ $(0 \leqslant i \leqslant l)$, $a, b, c_i \in \mathbb{R}$, 并且 a 可以为 $-\infty$, b 可以为 ∞,

- ψ 是形如 $x_i := c_i$ 的重置动作集合, 其中 $x_i \in X$, $c_i \in \mathbb{R}$.

• $V^0 \in V$ 为初始节点集合.

• α 是一个为 V 中所有节点添加节点不变式的标注函数, 节点不变式是形如 $a \leqslant \sum_{i=0}^{l} c_i x_i \leqslant b$ 的约束集合, 其中 $x_i \in X$ $(0 \leqslant i \leqslant l)$, $a, b, c_i \in \mathbb{R}$, 并且 a 可以为 $-\infty$, b 可以为 ∞.

• β 是一个为 V 中所有节点添加流条件的标注函数, 流条件是形如 $\dot{x_i} \in [a, b]$ 的变化率集合, 其中 $x_i \in X$ $(0 \leqslant i \leqslant l)$, $a, b \in \mathbb{R}$, $a \leqslant b$. 对任意节点 v 以及任意变量 $x_i \in X$, 有且仅有一个 x_i 的流条件 $\dot{x_i} \in [a, b] \in \beta(v)$.

• γ 是一个标注函数, 它将初始位置中每个位置映射到一组初始条件, 初始条件具有形式 $x_i := a$ 其中 $x_i \in X$ $(0 \leqslant i \leqslant l)$, $a \in \mathbb{R}$. 对任意初始节点 $v \in V^0$ 以及任意变量 $x_i \in X$, 有且仅有一个 x_i 的初始条件 $x_i := a \in \gamma(v)$.

给定线性混成自动机 $H = (X, \Sigma, V, E, V^0, \alpha, \beta, \gamma)$, 将形如 $\rho = \langle v_0 \rangle \xrightarrow[\sigma_0]{(\phi_0, \psi_0)} \langle v_1 \rangle \xrightarrow[\sigma_1]{(\phi_1, \psi_1)} \cdots \xrightarrow[\sigma_{n-1}]{(\phi_{n-1}, \psi_{n-1})} \langle v_n \rangle$ 的一个位置序列称为路径片段 (path segment), 其中对任意 $i (0 \leqslant i < n)$, 此序列满足 $(v_i, \sigma_i, \phi_i, \psi_i, v_{i+1}) \in E$. 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 (path).

通过给路径中每个节点赋一个非负实数 δ_i, 我们可以生成此路径的时间序列 (timed sequence):

$$\left\langle \begin{matrix} v_0 \\ \delta_0 \end{matrix} \right\rangle \xrightarrow[\sigma_0]{(\phi_0, \psi_0)} \left\langle \begin{matrix} v_1 \\ \delta_1 \end{matrix} \right\rangle \xrightarrow[\sigma_1]{(\phi_1, \psi_1)} \cdots \xrightarrow[\sigma_{n-1}]{(\phi_{n-1}, \psi_{n-1})} \left\langle \begin{matrix} v_n \\ \delta_n \end{matrix} \right\rangle.$$

此时间序列可以表示 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 $\zeta_i(x)$ 表示自动机沿着时间序列运行, 在位置 v_i 停留 δ_i 时间单位后变量 x 的值, 用变量 $\lambda_i(x)$ 表示自动机行为抵达位置 v_i 时变量 x 的值 $(0 \leqslant i \leqslant n, x \in X)$. 当 $x = a \in \gamma(v_0)$, $\lambda_0(x) = a$; 当 $0 \leqslant i < n$, 如果 $x := d \in \psi_i$, $\lambda_{i+1}(x) = d$, 否则 $\lambda_{i+1}(x) = \zeta_i(x)$.

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

$$\left\langle \begin{matrix} v_0 \\ \delta_0 \end{matrix} \right\rangle \xrightarrow[\sigma_0]{(\phi_0, \psi_0)} \left\langle \begin{matrix} v_1 \\ \delta_1 \end{matrix} \right\rangle \xrightarrow[\sigma_1]{(\phi_1, \psi_1)} \cdots \xrightarrow[\sigma_{n-1}]{(\phi_{n-1}, \psi_{n-1})} \left\langle \begin{matrix} v_n \\ \delta_n \end{matrix} \right\rangle$$

表示 H 的行为 (behavior) 当且仅当其满足以下条件:

• H 中存在路径 $\langle v_0 \rangle \xrightarrow[\sigma_0]{(\phi_0, \psi_0)} \langle v_1 \rangle \xrightarrow[\sigma_1]{(\phi_1, \psi_1)} \cdots \xrightarrow[\sigma_{n-1}]{(\phi_{n-1}, \psi_{n-1})} \langle v_n \rangle$;

• $\delta_1, \delta_2, \ldots, \delta_n$ 保证每个变量 $x \in X$ 在节点 v_i $(0 \leqslant i \leqslant n)$ 上的行为符合流条件的要求: 如果 $\dot{x} = [u_i, u_i'] \in \beta(v_i)$, 则 $u_i \delta_i \leqslant \zeta_i(x) - \lambda_i(x) \leqslant u_i' \delta_i$;

• $\delta_1, \delta_2, \ldots, \delta_n$ 满足转换卫式 ϕ_i $(1 \leqslant i \leqslant n-1)$ 中所有约束, 例如对 ϕ_i 中任一变量约束 $a \leqslant \sum_{k=0}^{m} c_k x_k \leqslant b$, 满足 $a \leqslant \sum_{k=0}^{m} c_k \zeta_i(x_k) \leqslant b$;

• $\delta_1, \delta_2, \ldots, \delta_n$ 满足节点 v_i $(1 \leqslant i \leqslant n)$ 状态不变式中所有的变量约束:

- 对 $\alpha(v_i)$ 中任意变量约束 $a \leqslant \sum_{k=0}^{m} c_k x_k \leqslant b$, 满足 $a \leqslant \sum_{k=0}^{m} c_k \zeta_i(x_k) \leqslant b$;

- 对 $\alpha(v_i)$ 中任意变量约束 $a \leqslant \sum_{k=0}^{m} c_k x_k \leqslant b$, 满足 $a \leqslant \sum_{k=0}^{m} c_k \lambda_i(x_k) \leqslant b$.

定义3 对线性混成自动机 $H=(X,\Sigma,V,E,V^0,\alpha,\beta,\gamma)$, 其可达性规约 $\mathcal{R}(v,\varphi)$ 由如下部分组成: 节点 $v\in V$, 变量约束集 φ, φ 为形如 $a\leqslant\sum_{i=0}^{l}c_ix_i\leqslant b$ 的约束集合, 其中 $x_i\in X$ $(0\leqslant i\leqslant l)$, $a,b,c_i\in\mathbb{R}$, 并且 a 可以为 $-\infty$, b 可以为 ∞.

2.2 组合线性混成自动机、路径、行为及面向路径可达性编码

定义4 给定线性混成自动机 $H_1=(X_1,\Sigma_1,V_1,V_1^0,E_1,\alpha_1,\beta_1,\gamma_1)$ 和 $H_2=(X_2,\Sigma_2,V_2,V_2^0,E_2,\alpha_2,\beta_2,\gamma_2)$, 其中 $X_1\cap X_2=\emptyset$. H_1 和 H_2 的组合 $H_1||H_2$, 是一个自动机 $N=(X,\Sigma,V,V^0,E,\alpha,\beta,\gamma)$, 其中

- $X=X_1\cup X_2,\Sigma=\Sigma_1\cup\Sigma_2,V=V_1\times V_2,V^0=V_1^0\times V_2^0,\alpha((v_1,v_2))=\alpha(v_1)\cup\alpha(v_2),\beta((v_1,v_2))=\beta(v_1)\cup\beta(v_2),\gamma((v_1,v_2))=\gamma(v_1)\cup\gamma(v_2)$;
- E 的定义如下:

- 对于 $a\in\Sigma_1\cap\Sigma_2$, 对于 E_1 中的每一个 (v_1,a,ϕ,ψ,v_1') 和 E_2 中的每一个 (v_2,a,ϕ,ψ,v_2'), E 包含 $((v_1,v_2),a,\phi_1\cup\phi_2,\psi_1\cup\psi_2,(v_1',v_2'))$;

- 对于 $a\in\Sigma_1\setminus\Sigma_2$, 对于 E_1 中的每一个 (v,a,ϕ,ψ,v') 和 V_2 中的每一个 t, E 包含 $((v,t,),a,\phi,\psi,(v',t))$;

- 对于 $a\in\Sigma_2\backslash\Sigma_1$, 对于 E_2 中的每一个 (v,a,ϕ,ψ,v') 和 V_1 中的每一个 t, E 包含 $((t,v),a,\phi,\psi,(t,v'))$.

对于 $m>2$, 线性混成自动机 $H_1,H_2,\ldots,H_m$ 的组合 $H_1||H_2||\cdots||H_m$ 可以被递归定义为 $H_1||H_2||\cdots||H_m=H_1||H'$, 其中 $H'=H_2||H_3||\cdots||H_m$.

给定组合线性混成自动机 $N=H_1||H_2||\cdots||H_m$, 其中每个成员 H_i 为线性混成自动机 $(X_i,\Sigma_i,V_i,V_i^0,E_i,\alpha_i,\beta_i,\gamma_i)(1\leqslant i\leqslant m)$, ρ 为 N 中一条路径, 形如: $\rho=\langle v_0\rangle\xrightarrow[\sigma_0]{(\phi_0,\psi_0)}\langle v_1\rangle\xrightarrow[\sigma_1]{(\phi_1,\psi_1)}\cdots\xrightarrow[\sigma_{n-1}]{(\phi_{n-1},\psi_{n-1})}\langle v_n\rangle$, 路径中每个节点 $v_i=(v_{i1},v_{i2},\ldots,v_{im})(0\leqslant i\leqslant n)$, 其中 $v_{ik}\in V_k$ $(1\leqslant k\leqslant m)$. 对于任意正整数 k $(1\leqslant k\leqslant m)$, 我们可以按照如下方式根据 ρ 构建节点序列 ρ_k: 将 ρ 中每个节点 v_i 替换为 v_{ik} $(0\leqslant i\leqslant n)$, 对任意转换 $\xrightarrow[\sigma_{i-1}]{(\phi_{i-1},\psi_{i-1})}\langle v_{ik}\rangle$ $(1\leqslant i\leqslant n)$, 如果 $(v_{i-1k},\sigma_{i-1},\phi,\psi,v_{ik})\in E_k$ 则将其替换为 $\xrightarrow[\sigma_{i-1}]{(\phi,\psi)}\langle v_{ik}\rangle$, 否则将其删去.

按照以上方法构造出的是节点序列 ρ_k 为自动机 H_k 中的一条路径, 我们称 ρ_k 为 ρ 在自动机 H_k 上的投影 (projection). 直观上而言, 当组合自动机 N 沿着路径 ρ 运行时, 其在成员自动机 H_k 上的运行路径即为 ρ_k.

上文定义了组合自动机中一条路径与各成员自动机的路径之间的投影关系. 接下来, 基于此关系, 我们将定义组合自动机中一条路径的可达性与各成员自动机中一个路径组的可达性之间的关系.

定义5 给定组合线性混成自动机 $N=H_1||H_2||\cdots||H_m$, 可达性规约 $\mathcal{R}(v,\varphi)$, 以及路径组 $P=\{\rho_1,\rho_2,\ldots,\rho_m\}$, 其中每条路径 ρ_i 为成员自动机 H_i $(1\leqslant i\leqslant m)$ 中的一条有限路径. P 满足 $\mathcal{R}(v,\varphi)$ 当且仅当 N 中存在路径 ρ 满足以下条件:

- 路径 ρ 在成员自动机 H_i 上的投影为路径 ρ_i $(1\leqslant i\leqslant m)$;
- 存在 N 基于路径 ρ 的一个行为满足 $\mathcal{R}(v,\varphi)$.

定义6 给定组合线性混成自动机 $N=H_1||H_2||\cdots||H_m$, 其中每个成员自动机 H_i 为线性混成自动机 $(X_i,\Sigma_i,V_i,V_i^0,E_i,\alpha_i,\beta_i,\gamma_i)(1\leqslant i\leqslant m)$. 定义 N 的运行轨迹 (Trail) τ 为多元组 $(\omega_1,\omega_2,\ldots,\omega_m)$, 其中 ω_i $(1\leqslant i\leqslant m)$ 为成员自动机 H_i 的行为

$$\left\langle\begin{matrix}v_{i0}\\ \delta_{i0}\end{matrix}\right\rangle\xrightarrow[\sigma_{i0}]{(\phi_{i0},\psi_{i0})}\left\langle\begin{matrix}v_{i1}\\ \delta_{i1}\end{matrix}\right\rangle\xrightarrow[\sigma_{i1}]{(\phi_{i1},\psi_{i1})}\cdots\xrightarrow[\sigma_{in_i-1}]{(\phi_{in_i-1},\psi_{in_i-1})}\left\langle\begin{matrix}v_{in_i}\\ \delta_{in_i}\end{matrix}\right\rangle,$$

且所有行为之间符合同步约束 (synchronization constraint), 如对任意 k,j $(1\leqslant k,j\leqslant m)$, $\delta_{k0}+\delta_{k1}+\cdots+\delta_{kn_k}=\delta_{j0}+\delta_{j1}+\cdots+\delta_{jn_j}$, 对在 ω_k 中第 d 次出现的集合 $\Sigma_k\cap\Sigma_j$ 中元素 σ_{kp} $(0\leqslant p\leqslant n_k)$, 必然在 ω_j 中同样存在第 d 次出现的集合 $\Sigma_k\cap\Sigma_j$ 中元素 $\sigma_{jq}(0\leqslant q\leqslant n_j)$, 并且 $\sigma_{kp}=\sigma_{jq}$, $\delta_{k0}+\delta_{k1}+\cdots+\delta_{kp}=\delta_{j0}+\delta_{j1}+\cdots+\delta_{jq}$.

定义7 给定组合线性混成自动机 $N=H_1||H_2||\cdots||H_m$, 其中 $H_i=(X_i,\Sigma_i,V_i,V_i^0,E_i,\alpha_i,\beta_i,\gamma_i)$ $(1\leqslant i\leqslant m)$, $\tau=(\omega_1,\omega_2,\ldots,\omega_m)$ 为 N 的一个轨迹, 其中 ω_i $(1\leqslant i\leqslant m)$ 形如

$$\left\langle\begin{matrix}v_{i0}\\ \delta_{i0}\end{matrix}\right\rangle\xrightarrow[\sigma_{i0}]{(\phi_{i0},\psi_{i0})}\left\langle\begin{matrix}v_{i1}\\ \delta_{i1}\end{matrix}\right\rangle\xrightarrow[\sigma_{i1}]{(\phi_{i1},\psi_{i1})}\cdots\xrightarrow[\sigma_{in_i-1}]{(\phi_{in_i-1},\psi_{in_i-1})}\left\langle\begin{matrix}v_{in_i}\\ \delta_{in_i}\end{matrix}\right\rangle.$$

给定可达性规约 $\mathcal{R}(v,\varphi)$, τ 满足 $\mathcal{R}(v,\varphi)$ 当且仅当 $v=(v_{1n_1},v_{2n_2},\ldots,v_{mn_m})$, 并且当自动机 H_i 在节点 v_{in_i} 停留 δ_{in_i} $(1\leqslant i\leqslant m)$ 时间单位后, 相关变量取值满足可达性约束 φ, 例如, 对 φ 中任意约束 $a\leqslant\sum_{k=0}^{l}c_kx_k\leqslant b$, 满足 $a\leqslant\sum_{k=0}^{l}c_k\zeta_n(x_k)\leqslant b$.

基于定义 2, 6 和 7, 给定组合线性混成自动机 $N=H_1||H_2||\cdots||H_m$ 以及一个路径组 $P=\{\rho_1,\rho_2,\ldots,\rho_m\}$, 其中 ρ_i 为成员自动机 H_i $(1\leqslant i\leqslant m)$ 中的一条路径. 对于可达性规约 $\mathcal{R}(v,\varphi)$, 我们可以将 P 是否满足 $\mathcal{R}(v,\varphi)$ 编码成一组线性约束的可满足性问题. 通过使用线性规划技术检验 $\Theta(P,\mathcal{R}(v,\varphi))$ 的可满足性来判定 P 是否满足可达性规约 $\mathcal{R}(v,\varphi)$.

2.3 组合线性混成自动机面向路径有界可达性检验

上节介绍了组合线性混成自动机的定义以及如何将其图结构里的一条路径组编码成一组线性约束, 通过判定这组约束的可满足性来决定在组合系统里是否存在一个对应的行为. 如果存在, 则组合线性混成系统里存在一个行为满足可达性规约, 即找到一个反例; 反之, 则该组合路径不可行 [18].

有界可达性分析 (bounded reachability analysis) 的目标是在给定阈值内找到系统里的一条满足可达性规约的轨迹. 既然我们已经可以验证组合线性混成自动机图结构上的一条路径组的可达性, 如果我们在系统的图结构上逐一遍历并验证每一条候选路径组的可满足性, 即可回答组合线性混成自动机的有界可达性问题. 此外, 由于图结构上在某个阈值下的候选路径的数量是有穷的, 该算法保证终止.

由于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, 我们称此为共享事件序列 (share label sequence, 简称 SLS). 为了遍历并枚举出符合条件的候选路径组, 一种共享事件序列制导的深度优先遍历 (share label sequence guided depth first search, 简称 SLS-DFS) 算法被提出以达到此目标 [12]. 该算法在处理组合线性混成自动机的图结构时, 只在第一个成员自动机上采用简单深度优先遍历, 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. 基于此算法, 我们可以在组合混成系统的图结构上遍历并枚举出符合同步规约的路径组.

图 1 展示的是一个核反应堆控制系统的混成自动机模型, 其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. 该系统的成员自动机包括一个控制器模型以及若干个控制棒模型. 我们将以此系统为例, 来阐述组合线性混成系统面向路径有界可达性分析的过程, 假定该系统里有两根控制棒和一个控制器, 阈值为 $\overline{k}=\langle5,6,8\rangle$, 其中 5, 6, 8 分别是系统内 3 个成员自动机的阈值. 第一条被枚举出的路径为

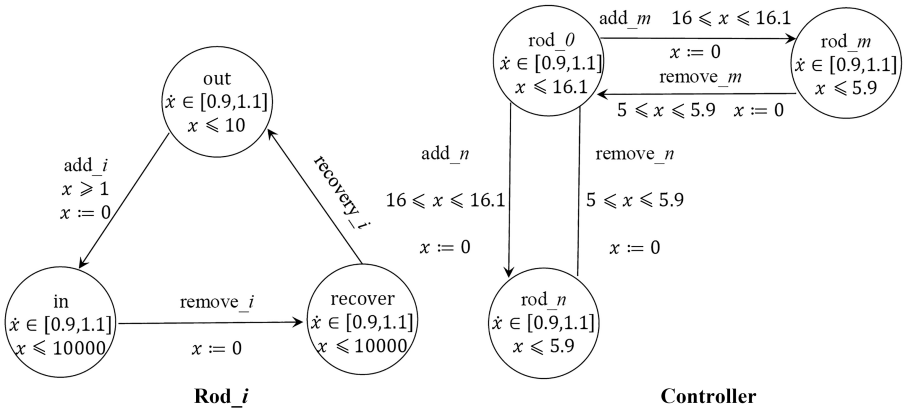

图 1　核反应堆控制系统混成自动机模型

Figure 1　Hybrid automata for Nuclear reactor system (NRS)

$\text{rod\_1} : \langle \text{out} \rangle \overset{\text{add\_1}}{\longrightarrow} \langle \text{in} \rangle \overset{\text{remove\_1}}{\longrightarrow} \langle \text{recover} \rangle$,

$\text{rod\_2} : \langle \text{out} \rangle \overset{\text{add\_2}}{\longrightarrow} \langle \text{in} \rangle \overset{\text{remove\_2}}{\longrightarrow} \langle \text{recover} \rangle \overset{\text{recover\_2}}{\longrightarrow} \langle \text{out} \rangle \overset{\text{add\_2}}{\longrightarrow} \langle \text{in} \rangle \overset{\text{remove\_2}}{\longrightarrow} \langle \text{recover} \rangle$,

$\text{controller} : \langle \text{rod\_0} \rangle \overset{\text{add\_2}}{\longrightarrow} \langle \text{rod\_2} \rangle \overset{\text{remove\_2}}{\longrightarrow} \langle \text{rod\_0} \rangle \overset{\text{add\_2}}{\longrightarrow} \langle \text{rod\_2} \rangle \overset{\text{remove\_2}}{\longrightarrow} \langle \text{rod\_0} \rangle \overset{\text{add\_1}}{\longrightarrow} \langle \text{rod\_1} \rangle \overset{\text{remove\_1}}{\longrightarrow} \langle \text{rod\_0} \rangle$,

组合混成系统里一条路径组的编码由其成员自动机的路径编码和时间同步编码构成. 对于成员自动机内的路径编码, 我们以 rod_1 自动机里的位置节点 out 为例来说明路径编码方法是如何工作的.

- 由于自动机在此节点的停留时间是非负的, 故有约束 $\delta_{\text{out}}^{\text{rod\_1}} \geqslant 0$.
- 根据其流条件 $\dot{x} = [0.9, 1.1]$ 将生成约束 $\zeta_{\text{out}}^{\text{rod\_1}}(x) - \lambda_{\text{out}}^{\text{rod\_1}}(x) \geqslant 0.9 \times \delta_{\text{out}}^{\text{rod\_1}}$, $\zeta_{\text{out}}^{\text{rod\_1}}(x) - \lambda_{\text{out}}^{\text{rod\_1}}(x) \leqslant 1.1 \times \delta_{\text{out}}^{\text{rod\_1}}$.
- 根据节点不变式 $x \leqslant 10$ 生成约束 $\lambda_{\text{out}}^{\text{rod\_1}}(x) \leqslant 10$, $\zeta_{\text{out}}^{\text{rod\_1}}(x) \leqslant 10$.

其中, $\lambda_{\text{out}}^{\text{rod\_1}}(x)$ 和 $\zeta_{\text{out}}^{\text{rod\_1}}(x)$ 分别表示自动机进入和离开节点 out 时变量 x 的值. $\delta_{\text{out}}^{\text{rod\_1}}$ 表示自动机在该节点的停留时间. 其他成员自动机和节点的约束生成规则依此类推. 接下来将介绍时间同步约束:

- 自动机 rod_1 和自动机 controller 的共享事件 add_1 的同步将会生成时间约束 $\delta_{\text{out}}^{\text{rod\_1}} = \delta_{\text{rod\_0}}^{\text{controller}} + \delta_{\text{rod\_2}}^{\text{controller}} + \delta_{\text{rod\_0}_2}^{\text{controller}} + \delta_{\text{rod\_2}_2}^{\text{controller}} + \delta_{\text{rod\_0}_3}^{\text{controller}}$.
- 自动机 rod_1 和自动机 controller 同时到达目标节点将会生成约束: $\delta_{\text{out}}^{\text{rod\_1}} + \delta_{\text{in}}^{\text{rod\_1}} + \delta_{\text{recover}}^{\text{rod\_1}} = \delta_{\text{rod\_0}}^{\text{controller}} + \delta_{\text{rod\_2}}^{\text{controller}} + \delta_{\text{rod\_0}_2}^{\text{controller}} + \delta_{\text{rod\_2}_2}^{\text{controller}} + \delta_{\text{rod\_0}_3}^{\text{controller}} + \delta_{\text{rod\_1}}^{\text{controller}} + \delta_{\text{rod\_0}_4}^{\text{controller}}$.

其他时间同步以此类推, 不再赘述.

经过编码得到一组线性约束后, 调用线性规划求解器证明此组约束不可解, 从而判定该路径组不可行. 接下来, 按照同样的步骤, 不断地遍历出候选路径组并进行编码求解. 最后, 算法将图结构上阈值内的 14 个候选路径组全部遍历并证明不可行后, 判定该组合系统在此组阈值下的可达性规约不可满足.

和传统的将系统阈值内所有行为编码成 SMT 约束再求解的方法相比, 面向路径的有界可达性检验方法有效地控制了单次验证的复杂度, 从而能够处理更大规模的组合线性混成系统. 在文献 [12] 中的实验评估里, 这种面向路径的有界可达性检验方法已经显示了较好的可扩展性 (scalability), 能够分析一些较大规模的组合线性混成系统.

然而, 这种面向路径的有界可达性检验方法的性能极其依赖于待验证的路径数量. 对于上述只有

3 个成员自动机的简单组合线性混成系统, 其在给定较小阈值 $\overline{k} = \langle 5, 6, 8\rangle$ 的情况下, 图结构上依然存在 14 个候选路径组. 可以预见, 当系统较为复杂或者给定较大阈值时, 候选路径的数量将会急剧上升, 基本的面向路径可达性检验方法的性能也会随之大幅下降.

3 基于组合 IIS 路径的状态空间约减优化

上一节介绍了一种基于深度优先搜索的组合线性混成自动机的有界可达性分析方法. 其主要思想是枚举并验证组合混成系统图结构里的每一条候选路径组, 由于阈值内的候选路径组数量是有穷的, 算法保证终止. 另一方面, 得益于每次只需要验证一条路径的可行性, 该方法有效地控制了单次验证的复杂度从而可以验证较大规模的系统.

然而在第 2.3 小节里的例子中, 对于一个只有 3 个成员自动机简单组合混成系统, 在给定较小阈值的情况下, 基本的面向路径有界可达性检验方法需要枚举并验证 14 个路径组的可达性. 事实上, 当系统规模较大或者给定较大阈值时, 该方法需要检测的路径数量将会急剧上升从而导致验证时间大幅增加.

3.1 单自动机 IIS 路径定位方法

在单个线性混成自动机的面向路径有界可达性检验过程中, 也同样遇到了类似的路径爆炸问题. 当系统模型的图结构较为简单或者给定较小阈值时, 基本的面向路径可达性检验方法展示了不错的性能. 而一旦待验证的候选路径数量上升后, 该方法的性能也会随之下降.

文献 [14] 提出了一种基于不可约不可解子集 (irreducible infeasible subset, 简称 IIS) [19] 技术的方法来解决此问题. 其基本思想是当一条路径被证明为不可行时, 利用 IIS 技术挖掘出造成其不可行的根本原因. 然后从中学习出经验教训, 从而在后续的路径遍历过程中规避一些必然不可行的候选路径. 通过这种方法, 可以一次判定同一类路径位不可行, 极大地约减了待检验路径的数量, 从而提升有界可达性分析的性能. 接下来, 我们将介绍 IIS 技术的定义并回顾如何从一条不可行的路径中定位出一条 IIS 路径片段.

定义8 一组不可约不可解子集是一个极小的不可协调集. 形式化地说, 线性约束集 $\mathbb{C}$ 的一个 IIS 是这样的一个集合 $\mathbb{C}' \subseteq \mathbb{C}$, 并且 $\mathbb{C}'$ 不可满足; 对于 $\mathbb{C}'$ 的任意真子集 $\mathbb{C}''$, $\mathbb{C}''$ 可满足.

直观地说, 一个不可解的线性约束集合的 IIS 是一个移除其中任意约束都会变得可解的子集. 另一方面, 根据文献 [19], 从一个不可解的线性约束里定位出 IIS, 是 “简单, 比较高效以及容易实现” 的. 目前有很多工具提供 IIS 分析功能, 既有商用的高性能求解器比如 CPLEX [1)], 也有开源的相关工具, 比如 Z3 [10] 等.

当一条路径 ρ 被判定为不可行时, 借助于 IIS 分析技术, 我们能从 ρ 对应的线性约束 $\mathbb{C}$ 里定位出一个 IIS, $\mathbb{C}'$. 根据定义 2, $\mathbb{C}$ 里的每个约束都是根据路径上的语法元素, 比如路径上每个节点的不变式, 转换上的卫式等顺序生成的. 根据上述编码规则, 我们可以将 $\mathbb{C}'$ 里的每个约束直接定位到生成该约束的源节点、转换. 只要定位出路径 ρ 中和 $\mathbb{C}'$ 相关的最前及最后节点, 即可直接截取以这两个节点为起始节点的路径片段. 很明显, 该路径片段就是要找的对应于 IIS 约束 $\mathbb{C}'$ 的不可行子路径片段. 任意包含相关不可行子路径片段的路径, 其对应的整体路径约束中必然包含相关不可行子路径片段对应的约束. 由定义 2 可知, 所有约束之间是合取关系, 因此整体约束集必然不可解, 无需进行遍历和判定.

1) CPLEX. http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/.

3.2 组合 IIS 路径定位

受到上述工作的启发, 我们尝试将此思想运用到组合线性混成系统的面向路径有界可达性检验里, 从而约减需验证的路径数量, 提升验证效率. 与工作 [14] 类似, 当一条候选路径组被判定为不可行时, 利用线性规划求解器可以从该路径组对应的线性约束编码里定位出一个 IIS. 然而, 在组合线性混成系统可达性分析里, 由于涉及到多组件间路径同步等问题, 单自动机上直接截取的方法不再适用. 因此依据组合自动机的同步语义, 接下来将介绍如何在保证同步语义正确的情况下, 将此 IIS 约束映射回系统图结构上, 得到一个组合 IIS 路径.

定义9 给定组合线性混成自动机 $N = H_1||H_2||\cdots||H_m$, 其中每个成员自动机 H_i 为线性混成自动机 $(X_i, \Sigma_i, V_i, V_i^0, E_i, \alpha_i, \beta_i, \gamma_i)(1 \leqslant i \leqslant m)$, 以及路径组 $P = \{\rho_1, \rho_2, \ldots, \rho_m\}$, 其中每条路径 ρ_i 为成员自动机 H_i $(1 \leqslant i \leqslant m)$ 中的一条有穷路径. 在根据路径组 P 编码生成的线性约束集 $\mathbb{C}$ 里, 对于涉及到成员自动机本地路径编码的每个约束 $\nabla_l \in \mathbb{C}$, ∇_l 在 P 里的源节点集合 $\mathbb{V}_{\nabla_l}$ 的定义如下:

- 如果 ∇_l 是根据自动机 $H_i(1 \leqslant i \leqslant m)$ 在节点 v_k 的停留时间生成的, $\delta_{v_k} \geqslant 0$, $v_k^i \in \mathbb{V}_{\nabla_l}$;
- 如果 ∇_l 是根据自动机 $H_i(1 \leqslant i \leqslant m)$ 里的转换 e_k 上的卫式 ϕ 生成的, $v_{k+1}^i \in \mathbb{V}_{\nabla_l}$, 如果 $i > 0$, $v_k^i \in \mathbb{V}_{\nabla_l}$;
- 如果 ∇_l 是根据自动机 $H_i(1 \leqslant i \leqslant m)$ 里的转换 e_k 上的重置式 ψ 生成的, $v_{k+1}^i \in \mathbb{V}_{\nabla_l}$, 如果 $i > 0$, $v_k^i \in \mathbb{V}_{\nabla_l}$;
- 如果 ∇_l 是根据自动机 $H_i(1 \leqslant i \leqslant m)$ 在节点 v_k 的流条件 $\beta(v_k)$ 生成的, $v_{k+1}^i \in \mathbb{V}_{\nabla_l}$, 如果 $i > 0$, $v_k^i \in \mathbb{V}_{\nabla_l}$;
- 如果 ∇_l 是根据自动机 $H_i(1 \leqslant i \leqslant m)$ 在节点 v_k 的状态不变式 $\alpha(v_k)$ 生成的:
  - 如果 ∇_l 是根据 $\zeta_k(x)$ 生成的, $v_k^i \in \mathbb{V}_{\nabla_l}$;
  - 如果 ∇_l 是根据 $\lambda_k(x)$ 生成的, $v_k^i \in \mathbb{V}_{\nabla_l}$, 如果 $i > 0$, $v_{k-1}^i \in \mathbb{V}_{\nabla_l}$.

其中, v_k^i 表示自动机 H_i 里的位置节点 v_k.

定义10 给定组合线性混成自动机 $N = H_1||H_2||\cdots||H_m$, 其中每个成员自动机 H_i 为线性混成自动机 $(X_i, \Sigma_i, V_i, V_i^0, E_i, \alpha_i, \beta_i, \gamma_i)(1 \leqslant i \leqslant m)$, 以及路径组 $P = \{\rho_1, \rho_2, \ldots, \rho_m\}$, 其中每条路径 ρ_i 为成员自动机 H_i $(1 \leqslant i \leqslant m)$ 中的一条有穷路径. 在根据路径组 P 编码生成的线性约束集 $\mathbb{C}$ 里, 对于涉及到路径同步语义的每个约束 $\nabla_s \in \mathbb{C}$, ∇_s 在 P 里的源节点集合 $\mathbb{V}_{\nabla_s}$ 的定义如下:

- 如果 ∇_s 是根据自动机 H_k 和 H_j 的同时到达目标节点的时间同步约束: $\delta_{k0} + \delta_{k1} + \cdots + \delta_{kn_k} = \delta_{j0} + \delta_{j1} + \cdots + \delta_{jn_j}$ 生成的, 则 $v_i^k \in \mathbb{V}_{\nabla_s}(0 \leqslant i \leqslant n_k)$, $v_i^j \in \mathbb{V}_{\nabla_s}(0 \leqslant i \leqslant n_j)$;
- 如果 ∇_s 是根据自动机 H_k 的转换 σ_{kp} 和 H_j 的转换 σ_{jq} 所形成的共享事件同步约束 $(\sigma_{kp} = \sigma_{jq})$: $\delta_{k0} + \delta_{k1} + \cdots + \delta_{kp} = \delta_{j0} + \delta_{j1} + \cdots + \delta_{jq}$ 生成的, 则 $v_i^k \in \mathbb{V}_{\nabla_s}(0 \leqslant i \leqslant p)$, $v_i^j \in \mathbb{V}_{\nabla_s}(0 \leqslant i \leqslant q)$.

定义11 给定组合线性混成自动机 $N = H_1||H_2||\cdots||H_m$ 以及路径组 $P = \{\rho_1, \rho_2, \ldots, \rho_m\}$, 其中 ρ_i 为成员自动机 H_i $(1 \leqslant i \leqslant m)$ 中的一条有穷路径. 对于根据路径组 P 的可满足性而编码生成的线性约束集 $\mathbb{C}$, 假定 $\mathbb{C}'$ 是 $\mathbb{C}$ 的一个 IIS, $\mathbb{C}'_l$, $\mathbb{C}'_s$ 分别是 $\mathbb{C}'$ 里对应于本地编码和同步编码的约束集. 对于集合 $\mathbb{C}'_l = \{\nabla_1, \nabla_2, \ldots, \nabla_m\}$, 其源节点集合 $\mathbb{V}_{\mathbb{C}'_l} = \mathbb{V}_{\nabla_1} \cup \mathbb{V}_{\nabla_2} \cup \cdots \cup \mathbb{V}_{\nabla_m}$. 与之类似, 对于集合 $\mathbb{C}'_s = \{\nabla_1, \nabla_2, \ldots, \nabla_m\}$, 其源节点集合 $\mathbb{V}_{\mathbb{C}'_s} = \mathbb{V}_{\nabla_1} \cup \mathbb{V}_{\nabla_2} \cup \cdots \cup \mathbb{V}_{\nabla_m}$. 而集合 $\mathbb{C}'$ 的源节点集合 $\mathbb{V}_{\mathbb{C}'} = \mathbb{V}_{\mathbb{C}'_l} \cup \mathbb{V}_{\mathbb{C}'_s}$.

给定组合线性混成自动机 $N = H_1||H_2||\cdots||H_m$ 以及路径组 $P = \{\rho_1, \rho_2, \ldots, \rho_m\}$, P 对应的线性约束编码 $\mathbb{C}_P$, 当 $\mathbb{C}_P$ 不可满足时, 借助 IIS 分析可得到一个 IIS, $\mathbb{C}'_P$. 根据定义 9, 10 和 11, 可根据约

束 $\mathbb{C}'_P$ 定位到组合系统里各个成员自动机上的关联源节点集合 $\mathbb{V}_{\mathbb{C}'}$. 然后在每个成员自动机 H_i 的路径 ρ_i 上找到 $\mathbb{V}_{\mathbb{C}'}$ 中的最前及最后节点, 截取以这两个节点为起始节点的路径片段 ρ'_i. 很明显, 这些成员自动机上的不可行子路径片段 ρ'_i 所组成的路径组, $P' = \{\rho'_1, \rho'_2, \ldots, \rho'_m\}$ 就是我们要找的对应于 IIS 约束 $\mathbb{C}'_P$ 的组合路径片段. 我们称其为组合 IIS 路径.

基于此, 一旦一个组合路径被判定为不可行时, 通过 IIS 分析可以从中定位出一个 IIS 约束, 然后还原成组合系统图结构上的一个组合 IIS 路径. 在后续的验证里, 如果一条候选组合路径包含了一个已发现的组合 IIS 路径, 可以直接判定其不可行. 因为依据此路径编码生成的线性约束, 将会包含结构不变的、只有变量名称改变的 IIS 约束, 而这样的约束显然是不可满足的. 通过这种方法, 可以快速判定一类组合路径的可行性而不需要进行编码并求解, 节省了大量的计算时间, 提高了验证效率.

在图 1 所示的核反应堆控制系统混成自动机模型里, 当判定以下候选组合路径不可行时,

$\text{rod\_1} : \langle \text{out} \rangle \xrightarrow{\text{add\_1}} \langle \text{in} \rangle \xrightarrow{\text{remove\_1}} \langle \text{recover} \rangle$,

$\text{rod\_2} : \langle \text{out} \rangle \xrightarrow{\text{add\_2}} \langle \text{in} \rangle \xrightarrow{\text{remove\_2}} \langle \text{recover} \rangle \xrightarrow{\text{remove\_2}} \langle \text{out} \rangle \xrightarrow{\text{add\_2}} \langle \text{in} \rangle \xrightarrow{\text{remove\_2}} \langle \text{recover} \rangle$,

$\text{controller} : \langle \text{rod\_0} \rangle \xrightarrow{\text{add\_2}} \langle \text{rod\_2} \rangle \xrightarrow{\text{remove\_2}} \langle \text{rod\_0} \rangle \xrightarrow{\text{add\_2}} \langle \text{rod\_2} \rangle \xrightarrow{\text{remove\_2}} \langle \text{rod\_0} \rangle \xrightarrow{\text{add\_1}} \langle \text{rod\_1} \rangle \xrightarrow{\text{remove\_1}} \langle \text{rod\_0} \rangle$.

在相应生成的线性约束 $\mathbb{C}$ 里, 通过 IIS 分析, 可以得到一组如下的约束集:

$$\begin{aligned}\mathbb{C}' =& \{\delta_{\text{out}}^{\text{rod\_2}} = \delta_{\text{rod\_0}}^{\text{controller}}, \zeta_{\text{rod\_0}}^{\text{controller}}(x) \geqslant 16, \zeta_{\text{out}}^{\text{rod\_2}}(x) \leqslant 10, \lambda_{\text{out}}^{\text{rod\_2}}(x) = 0, \lambda_{\text{rod\_0}}^{\text{controller}}(x) = 0,\\ & \zeta_{\text{out}}^{\text{rod\_2}}(x) - \lambda_{\text{out}}^{\text{rod\_2}}(x) \geqslant 0.9 * \delta_{\text{out}}^{\text{rod\_2}}, \zeta_{\text{out}}^{\text{rod\_2}}(x) - \lambda_{\text{out}}^{\text{rod\_2}}(x) \leqslant 1.1 * \delta_{\text{out}}^{\text{rod\_2}},\\ & \zeta_{\text{rod\_0}}^{\text{controller}}(x) - \lambda_{\text{rod\_0}}^{\text{controller}}(x) \geqslant 0.9 * \delta_{\text{rod\_0}}^{\text{controller}}, \zeta_{\text{rod\_0}}^{\text{controller}}(x) - \lambda_{\text{rod\_0}}^{\text{controller}}(x) \leqslant 1.1 * \delta_{\text{rod\_0}}^{\text{controller}}\},\end{aligned}$$

其中, $\delta_{\text{out}}^{\text{rod\_2}} = \delta_{\text{rod\_0}}^{\text{controller}}$ 是根据共享事件 add_2 的同步语义生成的约束. $\zeta_{\text{rod\_0}}^{\text{controller}}(x) \geqslant 16$ 是根据转换 add_2 上的卫式生成的, $\zeta_{\text{out}}^{\text{rod\_2}}(x) \leqslant 10$ 是根据节点 out 上的不变式生成的. $\lambda_{\text{out}}^{\text{rod\_2}}(x) = 0$ 和 $\lambda_{\text{rod\_0}}^{\text{controller}}(x) = 0$ 是根据相应自动机上的初始条件生成的. 后面 4 个约束则是根据 rod_2 成员自动机在位置节点 out 的流条件以及 controller 成员自动机在位置节点 rod_0 的流条件生成的.

根据上述的还原映射规则, 可以将此组 IIS 约束映射到如下路径:

$\text{rod\_2} : \langle \text{out} \rangle \xrightarrow{\text{add\_2}} \langle \text{in} \rangle$,

$\text{controller} : \langle \text{rod\_0} \rangle \xrightarrow{\text{add\_2}} \langle \text{rod\_2} \rangle$.

可以看到该组合路径片段较原路径相比, 长度已大幅缩短. 由于任意包含该组合路径片段的路径组皆不可行, 在路径遍历过程中一旦发现一个候选组合路径包含了一个 IIS 组合路径, 即可直接判定不可行, 而不需要进行验证, 从而节省大量的验证时间. 比如对于以下路径组:

$\text{rod\_1} : \langle \text{out} \rangle \xrightarrow{\text{add\_1}} \langle \text{in} \rangle \xrightarrow{\text{remove\_1}} \langle \text{recover} \rangle$,

$\text{rod\_2} : \langle \text{out} \rangle \xrightarrow{\text{add\_2}} \langle \text{in} \rangle \xrightarrow{\text{remove\_2}} \langle \text{recover} \rangle \langle \text{out} \rangle \xrightarrow{\text{add\_2}} \langle \text{in} \rangle \xrightarrow{\text{remove\_2}} \langle \text{recover} \rangle$,

$\text{controller} : \langle \text{rod\_0} \rangle \xrightarrow{\text{add\_2}} \langle \text{rod\_2} \rangle \xrightarrow{\text{remove\_2}} \langle \text{rod\_0} \rangle \xrightarrow{\text{add\_1}} \langle \text{rod\_1} \rangle \xrightarrow{\text{remove\_1}} \langle \text{rod\_0} \rangle \xrightarrow{\text{add\_2}} \langle \text{rod\_2} \rangle \xrightarrow{\text{remove\_2}} \langle \text{rod\_0} \rangle$.

由于其包含了上述组合 IIS 路径, 我们可以直接判定其为不可行而不需要编码和验证. 通过此法则, 可将上述组合系统里需要验证的路径组数量从 14 约减到 5, 提升非常明显.

3.3 基于 SMT 编码的有界图结构遍历

前面章节介绍了如何利用约束求解里的 IIS 分析技术, 从一个不可行的路径组里分析定位出一个组合 IIS 路径. 由于任意一个包含组合 IIS 路径的候选路径组皆不可行, 为了提高验证效率避免不必要的检验, 我们在路径遍历的过程中需要规避已发现的组合 IIS 路径. 目前来说, 规避的方法主要有两种: 一种是在遍历的过程中一旦发现包含了一个组合 IIS 路径, 则立即回溯. 但是由于深度优先遍

历算法是一种局部搜索算法, 每次只看下一个即将要遍历的节点, 无法提前预知是否将会遍历一个已有的 IIS 路径, 这将会导致路径遍历算法有可能会将大量时间浪费在遍历组合 IIS 路径上.

另一种方法是已在单自动机可达性分析里被证明是行之有效的方法. 文献 [13] 提出了一种基于布尔可满足性 (Boolean satisfiability, 简称 SAT) [20] 的有界图结构遍历算法, 其主要思想是将自动机的有界图结构编码成一组 SAT 约束, 然后使用高效的 SAT 求解器得出一组解, 最后将这组解对应到自动机的图结构, 还原出一条候选路径.

不同于传统的局部遍历算法, 上述方法是一种全局的有界图结构算法, 可以提前约减掉相关不可行 IIS 路径, 有效的缩减待遍历的状态空间, 加快路径枚举过程. 因此, 我们决定采用该方法的思想来枚举不包含组合 IIS 路径片段的候选路径组. 由于组合线性混成系统里各个成员自动机的同步要求, 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. 受限于此要求, 除了要将每个成员自动机的有界图结构编码成一组 SAT 约束, 我们还需要加入新的约束来满足路径同步要求.

首先简单回顾一下单自动机有界图结构编码. 给定阈值 k, 自动机的有界图结构编码 SBG^k 如下所示:

$$\mathrm{SBG}^k := \mathrm{INIT}^0 \wedge \bigwedge_{0\leqslant i\leqslant k-1} \mathrm{NEXT}^i \wedge \bigwedge_{1\leqslant i\leqslant k} \mathrm{EXCLUDE}^i \wedge \Big(\bigvee_{0\leqslant i\leqslant k} \mathrm{TARGET}^i\Big),$$

其包括 NEXT, EXCLUDE, INIT 和 TARGET 公式分集 (clause), 具体如下所示:

$$\mathrm{NEXT} := \bigwedge_{q\in V}\Big(\mathrm{loc}=q \rightarrow \bigvee_{(q,q')\in N} \mathrm{loc}'=q'\Big),$$

$$\mathrm{EXCLUDE} := \bigwedge_{q\in V}\Big(\mathrm{loc}=q \rightarrow \bigwedge_{q'\in V\wedge q'\neq q} \mathrm{loc}\neq q'\Big),$$

$$\mathrm{INIT} := (\mathrm{loc}=v_{\mathrm{I}}) \wedge \mathrm{EXCLUDE},$$

$$\mathrm{TARGET} := (\mathrm{loc}=v_{\mathrm{T}}),$$

其中离散变量 loc 和 loc' 分别表示自动机当前和下一时刻位置, v_{I} 和 v_{T} 分别表示自动机的初始和目标节点. 该编码目的是在单个自动机里寻找一条从初始节点开始最终到达目标节点的候选路径[2)].

给定组合线性混成系统 H 以及成员自动机的阈值 $\overline{k}=\langle k_1,k_2,\ldots,k_n\rangle$, H 的有界图结构可以编码为

$$\mathrm{BG}^{\overline{k}} := \bigwedge_{1\leqslant j\leqslant n} \mathrm{SBG}^{k_j}_{H_j} \wedge \mathrm{PATHSYNC}^k_H,$$

其中 PATHSYNC 编码是为了确保各个成员自动机的路径在共享事件上的映射是一致的, 具体编码如下所示:

$$\begin{aligned}\mathrm{PATHSYNC} := \bigwedge_{1\leqslant j\leqslant n}\bigwedge_{0\leqslant i\leqslant k_j} &\wedge \mathrm{SYNCSTEP}^i_j \wedge \mathrm{COUNTERINIT}_j \\ &\wedge\Big(\bigwedge_{0\leqslant i\leqslant k_j} \mathrm{COUNTERSTEP}^i_j\Big) \wedge \mathrm{FINALSYNC}_j,\end{aligned}$$

2) 由于篇幅限制, 该编码的详细描述请参考文献 [13].

其中 $\mathrm{SYNCSTEP}_j^i$ 表示如果在第 i 步, 第 j 个成员自动机里有一个共享事件 l 第 g 次出现, 那么该成员自动机的本地时间必然是 $\mathrm{occ\_time}_{g,l}$,

$$\mathrm{SYNCSTEP}_j^i := \bigwedge_{l\in U_j}(l_j^i = l) \to \bigwedge_{l\leqslant g\leqslant i}((\mathrm{count}_{l,j}^i = g) \to t_j^i = \mathrm{occ\_time}_{g,l}),$$

$$\mathrm{COUNTERSTEP}_j^i := \bigwedge_{l\in U_j}(l_j^i = l) \to (\mathrm{count}_{l,j}^{i+1} = \mathrm{count}_{l,j}^i + 1),$$

$$\mathrm{COUNTERINIT}_j := (\mathrm{count}_{l,j}^0 = 0),$$

COUNTERINIT 和 COUNTERSTEP 编码描述了共享事件计数器值的变化.

$$\mathrm{FINALSYNC}_j := \bigwedge_{l\in U_j}\mathrm{count}_{l,j}^{k_j} = l_{\mathrm{last}},$$

FINALSYNC 编码描述了每个共享事件计数器的最终值.

由于该编码里包含了用于对共享事件进行计数的变量, 超出了 SAT 求解器的处理范围, 本文选择使用 SMT 求解器来得到该组编码的一个可行解, 进而得到一条候选路径. 3.2 小节提到任意包含组合 IIS 路径的候选路径组皆不可行. 因此, 在路径枚举的过程中应该要求 SMT 求解器不要枚举包含组合 IIS 路径的候选路径组. 对于基于编码的图结构遍历算法而言, 可以直接将组合 IIS 路径的取反编码加入到图结构编码 $\mathrm{BG}^{\overline{k}}$ 里, 具体如下所示:

$$\mathrm{IIS} := \bigwedge_{\rho\in\mathrm{IISPath}}\mathrm{IIS}^{\overline{k}}(\rho), \quad \mathrm{BG}^{\overline{k}} := \mathrm{BG}^{\overline{k}} \wedge \mathrm{IIS},$$

其中 IISPath 是所有当前已发现的 IIS 路径, 而 $\mathrm{IIS}^{\overline{k}}(\rho)$ 是组合路径 ρ 的取反编码, 具体如下所示:

$$\mathrm{IIS}^{\overline{k}}(\rho) := \bigwedge_{\rho_i\in\rho}\mathrm{IIS}^{k_i}(\rho_i),$$

其中, ρ_i 是组合路径里的第 i 个子路径, k_i 是第 i 个成员自动机的阈值. 对于单自动机的 IIS 路径编码 $\mathrm{IIS}^{k_i}(\rho_i)$, 其基本方法是将路径里的节点依次取反编码. 假设 IIS 路径为$\rho_i = \langle v_3\rangle \xrightarrow[e_3]{} \langle v_4\rangle \xrightarrow[e_4]{} \langle v_1\rangle \xrightarrow[e_5]{} \langle v_5\rangle$, 其对应的编码 $\mathrm{IIS}^{k_i}(\rho_i)$ 如下所示:

$$\mathrm{IIS}^{k_i}(\rho_i) := \bigwedge_{0\leqslant i\leqslant k_i-\mathrm{len}+1}\left(v_3^i \wedge v_4^{i+1} \wedge v_1^{i+2} \to \neg v_5^{i+3}\right),$$

其中, len 表示路径 ρ_i 的长度而 k_i 表示给定阈值.

基于 IIS 路径抽取以及 SMT 图遍历的有界可达性分析方法的伪代码 (pseudocode) 如算法 1 所示. 给定一个线性组合混成系统以及相应的阈值, 首先将自动机的有界图结构按照 3.3 小节介绍的方法进行编码, 得到一组 SMT 约束, 然后调用一个高效的 SMT 求解器得到一组可行解. 接下来, 根据编码规则将这组可行解还原成自动机图结构上的一个候选组合路径, 然后使用面向路径的可达性分析方法去检测该路径组的可行性. 若不可行, 则利用 IIS 分析技术从中定位出一个组合 IIS 路径, 并且该组合 IIS 路径的取反编码加入到自动机的图结构编码里, 以提前约减掉所有包含这些组合 IIS 路径的路径组.

算法 1 组合 IIS 路径制导的组合线性混成系统有界可达性检验

```
BG^k ⇐ 组合系统有界图结构编码
IIS ⇐ ∅
while true do
  检验 BG^k 的可满足性
  if infeasible then
    return unsat
  else
    组合路径 P = (ρ_1, ρ_2, ..., ρ_n) ⇐ 可行解 BG^k 的解码;
    线性约束集 Θ(P, R(v_n, φ)) ⇐ P 的行为编码;
    检验 Θ(P, R(v_n, φ)) 的可满足性;
    if feasible then
      return sat
    else
      从 P 中定位出组合 IIS 路径 P';
      IIS ⇐ IIS ∧ IIS^k(P')
      BG^k ⇐ BG^k ∧ IIS
    end if
  end if
end while
```

3.4 多重 IIS 路径定位技术

3.2 小节介绍了如何利用 IIS 分析技术从一个不可行的路径组里定位出一个组合 IIS 路径, 而上一小节介绍了一种高效的规避组合 IIS 路径的有界图结构遍历算法, 从而利用组合 IIS 路径来指导后续的路径枚举过程. 当得到组合 IIS 路径的编码反馈后, 通过 SMT 求解里的优化技术, 可以提前约减掉图结构里包含相关路径的分支, 缩减待遍历的状态空间, 加速可达性分析过程.

在上述的优化技术里, 直观地说一条更短的组合 IIS 路径, 可以帮助约减更多的候选组合路径, 从而更大程度地加速验证过程. 由于组合 IIS 路径是由约束里的 IIS 映射而来, 那么我们可否得到一个更 “好” 的 IIS 呢? 不幸的是, 一个不可解的线性约束里可能存在多个 IIS, 每个 IIS 都是极小的集合, 但是无法保证最小. 现有的技术大多是通过启发式搜索来给出一个 IIS, 无法保证其是最小或是相对更小的.

受限于当前技术的不足, 我们选择从另一个方向来解决此问题. 既然现有算法无法找出最小甚至较小的 IIS, 如果我们找出多个 IIS, 就可以通过比较选出较好的 IIS 组合路径. 由于目前尚无工具可以从一组不可解的线性约束里提取多个 IIS, 本文决定基于现有的工作进行扩展.

文献 [15] 提出了一种可以从一组不可解的 SAT 约束里分析定位出多个不可满足核心 (unsatisfiable core, 简称 UC) 的 MARCO 算法. 现有的 SAT 求解器可以给出一个 UC, MARCO 算法只需要基于现有的 SAT 求解器就能给出多个 UC. 这和目前遇到的问题非常类似, 现有的线性规划求解器可以给出一个 IIS, 而我们想要得到多个 IIS. 受此启发, 本文决定基于 MARCO 算法实现一个基于现有线性规划求解器的可以给出多个 IIS 的方法.

MARCO 算法的基本思想十分简单, 当一个约束不可解时, 从 SAT 求解器得到一个 UC, 然后从原约束里分割出一个真子集, 若该子集可解则继续从原约束里找出一个真子集, 若不可解则又可以从 SAT 求解器里得到一个 UC. 如此反复, 即可从一个不可解的 SAT 约束里得到多个 UC. 需要注意的是, 由于每次从原约束里找出的子集都不能包含已经发现的 UC, 可以保证每次得到的 UC 都不重复.

文献 [15] 里的实验评估证明了该算法简单而高效. 由于该算法和 SAT 求解器几乎没有藕合, 具有一般普适性. 我们将 SAT 求解器换成了线性规划求解器, 并且对于线性约束集里的每个约束使用一个布尔变量来蕴含 (imply) 它. 当该布尔变量为真时, 表示该约束存在于约束集里; 反之, 则表示该约束被移除. 通过此种方式, 我们基于 MARCO 算法, 仅使用一个现有的线性规划求解器, 并将其集成到算法 1 中, 实现了一个可以从一组不可解的线性约束里定位出多个 IIS 的算法, 从而在一次验证中发现多个 IIS, 并将其对应的 IIS 路径片段反馈给路径枚举过程, 加速后续验证过程.

4 工具实现与实验评估

4.1 工具实现

本文所述的组合 IIS 路径制导的组合线性混成自动机有界可达性验证方法已经在线性混成系统验证工具 BACH [16] 里进行了实现. BACH 是针对线性混成自动机的有界可达性检验工具集, 其使用 C++ 语言实现, 运行在 Linux 平台. 在 BACH 中, 选用 MathSAT [9] 来对自动机图结构约束进行求解, 从而枚举出候选路径. MathSAT 可以高效地进行增量求解 (incremental solving), 而枚举路径的过程就是对图结构的约束集进行不断求解的过程. 因而选用 MathSAT 来完成此任务. 在线性约束求解方面, 选用 Z3 [10] 来从一组不可解的线性约束里抽取出一个 IIS, 并使用 Z3 来实现多重 IIS 定位的 MARCO [15] 算法. 得益于微软研究院的不断开发与完善, Z3 对线性约束的求解与分析能力十分优异.

BACH 主要包括以下功能:

- 线性混成自动机图形化建模和编辑.
- 单个线性混成自动机有界可达性分析, 其分析算法在文献 [13] 中有详细介绍.
- 单个线性混成自动机有界可达性结果增强, 其主要思想是基于有界模型检验的结果进行推理分析, 从而在一定情况下得到全局的结果 [21].
- 组合线性混成系统有界可达性分析, 其分析算法如本文所述.

4.2 实验配置

为了评估本文方法的有效性, 我们在一系列被相关研究领域广泛使用的组合线性混成系统上完成了相关对比实验. 主要包括:

- Star-shape Fischer: 这是互斥协议里的一种混合 Fischer 算法, 模型结构如图 2 所示, 其用一个共享变量来控制对关键区域的访问.
- Ring-shape Fischer: 这是 Fischer 算法的一个变种, 包含一个环状的进程集合, 每个进程和它的左右邻居共用一个共享变量, 这个变量用于控制相邻的两个进程对关键区域的访问.
- FDDI Protocol: 这是一个基于文献 [22] 里的系统模型改编的一个环状的拓扑结构模型. 这是一个局域网里基于光纤的数据传输标准, 系统里的每个组件需要等待前一个组件的信号才能传输数据.
- Motorcycle: 这个例子是根据高速公路系统模型 [23] 改编的. 系统对 n 辆摩托车组成的车队进行建模, 每辆摩托车需要等待前一辆车的信号才能移动, 它需要通过和相邻车辆共享事件进行同步来保持队形.
- Nuclear Reactor System (NRS): 这个例子来源于文献 [24], 其模型结构如图 1 所示. 系统控制着原子反应堆的 n 个控制棒, 并且用这些控制棒来一个一个吸收中子. 每个移出的控制棒一定不能接触水并且需冷却几个时间单元.

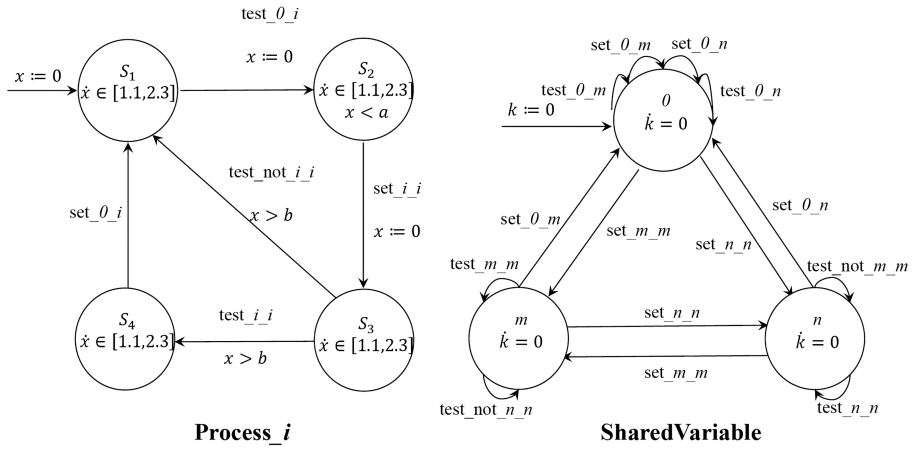

图 2　Fischer 互斥系统混成自动机模型

Figure 2　Hybrid automata for Fischer mutual exclusion system (FME)

为了更清晰地展示我们的工具 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实验中的各个系统构造了两个版本: “可达版” 和 “不可达版”. “可达版” 表示系统中存在一个可达的路径组, 反之, “不可达版” 表示系统中不存在可达的路径组.

为了说明 IIS 优化技术对 BACH 性能的影响, 本文分别给出了在使用 IIS 和不使用 IIS 技术时 BACH 的性能数据, 它们分别被标记为 BACH 和 BACH (IIS). 更进一步, 为了说明多重 IIS 定位技术对本文所述的可达性分析算法性能的影响, 本文也给出了在使用多重 IIS 定位技术时 BACH 的性能数据, 其被标记为 BACH (MIIS).

此外, 本文还将 BACH 的实验结果与基于 SMT 编码的组合线性混成系统验证最新代表性工具, HyCOMP [17] 进行了全面地比较. 对于此类基于 SMT 的验证工具, 编码方法是影响其性能的决定因素. HyCOMP 拥有 7 种配置, 分别对应于 7 种不同的编码方式. 最常用和可靠的是基于交叠 (interleaving) 语义的编码方法, 它是工具作者推荐使用的经典编码方式. 除此之外, HyCOMP 还有 5 种不同的基于浅同步 (shallow synchronization) [25] 语义的编码方式: shallow_ru, shallow_rf, shallow_ef, shallow_tu 和 shallow_tf. 这些编码的方法整体思想一致, 区别在于添加路径同步约束的时间点, 以及约束的表示形式不同. 实验发现, 基于 shallow_ru 和 shallow_rf 的编码验证算法给出了错误的结果, 经与工具作者沟通, 确认为工具 bug, 故实验中未列出其实验数据. 使用基于浅同步和交叠语义编码的 HyCOMP 配置, 依次被标记为 HyCOMP (shallow_ef), HyCOMP (shallow_rf), HyCOMP (shallow_ru) 和 HyCOMP (interleaving). 此外, 在交叠语义下存在一种被称为分布语义 (step semantics) [26] 的变种. 此语义与标准交叠语义的区别在于当某个进程触发了非共享事件后, 并不强制另一个进程触发空转换, 它可以同样触发一个本地的非强制转换, 来保证在当前步长所有自动机均触发的离散转换, 并达到规避中间变量的效果. 这里也列出了基于该编码方法的实验数据, 标记为 HyCOMP (step).

有界模型检验工具的性能和用户设定的阈值密切相关, 由于本次实验需要比较两种不同类型的工具的性能. 我们分别针对不同的系统模型计算出等价的阈值, 详细对应关系如表 1 所示, 表格中的 n 表示相应模型里成员自动机的数目. 由于不同配置的 BACH 的阈值相同, 故用 BACH 来表示所有配置. HyCOMP 的各种浅同步编码的阈值相同, 用 HyCOMP (shallow) 表示; 而 interleaving 和 step 编码的阈值相同, 使用 HyCOMP (interleaving) 表示. 从表中可以看出 BACH 的阈值是 HyCOMP (shallow) 的阈值的一半再加上 1. 这是因为 HyCOMP 的阈值包括连续的时间步和离散的跳转步, 而 BACH 的

表 1 不同系统模型的阈值设定

Table 1 Bound setting for different models

System	Type	Bound		
		BACH	HyCOMP (shallow)	HyCOMP (interleaving)
NRS	Reachable	$2n+1$	$4n$	$4n$
	Unreachable	$2n+1$	$4n$	$4n$
FDDI Protocol	Reachable	3	5	$2n+1$
	Unreachable	11	20	$10n+5$
Motorcycle	Reachable	5	9	$4n+3$
	Unreachable	9	16	$4n+3$
Star-shape Fischer	Reachable	$3n+1$	$6n$	$6n$
	Unreachable	$3n+1$	$6n$	$6n$
Ring-shape Fischer	Reachable	8	13	$7n+2$
	Unreachable	8	13	$7n+2$

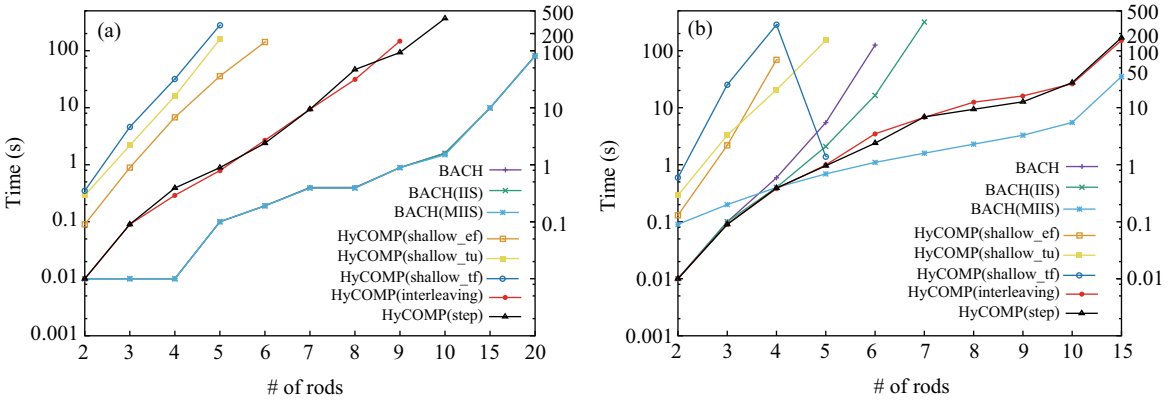

图 3 (网络版彩图) NRS 模型实验数据

Figure 3 (Color online) Experiment data for (a) reachable NRS model and (b) unreachable NRS model

阈值只包括离散步. HyCOMP (shallow) 和 HyCOMP (interleaving) 的阈值之间的对应关系依赖于待分析的系统模型, 文献 [25] 中有详细描述, 这里不再赘述. 为了保持一致性, 本次实验中各个模型的阈值设定和文献 [25] 里的一致.

所有实验都是在一台 ThinkPad 工作站上完成的, 其拥有 Intel 四核 3.1 GHz CPU, 8 GB 内存, 运行 Ubuntu 15.04 64 位系统. 每次实验的时间上限设为 500 s, 而内存上限则设定为 4 GB. 实验程序、脚本和模型输入文件都可以通过此链接 http://seg.nju.edu.cn/BACH/composed16/下载.

4.3 实验评估

通过在 5 个不同的系统及可达、不可达配置共计 10 套系统上运行不同配置的 BACH 和 HyCOMP 工具, 得到了 10 组实验数据. 本文给出了其中比较有代表性的两个模型的实验数据[3], 分别如图 3 和 4 所示. 实验主要从验证效率和可扩展性上来评估工具, 其中验证效率在纵坐标上体现, 表示完成相关

3) 完整的实验数据请参照链接 http://seg.nju.edu.cn/BACH/composed16.

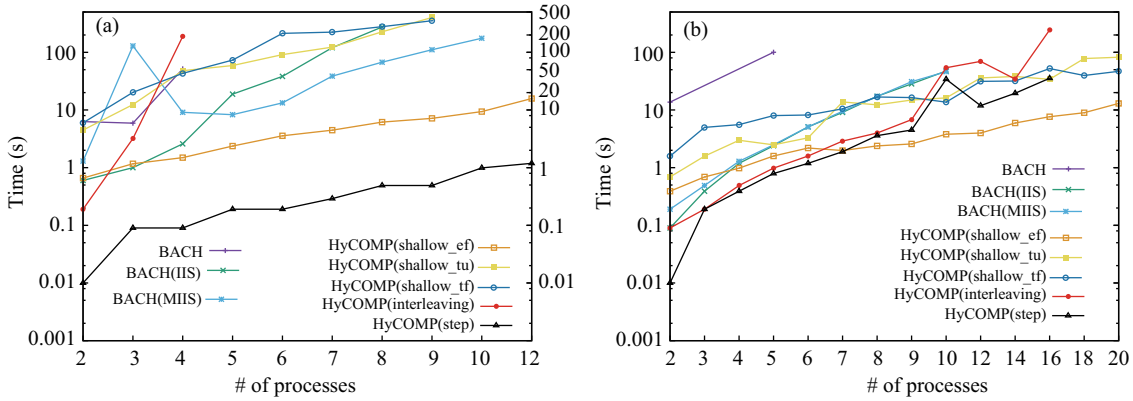

图 4 (网络版彩图) Ring-shape Fischer 模型实验数据

Figure 4 (Color online) Experiment data for (a) reachable Ring-shape Fischer model and (b) unreachable Ring-shape Fischer model

检验所耗费的时间, 可扩展性上主要在横坐标上体现, 表示所验证系统中含有组件的规模.

首先来分析 IIS 技术对 BACH 性能的影响. IIS 定位技术对 BACH 的性能起到了极大的优化作用, 尤其是对于需要验证阈值内所有路径可行性的不可达版系统模型, IIS 技术对性能起到了决定性的影响. 例如对于不可达版的 FDDI Protocol 系统模型, 基本版的 BACH 能处理的系统规模为 10, 而 BACH (IIS) 可以在不到 1 s 内分析完拥有 20 个成员的该系统, 差距非常明显. 另一方面, 多重 IIS 分析技术在很多情况下也进一步增强了 BACH 的性能. 这是因为组合线性混成系统里的路径编码生成的线性约束很大, 其中存在极多的 IIS, 有些 IIS 可以很大程度地约减图结构而有些作用就很小. 目前的技术无法做到只定位出 “好” 的 IIS, 作为一种折衷的办法, 定位出多个 IIS 则有很大概率得到 “好” 的 IIS. 例如, 对于不可达版的 NRS 系统 (如图 3(b) 所示), BACH (IIS) 只能处理规模为 7 的系统, 提升有限. 而基于多重 IIS 定位技术的 BACH (MIIS) 却可以很轻松地处理规模为 15 的该系统.

从另外一个角度, 由于 BACH 是通过验证阈值内所有路径可行性的方法来解决有界可达性分析问题的, 可以用需验证的路径数量来衡量其性能. 很明显, 需要验证的路径数量越多则性能越差. 因此, 用需验证的路径数量来直观地比较 IIS 分析技术以及多重 IIS 定位技术对 BACH 性能的影响. 表 2 给出了 BACH 在分析不可达版的系统模型时需要验证的路径数量[4]. 从这些数据我们可以看出:

- 在大多数 (4/5) 模型里, IIS 分析技术极大地约减了待验证的路径数量, 例如对于大小为 4 的 Motorcycle 系统, 可以将待验证的路径数量从 2379 减到 13, 约减了 99% 的待验证路径, 极大地提升了 BACH 的性能.
- 在部分模型, 如 NRS 上, 多重 IIS 技术起到了进一步的优化作用. 例如在包含 6 个成员的 NRS 系统上, BACH 和 BACH (IIS) 需检验的路径数量分别是 720 和 326, 而 BACH (MIIS) 只需要检验 6 条路径的可行性, 提升效果十分显著. 使用多重 IIS 抽取的目的在于尝试挖掘出更好的 IIS, 如果当前发现的 IIS 已足够好, 使用多重 IIS 抽取技术反而会给系统带来额外的负担. 例如, 在 FDDI 和不可达版的 Motorcycle 模型上, BACH (IIS) 比 BACH (MIIS) 的性能稍好. 但这部分的开销同整体分析时间相比并不是很大, 在所有实验中最坏情况仅相差 5 s, 基本可忽略不计.

从上面的实验可以发现 IIS 技术可以有效地提升 BACH 的性能. 接下来, 我们将基于 IIS 抽取技

4) 之所以只给出不可达模型上的数据, 是因为只有不可达的模型上, BACH 才需要搜索完整状态空间, 才能体现 IIS 技术在约减待验证路径方面的优势.

表 2 在不可达模型里, BACH 需要验证的路径数量

Table 2 Number of paths verified by BACH in unreachable models

System	#automata	Number of paths				
		BACH	BACH (IIS)	Reduction	BACH (MIIS)	Reduction
NRS	4	24	16	33%	4	83%
	6	720	326	55%	6	99%
	8	N/A	N/A	N/A	9	N/A
FDDI Protocol	4	16	3	81%	2	88%
	6	64	2	97%	2	97%
	10	1024	2	99%	2	99%
Motorcycle	2	75	13	83%	13	83%
	3	421	13	97%	13	97%
	4	2379	13	99%	13	99%
Star-shape Fischer	2	2	2	0	2	0
	3	12	12	0	12	0
	4	144	144	0	144	0
Ring-shape Fischer	2	324	1	99%	1	99%
	3	N/A	1	N/A	1	N/A
	10	N/A	1	N/A	1	N/A

术的 BACH 和领域内最具代表性的工具 HyCOMP 进行比较.

• 基于 IIS 技术的 BACH 的性能整体优于 HyCOMP, 尤其明显优于其最为依赖的基于交叠语义的编码方法. 在 10 个模型中, BACH 的性能在可达版的 NRS, Motorcycle 和不可达版的 NRS, FDDI 等 4 个模型上全面超越 HyCOMP, 比如对于可达版的 NRS 模型 (如图 3(a) 所示), BACH 可以在 80 s 内完成对拥有 20 个成员自动机的 NRS 系统的分析, 而 HyCOMP 只能在 500 s 时间上限内处理包含 10 个成员自动机的该系统. 在可达版的 FDDI 和不可达版的 Motorcycle, Star-shape Fischer 等 3 个模型上的性能和 HyCOMP 处于同一级别, 两个工具能处理的最大问题的规模相同, 并且最大验证时间差别都在几秒内. 而在余下的 3 个 Fischer 模型上, BACH 的性能落后于 HyCOMP, 下一段会给出详细分析. 由此可见, 在 IIS 技术的帮助下, BACH 的性能得到显著提升, 并且能处理更大规模的系统.

• BACH 在 Ring-shape Fischer (如图 4 所示) 和 Star-shape Fischer 系统上的性能落后于 HyCOMP, 这是因为 Fiscer 系统上的每个节点都包含自环, 图结构即为复杂包含大量的路径, 给我们的路径枚举带来极大的压力. 采用交叠语义编码的 HyCOMP 不需要枚举路径, 故在性能上有天生的优势. 但是这样的包含大量自环的特殊系统模型在实际应用中并不常见. 另一方面, 由于可达的模型里存在多条可行路径, 而对于可达性分析来说只要找出一条可行路径即可. 因此, 对可达模型进行分析, 其结果带有一定的随机性. 例如, 对于可达版的 Star-shape Fischer 系统, BACH 的性能落后于 HyCOMP, 但是对于需要搜索完整状态空间的不可达版的该系统, BACH 的性能反而有所超越.

• 上述讨论都是将 BACH 的性能同 HyCOMP 的最好配置相比, 与基于交叠语义的 HyCOMP 相比, 我们只在自环较多的系统上落后. 但同样可以发现, HyCOMP 中基于浅同步编码的各种不同实现的性能并不稳定, 在一些模型上比较好, 在另一些上却很差, 波动较大. 在各种浅同步编码的实现中, shallow_ef 的效果最好, 其他编码的性能均大幅落后于 BACH. 此外, 由于其中的两种浅同步编码给出

了错误结果, HyCOMP 作者确认其为工具 bug, 并推荐我们使用更为常用可靠的交叠语义编码. 反观本文方法的性能, 则比较稳定、平缓, 整体上优于 HyCOMP. 通过进一步的分析, 我们发现在少数模型上的图结构约束求解在 BACH 的计算时间里占据了很大比重. 事实上, 我们的图结构编码是标准的浅同步编码, 只是移除了自动机的连续语义和时间同步语义. 而 shallow_ef 在少数例子上的性能微弱领先于 BACH 的原因是该工具由 MathSAT 团队开发, 和底层求解器进行了深度整合, 采用了 incremental unrolling 等优化技术来增强性能. 而目前, BACH 的图结构约束求解只是简单的调用了约束求解器, 并未使用相关优化技术. 相信将 shalllow_ef 里使用的底层求解器实现层面的优化技术加入到 BACH 中后, 本文的方法会有很大的提升空间.

5 相关工作

线性混成自动机是混成自动机 [2] 的一个简单子类, 其可达性分析问题是不可判定的 [3]. 经典的线性混成自动机可达性分析方法大多基于多面体计算, 通过不断的计算系统的下一状态可达集, 从而遍历系统的状态空间. 但这一计算过程是不保证终止的, 而且由于多面体计算的指数级复杂度, 基于此类技术的验证工具 HyTech [4] 以及 PHAVer [5] 能处理的系统规模和实际的需求尚有一定的距离. 近年来, 相关学者在 PHAVer 的基础上开发出了新的工具 SpaceEx [6]. 和 PHAVer 相比, SpaceEx 能够处理一类非线性混成自动机, 但并未提升其处理线性混成自动机的能力.

近年来, 作为经典模型检验的一种补充方法, 有界模型检验技术 (bounded model checking, BMC) [7] 被学者提出并得到了广泛的应用. 混成自动机有界模型检验的基本思想是通过 SMT (satisfiability modulo theories) [8] 技术对系统在阈值内的行为进行编码并求解, 以检验系统是否满足相关性质. 但是由于这种方法需要提前一次性编码并计算系统阈值内所有的状态空间, 如此高的计算复杂性限制了其能处理的系统规模, 难以应用到工业界的实际系统.

另一种解决问题的思路是将整个系统的有界模型检验问题分解成系统内各路径的检验问题来控制单次验证的复杂度 [11], 在此基础上遍历系统内路径完成整个系统的有界模型检验. 该方法有效地控制了单次验证的复杂度, 从而能够验证较大规模的系统 [16].

上述工作主要是针对单个线性混成自动机, 对于组合线性混成系统, 经典方法是将系统内的成员自动机做笛卡尔乘积后得到一个自动机, 再使用上述分析方法. 值得一提的是, 对于组合线性混成系统的有界可达性检验, 常用方法是基于交叠 (interleaving) 语义, 将系统阈值内的行为编码成一组 SMT 约束后再调用相关求解器进行求解 [9]. 然而, 由于交叠语义包含大量的空转换 (stutter transition), 编码生成的 SMT 问题规模很大导致难以求解, 极大地限制了该方法能处理的问题规模. 如在上一章节实验中, 我们比对的 HyCOMP 工具, 其 interleaving 与 step 两种配置就是典型的基于交叠语义的实现, 实验数据显示其性能整体落后于 BACH.

文献 [18] 提出了一种基于浅同步语义的组合线性混成系统可达性检验方法, 通过将一个组合路径编码成线性约束加以求解, 从而判断该路径的可行性. 由于每次只需考虑一个组合路径的可行性, 其能验证的路径规模很大. 基于此方法, 文献 [12] 提出了一种共享事件制导的深度优先算法用以枚举组合系统图结构上的候选组合路径, 然后使用上述面向路径的方法检验该路径的可行性. 通过逐一枚举并验证组合线性混成系统在给定阈值内的所有候选路径组, 从而回答该系统的有界可达性问题. 然而, 当组合系统较为复杂, 或者给定阈值较大时, 系统图结构上的候选路径数量将会急剧增加, 逐一地枚举并验证相关路径将会变得十分耗时, 严重制约了相关方法的性能.

文献 [25] 提出了一种基于浅同步语义的 SMT 编码方法, 以避免做笛卡尔乘积而导致的组合状态

空间爆炸. 虽然该编码有效地缩减了目标约束的大小, 但是 SMT 风格的先编码再求解的方式会生成较大的约束, 从而难以求解. 第 4 节已将本文所述方法和同样实现了此技术的工具 HyCOMP 做了详细的性能比较. 从实验数据上看, 本文的方法在整体上具有优势.

6 全文总结

组合线性混成系统的可达性分析问题十分困难, 现有的技术难以处理大规模的系统. 虽然面向路径的有界可达性分析方法可以很好地控制单次验证的复杂度, 但在面对大规模复杂系统时, 待验证的路径数量急剧上升, 极大地制约了该方法的性能.

为解决此问题, 本文提出了一种基于组合 IIS 路径抽取的优化分析技术, 当一条路径被判定为不可行时, 利用 IIS 分析技术从中定位出一个组合 IIS 路径, 并在后续路径遍历过程中规避相关组合 IIS 路径. 为了定位出更 “好” 的 IIS 组合路径, 更进一步约减状态空间, 我们基于已有算法实现了一种多重 IIS 定位技术. 此外, 为了高效地规避组合 IIS 路径, 本文设计了一种全新的基于 SMT 编码的有界图结构遍历算法, 可以根据已发现的组合 IIS 路径, 对图结构进行高效剪枝, 从而加快路径遍历速度, 提升验证效率. 实验表明, 该分析方法大幅提升了面向路径有界可达性检验的性能, 而且整体性能领先当前最先进的同类工具.

参考文献

1 Clarke E, Grumberg O, Peled D. Model Checking. Cambridge: MIT Press, 1999

2 Henzinger T A. The theory of hybrid automata. In: Proceedings of the 11st Annual IEEE Symposium on Logic in Computer Science, New Brunswick, 1996. 278–292

3 Henzinger T A, Kopke P W, Puri A, et al. What's decidable about hybrid automata? J Comput Syst Sci, 1998, 57: 94–124

4 Henzinger T A, Ho P H, Wong-Toi H. HYTECH: a model checker for hybrid systems. Softw Tools Techn Transfer, 1997, 1: 110–122

5 Frehse G. PHAVer: algorithmic verification of hybrid systems past HyTech. In: Hybrid Systems: Computation and Control. Berlin: Springer, 2005. 258–273

6 Frehse G, Guernic C L, Donzé A, et al. SpaceEx: scalable verification of hybrid systems. In: Computer Aided Verification. Berlin: Springer, 2011. 379–395

7 Biere A, Cimatti A, Clarke E, et al. Bounded model checking. Advance Comput, 2003, 58: 118–149

8 Barrett C W, Sebastiani R, Seshia S, et al. Satisifiability modulo theories. Handbook of Satisfiability, 2009, 185: 825–885

9 Audemard G, Bozzano M, Cimatti A, et al. Verifying industrial hybrid systems with MathSAT. Electron Notes Theor Comput Sci, 2005, 119: 17–32

10 de Moura L, Bjørner N. Z3: an efficient SMT solver. In: Tools and Algorithms for the Construction and Analysis of Systems. Berlin: Springer, 2008. 337–340

11 Li X D, Jha S K, Bu L. Towards an efficient path-oriented tool for bounded reachability analysis of linear hybrid systems using linear programming. Electron Notes Theor Comput Sci, 2007, 174: 57–70

12 Bu L, Li Y, Wang L Z, et al. BACH 2: bounded reach ability checker for compositional linear hybrid systems. In: Proceedings of the 13th Design Automation & Test in Europe Conference, Leuven, 2010. 1512–1517

13 Xie D B, Bu L, Zhao J H, et al. SAT-LP-IIS joint-directed path-oriented bounded reachability analysis of linear hybrid automata. Formal Methods Syst Design, 2014, 45: 42–62

14 Bu L, Yang Y, Li X D. IIS-guided DFS for efficient bounded reachability analysis of linear hybrid automata. In: Proceedings of the 7th International Haifa Verification Conference, Haifa, 2011. 35–49

15 Mark L, Ammar M. Enumerating infeasibility: finding multiple MUSes quickly. In: Proceedings of the 10th International Conference on Integration of Artificial Intelligence and Operations Research Techniques in Constraint Programming, Yorktown Heights, 2013. 160–175
16 Bu L, Li Y, Wang L Z, et al. BACH: bounded reachability checker for linear hybrid automata. In: Proceedings of the 8th International Conference on Formal Methods in Computer Aided Design, Portland, 2008. 65–68
17 Cimatti A, Griggio A, Mover S, et al. HyComp: an SMT-based model checker for hybrid systems. In: Tools and Algorithms for the Construction and Analysis of Systems. Berlin: Springer, 2015. 52–67
18 Bu L, Li X D. Path-oriented bounded reachability analysis of composed linear hybrid systems. Int J Softw Tools Tech Transfer, 2011, 13: 307–317
19 Chinneck J, Dravnieks E. Locating minimal infeasible constraint sets in linear programs. ORSA J Comput, 1991, 3: 157–168
20 Biere A, Clarke E, Zhu Y S. Symbolic model checking without BDDs. In: Tools and Algorithms for the Construction and Analysis of Systems. Berlin: Springer, 1999. 193–207
21 Xie D B, Bu L, Li X D. Deriving unbounded proof of linear hybrid automata from bounded verification. In: Proceedings of Real-Time Systems Symposium (RTSS), Rome, 2014. 128–137
22 Zhao J H, Li X D, Zheng T, et al. Removing irrelevant atomic formulas for checking timed automata efficiently. In: Formal Modeling and Analysis of Timed Systems. Berlin: Springer, 2004. 34–45
23 Jha S, Krogh B H, Weimer J E, et al. Reachability for linear hybrid automata using iterative relaxation abstraction. In: Hybrid Systems: Computation and Control. Berlin: Springer, 2007. 287–300
24 Wang F. Symbolic parametric safety analysis of linear hybrid systems with BDD-like data structures. IEEE Trans Softw Eng, 2005, 31: 38–51
25 Bu L, Cimatti A, Li X D, et al. Model checking of hybrid systems using shallow synchronzation. In: Formal Techniques for Distributed Systems. Berlin: Springer, 2010. 155–169
26 Heljabko K, Niemala I. Bounded LTL model checking with stable models. In: Theory and Practice of Logic Programming. Cambridge: Cambridge University Press, 2003. 519–550

Composed IIS path locating based optimization for bounded reachability analysis of compositional linear hybrid systems

Dingbao XIE[1], Yuexiang ZHOU[1], Lei BU[1,2*], Linzhang WANG[1,2] & Xuandong LI[1,2]

1. *State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing* 210023*, China*;
2. *Collaborative Innovation Center of Novel Software Technology and Industrialization, Nanjing* 210023*, China*
*Corresponding author. E-mail: bulei@nju.edu.cn

Abstract Hybrid systems include both discrete and continuous behavior and are widely used to model control systems. The reachability analysis of its unsafe state is an important method for guaranteeing the safety of a system. However, the current techniques do not scale well to the problems of practical interest. Due to the synchronization of components and the combinatorial explosion of state space, the reachability analysis of compositional linear hybrid system is extremely complex. In order to reduce the complexity, a path-oriented approach was proposed in a previous work, which conducted bounded reachability analysis of a compositional linear hybrid system. By enumerating and verifying each potential path one by one, the size of the problem that can be solved will be increased substantially. This path-oriented approach will become quite inefficient due to a sharp increase in the number of candidate paths when analyzing complex systems. The path explosion problem in model checking is also famous. To solve this problem, we propose a state-space reduction technique, which accelerates the verification process. We propose a method to locate the cause of infeasibility, when a composed infeasible path segment after a path set is proved to be infeasible. As we can simply falsify a path set that contains a composed infeasible path segment, the number of candidate paths can be reduced significantly. Furthermore, to avoid such composed path segments efficiently, we propose an approach based on satisfiability modulo theories (SMT), to traverse the bounded graph structure of the composed linear hybrid system. The results of the experiment show that the performance of the path-oriented bounded reachability analysis can be optimized significantly and that the overall performance of the proposed approach is better than that of the state-of-the-art competitor.

Keywords hybrid system, bounded model checking, reachability analysis, compositional linear hybrid automata, satisfiability modulo theories, irreducible infeasible subset

Dingbao XIE was born in 1988. He received his B.S. degree in computer science from Nanjing University in 2011. He is currently pursuing his Ph.D. degree from the Department of Computer Science and Technology, Nanjing University. His research interests mainly focus on verification of hybrid systems.

Lei BU was born in 1983. He received his B.S. degree and Ph.D. degree in computer science from Nanjing University in 2004 and 2010, respectively. He is an associate professor in the Department of Computer Science and Technology at Nanjing University. His main research interests include formal method, model checking, especially the verification of hybrid systems and cyber-physical systems.

Linzhang WANG was born in 1973. He received his Ph.D. degree from Nanjing University in 2005. He is a full professor in the Department of Computer Science and Technology at Nanjing University. His research interests span modeling, analysis, testing and verification in software engineering area.

Xuandong LI was born in 1963. He received his M.S. and Ph.D. degrees from Nanjing University, China, in 1991 and 1994, respectively. He is a full professor at the Computer Science and Technology Department of Nanjing University. His research interests include formal support for design and analysis of reactive, disturbed, real-time, hybrid, and cyber-physical systems, software testing and verification.