软件学报 ISSN 1000-9825, CODEN RUXUEW
2025,36(8):3587−3603 [doi: 10.13328/j.cnki.jos.007352] [CSTR: 32375.14.jos.007352]

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

基于混成自动机路径过滤与动态选择的 CPS 系统反例生成*

王佳宛[1,2], 刘熹橦[2], 卜 磊[1,3], 李宣东[1,2]

[1](计算机软件新技术全国重点实验室 (南京大学), 江苏 南京 210023)

[2](南京大学 计算机学院, 江苏 南京 210023)

[3](南京大学 软件学院, 江苏 南京 210093)

通信作者: 卜磊, E-mail: bulei@nju.edu.cn

摘 要: 信息物理融合系统 (cyber-physical system, CPS) 在安全攸关领域具有广泛的应用, 保障其安全性至关重要. 形式化验证是证明系统安全性的有效手段, 但在现实世界中的复杂 CPS 系统上应用仍面临挑战. 因此, 反例生成的方法被提出, 旨在通过寻找系统中违背安全规约的反例行为来证明系统的不安全. 现有的基于路径的 CPS 系统反例生成方法采用分治策略, 针对系统模型中各条路径上的行为空间分别进行探索, 能够有效发现系统中的不安全行为. 然而, 在大规模系统中, 这类方法面临严重的路径爆炸问题, 路径数量随着离散状态数目指数级增长, 反例生成的效率和性能显著下降. 为缓解反例生成中的路径爆炸问题, 基于系统模型与规约提出路径过滤与动态选择两种技术. 首先, 设计面向规约的路径过滤方法, 通过分析系统规约的语法树和路径上的行为约束, 快速过滤不可能包含不安全行为的路径. 其次, 引入多臂老虎机算法来指导反例生成过程中路径的动态选择, 动态调整用于探索不同路径的计算资源以最大化反例生成的性能. 在一系列经典测试案例上的实验结果表明, 所提方法显著缓解了路径爆炸问题, 提高了 CPS 系统反例生成的效率和性能, 将发现不安全行为的成功率提升了两倍以上.

关键词: 信息物理融合系统; 混成自动机; 反例生成; 面向路径

中图法分类号: TP311

中文引用格式: 王佳宛, 刘熹橦, 卜磊, 李宣东. 基于混成自动机路径过滤与动态选择的CPS系统反例生成. 软件学报, 2025, 36(8): 3587–3603. http://www.jos.org.cn/1000-9825/7352.htm

英文引用格式: Wang JW, Liu XT, Bu L, Li XD. CPS Falsification Based on Hybrid Automata Path Filtering and Dynamic Selection. Ruan Jian Xue Bao/Journal of Software, 2025, 36(8): 3587–3603 (in Chinese). http://www.jos.org.cn/1000-9825/7352.htm

CPS Falsification Based on Hybrid Automata Path Filtering and Dynamic Selection

WANG Jia-Wan[1,2], LIU Xi-Tong[2], BU Lei[1,3], LI Xuan-Dong[1,2]

[1](State Key Laboratory for Novel Software Technology (Nanjing University), Nanjing 210023, China)

[2](School of Computer Science and Technology, Nanjing University, Nanjing 210023, China)

[3](Software Institute, Nanjing University, Nanjing 210093, China)

Abstract: Cyber-physical system (CPS) is widely employed in safety-critical domains, making its safety assurance a critical concern. Formal verification serves as an effective approach for proving system safety but encounters challenges when applied to complex real-world CPS. Falsification has been proposed as an alternative, aiming to demonstrate system unsafety by identifying counterexample behaviors that violate specified safety properties. Existing path-oriented falsification methods for CPS utilize divide-and-conquer strategies to explore system behaviors along individual paths, effectively uncovering unsafe behaviors. However, in large-scale CPS, these methods are hindered by the path explosion problem, where the number of paths grows exponentially with the system's discrete system modes, leading to significant reductions in falsification efficiency and performance. To address the path explosion issue in path-oriented falsification, this study introduces two novel techniques based on system models and specifications: path filtering and dynamic path

* 基金项目: 国家自然科学基金 (62232008, 62172200); 江苏省前沿引领技术基础研究专项 (BK20202001)
本文由"形式化方法与应用"专题特约编辑陈明帅研究员、田聪教授、熊英飞副教授推荐.
收稿时间: 2024-08-26; 修改时间: 2024-10-14; 采用时间: 2024-11-26; jos 在线出版时间: 2024-12-10
CNKI 网络首发时间: 2025-04-03

selection. First, a specification-driven path filtering method is proposed to rapidly eliminate paths unlikely to contain unsafe behaviors by analyzing the syntax tree of system specifications and the behavioral constraints of each path. Second, a multi-armed bandit algorithm is adopted to guide the dynamic selection of paths during the falsification process, adaptively reallocating computational resources to optimize performance. Experimental evaluations on several classical benchmark cases demonstrate that the proposed techniques effectively mitigate the path explosion problem, significantly enhancing the efficiency and performance of CPS falsification. The success rate of identifying unsafe system behaviors improves by more than twofold.

Key words: cyber-physical system (CPS); hybrid automata; falsification; path-oriented

信息物理融合系统 (cyber-physical system, CPS) 集成了计算、通信和控制这 3 大技术, 紧密融合计算过程与物理过程, 是一类复杂的系统[1,2]. 这些系统被广泛应用于国防军工、航空航天、智能交通、医疗救助等诸多安全攸关领域. 其行为的复杂性以及应用场景的广泛与重要性, 使得确保这类系统能够安全可靠运行至关重要.

混成自动机 (hybrid automata)[3]作为一种强大的数学建模工具, 能够形式化地描述 CPS 系统中离散计算和连续物理过程紧密交织的混合动态行为. 然而, 现有的混成自动机的形式化验证技术[4]大多局限于低维度, 线性或特定动态行为特性的混成自动机模型[5], 难以应用于现实世界中日益复杂的大规模 CPS 系统. 因此, 近年来, 对 CPS 系统可靠性进行反例生成 (falsification) 被大量研究并在工业界应用.

CPS 系统反例生成方法旨在从系统模型中寻找违反系统规约的不安全行为, 从而证明系统不安全. 当前, 主流的反例生成技术大多采用基于优化的方法, 在系统行为空间中寻找能够最小化系统规约鲁棒的行为[6−12]. 在此基础上, 基于路径的策略被应用于反例生成: 通过对系统的混成自动机模型进行遍历来生成路径, 再针对每条路径进行基于优化的反例生成来寻找路径上的不安全的行为[13,14]. 面向路径的反例生成方法尤其适用于内部具有不确定性的 CPS 系统. 由于系统在不同路径选择下行为差异显著, 路径有助于更精确地控制不确定性系统的行为, 进而更全面地探索覆盖其行为空间. 此外, 面向路径的方法通过分治策略层次化地求解问题, 将系统的复杂行为空间基于离散路径分解为多个较小子空间并独立求解, 降低了优化问题的维度和复杂性, 进而提高了反例生成的性能. 目前, 较为成熟的 CPS 反例生成工具包括基于优化的工具 S-TaLiRo[10]、Breach[11]以及面向路径和优化的工具 SNIFF[14]等.

然而, 对于现实世界中的大规模 CPS 系统, 基于路径的反例生成方法容易面临严重的路径爆炸问题. 当系统模型中存在大量离散状态时, 系统路径数量往往会呈指数级增长. 由于对每条路径进行基于优化的反例生成较为耗时, 每次优化迭代都需要对系统的行为轨迹进行模拟计算, 因此, 当系统路径数量较大时, 系统整体反例生成的耗时无疑会大幅增加. 而实际应用中, 对系统的可靠性进行反例生成分析往往会有时间或系统行为模拟次数的限制. 当系统路径爆炸时, 每条路径被分配到的时间和计算资源都会非常有限, 导致路径上的系统行为空间难以被充分探索, 系统整体的反例生成成功率也会相应降低.

在此背景下, 本文尝试从减少路径数目以及为路径合理分配计算资源这两个角度出发, 缓解 CPS 反例生成中的路径爆炸问题, 提高反例生成性能. 首先, 传统的基于路径的反例生成方法通常仅依赖自动机的离散图结构来生成路径, 然后对每条路径上的行为都进行探索检查. 但实际上, 系统在某些路径上的行为可能由于受路径上的约束的限制, 一定不会违背规约. 因此, 本文提出了一种面向规约的路径快速探索与过滤方法, 根据系统规约的语法树和路径上的约束, 基于约束求解器快速过滤掉一定不包含不安全行为的路径, 从而缓解路径爆炸. 其次, 传统方法中通常会使所有路径平分计算资源, 逐条路径依次探索, 即每条路径进行反例生成时优化算法的迭代次数 (系统行为的模拟次数) 相同. 因此, 本文进一步提出了基于多臂老虎机算法指导的路径动态选择的反例生成方法. 该方法首先对各条路径都进行初步探索, 然后利用已探索的信息不断地动态选择更具有潜力的路径执行基于优化的反例生成, 自适应地调整不同路径的计算资源分配, 从而最大化反例生成性能.

总之, 本文为提高 CPS 系统的安全性和可靠性, 提出了基于混成自动机路径过滤与动态选择的反例生成方法. 本文的主要贡献可以总结为以下几点.

• 针对基于路径的 CPS 系统反例生成方法, 提出了面向系统时序规约的路径过滤技术, 减少冗余路径, 缓解路径爆炸问题;

• 针对需要反例生成的多条路径, 提出了基于多臂老虎机的路径动态选择策略, 充分利用系统路径的探索信息, 智能化地为路径动态分配反例生成的计算资源, 提升系统反例生成的性能;

• 将本文提出的基于路径过滤与动态选择的反例生成技术被扩展实现在工具 SNIFF 上, 并在混成系统的经典案例上分析验证了该方法的有效性. 实验结果表明该方法可以有效缓解路径爆炸问题, 将 CPS 系统反例生成的成功率提高到两倍以上并减少时间开销.

本文第 1 节介绍 CPS 系统反例生成的相关工作和研究现状. 第 2 节介绍 CPS 系统的形式化建模与规约描述语言以及本文涉及的多臂老虎机相关技术的基础知识. 第 3 节介绍本文提出的基于混成自动机路径过滤与动态选择的 CPS 系统反例生成方法. 第 4 节通过实验具体分析本文技术的有效性. 第 5 节对本文进行总结.

1 CPS 系统反例生成相关工作

CPS 系统反例生成是检测系统行为是否会违背目标规约的有效方法. 现有的反例生成工作大多采用各种优化算法, 通过最小化目标规约的鲁棒性[15,16]来探索系统的状态空间, 寻找系统不安全行为[6]. Nghiem 等人[7−9]提出了使用各种不同的经典的启发式优化搜索算法, 如蒙特卡洛算法、蚁群算法、交叉熵算法等, 来进行反例生成, 形成了较为成熟的基于优化算法的 CPS 反例生成工具, 例如 S-TaLiRo[10]和 Breach[11]. 随后, 一系列工作根据 CPS 系统反例生成这一具体问题来设计各种不同的状态空间解耦与搜索方法, 以优化反例生成的性能. Mathesen 等人[17]基于信任区间提出了将全局随机优化方法与局部自适应可重启的优化搜索相结合的反例生成方法, Yaghoubi 等人[18]提出了将全局的模拟退火优化搜索与基于梯度下降的局部搜索相结合的反例生成方法, Zhang 等人[19]则提出了通过网格分区将全局的蒙特卡洛树搜索和局部搜索相结合的双层反例生成方法. 在此基础上, Wang 等人[13,14]将形式化领域面向路径的方法应用于 CPS 系统反例生成问题, 基于系统模型中的路径对状态空间解耦, 在全局基于路径遍历探索系统的离散空间, 在局部采用基于分类模型的学习优化方法来搜索各路径上的行为. 该类方法既提高了反例生成的性能, 又可以有效地支持非确定性系统; 但面向的是具有混成自动机模型的 CPS 系统, 且当模型规模较大时可能会遇到路径爆炸问题. 相关工具包括 PDF[13]和 SNIFF[14].

除了主流的基于优化的反例生成以外, 早期 Plaku 等人[20]还提出了基于运动规划的反例生成, 通过提出并使用模型状态空间覆盖率, 安全规约鲁棒性等指标来指导快速扩展随机树的构造[21,22], 从而探索系统模型中的状态空间, 寻找系统不安全行为. 此外, Zutshi 等人[23,24]提出了一类基于梯度的反例生成方法, 基于梯度信息不断优化调整随机生成的系统行为轨迹片段, 直至最终形成一条完整且不安全的系统行为轨迹. 该方法还与符号执行技术相结合, 通过分析系统的可达性约束来加快循环迭代找到系统不安全行为的效率[25]. 由于此类方法需要梯度信息和可达性分析, 因此通常对系统模型有一定的限制, 难以处理任意非线性系统.

与本文工作最相近的是文献 [13,14] 中提出的基于路径解耦状态空间并基于优化搜索每条路径上的行为的反例生成工作. 这类基于路径的方法通过状态空间解耦降低了搜索问题的维度和复杂性, 进而提高了反例生成的性能, 但正如前文所述, 也容易在大规模系统中遇到路径爆炸的问题. 在形式化领域, 路径爆炸是一个常见问题, 指的是系统中存在大量可能的执行路径使得全面的系统验证或测试变得困难. 为缓解路径爆炸问题, 基于路径剪枝[26,27], 系统模型离散状态合并[28], 路径并行探索[29]等技术被提出使用. 本文旨在特别针对包含混成行为的 CPS 系统在时序规约下的反例生成问题, 在基于路径的反例生成工作的基础上, 通过减少需探索的路径数目以及为各路径合理分配用于探索的资源, 来缓解这类方法中的路径爆炸问题, 进而提升 CPS 系统反例生成的效率和成功率. 实际上, 在 CPS 领域, 基于路径来解耦问题并高效探索状态空间的思路还被应用在相关系统的形式化验证[30,31]和控制生成[32]等问题上. 本文提出的路径过滤以及动态选择的优化技术也都可以应用于相关方法, 缓解其中的路径爆炸问题.

2 基础知识

第 2.1 节介绍 CPS 系统反例生成基础知识, 第 2.2 节介绍本文方法涉及的多臂老虎机问题及相关算法.

2.1 CPS 系统反例生成

混成自动机 (hybrid automata)[3]是对 CPS 系统中紧密交织的离散和连续行为进行形式化建模的常用语言. 时序逻辑 (temporal logic) 是一类可以通过逻辑公式精准刻画系统行为与时间相关性质的规约描述语言, 其中, 信号

时序逻辑 (signal temporal logic, STL)[33]可以描述连续时间下系统中的实值信号的属性, 常用于描述包含连续行为的 CPS 系统的行为规约. 混成自动机及信号时序逻辑的形式化定义如下.

定义 1. 混成自动机. 混成自动机可以由一个多元组 $H=(X,Q,F,Inv,E,G,R,Q^0,Q^f)$ 定义, 其中,

1) X 是连续变量的集合;

2) Q 是离散节点的集合, 其中,

• 流函数 $F=\{F_q|q\in Q\}$ 定义各个离散节点下连续变量的动态行为, 通常为微分方程 $\dot{X}=F_q(X)$;

• 不变式 $Inv=\{Inv_q|q\in Q\}$ 定义各个离散节点下连续变量需要满足的不变式约束 $Inv_q(X)$;

3) $E\subseteq Q\times Q$ 是离散节点的跳转集合, 其中,

• 转移卫式 $G=\{G_e|e\in E\}$ 定义各个离散跳转发生时连续变量需要满足的约束 $G_e(X)$;

• 重置条件 $R=\{R_e|e\in E\}$ 定义各个离散跳转发生后连续变量需执行的重置动作 $X:=R_e(X)$;

4) Q^0 是初始节点的集合, 表示系统可以从哪些离散节点开始;

5) Q^f 是终止节点的集合, 表示系统的目标节点或接收节点.

混成自动机的状态由当前所处的离散节点以及当前连续变量的取值两者决定, 可由二元组 $s=(q,x)\in S$ 表示, 其中, $S=Q\times X$ 为混成自动机的状态空间. 混成自动机在状态 $s=(q,x)$ 下有两种可能的行为: 可以继续停留在节点 q 下 (需满足不变式约束 Inv_q), 变量根据流函数 F_q 随时间连续变化; 也可以沿着离散跳转 $e=(q,q')\in E$ 跳到节点 (需满足转移卫式 G_e), 变量根据重置条件 R_e 更新, 状态即刻变化为 $(q',R_e(x))$.

定义 2. 路径. 混成自动机 $H=(X,Q,F,Inv,E,G,R,Q^0,Q^f)$ 的路径是由离散节点构成的序列 $p=\{q_i\}_0^N$, 其中,

• 序列从初始节点开始, 由终止节点结束, 即: $q_0\in Q^0,q_N\in Q^f$;

• 对于序列上任意两相邻节点, 自动机中都存在从前者跳到后者的跳转, 即: $\forall i\in[0,N),(q_i,q_{i+1})\in E$.

混成自动机的路径描述了系统在离散状态空间中的演化序列, 但并未涉及混成自动机的状态在各个节点下随时间连续变化的过程. 因此, 只有进一步给定系统变量的初始值 x_0 以及系统在离散路径 p 中的每个节点上停留的时间 τ 时, 混成自动机的动态行为才可以被完整定义且唯一确定.

值得注意的是, 并不是任意的三元组 $(x_0,p=\{q_i\}_0^N,\tau=\{\tau\}_0^N)$ 都对应混成自动机上的合法行为. 只有当系统按照给定变量初值 x_0, 路径 p, 以及节点停留时间 τ 运行, 节点上的不变式约束以及跳转上的转移卫式约束从未被违背时, 该三元组才定义了一个合法的系统行为. 下面给出混成自动机上的合法行为的形式化定义.

定义 3. 合法行为. 在混成自动机 $H=(X,Q,F,Inv,E,G,R,Q^0,Q^f)$ 中, 给定 x_0 为变量 X 的初值, $p=\{q_i\}_0^N$ 为混成自动机的路径, $\tau=\{\tau_i\}_0^N$ 为在路径 p 的每个节点上的停留时间, 令此时混成自动机 H 根据其流函数 F 及重置条件 R 运行的行为轨迹为 $w^S_{(x_0,p,\tau)}:[0,\tau_N]\to Q\times X$, 则行为 $w^S_{(x_0,p,\tau)}=(w^Q_{(x_0,p,\tau)},w^X_{(x_0,p,\tau)})$ 是 H 中的合法行为当且仅当:

• 在每个节点下, 状态总满足 H 中相应的不变式约束, 即: $\forall i\in[0,N],\forall t\in[\tau_i,\tau_{i+1}),Inv_{q_i}(w^X_{(x_0,p,\tau)}(t))$;

• 每次跳转时, 状态总满足 H 中相应的转移卫式约束, 即: $\forall i\in[0,N),G_{(q_i,q_{i+1})}(w^X_{(x_0,p,\tau)}(\tau_{i+1}))$ 总成立.

在本文中, 我们用符号 C_p 表示上述路径 p 上的系统合法行为需要满足的所有不变式和转移卫式约束.

定义 4. 信号时序逻辑. 给定 X 为 $\overline{\mathbb{R}}^n$ 上定义的 n 维变量 ($\overline{\mathbb{R}}=\mathbb{R}\cup\{\top,\bot\}$ 为包含所有实数和布尔值的扩展实数域, $\top,\bot$ 分别为最大和最小元素), 信号 $w^X:T\to\overline{\mathbb{R}}^n$ 为变量 X 在时间域 $T=[0,t)\subseteq\mathbb{R}$ 上的值, 则表示信号 w^X 的时序性质的信号时序逻辑公式的语法为: $\varphi:=\text{true}|\theta(X)\geqslant d|\neg\varphi_1|\varphi_1\vee\varphi_2|\varphi_1\mathbf{U}_I\varphi_2$. 其中, $\theta:\overline{\mathbb{R}}^n\to\mathbb{R}$ 是将变量 X 映射到实数的函数, $d\in\mathbb{R}$ 是常数; U 是表示“直到 (Until)”的时序算子, 其下标 $I\subseteq\mathbb{R}^+$ 表示时间区间, $I=[0,\infty)$ 时可省略. 其他常见的逻辑常数, 逻辑连接词和时序算子都可以用该语法表示, 譬如: $\text{false}\equiv\neg\text{true}$, $\varphi_1\wedge\varphi_2\equiv\neg(\neg\varphi_1\vee\neg\varphi_2)$, $\varphi_1\to\varphi_2\equiv\neg\varphi_1\vee\varphi_2$, $\mathbf{F}_I\varphi\equiv\text{true}\mathbf{U}_I\varphi$, $\mathbf{G}_I\varphi\equiv\neg(\mathbf{F}_I\neg\varphi)$, $\varphi_1\mathbf{R}_I\varphi_2\equiv\neg(\neg\varphi_1\mathbf{U}_I\neg\varphi_2)$ 等.

信号时序逻辑公式具有布尔语义和定量语义[15,16], 其递归定义如表 1. 其中, 布尔语义描述信号 w 在给定时刻 t 是否满足公式 φ, 定量语义 $\rho(\varphi,w,t)$ 则描述信号 w 在时刻 t 满足公式 φ 的程度 (鲁棒性). 鲁棒性为正表示信号满足公式, 鲁棒性为负则表示信号不满足公式, 并且值越小表示信号满足公式的程度越低. 其中, 在信号初始时刻 (即 $t=0$ 时), 信号的鲁棒性 $\rho(\varphi,w,0)$ 通常被简写为 $\rho(\varphi,w)$.

表 1 信号时序逻辑公式的布尔语义和定量语义

布尔语义	定量语义 (鲁棒性)
$w,t \vDash \text{true}$	$\rho(\text{true},w,t)=\top$
$w,t \vDash \theta(X) \geqslant d \Leftrightarrow \theta(w(t)) \geqslant d$	$\rho(\theta(X) \geqslant d,w,t)=\theta(w(t))-d$
$w,t \vDash \neg\varphi_1 \Leftrightarrow w,t \nvDash \varphi_1$	$\rho(\neg\varphi_1,w,t)=-\rho(\varphi_1,w,t)$
$w,t \vDash \varphi_1 \vee \varphi_2 \Leftrightarrow w,t \vDash \varphi_1 \vee w,t \vDash \varphi_2$	$\rho(\varphi_1 \vee \varphi_2,w,t)=\max\{\rho(\varphi_1,w,t),\rho(\varphi_2,w,t)\}$
$w,t \vDash \varphi_1 \mathbf{U}_I \varphi_2 \Leftrightarrow \exists t' \in \{t+i \mid i \in I\}, w,t' \vDash \varphi_2 \wedge \forall t'' \in [t,t'), w,t'' \vDash \varphi_1$	$\rho(\varphi_1 \mathbf{U}_I \varphi_2,w,t)=\sup\limits_{t' \in \{t+i \mid i \in I\}} \min\{\rho(\varphi_2,w,t'), \inf\limits_{t'' \in [t,t')} \rho(\varphi_1,w,t'')\}$

在此基础上, 我们给出 CPS 系统的反例生成 (falsification) 问题的定义.

问题定义: CPS 系统反例生成. 给定 CPS 系统的模型 $\mathcal{M}$ 与规约 φ, 反例生成旨在模型 $\mathcal{M}$ 中找到违背规约 φ 的不安全行为. 譬如给定 CPS 系统的混成自动机模型 $H=(X,Q,F,Inv,E,G,R,Q^0,Q^f)$ 和信号时序逻辑规约 φ, 反例生成的目标为在混成自动机 H 中找到合法的系统行为, 使得此时系统的行为轨迹 (即系统状态的信号) $w^S \nvDash \varphi$.

针对上述 CPS 系统的反例生成问题, 基于路径的反例生成方法的基本框架如图 1 所示. 这类方法在底层根据系统的混成自动机模型生成从 Q^0 中的节点出发到达 Q^f 中的节点的系统路径, 在顶层针对每条系统路径依次进行基于优化的路径反例生成, 搜索寻找该路径上合法但违背目标规约的不安全行为.

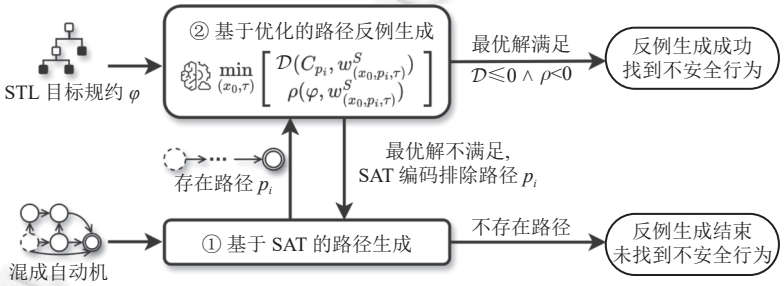


图 1 基于路径的 CPS 系统反例生成方法框架

具体来说, 底层的路径生成可以通过对自动机的离散图结构进行可满足性问题 (SAT) 编码和求解得到, 详细的形式化定义可见文献 [27]. 通常情况下, 基于路径的方法仅考虑有限步数内的路径以控制复杂度, 即限制路径中离散跳转次数上限, 这与实际应用中系统通常在有限的时间和资源内运行也是相符合的.

顶层的路径反例生成则是基于路径约束的不满足度[13]以及信号时序逻辑的定量语义, 将路径上合法但违背 STL 规约的行为搜索问题转化成了多目标优化问题进行求解的. 其中, 路径约束不满足度 $\mathcal{D}(C_{p_i},w^S_{(x_0,p_i,\tau)})$. 用于衡量系统行为 $w^S_{(x_0,p_i,\tau)}$ 是否是路径 p_i 上的合法行为, 即衡量该行为不满足定义 3 中的路径 p_i 上的所有不变式及转移卫式约束 C_{p_i} 的程度. 路径约束不满足度越大则路径约束的违背程度越严重, 当且仅当不满足度 $\mathcal{D}(C_{p_i},w^S_{(x_0,p_i,\tau)}) \leqslant 0$ 时, 系统行为 $w^S_{(x_0,p_i,\tau)}$ 是合法行为. 关于不满足度更详细的形式化定义可见文献 [13]. 而 STL 规约 φ 的违背程度则可基于信号时序逻辑的定量语义, 由规约 φ 在系统行为 $w^S_{(x_0,p_i,\tau)}$ 下的鲁棒性 $\rho(\varphi,w^S_{(x_0,p_i,\tau)})$ 来衡量. 因此, 针对底层生成的每条路径 p_i, 顶层采用优化算法最小化路径约束 C_{p_i} 的不满足度 $\mathcal{D}(C_{p_i},w^S_{(x_0,p_i,\tau)})$ 以及系统规约 φ 的鲁棒性 $\rho(\varphi,w^S_{(x_0,p_i,\tau)})$.

如图 1 所示, 若顶层的路径反例生成模块在给定的优化算法迭代次数内, 找到一组由系统变量初值以及路径节点停留时间构成的最优解 (x_0,τ), 使得 $\mathcal{D}(C_{p_i},w^S_{(x_0,p_i,\tau)}) \leqslant 0$ 且 $\rho(\varphi,w^S_{(x_0,p_i,\tau)})<0$, 则反例生成成功, 找到了违背目标规约的系统合法行为; 否则, 将该路径的 SAT 编码后取反, 加入底层路径生成模块的 SAT 问题编码中, 然后在底层继续通过 SAT 求解来生成还未检查的路径. 若底层 SAT 求解找到路径, 则顶层继续进行基于优化的反例生成, 否则, 说明所有系统路径都已被检查, 反例生成结束, 未找到不安全的系统行为.

2.2 多臂老虎机问题

多臂老虎机 (multi-armed bandit) 问题是概率论中的一个经典问题[34], 其目标是在多个选择 (即多个老虎机的

“臂”) 中找到策略, 从而最大化累积奖励. 每个选择的奖励符合某个未知的概率分布, 因此, 需要通过尝试不同的选择来估计分布, 优化选择. 该问题的形式化定义如下.

定义 5. 多臂老虎机问题. 给定 n 个老虎机的臂 $\{A_i\}_1^n$, 每个老虎机的臂 A_i 每次被选择会获得符合一个未知的分布 μ_i 的奖励, 多臂老虎机问题旨在最大化 T 次选择的累计奖励.

多臂老虎机问题可以被视为一种特殊的优化问题. 与广义的优化问题相比, 多臂老虎机问题更强调环境的动态不确定, 即每个选择获得的奖励均具有一定的随机性且其奖励的分布未知. 因此, 与梯度优化 (如梯度下降和随机梯度下降)、全局优化 (如遗传算法和模拟退火) 等常见的优化算法相比, 多臂老虎机问题的求解方法更强调适应奖励的不确定性并要求在线高效决策, 其核心是需要有效平衡“探索” (试验新的或不被经常选择的臂, 从而获取更多的信息) 和“利用” (选择当前估计最优的臂, 从而获取高奖励).

多臂老虎机问题的常见算法包括 ε-贪心算法和上置信界 (upper confidence bound, UCB) 算法等[35]. 其中, ε-贪心算法每次选择时都以一个较小概率 ε 进行探索, 以 1–ε 的概率选择当前估计最优的臂, 该方法简单直接但容易陷入次优解, 尤其是 ε 取固定值时该算法无法保证收敛. UCB 算法相对复杂但有较好的理论保障, 其核心思想是每次选择时都同时考虑选择的平均奖励以及不确定性, 选择具有最大上置信界的臂. 具体来说, 臂 A_i 的上置信界 $UCB_i = R_i + c\sqrt{(2\ln t/N_i)}$, 其中, R_i 是臂 A_i 的平均奖励, c 是 UCB 算法调节平衡探索和利用的参数, t 标记当前选择是第几次选择, N_i 是臂 A_i 已被选择的次数.

与 ε-贪心算法相比, 虽然 UCB 算法每轮选择的计算代价更大, 但它具有次线性遗憾的理论保证. 其中, 遗憾评估的是算法由于不选择最优臂而造成的累计损失, 即算法策略下期望总奖励与理论最优奖励之间的差值. UCB 算法在 t 次选择后的遗憾已被证明不超过 $O(\sqrt{(nt\ln t)})$. 即累计遗憾量随着选择的次数 t 的增加而增长, 但其增长速度低于线性, 这意味着随着时间的推移, UCB 算法表现会越来越接近最优.

3 基于路径过滤与动态选择的 CPS 系统反例生成

本节介绍我们针对 CPS 系统的反例生成问题, 提出的基于混成自动机路径快速探索过滤与动态选择的反例生成方法, 方法框架如图 2 所示.

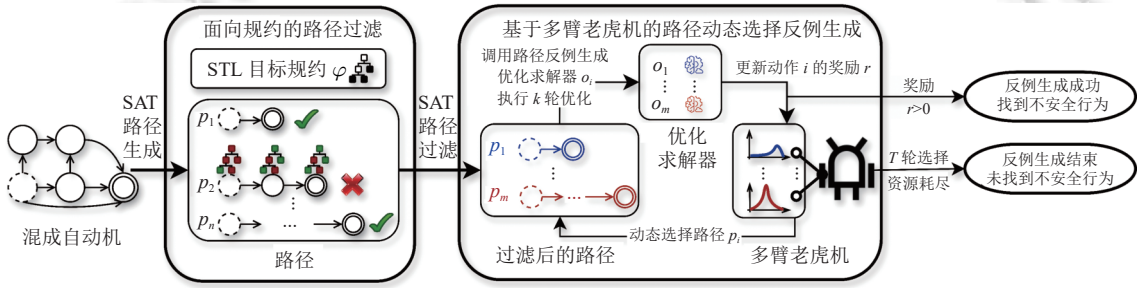


图 2 基于混成自动机路径过滤与动态选择的 CPS 系统反例生成方法框架

首先, 我们针对系统的混成自动机模型生成系统所有可能执行的离散路径. 由于生成的路径可能数量规模巨大, 且后续针对路径进行反例生成过程涉及根据常微分方程对系统行为轨迹进行密集计算, 非常耗时, 因此, 路径爆炸会限制方法性能. 针对这一障碍, 我们设计了面向规约的路径快速探索与过滤方法. 根据目标规约的语法解析树以及路径上的约束, 我们基于时序逻辑的语义以及 SMT (satisfiability modulo theories) 求解器快速过滤一定不存在违背目标规约的合法行为的路径. 第 3.1 节将具体介绍我们面向规约的路径过滤算法.

针对过滤后的路径, 传统方法会逐条路径基于优化算法依次进行反例生成. 但无论采取什么固定次序对路径进行优化求解和反例生成, 都可能导致在初始阶段就将过多计算资源分配给不太可能包含不安全行为的路径, 而实际存在不安全行为的路径对应的状态空间较晚才被探索到, 浪费时间和资源. 不仅如此, 由于缺乏足够的信息来判断各条路径上存在不安全行为的潜力, 导致传统的方法只能对每条路径平均划分计算资源, 即每条路径的反例

生成优化算法的迭代次数上限相同, 这导致具有潜力的路径对应的状态空间可能不能被充分探索. 针对这一问题, 我们提出了多臂老虎机问题算法指导的路径动态选择的反例生成方法, 通过在每一轮选择上置信界更大、更有潜力的路径来进行基于优化的反例生成, 从而尽可能地平衡对系统状态空间的探索和利用, 自适应地动态调整对不同路径上的行为进行探索的资源分配, 最大化状态空间的探索效率. 第 3.2 节将具体介绍我们的路径动态选择反例生成算法.

3.1 面向目标规约的路径快速探索与过滤

在基于路径的反例生成方法里, 针对底层生成的路径进行基于优化的反例生成过程通常是该方法的主要时间开销来源. 如果能在反例生成前就排除部分路径, 可以有效缓解路径爆炸问题, 提高反例生成的效率. 本节将介绍基于如何根据反例生成的目标规约 φ, 高效地过滤一定不存在违背目标规约的合法行为的系统路径.

根据反例生成的问题定义, 需要寻找系统合法但不安全的行为 $w^S_{(x_0,p,\tau)} \nvDash \varphi$, 即 $w^S_{(x_0,p,\tau)} \vDash \neg\varphi$. 而根据定义 3, 给定系统路径 $p=\{q_j\}_0^N$, 其上行为 $w^S_{(x_0,p,\tau)}$ 是系统合法行为要求 $w^S_{(x_0,p,\tau)} \vDash C_p$, 其中, C_p 为在路径 p 上各节点时的不变式和转移卫式约束. 因此, 当 C_p 中的约束与时序规约 $\neg\varphi$ 冲突时, 路径 p 上一定不存在违背目标规约的合法系统行为, 即该路径可以被直接过滤, 无需进行反例生成.

基于信号时序逻辑的语义, 我们设计了根据检测路径约束 C_p 与时序规约 $\neg\varphi$ 是否一定冲突来过滤路径的算法, 伪代码见算法 1. 该算法采用动态规划的思想, 为路径上的每个节点以及规约 $\neg\varphi$ 的语法解析树中的所有子式分别判断, 子式在节点是否一定成立或冲突. 每个子问题的判断都基于 SMT 求解器的结果或其他子问题的判断结果. 其中, SMT 求解器是一种用于确定逻辑公式在给定理论下是否可满足的工具, 其返回结果通常有以下几种情况: SAT 表示给定的逻辑公式是可满足的, 即存在一种变量赋值使得所有的约束条件都能被满足; UNSAT 表示给定的逻辑公式是不可满足的, 即一定不存在任何一种变量赋值能使所有的约束条件都被同时满足; UNKNOWN 表示求解器无法确定给定的逻辑公式是否可满足.

算法 1. 面向目标规约的路径过滤.

输入: 待检查路径 $p=\{q_i\}_0^N$, 反例生成的目标规约 φ;
输出: 是否过滤路径 p, true 表示过滤, false 表示保留.

1. **Function** pathFilter(φ, p)
2. φ_{neg} = transform($\neg\varphi$); ▷ 通过等价转换将否定符号 ¬ 移至谓词前
3. ϕ = parse(φ_{neg}); ▷ 存储 φ_{neg} 的语法树中的所有公式 (沿语法树自顶向下)
4. **init** $\mathcal{R}[|\phi|][N+1]$; ▷ 公式在节点上的冲突表, 1: 总成立, 0: 未知, –1: 冲突
5. **for** $i=|\phi|-1$ **to** 0 **do** ▷ 遍历所有公式, 从底层的谓词到顶层的 φ_{neg} 自身
6. **for** $j=N$ **to** 0 **do** ▷ 从路径的最后一个节点到第 1 个节点
7. **if** $\phi[i]$:= true **then** $\mathcal{R}[i,j]=1$; ▷ 总成立
8. **else if** $\phi[i] := \theta(X) \geqslant d$ **then** ▷ SMT 检查公式 $\phi[i]$ 与节点 q_j 是否冲突
9. **if** SMTsolver($Inv_{q_j}, Q=q_j, \phi[i]$)=UNSAT **then** $\mathcal{R}[i,j]=-1$;
10. **else if** SMTsolver($Inv_{q_j}, Q=q_j, \neg\phi[i]$) =UNSAT **then** $\mathcal{R}[i,j]=1$;
11. **else** $\mathcal{R}[i,j]=0$; ▷ 未知
12. **else if** $\phi[i] := \neg\phi[k]$ **then** $\mathcal{R}[i,j]=-\mathcal{R}[k,j]$; ▷ 根据表 1
13. **else if** $\phi[i] := \phi[k_1] \vee \phi[k_2]$ **then** $\mathcal{R}[i,j]=\max(\mathcal{R}[k_1,j], \mathcal{R}[k_2,j])$;
14. **else if** $\phi[i] := \phi[k_1]\ \mathbf{U}_I\ \phi[k_2]$ **then** ▷ 根据表 1
15. $\mathcal{R}[i,j]=\min(0, \mathcal{R}[k_2,j])$;
16. **if** $j<N$ **then** $\mathcal{R}[i,j]=\max(\mathcal{R}[i,j], \mathcal{R}[i,j+1])$;

17. **if** $\mathcal{R}[0,0]=-1$ **then return** true; ▷ 路径 p 与 $\neg\varphi$ 冲突, 过滤该路径
18. **return** false; ▷ 保留该路径

具体来说, 面向目标规约的路径过滤算法 1 在第 4 行初始化了一个动态规划表 $\mathcal{R}$, 其中 $\mathcal{R}[i,j]$ 将用来存储公式 $\phi[i]$ 在节点 q_j 是否冲突, 其中 1 表示 $\phi[i]$ 总成立 (一定不冲突), −1 表示一定冲突, 否则为 0. 该算法自底向上依次计算 $\neg\varphi$ 的语法解析树中的原子谓词, 子公式和最顶层公式在路径各节点的冲突情况 (第 5–16 行), 每次都从路径上的最后一个节点向前计算到初始节点 (第 6–16 行). 该动态规划算法在计算每个公式在每个节点上的冲突情况时, 通过动态规划表 $\mathcal{R}$ 重用了该公式在后续节点的冲突计算结果以及其子公式的冲突计算结果, 从而避免了重复计算. 我们根据公式 $\phi[i]$ 的语法分情况讨论.

• $\phi[i]:=\text{true}$ 时, $\phi[i]$ 在节点 q_j 上总成立 (第 7 行);

• $\phi[i]:=\theta(X)\geqslant d$ 时, 使用 SMT 求解器判断节点 q_j 上的不变式等约束与 $\phi[i]$ 是否总冲突, 若总冲突说明 $\phi[i]$ 在该节点一定不成立 (默认节点内的不变式等约束有解, 否则该节点可以被提前删去). 类似地, 判断节点上的约束与 $\neg\phi[i]$ 是否总冲突, 若总冲突则说明 $\phi[i]$ 在该节点一定成立. 否则, $\phi[i]$ 在该节点的冲突情况为未知 (第 8–11 行);

• $\phi[i]:=\neg\varphi_1$ 时, 若 φ_1 在节点 q_j 上总成立则 $\phi[i]$ 在节点 q_j 上总冲突, 若 φ_1 在节点 q_j 上总冲突则 $\phi[i]$ 在节点 q_j 上总成立, 否则未知 (第 12 行);

• $\phi[i]:=\varphi_1\vee\varphi_2$ 时, $\phi[i]$ 的冲突情况为 φ_1 或 φ_2 在节点 q_j 上的最好情况 (第 13 行);

• $\phi[i]:=\varphi_1\mathbf{U}_I\varphi_2$ 时, 若 φ_2 在当前节点以及后续所有节点都冲突, 则 $\phi[i]$ 在当前节点一定冲突; 否则, 由于路径节点与时间区间 I 的关系无法明确, 所以冲突情况未知 (第 14–16 行).

譬如给定在图 3 所示的地图内移动的车辆导航系统, 车辆在不同网格有不同的期望速度, 网格内的蓝色箭头代表期望速度的方向. 绿色网格为车辆出发点, 蓝色网格为目的点, 红色区域为危险区域. 因此, 导航系统的反例生成目标规约为 $\neg\mathbf{F}(|x-2|<0.5\wedge|y-2|<0.5)$, 即车辆应总避免进入地图上以 $(2,2)$ 为中心的边长为 1 的红色正方形危险区域. 在该导航系统对应的混成自动机中, 连续变量包括车辆的位置 x,y 等, 离散节点则与 25 个网格对应, 标为 q_1-q_{25}, 相邻网格对应的节点间具有离散跳转. 在每个节点上, 不变式约束由网格位置决定. 由于该混成自动机的初始节点为 q_{19}, 终止节点为 q_2, 因此, 图上的黑色路径 $p=\{q_{19},q_{14},q_9,q_4,q_3,q_2\}$ 即为一条系统路径.

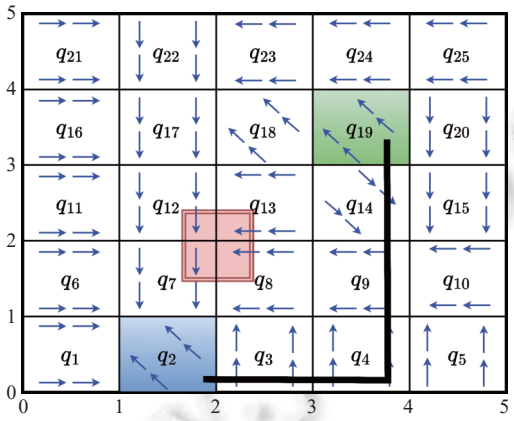


图 3 导航系统的二维网格地图

根据算法 1 计算路径 p 是否需要被过滤的流程如表 2 所示. 对目标规约取反得到 $\varphi_{\text{neg}}=\mathbf{F}(|x-2|<0.5\wedge|y-2|<0.5)$. 针对 φ_{neg} 中包含的所有公式, 依次 (逐行) 计算公式在路径 p 每个节点上的约束满足情况, 从节点 q_2 自后向前依次计算. 譬如节点 q_2 的不变式约束为 $1<x<2\wedge 0<y<1$, 与公式 $|x-2|<0.5$ 不冲突, 与公式 $\neg(|x-2|<0.5)$ 也不冲突, 故约束满足情况未知, 记为 0; 而节点 q_4 的不变式约束为 $3<x<4\wedge 0<y<1$, 与公式 $|x-2|<0.5$ 冲突, 故约束满足情况记为–1. 最终, 最后一列的公式 φ_{neg} 在第 1 个节点 q_{19} 上沿该路径的约束满足情况为–1, 即冲突. 因

此, 图上的黑色路径上一定不存在违背规约的合法行为, 可以被过滤. 该算法可以保证过滤的路径上一定不存在违背目标规约的合法行为, 但该算法并不能保证完备. 一方面, SMT 求解器检查谓词公式与约束是否冲突时, 基于的是其 UNSAT (不可满足) 判断结果, 该结果是可靠但不保证完备的; 另一方面, 由于路径节点和时序算子的时间区间并无直接对应关系, 所以我们对于包含时序算子的公式的冲突计算也是不完备的. 但值得注意的是, 我们在该算法第 2 行通过等价转换来移动 $\neg\varphi$ 的否定符号至谓词前, 形成公式 φ_{neg}, 尽可能地检查出更多的可过滤路径. 此外, 在算法 1 伪代码中, 为与定义 4 中 STL 公式语法相对照, 我们统一给出了包含时序算子的公式的冲突情况计算方法. 但在实际实现中, 我们可以对包含时序算子 U, G, F, 以及时序算子下标的时间区间 I 是否包含时间 0 等情况进行分类处理, 从而最大化地过滤路径.

表 2 路径 p 面向规约的过滤计算动态规划表

公式	q_{19}	q_{14}	q_9	q_4	q_3	q_2
$\lvert x-2\rvert<0.5$	−1	−1	−1	−1	0	0
$\lvert y-2\rvert<0.5$	−1	0	0	−1	−1	−1
$\lvert x-2\rvert<0.5 \wedge \lvert y-2\rvert<0.5$	−1	−1	−1	−1	−1	−1
$\mathbf{F}(\lvert x-2\rvert<0.5 \wedge \lvert y-2\rvert<0.5)$	−1	−1	−1	−1	−1	−1

在算法时间开销方面, 该算法采用动态规划思想, 将公式 $\neg\varphi$ 与路径 $p=\{q_j\}_0^N$ 上的约束 C_p 的冲突情况计算分解为各个子式在各个节点上的冲突情况计算. 一共有 $(N+1)\times(m_1+m_2)$ 个子问题, 其中 m_1 为 φ_{neg} 的语法树上形如 $\theta(X)\geqslant d$ 的谓词的个数, m_2 为 φ_{neg} 的语法树上其他公式的个数. 对于非谓词公式对应的 $(N+1)\times m_2$ 子问题, 算法复杂度均为 O(1); 对于剩余的谓词公式对应的每个子问题, 算法效率取决于 SMT 求解器的约束求解效率.

由于不同理论的复杂度对 SMT 求解器的性能影响显著: 只包含布尔公式时, 求解效率较高; 包含线性整数算术时, 复杂度通常为多项式级别; 而涉及非线性算术时, 复杂度显著提升. 因此, 为了能够在不同场景下平衡路径过滤算法的效率和效果, 在算法 1 的第 9–10 行实现中, 我们提供了路径过滤的配置选项. 用户可根据实际需求自主选择是否将布尔、线性以及非线性类型的约束交给 SMT 求解器处理. 若选择将所有约束都交给 SMT 求解器求解, 则可以最大化地过滤路径, 但求解耗时可能较多; 若仅将部分约束交给求解器求解, 则不交给 SMT 处理的约束冲突情况将被记录为未知, 尽管这可能降低路径过滤算法的完备性, 但可在确保过滤算法正确性 (即过滤的路径上一定不存在违背目标规约的合法行为) 的同时减少算法耗时. 本文在后续第 4.3 节中将结合实验数据进一步探讨约束求解与过滤算法的性能权衡. 总体而言, 在实际实验中, 我们发现只考虑布尔或线性约束时, 路径过滤算法通常都可以在毫秒级完成对路径的快速探索与过滤, 其时间消耗远小于对路径进行基于优化的反例生成, 可以有效缓解路径爆炸.

3.2 多臂老虎机算法指导的路径动态选择

系统在不同路径上的状态空间通常差异较大, 存在不安全的行为的概率分布也有所不同. 因此, 在探索多条路径的状态空间以寻找系统不安全行为的过程中, 动态地选择更值得探索的路径进行反例生成, 可以提高发现不安全行为的效率和成功率.

路径的动态选择可以被视为一个在线优化过程, 每轮优化都需要根据已有信息做出决策, 选择一条路径在该轮进行探索. 不仅如此, 该优化问题的奖励具有不确定性, 在每次选择路径后采样评估该路径上存在不安全行为的概率 (奖励) 时, 评估结果是具有不确定性且分布未知的. 因此, 路径的动态选择是一个典型的奖励具有不确定性的在线优化问题, 传统的梯度下降等优化算法难以适用. 本节将基于定义 5 中介绍的更适用于该类问题的多臂老虎机经典算法, 提出多臂老虎机算法指导的路径动态选择反例生成方法, 以提高反例生成的性能.

多臂老虎机算法的核心在于解决在空间探索过程中, 对未充分了解的空间的探索与对已获取的探索信息的利用之间的权衡问题. 类似地, 在基于路径的反例生成问题里, 每一条路径可以视作一个选择, 而对路径上的行为进行一系列采样模拟并评估行为是否违背规约可以被看作是对该选择的探索. 选择行为空间尚未被充分探索的路径进行探索, 还是选择已发现很接近不安全行为的路径进行探索, 需要充分权衡. 为使用已有广泛研究和充分理论支

撑的多臂老虎机算法来指导我们在路径选择过程中做出最佳权衡, 我们对基于路径的反例生成方法进行分析与调整.

具体来说, 所有路径都有各自的用于行为空间探索的优化求解器. 为对每次选择探索的路径进行评估, 我们对该路径执行 k 轮基于优化的反例生成, 即基于其优化求解器采样 k 条该路径上的系统行为进行评估, k 为远小于系统反例生成全过程的行为模拟次数总上限 K 的固定值. 当某条路径被再次被选择时, 其优化求解器中保留上一轮探索的最优值等信息. 由于我们的反例生成被转化为了多目标优化问题, 需要同时最小化 $\mathcal{D}(C_{p_i}, w^S_{(x_0,p_i,\tau)})$ 和 $\rho(\varphi, w^S_{(x_0,p_i,\tau)})$, 优化目标为 $\mathcal{D}(C_{p_i}, w^S_{(x_0,p_i,\tau)}) \leqslant 0$ 且 $\rho(\varphi, w^S_{(x_0,p_i,\tau)}) < 0$, 因此, 我们将每次路径选择的奖励值设为 $r(\tilde{w}) = -\max(0, \mathcal{D}(C_{p_i}, \tilde{w})) - \max(-\epsilon, \rho(\phi, \tilde{w})), \epsilon \to 0^+$, 即 k 轮优化后找到的最优行为 $\tilde{w} = w^S_{(\tilde{x}_0,p_i,\tilde{\tau})}$ 对应的函数值. 奖励值越大, 则路径上的行为越接近合法且不安全的系统行为.

由于每轮路径选择后, 基于优化计算路径奖励的代价较大, 涉及对 CPS 系统复杂混成行为的迭代模拟计算, 远大于每一轮的路径选择时对所有路径的奖励分布进行计算排序的耗时, 所以我们采用第 2.2 节介绍的策略计算复杂度更高但具有更强理论保障的上置信界 (UCB) 算法来指导每轮的路径选择. 我们的路径动态选择的反例生成过程如图 2 相关部分所示, 对应的算法伪代码见算法 2.

算法 2. 基于多臂老虎机算法指导的路径动态选择的反例生成.

输入: 路径集合 $P = \{p_i\}_1^n$, 反例生成的目标规约 φ, 动态选择轮数 T, 动态选择的平衡参数 c, 系统行为模拟的次数上限 K;
输出: 若反例生成成功则返回找到的系统不安全行为, 否则返回空指针.

```
1. Function pathFalsify(P, φ, T, c, K)
2.     for p_i ∈ P do                                  ▷ 为每条路径初始化
3.         o_i = initOptimizer(C_{p_i}, φ);            ▷ 根据路径约束及目标规约初始化路径的反例生成优化求解器
4.         N_i = 0, R_i = 0;                           ▷ 初始化路径选择次数和奖励估计值
5.     for t = 1 to T do                               ▷ 基于 UCB 算法动态选择 T 轮路径进行反例生成
6.         for p_i ∈ P do                              ▷ 计算每条路径的上置信界
7.             if N_i = 0 then UCB_i = ∞;              ▷ 路径还未被选择
8.             else UCB_i = R_i + c × sqrt(2 ln t / N_i);   ▷ 根据公式计算上置信界
9.         s = argmax_i UCB_i;                         ▷ 选择上置信界最大的路径 p_s
10.        w̃ = −o_s.run(K/T);                          ▷ 继续执行 K/T 轮优化记录路径上最优行为
11.        r = −max(0, D(C_{p_i}, w̃)) − max(−ε, ρ(φ, w̃));   ▷ 计算奖励
12.        if r > 0 then return w̃;                     ▷ 反例生成成功, 找到系统中的不安全行为
13.        N_s = N_s + 1;
14.        R_s = R_s + (r − R_s)/N_s;                  ▷ 更新选择的路径的奖励估计值
15.    return NULL;                                    ▷ 未找到不安全行为
```

具体来说, 本文提出的基于路径动态选择的反例生成算法的输入包括路径集合 P, 反例生成的目标规约 φ, 动态选择轮数 T, 平衡参数 c, 以及系统行为模拟的次数总上限 K. 首先, 算法为每条路径 p_i 初始化优化求解器 o_i, 并将路径选择次数 N_i 和奖励估计值 R_i 均初始化为 0 (第 2–4 行). 然后, 基于多臂老虎机问题的 UCB 算法动态选择 T 轮路径并进行基于优化的反例生成 (第 5–14 行). 在每轮动态选择过程中, 算法计算每条路径的奖励上置信界 UCB (第 6–8 行), 并选择 UCB 值最大的路径 p_s 进行反例生成. 选定路径 p_s 后, 算法运行其反例生成的优化求解器 o_s, 进行 $k = K/T$ 轮迭代优化以计算选择路径 p_s 获得的奖励 r (第 10, 11 行). 随后, 更新选择路径的选择次数 N_s 和平均奖励 R_s (第 13, 14 行). 如果在某轮迭代中, 反例生成的优化求解器找到了系统的不安全行为, 即奖励 $r > 0$, 则反例生成成功 (第 12 行); 否则, 在所有路径轮次结束后返回反例生成失败, 未找到系统中的不安全行为

(第 15 行). 该算法通过多臂老虎机问题的 UCB 算法, 尽可能地平衡对不同路径下系统的行为空间的探索和利用, 自适应地动态调整对不同路径的反例生成的资源分配, 从而提高了反例生成的性能.

4 实验分析

4.1 工具实现

本文提出的路径过滤与动态选择算法可以普遍适用于基于路径的 CPS 系统反例生成中. 正如在本文相关工作章节中所讨论的, 现有的基于路径的 CPS 系统反例生成工具主要包括 PDF[13]和 SNIFF[14]. 其中, PDF 仅支持对系统的可达性规约 (reachability property) 进行反例生成, 而 SNIFF 则支持更加广泛的信号时序逻辑规约. 鉴于本文方法同样可以支持表达能力更强的信号时序逻辑规约, 我们将本文方法在工具 SNIFF 的基础上采用 C++进行了扩展实现.

具体来说, 在面向目标规约的路径过滤算法实现中, 我们采用了较为成熟的 Z3[36]求解器进行 SMT 约束求解. 并且, 路径过滤算法的实现为用户提供了参数选项, 支持用户选择针对布尔、线性、非线性等类型的约束进行基于 SMT 求解的路径过滤检查 (默认参数设置为仅检查布尔和线性约束). 此外, 动态路径选择算法的实现也为用户提供了参数选项, 支持用户对 UCB 算法中平衡空间探索与利用的参数 c 进行设置 (默认值为 1).

4.2 实验设置

为了评估基于路径过滤和动态选择的 CPS 系统反例生成方法的有效性, 我们研究了以下 3 个问题.

• RQ1: 面向规约的路径过滤算法能否有效过滤路径, 能否优化基于路径的反例生成的性能?

• RQ2: 多臂老虎机算法指导的动态路径选择能否提高基于路径的反例生成的成功率和效率?

• RQ3: 当路径过滤优化和路径动态选择优化组合使用时, 我们反例生成方法的最终性能优化如何?

为了实验评估上述问题, 我们将工具 SNIFF 以及我们扩展后的不同版本, 分别应用在一组经典的混成系统反例生成的测试用例上. 这组测试用例中的系统均规模较大或具有复杂的非线性行为. 表 3 中列出了涉及的每个系统的名称, 系统对应的混成自动机模型上的连续变量数, 离散节点数, 离散跳转数等信息, 最后一列还标出了从自动机模型的初始节点出发到达目标节点的有界路径的数目, 路径数在 6–1 000 条不等.

表 3 测试用例系统的混成自动机模型信息

系统	连续变量数	离散节点数	跳转数	初始节点	目标节点	路径数目
血糖控制系统	4	6	10	q_1	q_5	6
导航系统 (1个车辆)	5	25	80	q_2	q_{19}	10
导航系统 (2个车辆)	10	625	160	(q_2,q_2)	(q_{19},q_{19})	100
导航系统 (3个车辆)	15	15 625	240	(q_2,q_2,q_2)	(q_{19},q_{19},q_{19})	1 000

表 3 中, 血糖控制系统[37]模拟了糖尿病患者的人体血液系统中葡萄糖与胰岛素相互作用的动力学模型, 考虑了胰岛素注射和碳水化合物摄入对血糖水平的影响. 具体而言, G 为高于基础值 G_B 的血浆葡萄糖浓度, I 为高于基础值 I_B 的血浆胰岛素浓度, X 为间质室中的胰岛素浓度, 其行为可根据胰岛素注射策略以及碳水化合物摄入后的时间划分成 6 个不同的模式 (离散节点), 每个模式下其连续行为都由非线性的多项式模拟.

在糖尿病患者的血糖控制系统中, 系统规约共有 8 条, 主要包括以下 3 大类要求.

• 避免严重高血糖: $\mathbf{G}_{[0,360]}(G < G_{\text{serveHyper}})$, 6 h 内血浆中的葡萄糖浓度总低于浓度上限 $G_{\text{serveHyper}}$;

• 避免长期较高血糖: $\neg\mathbf{F}\,(\mathbf{G}_{[0,120]}(G > G_{\text{hyper}}))$, 血糖浓度避免连续 2 h 都高于高血糖浓度常数 G_{hyper};

• 避免长期低血糖: $\neg\mathbf{F}\,(\mathbf{G}_{[0,20]}(G < G_{\text{hypro}}))$, 血糖浓度避免连续 20 min 都低于低血糖浓度常数 G_{hypro}.

导航系统[20,38]模拟移动实体 (例如车辆) 在一个网格化的二维地图内移动的动态行为. 地图里包含实体的初始网格, 要到达的目标网格, 以及需要避免的不安全网格. 实体在每个网格下都有一个特定的期望速度 v_d, 实体的实际速度由微分方程 $\dot{v} = A(v - v_d)$ 确定, 其中 A 是一个矩阵, 保证速度收敛到期望速度. 文献 [38] 中提供了一系列地

图测试用例, 我们选取在该论文中展示的 Nav11 地图进行实验. 该案例的地图上包含 5×5 个网格. 因此, 当地图上有 1 个车辆时, 对应的混成自动机模型有 25 个离散节点, 车辆中有 5 个连续变量. 而当考虑 2 个车辆在地图上的组合行为时, 对应的混成自动机模型有 25×25=625 个离散节点, 当考虑 3 个车辆在地图上的组合行为时, 对应的混成自动机模型有 25×25×25=15625 个离散节点, 此时路径爆炸问题带来的挑战巨大.

在导航系统中, 1 个、2 个和 3 个车辆的场景下, 系统规约共有 17 条, 主要包括以下 4 大类要求.

• 车辆不进入危险网格: $\neg\mathbf{F}_{[t_1,t_2]}(Q=q_i)$, 车辆避免在危险时间区间 $[t_1,t_2]$ 进入地图上的 i 号网格;

• 车辆不进入危险区域: $\neg\mathbf{F}_{[t_1,t_2]}(|x-x_d|<l/2 \wedge |y-y_d|<m/2)$, 车辆避免在危险时间区间 $[t_1,t_2]$ 进入地图上以 (x_d,y_d) 为中心的, 大小为 $l\times m$ 的矩形危险区域;

• 车辆远离危险点: $\mathbf{G}_{[0,60]}\left(\sqrt{(x-x_d)^2+(y-y_d)^2}>D_d\right)$, 1 h 内车辆距危险点总远于危险距离常数 D_d;

• 车辆间防止碰撞: $\neg\mathbf{F}\left(\sqrt{(x_1-x_2)^2+(y_1-y_2)^2}\leqslant D_{\text{safe}}\right)$, 两车距离避免小于等于安全距离常数 D_{safe}.

我们对上述系统的 25 个系统规约进行反例生成实验. 所有实验都在同一台 PC 机上进行 (Intel Core i5-12500, 16 GB RAM, Ubuntu 22.04.4 LTS). 由于反例生成过程中使用的优化求解算法的效果具有随机性, 所以在 25 个案例里, 每个算法都被执行了 100 次, 以对算法性能进行准确可靠的评估. 所有反例生成算法执行时, 其中的优化算法的总迭代次数 (即系统行为的仿真模拟次数上限) 在血糖控制系统以及 1 个和 2 个车辆的导航系统中均为 10 000, 在规模和路径数目都巨大的导航系统 (3 个车辆) 中为 100 000. 此外, 在实验中, 路径过滤算法采用默认配置, 仅对布尔和线性约束进行 SMT 求解. 路径动态选择算法中的 UCB 探索利用平衡参数 c 统一配置为常数值 1. 尽管在实践中, 我们发现在高维问题上较大的 c 值通常能使置信区间更宽, 效果更好; 而在低维问题上较小的 c 值往往效果更佳, 但我们在实验中选择统一设置参数值, 以评估路径动态选择优化技术的通用性, 确保实验结果具有尽可能广泛的适用性.

4.3 实验结果与分析

在 4 个系统的 25 条规约下, 反例生成实验结果见表 4, 包括使用 SNIFF 中原始的基于路径进行反例生成的基本方法, 加上面向规约的路径过滤优化算法, 加上路径动态选择的优化算法, 以及同时加上两个优化技术扩展后的反例生成方法的实验结果. 每种情况我们都列出了其 100 次反例生成的成功率, 成功找到不安全行为的平均耗时, 以及成功找到不安全行为的平均需要消耗的优化算法的总迭代次数 (即系统行为的仿真模拟次数).

表 4 反例生成性能比较实验结果

系统	规约	基本方法			路径过滤				路径动态选择					路径过滤和动态选择		
		成功率 (%)	耗时 (s)	迭代次数	成功率 (%)	耗时 (s)	迭代次数	过滤路径数/总路径数	成功率 (%)	耗时 (s)	迭代次数	成功率提升倍数	迭代次数消耗倍数	成功率 (%)	耗时 (s)	迭代次数
血糖控制系统	s1	83	5.9	6 673	74	5.7	6 374	0/6	**100**	2.3	2 770	1.2	0.4	**100**	2.0	2 310
	s2	45	5.4	6 288	40	5.1	6 017	0/6	**94**	3.0	3 666	2.1	0.6	89	3.2	4 038
	s3	21	5.2	6 228	28	6.1	6 665	0/6	**70**	3.5	4 471	3.3	0.7	66	3.9	5 077
	s4	12	5.3	6 198	9	4.5	6 184	0/6	36	4.1	5 384	3.0	0.9	**52**	4.3	5 841
	s5	52	6.0	3 605	47	5.5	3 395	0/6	**96**	7.9	3 344	1.8	0.9	**96**	9.3	3 913
	s6	91	5.1	3 394	92	6.0	3 724	0/6	**100**	4.2	1 879	1.1	0.6	**100**	4.3	1 953
	s7	**100**	3.5	4 429	**100**	0.9	1 405	3/6	**100**	0.9	950	1.0	0.2	**100**	0.3	256
	s8	79	7.7	5 864	96	6.8	3 577	3/6	97	7.0	2 991	1.2	0.5	**99**	6.7	2 298
导航系统 (1个车辆)	s9	20	0.3	3 300	57	0.3	2 722	7/10	86	0.5	4 272	4.3	1.3	**89**	0.4	3 747
	s10	36	0.3	3 468	81	0.4	3 358	7/10	98	0.4	3 581	2.7	1.0	**99**	0.3	2 584
	s11	59	0.3	3 099	93	0.2	2 060	7/10	**100**	0.3	2 698	1.7	0.9	**100**	0.2	1 844
	s12	17	0.4	3 218	33	0.4	3 297	7/10	60	0.6	5 377	3.5	1.7	**67**	0.5	4 334
	s13	26	0.4	3 630	75	0.4	3 531	7/10	93	0.4	3 907	3.6	1.1	**97**	0.3	2 844

表 4 反例生成性能比较实验结果 (续)

系统	规约	基本方法			路径过滤				路径动态选择					路径过滤和动态选择		
		成功率 (%)	耗时 (s)	迭代次数	成功率 (%)	耗时 (s)	迭代次数	过滤路径数/总路径数	成功率 (%)	耗时 (s)	迭代次数	成功率提升倍数	迭代次数消耗倍数	成功率 (%)	耗时 (s)	迭代次数
导航系统 (2个车辆)	s14	35	0.7	3 050	78	1.0	3 435	70/100	97	1.0	3 839	2.8	1.3	**99**	0.9	2 757
	s15	35	0.6	2 666	95	0.8	2 915	70/100	**100**	0.8	3 143	2.9	1.2	**100**	0.6	2 180
	s16	37	0.7	3 024	98	0.6	1 895	70/100	99	0.7	2 954	2.7	1.0	**100**	0.5	1 857
	s17	26	0.6	3 278	43	1.0	3 195	91/100	30	0.8	3 026	1.2	0.9	**48**	0.5	1 396
	s18	23	0.6	3 616	52	1.3	4 159	91/100	27	0.8	3 034	1.2	0.8	**54**	0.6	1 472
	s19	38	0.7	3 510	67	1.1	4 083	70/100	81	1.7	5 325	2.1	1.5	**87**	1.4	4 160
	s20	31	0.6	2 603	74	1.1	3 898	70/100	92	1.1	3 819	3.0	1.5	**100**	0.9	2 859
	s21	41	0.7	2 947	99	0.6	2 153	70/100	99	0.7	2 930	2.4	1.0	**100**	0.6	2 029
导航系统 (3个车辆)	s22	11	5.2	25 156	83	7.4	22 610	973/1 000	19	9.5	32 263	1.7	1.3	**87**	7.0	13 354
	s23	19	8.5	31 654	61	8.7	27 235	700/1 000	63	9.9	28 165	3.3	0.9	**75**	8.1	23 701
	s24	13	6.3	23 545	22	7.2	25 235	400/1 000	23	8.8	26 580	1.8	1.1	**26**	8.0	25 512
	s25	4	11.9	56 162	65	10.4	24 446	973/1 000	24	10.6	31 143	6.0	0.6	**71**	7.6	14 918
平均		38	3.3	8 824	66	3.3	7 103	54%	75	3.2	7 660	2.5	0.9	**84**	2.9	5 489

RQ1: 面向规约的路径过滤算法能否有效过滤路径, 能否提升基于路径的反例生成的性能?

表 4 中的第 6–9 列为仅采用路径过滤优化后的反例生成实验数据. 其中, 第 9 列展示了 4 个系统的总路径数以及面向每条规约可以过滤的一定不存在不安全行为的路径数目. 可以看到, 实验中的 4 个系统场景分别有 6 条、10 条、100 条和 1 000 条系统路径. 其中, 在包含多个车辆的导航系统中, 由于组合爆炸效应, 系统总路径数较多. 根据第 9 列数据, 可以看出面向规约的路径过滤算法并不能保证在所有场景规约下都能过滤路径. 例如, 在规约 s1–s6 中, 规约与每条路径上的约束都不存在直接冲突, 因此没有路径被过滤. 然而, 在绝大多数例子中, 该算法都检测到了可以提前过滤的路径. 该列数据显示, 在我们实验的 25 个案例中, 默认配置下的路径过滤算法平均能够过滤一半以上 (54%) 的路径, 大幅减少了需要通过耗时的优化算法进行反例生成的路径数量.

此外, 根据第 4.2 节的实验设置, 实验中的路径过滤算法采用默认配置 (仅对布尔和线性约束进行 SMT 求解), 这可能导致部分通过非线性约束求解可被过滤的路径未被检测到. 因此, 我们进行了额外的补充实验, 评估忽略非线性约束对路径过滤效果和效率的影响. 我们对表 4 中 25 个案例的统计显示, 算法 1 中需进行 SMT 求解的约束里, 非线性约束占比 32%. 若在实验的路径过滤过程中, 对这部分非线性约束也进行 SMT 求解, 则平均可过滤的路径比例可以从 54% 进一步提升至 58%, 但整体耗时也将增加至原来的 10 倍以上. 我们通过进一步分析发现, 非线性约束求解对路径过滤有效性提升增益较小主要是因为对非线性约束进行 SMT 求解返回 UNKNOWN 结果的概率较高, 而返回 UNSAT 结果的概率非常低. 尽管如此, 为了适应用户的系统规约与场景需求的多样性, 我们路径过滤算法的实现上依然提供了配置选项, 允许用户自主选择是否对非线性等复杂约束进行 SMT 求解.

而从反例生成的总体性能来看, 与基础方法相比, 加上我们的路径过滤优化后, 平均成功率从 38% 提升到了 66%, 并且耗时并未因为路径过滤算法导致明显的增加. 反例生成成功率的提升主要是由于我们实验设置了对系统进行反例生成的系统行为仿真次数上限, 即所有路径的反例生成优化算法的总迭代次数上限. 因此, 当需反例生成的路径个数大幅减少时, 每条需进行反例生成的路径可以执行的优化迭代次数就会大幅增加.

综上所述, 实验结果表明, 面向规约的路径过滤算法能够有效过滤路径, 并提升基于路径的反例生成性能.

RQ2: 多臂老虎机算法指导的动态路径选择能否提高基于路径的反例生成的成功率和效率?

表 4 的第 10–14 列为仅采用路径动态选择优化后的反例生成实验数据. 从最后一行的平均数据来看, 加上路径动态选择优化后, 反例生成的成功率有高达 2.5 倍的性能提升, 而找到系统不安全行为所消耗的平均迭代次数反而下降到了原先的 0.9 倍. 在绝大多数例子里, 找到不安全行为需要的迭代次数都是下降或者几乎不变的, 而在

为数不多的消耗的迭代次数增加的例子里, 成功率的提升都是超过 70% 的, 即提升到了原来的 1.7 倍以上. 由于我们统计的是成功找到不安全行为时平均所需要消耗的迭代次数, 所以找到解的成功率和需要的迭代次数同时提升的案例意味着优化后的算法在局部优化和全局优化之间取得了更好的平衡.

除此之外, 我们也尝试分析了路径动态选择技术在哪些案例上对成功率提升效果更明显. 通过具体分析我们发现, 该技术的优化效果与被选择的路径的数目关系不明显, 与不同路径上的行为距离违背规约的程度分布差异更相关. 譬如导航系统在 s9 的目标规约下, 有一条路径上的行为明显更容易违背规约, 而其他路径上的行为很难违背规约, 较大的奖励分布差异使得 UCB 算法能够更快地分辨出更具有潜力的路径, 从而更多地探索该路径上的行为空间, 更快地找到不安全行为. 此外, 像规约 s6, s7 对应的反例生成问题, 由于问题本身比较简单, 采用 SNIFF 的基本方法就已经能达到 90% 以上的成功率, 因此动态选择技术在这些案例上对成功率的提升效果比较有限, 但此时找到不安全行为所需要的平均迭代次数大幅减少.

总之, 我们基于多臂老虎机算法指导的动态路径选择技术既减少了反例生成的耗时, 还大幅提高了基于路径的反例生成的成功率, 性能提升非常明显.

RQ3: 当路径过滤优化和路径动态选择优化组合使用时, 我们反例生成方法的最终性能优化如何?

总体来说, 在实验的 4 个算法设置下, 我们基于路径过滤与动态选择的 CPS 系统反例生成方法的效果是最优的, 优于 SNIFF 中的基础方法以及只加任意一种优化技术的方法. 在表 4 中, 我们将 25 个反例生成问题获得的最高的成功率都加粗显示, 可以发现我们的基于路径过滤与动态选择的方法在 23 个例子上都有最高的成功率, 甚至在一半以上的例子上的成功率都不少于 96%. 根据我们的实验数据, 加上两个缓解路径爆炸的优化技术后, 我们可以将 SNIFF 基础方法的成功率提升两倍, 从平均成功率 38% 提升至 84%. 与此同时, 我们的方法也没有因为优化技术而导致耗时增加, 反而因为需要的迭代次数减少而只需要更少的总耗时.

尽管同时加上两个优化技术后, 我们方法最终的反例生成效果优于只加任意一个优化技术的效果, 但我们也发现两个优化技术同时使用对反例生成效果的提升要少于两者单独使用效果提升的总和. 譬如在基本方法上加入路径过滤优化后成功率平均提升了 28% (从 38% 到 66%), 但在已经进行了路径动态选择优化后的算法上再加入路径过滤优化, 成功率仅提升了 9% (从 75% 到 84%). 我们认为这主要是因为路径过滤和路径动态选择这两种优化技术本质都在试图解决类似的问题, 即通过减少分配 (或不分配) 给不具有潜力的路径探索状态空间的计算资源来缓解路径爆炸问题. 因此, 两种优化技术对反例生成性能增益部分有所重叠, 从而减少了总体的额外增益. 但两者评估路径潜力的方法角度不同: 路径过滤采用的是 SMT 技术验证路径上的部分约束与目标规约之间的约束满足关系, 而路径动态选择是对不同路径上行为的奖励分布进行区分, 前者在存在简单直接的约束冲突时表现更好 (譬如表 4 中的规约 s18, s22), 而后者在奖励分布差异较大时表现更好. 因此, 两个优化技术叠加使用可以最大化地缓解路径爆炸问题, 以确保最终能够尽可能地达到最佳的反例生成性能和效果.

5 结论及未来工作

本文提出了一种基于混成自动机路径过滤和动态选择的 CPS 系统反例生成方法, 以应对系统反例生成过程中的路径爆炸问题. 首先, 我们提出了面向规约的路径过滤方法, 通过分析系统规约的语法树及系统路径上的行为约束, 快速过滤不可能包含不安全行为的路径, 从而有效减少需要探索的路径数量. 其次, 我们引入了多臂老虎机算法, 用于指导反例生成过程中的路径动态选择, 根据已探索路径的信息动态调整计算资源的分配, 尽可能将资源分配给更有潜力的路径, 最大化反例生成的性能和效率. 实验结果表明, 本文方法显著提升了 CPS 系统反例生成的性能, 将成功率提高了两倍以上.

由于本文方法是基于混成自动机上的路径的, 故面向的是具备混成自动机模型的 CPS 系统, 或者至少需具备系统的路径结构信息. 然而, 混成自动机等形式化模型在工业界的应用仍较为有限, 复杂 CPS 系统的混成自动机模型往往难以获取. 目前, 已有一些研究关注如何自动从系统的 Simulink 模型转换获取混成自动机模型 [39], 或从系统的时序轨迹中学习生成混成自动机等形式化模型 [40–44]. 因此, 当无法获取 CPS 系统的形式化模型时, 可以先

借助这类研究方法来生成系统的混成自动机模型, 再应用本文的方法进行基于路径的系统反例生成. 我们未来的工作也计划进一步探索如何结合深度学习等技术, 学习系统模型的关键结构信息以用于路径生成与过滤, 以应对日益复杂的 CPS 系统在缺乏形式化模型信息时对反例生成带来的挑战.

References:

[1] Alur R. Principles of Cyber-physical Systems. London: The MIT Press, 2015.

[2] Lee EA, Seshia SA. Introduction to Embedded Systems: A Cyber-physical Systems Approach. 2nd ed., Cambridge: MIT Press, 2017.

[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(1): 3–34. [doi: 10.1016/0304-3975(94)00202-T]

[4] Doyen L, Frehse G, Pappas GJ, Platzer A. Verification of hybrid systems. In: Clarke EM, Henzinger TA, Veith H, Bloem R, eds. Handbook of Model Checking. Cham: Springer, 2018. 1047–1110. [doi: 10.1007/978-3-319-10575-8_30]

[5] Henzinger TA, Kopke PW, Puri A, Varaiya P. What's decidable about hybrid automata? In: Proc. of the 27th Annual ACM Symp. on Theory of Computing. Las Vegas: ACM, 1995. 373–382. [doi: 10.1145/225058.225162]

[6] Kapinski J, Deshmukh JV, Jin XQ, Ito H, Butts K. Simulation-based approaches for verification of embedded control systems: An overview of traditional and advanced modeling, testing, and verification techniques. IEEE Control Systems Magazine, 2016, 36(6): 45–64. [doi: 10.1109/MCS.2016.2602089]

[7] Nghiem T, Sankaranarayanan S, Fainekos G, Ivancić F, Gupta A, Pappas GJ. Monte-Carlo techniques for falsification of temporal properties of non-linear hybrid systems. In: Proc. of the 13th ACM Int'l Conf. on Hybrid Systems: Computation and Control. Stockholm: ACM, 2010. 211–220. [doi: 10.1145/1755952.1755983]

[8] Annapureddy YSR, Fainekos GE. Ant colonies for temporal logic falsification of hybrid systems. In: Proc. of the 36th Annual Conf. on IEEE Industrial Electronics Society. Glendale: IEEE, 2010. 91–96. [doi: 10.1109/IECON.2010.5675195]

[9] Sankaranarayanan S, Fainekos G. Falsification of temporal properties of hybrid systems using the cross-entropy method. In: Proc. of the 15th ACM Int'l Conf. on Hybrid Systems: Computation and Control. Beijing: ACM, 2012. 125–134. [doi: 10.1145/2185632.2185653]

[10] Annpureddy Y, Liu C, Fainekos G, Sankaranarayanan S. S-TALIRO: A tool for temporal logic falsification for hybrid systems. In: Proc. of the 17th Int'l Conf. on Tools and Algorithms for the Construction and Analysis of Systems. Saarbrücken: Springer, 2011. 254–257. [doi: 10.1007/978-3-642-19835-9_21]

[11] Donzé A. Breach, a toolbox for verification and parameter synthesis of hybrid systems. In: Proc. of the 22nd Int'l Conf. on Computer Aided Verification. Edinburgh: Springer, 2010. 167–170. [doi: 10.1007/978-3-642-14295-6_17]

[12] Akazaki T, Liu S, Yamagata Y, Duan YH, Hao JY. Falsification of cyber-physical systems using deep reinforcement learning. In: Proc. of the 22nd Int'l Symp. on Formal Methods. Oxford: Springer, 2018. 456–465. [doi: 10.1007/978-3-319-95582-7_27]

[13] Wang JW, Bu L, Xing SP, Li XD. PDF: Path-oriented, derivative-free approach for safety falsification of nonlinear and nondeterministic CPS. IEEE Trans. on Computer-aided Design of Integrated Circuits and Systems, 2022, 41(2): 238–251. [doi: 10.1109/TCAD.2021.3056360]

[14] Wang JW, Liu WX, Zhang MM, Wei JQ, Shi YH, Bu L, Li XD. Scenario-based flexible modeling and scalable falsification for reconfigurable CPSs. In: Proc. of the 36th Int'l Conf. on Computer Aided Verification. Montreal: Springer, 2024. 329–355. [doi: 10.1007/978-3-031-65633-0_15]

[15] Fainekos GE, Pappas GJ. Robustness of temporal logic specifications for continuous-time signals. Theoretical Computer Science, 2009, 410(42): 4262–4291. [doi: 10.1016/j.tcs.2009.06.021]

[16] Donzé A, Maler O. Robust satisfaction of temporal logic over real-valued signals. In: Proc. of the 8th Int'l Conf. on Formal Modeling and Analysis of Timed Systems. Klosterneuburg: Springer, 2010. 92–106. [doi: 10.1007/978-3-642-15297-9_9]

[17] Mathesen L, Yaghoubi S, Pedrielli G, Fainekos G. Falsification of cyber-physical systems with robustness uncertainty quantification through stochastic optimization with adaptive restart. In: Proc. of the 15th IEEE Int'l Conf. on Automation Science and Engineering (CASE). Vancouver: IEEE, 2019. 991–997. [doi: 10.1109/COASE.2019.8843005]

[18] Yaghoubi S, Fainekos G. Falsification of temporal logic requirements using gradient based local search in space and time. IFAC-PapersOnLine, 2018, 51(16): 103–108. [doi: 10.1016/j.ifacol.2018.08.018]

[19] Zhang ZY, Ernst G, Sedwards S, Arcaini P, Hasuo I. Two-layered falsification of hybrid systems guided by Monte Carlo tree search. IEEE Trans. on Computer-aided Design of Integrated Circuits and Systems, 2018, 37(11): 2894–2905. [doi: 10.1109/TCAD.2018.2858463]

[20] Plaku E, Kavraki LE, Vardi MY. Hybrid systems: From verification to falsification by combining motion planning and discrete search.

Formal Methods in System Design, 2009, 34(2): 157–182. [doi: 10.1007/s10703-008-0058-5]

[21] Dang T, Nahhal T. Coverage-guided test generation for continuous and hybrid systems. Formal Methods in System Design, 2009, 34(2): 183–213. [doi: 10.1007/s10703-009-0066-0]

[22] Dreossi T, Dang T, Donzé A, Kapinski J, Jin XQ, Deshmukh JV. Efficient guiding strategies for testing of temporal properties of hybrid systems. In: Proc. of the 7th Int'l Symp. on NASA Formal Methods. Pasadena: Springer, 2015. 127–142. [doi: 10.1007/978-3-319-17524-9_10]

[23] Zutshi A, Sankaranarayanan S, Deshmukh JV, Kapinski J. A trajectory splicing approach to concretizing counterexamples for hybrid systems. In: Proc. of the 52nd IEEE Conf. on Decision and Control. Firenze: IEEE, 2013. 3918–3925. [doi: 10.1109/CDC.2013.6760488]

[24] Zutshi A, Deshmukh JV, Sankaranarayanan S, Kapinski J. Multiple shooting, CEGAR-based falsification for hybrid systems. In: Proc. of the 14th Int'l Conf. on Embedded Software. New Delhi: ACM, 2014. 5. [doi: 10.1145/2656045.2656061]

[25] Bogomolov S, Frehse G, Gurung A, Li DX, Martius G, Ray R. Falsification of hybrid systems using symbolic reachability and trajectory splicing. In: Proc. of the 22nd ACM Int'l Conf. on Hybrid Systems: Computation and Control. Montreal: ACM, 2019. 1–10. [doi: 10.1145/3302504.3311813]

[26] Bardin S, Herrmann P. Pruning the search space in path-based test generation. In: Proc. of the 2009 Int'l Conf. on Software Testing Verification and Validation. Denver: IEEE, 2009. 240–249. [doi: 10.1109/ICST.2009.15]

[27] Xie DB, Bu L, Zhao JH, Li XD. SAT-LP-IIS joint-directed path-oriented bounded reachability analysis of linear hybrid automata. Formal Methods in System Design, 2014, 45(1): 42–62. [doi: 10.1007/s10703-014-0210-3]

[28] Kuznetsov V, Kinder J, Bucur S, Candea G. Efficient state merging in symbolic execution. In: Proc. of the 33rd ACM SIGPLAN Conf. on Programming Language Design and Implementation. Beijing: ACM, 2012. 193–204. [doi: 10.1145/2254064.2254088]

[29] Staats M, Păsăreanu CS. Parallel symbolic execution for structural test generation. In: Proc. of the 19th Int'l Symp. on Software Testing and Analysis. Trento: ACM, 2010. 183–194. [doi: 10.1145/1831708.1831732]

[30] Bu L, Zhao JH, Li XD. Path-oriented reachability verification of a class of nonlinear hybrid automata using convex programming. In: Proc. of the 11th Int'l Conf. on Verification, Model Checking, and Abstract Interpretation. Madrid: Springer, 2010. 78–94. [doi: 10.1007/978-3-642-11319-2_9]

[31] Bu L, Wang JW, Wu YM, Li XD. From bounded reachability analysis of linear hybrid automata to verification of industrial CPS and IoT. In: Proc. of the 5th Int'l School on Engineering Trustworthy Software Systems. Chongqing: Springer, 2020. 10–43. [doi: 10.1007/978-3-030-55089-9_2]

[32] Xing SP, Wang JW, Bu L, Chen X, Li XD. Approximate optimal hybrid control synthesis by classification-based derivative-free optimization. In: Proc. of the 24th Int'l Conf. on Hybrid Systems: Computation and Control. Nashville: ACM, 2021. 7. [doi: 10.1145/3447928.3456658]

[33] Maler O, Nickovic D. Monitoring temporal properties of continuous signals. In: Proc. of the 2004 Joint Int'l Conf. on Formal Modeling and Analysis of Timed Systems, FORMATS 2004 and Formal Techniques in Real Time and Fault-tolerant Systems, FTRTFT 2004. Grenoble: Springer, 2004. 152–166. [doi: 10.1007/978-3-540-30206-3_12]

[34] Berry DA, Fristedt B. Bandit Problems: Sequential Allocation of Experiments (Monographs on Statistics and Applied Probability). Dordrecht: Springer, 1985. 1–274. [doi: 10.1007/978-94-015-3711-7]

[35] Auer P, Cesa-Bianchi N, Fischer P. Finite-time analysis of the multiarmed bandit problem. Machine Learning, 2002, 47(2): 235–256. [doi: 10.1023/A:1013689704352]

[36] De Moura L, Bjørner N. Z3: An efficient SMT solver. In: Proc. of the 14th Int'l Conf. on Tools and Algorithms for the Construction and Analysis of Systems. Budapest: Springer, 2008. 337–340. [doi: 10.1007/978-3-540-78800-3_24]

[37] Fisher ME. A semiclosed-loop algorithm for the control of blood glucose levels in diabetics. IEEE Trans. on Biomedical Engineering, 1991, 38(1): 57–61. [doi: 10.1109/10.68209]

[38] Fehnker A, Ivančić F. Benchmarks for hybrid systems verification. In: Proc. of the 7th Int'l Workshop on Hybrid Systems: Computation and Control. Philadelphia: Springer, 2004. 326–341. [doi: 10.1007/978-3-540-24743-2_22]

[39] Minopoli S, Frehse G. SL2SX translator: From simulink to SpaceEx models. In: Proc. of the 19th Int'l Conf. on Hybrid Systems: Computation and Control. Vienna: ACM, 2016. 93–98. [doi: 10.1145/2883817.2883826]

[40] García Soto M, Henzinger TA, Schilling C, Zeleznik L. Membership-based synthesis of linear hybrid automata. In: Proc. of the 31st Int'l Conf. on Computer Aided Verification. New York: Springer, 2019. 297–314. [doi: 10.1007/978-3-030-25540-4_16]

[41] Jin XY, An J, Zhan BH, Zhan NJ, Zhang MM. Inferring switched nonlinear dynamical systems. Formal Aspects of Computing, 2021, 33(3): 385–406. [doi: 10.1007/s00165-021-00542-7]

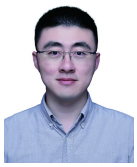
[42] Yang X, Beg OA, Kenigsberg M, Johnson TT. A framework for identification and validation of affine hybrid automata from input-output

traces. ACM Trans. on Cyber-physical Systems, 2022, 6(2): 13. [doi: 10.1145/3470455]

[43] García Soto M, Henzinger TA, Schilling C. Synthesis of parametric hybrid automata from time series. In: Proc. of the 20th Int'l Symp. on Automated Technology for Verification and Analysis. Springer, 2022. 337–353. [doi: 10.1007/978-3-031-19992-9_22]

[44] Gurung A, Waga M, Suenaga K. Learning nonlinear hybrid automata from input-output time-series data. In: Proc. of the 21st Int'l Symp. on Automated Technology for Verification and Analysis. Singapore: Springer, 2023. 33–52. [doi: 10.1007/978-3-031-45329-8_2]

王佳宛(1997－), 女, 博士生, CCF 学生会员, 主要研究领域为复杂软件系统形式化分析与验证.



卜磊(1983－), 男, 博士, 教授, 博士生导师, CCF 杰出会员, 主要研究领域为形式化方法, 复杂软件系统分析与验证.

刘熹橦(2003－), 男, 本科生, 主要研究领域为机器学习.

李宣东(1963－), 男, 博士, 教授, 博士生导师, CCF 会士, 主要研究领域为软件工程, 系统软件, 可信软件, 形式化方法.