软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2023,34(7):3241−3255 [doi: 10.13328/j.cnki.jos.006532]

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

主动自动机学习中的等价查询算法优化*

潘 雁, 祝跃飞

(数学工程与先进计算国家重点实验室 (信息工程大学), 河南 郑州 450001)

通信作者: 祝跃飞, E-mail: yfzhu17@sina.com

摘 要: 模型学习是一种获取黑盒软件系统行为模型的有效方法, 可分为主动学习和被动学习. 主动学习是基于字母表构造测试用例, 通过与黑盒系统主动交互, 可在多项式时间内得到目标系统的最小完备自动机, 其中等价查询仍是开发和应用主动自动机学习工具的障碍之一. 通过探讨反例对于学习算法的影响, 定义假设的比较规则, 提出测试用例构造的两个原则, 同时依据原则对 Wp-method 等价查询算法改进, 产生更优的假设, 有效降低查询的数量, 并基于 LearnLib 开源工具, 分别以 3 类自动机为实验对象验证原则和改进算法的有效性.

关键词: 模型学习; 自动机; 成员查询; 等价查询

中图法分类号: TP18

中文引用格式: 潘雁, 祝跃飞. 主动自动机学习中的等价查询算法优化. 软件学报, 2023, 34(7): 3241–3255. http://www.jos.org.cn/1000-9825/6532.htm

英文引用格式: Pan Y, Zhu YF. Optimization of Equivalence Query Algorithm in Active Automata Learning. Ruan Jian Xue Bao/Journal of Software, 2023, 34(7): 3241–3255 (in Chinese). http://www.jos.org.cn/1000-9825/6532.htm

Optimization of Equivalence Query Algorithm in Active Automata Learning

PAN Yan, ZHU Yue-Fei

(State Key Laboratory of Mathematical Engineering and Advanced Computing (Information Engineering University), Zhengzhou 450001, China)

Abstract: As an effective technique for black-box state machine models of software systems, model learning (a.k.a. automata learning) can be divided into active and passive learning. Based on given input and output alphabets, the minimum complete state machine of the target system can be obtained in polynomial time through active interaction with the black box system. And the algorithm of equivalence query is still a big obstacle to the development and application of active automata learning tools. This study discusses the influence of counterexamples on the learning algorithms with the discrimination tree, and defines the comparison rules of hypotheses, and proposes two principles of constructing test cases. According to the principle, the Wp-method equivalence query algorithm is improved to produce better hypotheses and effectively reduce the number of queries and symbols. Based on the LearnLib, three kinds of automata are used as experimental objects to verify the effectiveness of the principle and the improved algorithm.

Key words: model learning; automata; membership query; equivalence query

计算机技术的发展使得软件系统在日常生活中无处不在, 软件实现的正确性是极其重要的, 而获取未知软件系统的行为模型是极具挑战性的, 其中模型学习[1]是一种获取未知软件系统的行为模型的常用方法, 其对象是黑盒系统, 一般为具有明确输入输出的软件, 如智能卡片[2]、协议实现[3]、遗留软件[4]等. 根据学习的策略, 模型学习一般分为两种: 一是基于软件系统已记录的输入输出数据进行分析, 称为被动学习, 其可以抽象为基于正反例数据进行正则语言的推断, 该问题被证明是 NP 难问题[5], 以协议实现为例, 对于某个协议实现, 已有丰富且覆盖面较广的数据包, 被动学习即是通过已有的数据包推断协议自动机, 但已有的数据包都是问题描述中的正例数据, 没有反

* 基金项目: 国家重点研发计划 (2019QY1300)
收稿时间: 2021-07-26; 修改时间: 2021-09-27, 2021-11-01; 采用时间: 2021-11-09; jos 在线出版时间: 2022-09-23
CNKI 网络首发时间: 2022-11-15

例数据, 这也是实际应用中广泛存在的问题; 二是基于已知字母表构造测试用例, 通过与黑盒系统主动交互来推断自动机, 称为主动自动机学习 (亦称为主动学习), 能在一定程度上克服被动学习仅具有正样本的缺点, 同时被证明可以在多项式时间内得到目标系统的最小完备自动机.

主动学习广泛使用的框架是 Angluin 提出的 MAT (minimal adequete teacher) 框架[6], 主要分为两个步骤: 成员查询和等价查询, 成员查询是构造输入并获得程序的输出, 得到结果后观察是否符合一定条件, 若不符合条件, 则需继续进行查询直至满足一定的条件, 而后依据此结果构建一个假设. 由于只能获取部分输入的响应, 超出此范围的输入并不能保证其假设的正确性, 因此得到假设后需要通过等价查询判断假设的自动机是否正确, 如果不正确, 等价查询将返回一个使假设不成功的反例. 而后算法需要依据反例对原始的分类策略和成员查询的空间进行更新, 同样需要使其符合一定条件, 循环上述过程直至构建的假设与实际系统的自动机相同, 该方法的前提条件是目标自动机的状态数已知.

但是在实际系统中, 具有无穷知识的预言机并不存在, 使得理想的等价查询难以实现. 而是通常基于一致性检测的方法, 通过精心构造的成员查询对等价查询进行模拟, 常用算法包括 W-method[7]和 Wp-method (partial W-method) 算法[8], 以及基于两种算法的随机化算法 random W-method 和 random Wp-method, 但是随机化方法的效果存在不确定性, 因此实际过程中仍以 W-method 和 Wp-method 算法为主[9−12], 其中 Wp-method 算法所需的测试例理论上更少. 由于等价查询的复杂度相较于成员查询更高, 且随着目标状态数呈指数增长, Aarts 等人认为等价查询的测试选择和覆盖仍然是开发和应用主动学习工具的主要问题, 因此, 需要更多的研究以期提高等价查询的测试效率[13].

由于 Wp-method 算法的初衷是全覆盖的一致性测试, 构造的集合是固定的, 不关心测试顺序, 而在模型学习的等价查询阶段执行顺序是重要的, 如 random Wp-method 本质上是随机选取 Wp-method 集合中的测试用例进行查询, 直至找到反例或验证假设, 这些改进算法探讨的更多是在一次等价查询过程中如何更早地找到反例. 而本文尝试探讨不同反例对于所构造的假设的影响, 从而能在后续的学习过程中减少测试用例, 并通过分析优化 Wp-method 构造的测试用例所执行的测试顺序使第一个反例尽可能地好. 本文的主要贡献如下.

(1) 探讨反例对于学习算法的影响, 定义了反例和假设的对比方法.

(2) 基于测试用例构造的两个原则优化 Wp-method 算法的测试顺序, 提高等价查询算法的效率, 并通过实验验证优化效果.

(3) 基于 LearnLib 框架修改的代码开源 (https://gitee.com/z11panyan/learnlib.git, 文中设计的算法为 RefineWpmethod).

本文第 1 节介绍相关工作. 第 2 节介绍主动自动机学习的基础理论. 第 3 节阐述本文提出的优化方法. 第 4 节展示相关的实验结果. 第 5 节总结全文.

1 相关工作

基于 MAT 框架的主动学习的通用流程如图 1 所示, 框架中包含一个学习器与一个预言机, 学习器通过向预言机提出查询来学习未知的模型, 预言机了解 SUL (system under learning) 的所有内容, 但学习者仅知道 SUL 的输入/输出字母表. 为了学习未知自动机, 学习器提出查询, 即发送一个序列至 SUL, 若 SUL 接受则返回“是”, 否则返回“否”; 然后通过接收到的查询响应尝试构造一个行为与目标自动机相匹配的未知自动机, 并提交至预言机, 由预言机进行判断自动机是否正确并给出反例.

为了将模型学习技术应用到具有大量输入和输出的实际系统中, Aarts 等人[14]在 MAT 框架上增加了组件: 映射器 (Mapper), 以便生成具有大消息字母表的组件, 其位于学习器和 SUL 之间, 起着抽象与具象的作用, 学习器向映射器发送抽象消息, 映射器基于输入字母表将其转换为具体的消息并将其转发给 SUL, 同时, 映射器将 SUL 的具体响应转换回抽象消息并发送给学习器. Shahbaz 等人[15]以 Mealy 自动机为目标提出了 L_M^*, 其关注的响应是由输出字母表构成的输出序列. 基于观测表的方法将反例的后缀添加至观测表的列中, 会导致较大的冗余, Kearns 等人[16]构造了一种判别树的数据结构, 替代观测表, 判别树中的叶节点代表状态, 内部节点是两个状态的区分后缀, 保证了由判别树生成的辨别串是无冗余的, 有效降低了测试序列的规模.

等价查询算法方面, 最早采用的是 1978 年 Chow[7]提出的 W-method 算法, 通过构建 n-switch 树, 生成测试序列集合, 并从理论上证明了方法所构建的测试序列集合能有效验证自动机假设与实际自动机的等效性. Fujiwara 等人[8]提出了 Wp-method 算法, 从理论上将全局的特征集合缩小为局部的某个状态的特征集合, 减少了测试序列的数量. 文献 [17] 中的 UIO (unique input/output) 方法是一种更特殊的情形下采用的, 其 UIO 序列是一个可以区分某个状态与其他所有状态的输入序列, 但是这个序列不是每一个状态都存在的. 崔玲等人[18]提出一种基于集合覆盖的测试集约简方法, 通过分析 Wp-method 方法的特点, 找出测试序列之间包含关系的规律, 删除冗余的测试用例. Yang 等人[19]提出主被动结合的等价查询算法, 增加基于日志的被动学习环节与查询缓存, 减少与 SUL 的交互次数.

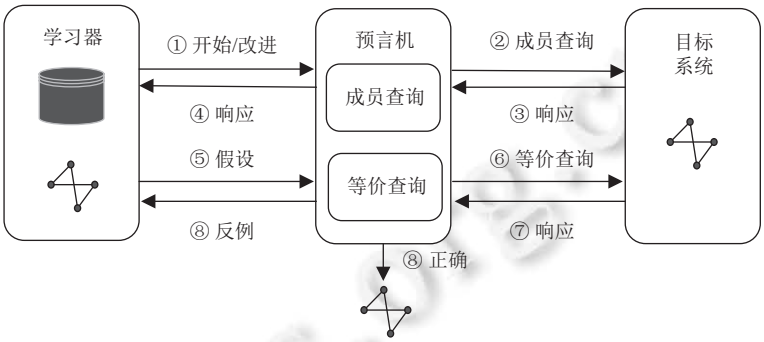

图 1 基于 MAT 框架的通用流程

反例分析算法方面, L*算法是将反例的所有后缀加入观测表的行中, 导致观测表的规模较大, 而 Rivest 等人[20]提出一种二叉搜索的方法, 有效提升了从反例中提取区分后缀的效率. 文献 [21,22] 基于判别树提出 Observation Pack 算法, 通过不断的 Sift 与 Split 操作构建判别树, 其中叶节点使用状态接受序列进行标记, 内部节点使用区分其叶节点代表的状态的后缀序列. 文献 [23,24] 在此基础上, 针对状态图中存在的自循环和状态循环产生的长反例, 提出 TTT 算法, 有效地消除了在分析反例时的过长前缀, 该算法也被各类开源工具采纳并实现.

在自动机类型方面, 由于 Mealy 自动机的表达能力有限, 文献 [25,26] 将主动自动机学习推广到寄存器自动机, 一种能够表达数据对控制流影响的自动机模型, 其值可以分配给寄存器并对下一次输入进行比较, 可直接推断出一部分数据值对控制流的影响, 其表达能力相较于 Mealy 自动机有一定的提升. Argyros 等人[27]采用的符号自动机是寄存器自动机的另一种形式, 其基于符号有限自动机的学习设计了一种黑盒差分测试框架 SFADiff.

实际应用中, Raffelt 等人于 2006 年构建了基于 Java 的开源框架 LearnLib[28], 是被最为广泛使用的框架, 其主要学习的模型为 Moore 与 Mealy 自动机. 其所在团队在 LearnLib 的基础上实现了 RALib[29], 其基于 SL*算法对寄存器自动机进行学习. Aarts 等人于 2012 年在 LearnLib 的基础上实现了 TOMTE, 其关注点在映射器的实现, 其在文献 [30] 与其博士论文中对映射器的抽象与具象行为进行了形式化描述, 并在 TOMTE 中予以实现, 其学习模型主要为寄存器自动机, 上述框架都在持续更新中. 在网络协议领域, Cho 等人[31]基于 L*算法提出了一种预测算法, 采用观测表中的查询结果对新的输入序列的观测结果进行预测, 降低了成员查询的数量级, 并将其应用于僵尸网络命令与控制协议的分析. Ruiter 所在团队分别对 TLS、OpenVPN[9]、QUIC 协议[32]的相关实现进行了分析; Fiterău-Broştean 团队分别针对 TCP[33]、SSH[34]、DTLS 协议[11]进行研究, 并发现了相应的漏洞; 申莹珠等人基于 LearnLib 平台对 OpenVPN[35,36]、IPSec 协议进行了相关的研究; 王辰等人[37]结合协议特点提出了一种基于域知识的协议自动机主动学习算法. 除此之外, 很多研究者将模型学习与模型检测[38]、白盒测试[39]、模糊测试[40]结合以解决新的问题.

2 基础知识

主动学习可以抽象为主动构造一系列可观测的输入输出序列, 并基于此构建与软件实现逻辑相同的最小自动机, 下面对文献 [24] 中的定义和算法进行一定的简化, 表 1 是过程中常使用到的相关符号.

表 1 符号描述表

符号	a,b	u,v,w	$u\cdot v$	$U\cdot V$	s	$\mathbb{B}$	$[w]_{\approx}$	$\lfloor s\rfloor$
意义	字母	串	串的连接	集合的连接	状态	$\{0,1\}$	等价类	访问字符串

2.1 基础定义

定义 1. 字母表. 单个字母的有穷非空集合, 记为 Σ, 如 $\Sigma=\{a,b,\ldots,z\}$; 由字母构成的有穷序列, 称为串, 记为 w, 其构成的集合记为 Σ^*, 串的连接记为 $w_1\cdot w_2$, 串的长度记为 $|w|$. 特别地, ε 为空串, 长度为 0. 长度为 n 的串构成的集合为 $\Sigma^n=\{w\in\Sigma^*||w|=n\}$, 长度不大于 n 的串构成 $\Sigma^{[n]}=\{w\in\Sigma^*||w|\leqslant n\}$. 同时定义集合的连接, $\forall U,V\subset\Sigma^*, U\cdot V=\{u\cdot v|u\in U,v\in V\}$.

给定 $w,u\in\Sigma^*$, u 为 w 的前缀, 当且仅当 $\exists v\in\Sigma^*$, 使得 $w=u\cdot v$, 记为 $u\sqsubseteq_{\text{pref}}w$, 当 $|u|<|w|$, 称严格前缀, 记为 $u\sqsubsetneq_{\text{pref}}w$; 同理, 给定 $w,u\in\Sigma^*$, v 为 w 的后缀, 当且仅当 $\exists u\in\Sigma^*$, 使得 $w=u\cdot v$, 记为 $v\sqsubseteq_{\text{suff}}w$, 当 $|u|<|w|$, 称严格后缀, 记为 $v\sqsubsetneq_{\text{suff}}w$. 给定 $U\subset\Sigma^*$, 若 $\forall w\in U,\forall u\sqsubseteq_{\text{pref}}w$, 满足 $u\in U$, 则称 U 满足前缀闭合性.

定义 2. 字母表 Σ 上的确定性有限自动机使用四元组 $\mathcal{A}=\langle S,s_0,\delta,F\rangle$ 表示, 其中,

(1) S 是有限的、非空状态的集合;

(2) $s_0\in S$, 是初始状态;

(3) $F\subseteq S$, 是接受状态的集合;

(4) $\delta:S\times\Sigma\to S$, 是状态转移函数, $\delta(s,a)=s'$ 表明自动机在输入 a 时, 其状态由 s 转向 s'.

后文中的自动机均为确定性有限自动机, 并为区分不同自动机, 添加下角标区分, $\mathcal{A}=\langle S_{\mathcal{A}},s_{0,\mathcal{A}},\delta_{\mathcal{A}},F_{\mathcal{A}}\rangle$.

定义 3. 将定义 δ 扩展至 Σ^*, 定义 $\delta^*:S\times\Sigma^*\to S$,

$$\begin{cases}\delta^*(s,\varepsilon)=s, & \forall s\in S\\ \delta^*(s,a\cdot w)=\delta^*(\delta^*(s,a),w), & \forall s\in S,a\in\Sigma,w\in\Sigma^*\end{cases}.$$

其中, δ 是 δ^* 在第 2 个参数长度为 1 时的特殊形式, 由于不会造成二义性, 后文使用 δ 替换 δ^* 进行表述. 特别的, 可使用符号 $[w]$ 代表字符串 w 由初始状态达到的状态 $[w]=\delta(s_0,w),\forall w\in\Sigma^*$.

定义 4. 给定 $\mathcal{A}$ 是字母表 Σ 上的自动机, 定义映射 λ^s 如下:

$$\lambda^s=\Sigma^*\to\mathbb{B},\ \lambda^s(w)=\begin{cases}1, & \text{if } \delta(s_0,w)\in F_{\mathcal{A}}\\ 0, & \text{otherwise}\end{cases},\forall w\in\Sigma^*.$$

特别的, 当 $s=s_0$ 时, λ^{s_0} 记为 λ, 称为自动机的输出函数.

定义 5. Nerode 一致性. 给定映射 $\phi:\Sigma^*\to\mathbb{B}$, 可构建字母表上的二元关系 $\cong_\phi=\{(u,u')|\ \phi(u)=\phi(u')\}$, 若 $\forall u,u'\in\Sigma^*$, $u\cong_\varphi u'\Leftrightarrow(\forall v\in\Sigma^*:\varphi(u\cdot v)=\varphi(u'\cdot v))$, 称该二元关系满足 Nerode 一致性. 易知, 该关系是等价关系, 且具有性质: $\forall u,u',v\in\Sigma^*:u\cong_\varphi u'\Rightarrow(u\cdot v\cong_\varphi u'\cdot v)$. 特别的, 给定 $\mathcal{A}$ 的输出函数 λ, 其构建的 $\cong_\lambda=\{(u,u')|\ \lambda(u)=\lambda(u')\}$, 满足 $u\cong_\lambda u'\Leftrightarrow(\forall v\in\Sigma^*,\lambda(u\cdot v)=\lambda(u'\cdot v))$, 也即满足 Nerode 一致性.

由文献 [24] 依据该关系化简自动机得到的自动机 $\mathcal{M}=\langle S_{\mathcal{M}},s_{0,\mathcal{M}},\delta_{\mathcal{M}},F_{\mathcal{M}}\rangle$ 是最小自动机, 构建方法如下.

(1) $S_{\mathcal{M}}=\Sigma^*/\cong_\lambda$;

(2) $s_{0,\mathcal{M}}=[\varepsilon]_{\cong_\lambda}$;

(3) $\delta_{\mathcal{M}}([w]_{\cong_\lambda},a)=[w\cdot a]_{\cong_\lambda}$;

(4) $F_{\mathcal{M}}=\{[w]_{\cong_\lambda}|\lambda(w)=1\}$.

MAT 学习框架则是通过获取部分输入串的输出映射关系, 逼近上述等价关系, 其划分的等价类数量最小, 也就达成了最小自动机的目的. 后文中的自动机都是最小自动机.

定义 6. 等价自动机和等价状态. 给定两个自动机 $\mathcal{A}$ 和 $\mathcal{H}$, $s\in S_{\mathcal{H}}$, $s'\in S_{\mathcal{A}}$, $V\subset\Sigma^*$.

如果 $\forall v\in\Sigma^*$, $\lambda^s_{\mathcal{H}}(v)=\lambda^{s'}_{\mathcal{A}}(v)$, 称 s 与 s' 关于集合 V 等价, 记为 $s\equiv_V s'$.

特别, 取 $V=\Sigma^*$, $s\equiv_V s'$, 称 s 与 s' 等价, 记为 $s\equiv s'$; 若 $s=s_{0,\mathcal{H}}$, $s'=s_{0,\mathcal{A}}$, 即 $s_{0,\mathcal{H}}\equiv s_{0,\mathcal{A}}$, 则称 $\mathcal{A}$ 和 $\mathcal{H}$ 等价, 记为 $\mathcal{A}\equiv\mathcal{H}$.

给定映射 $f:\Sigma^*\to\{-1,0,1\}$, 记 $domf=\{u\in\Sigma^*|f(u)=0\vee f(u)=1\}$, 由映射 f 构成的集合记为 $\mathcal{F}$.

定义 7. 给定自动机 $\mathcal{A}$, $\mathcal{U}\subset\Sigma^*$, λ 是的输出函数, 映射 $\kappa:\Sigma^*\to\mathcal{F}$, 若满足 $\forall u\in\mathcal{U}$, $v\in domκ(u):\kappa(u)(v)=\lambda(u\cdot v)$, 称 κ 为 λ 的有效近似, $\mathcal{R}=\langle\mathcal{U},\kappa\rangle$ 为 λ 的抽象.

对于 λ 的抽象 $\mathcal{R}=\langle\mathcal{U},\kappa\rangle$, 字母表上的二元关系 $\sim_\kappa\subseteq\Sigma^*\times\Sigma^*$, 使其满足 $\forall u,u'\in\Sigma^*$, $u\sim_\kappa u'\Leftrightarrow(\forall v\in domκ(u)\cap domκ(u'):\kappa(u)(v)=\kappa(u')(v))$, 显然 $\sim_\kappa$ 是 Σ^* 上的等价关系; 如果满足:

(1) $\forall u\in\mathcal{U}$, $a\in\Sigma$, 存在 $u'\in\mathcal{U}$, 使得 $ua\sim_\kappa u'$, 称 $\mathcal{R}$ 具备封闭性;

(2) $\forall u,u'\in\mathcal{U}$, $a\in\Sigma$, 都有 $u\sim_\kappa u'\Rightarrow ua\sim_\kappa u'a$, 称 $\mathcal{R}$ 具备确定性.

定义 8. 给定 $\mathcal{R}=\langle\mathcal{U},\kappa\rangle$ 为 λ 的抽象, 若 $\forall u\in\mathcal{U}$, $v\in domκ(u)$, 满足 $v=a\cdot v'$, $a\in\Sigma$, 并且 $v'\in domκ(u\cdot a)$, 则称 $\mathcal{R}$ 满足语义后缀闭合性.

引理 1[24]. 若 $\mathcal{R}=\langle\mathcal{U},\kappa\rangle$ 满足语义后缀闭合性和 $\mathcal{U}$ 的前缀闭合性时, 则 $\mathcal{R}$ 具备封闭性和确定性.

若 $\mathcal{R}=\langle\mathcal{U},\kappa\rangle$ 满足封闭性和确定性, 可如下构建自动机 $\mathcal{H}=\mathrm{DFA}(\mathcal{R})$:

(1) $S=\{[u]\sim_\kappa|u\in\mathcal{U}\}$;

(2) $s_0=[\varepsilon]\sim_\kappa$;

(3) $\delta([u]\sim_\kappa,a)=[ua]\sim_\kappa,\forall u\in\mathcal{U},a\in\Sigma$;

(4) $F=\{[u]\sim_\kappa|u\in\mathcal{U},\kappa(u)(\varepsilon)=1\}$.

对于自动机 $\mathcal{H}=\mathrm{DFA}(\mathcal{R})$, 若 $v\notin domκ(u)$, $\lambda_{\mathcal{H}}^{[u]\sim_\kappa}(v)=\lambda_{\mathcal{H}}^{[u]\sim_\kappa}(a\cdot v')=\lambda_{\mathcal{H}}^{[ua]\sim_\kappa}(v')$, 可依据上述规则对 v 化简, 使其长度不断减小, 直至 $v'\in domκ(u')$, $u'\in[ua]\sim_\kappa\cap\mathcal{U}$. 特别的, 将 $\mathcal{R}=\langle\mathcal{U},\kappa\rangle$ 形成的等价类个数, 也即自动机状态数, 记为 $ind(\mathcal{R})$.

定义 9. 辨别串.

(1) 给定 $\mathcal{A}$ 和 $\mathcal{H}$, $\exists w\in\Sigma^*$, 使得 $\lambda_{\mathcal{A}}(w)\neq\lambda_{\mathcal{H}}(w)$, 则称 w 为 $\mathcal{A}$ 和 $\mathcal{H}$ 的辨别串, 后文也称为反例;

(2) 给定自动机 $\mathcal{A}$ 和 s, $s'\in S$, $\exists w\in\Sigma^*$, 使得 $\lambda^s(w)\neq\lambda^{s'}(w)$, 则称 w 为 s 和 s' 的辨别串.

给定 $s,s'\in S$, 其辨别串构成的集合记为 $Seps(s,s')=\{w\in\Sigma^*|\ \lambda^s(w)\neq\lambda^{s'}(w)\}$, 特别的, 取定一个元素, 构成仅具有单个元素的集合, 记为 $sep(s,s')$, 满足 $sep(s,s')\in Seps(s,s')$ 且 $|sep(s,s')|=1$.

2.2 等价查询算法

模型学习在初始化后进入学习阶段和等价查询阶段, 通过反复的学习和等价查询, 构建与 $\mathcal{A}$ 等价的自动机. 学习阶段是基于反例对 $\mathcal{U}$ 进行扩展, 并调整 λ 的有效近似 κ, 以改进抽象 $\mathcal{R}=\langle\mathcal{U},\kappa\rangle$, 并构建自动机 $\mathcal{H}=\mathrm{DFA}(\mathcal{R})$, 称之为假设; 等价查询阶段即通过等价查询算法判断 $\mathcal{H}$ 与 $\mathcal{A}$ 是否等价, 若等价则算法终止, 输出 $\mathcal{H}$; 否则返回反例, 再次进入学习阶段. 等价查询算法可分为两类, 一类是基于监控的, 长期监控系统运行的输入输出序列, 对模型假设进行修正, 称为终身学习; 另一类是基于一定策略构建测试用例, 在给定状态数的前提下, 保证假设与实际系统的等价性, 实际过程中多采用第 2 种更为主动的方法. 本节主要介绍等价查询算法中使用到的相关定义.

定义 10. 给定自动机 $\mathcal{A}$, $s\in S$, 称集合 $arrive(s)=\{w|w\in\Sigma^*,\delta(s_0,w)=s\}$ 为状态 s 的到达字符串集合, 记 $\lfloor s\rfloor$ 为 $arrive(s)$ 中任一最短的元素, 即 $\|\lfloor s\rfloor\|\leqslant|w|,w\in arrive(s)$, 称 $\lfloor s\rfloor$ 为状态 s 的访问字符串. 常记 $Q=\{\lfloor s\rfloor|\forall s\in S\}$, 为 $\mathcal{A}$ 的一个访问字符串的集合; $P=Q\cdot\Sigma$, 为 $\mathcal{A}$ 的一个状态转移字符串的集合.

特别的, 为便于后文表达, 针对假设 $\mathcal{H}$, 将 $\lfloor[u]\sim_\kappa\rfloor_{\mathcal{H}}$ 简记为 $\lfloor u\rfloor_{\mathcal{H}}$.

定义 11. 给定自动机 $\mathcal{A}$, 定义 $\cup_{s,s'\in S,s\neq s'}sep(s,s')\subseteq\Sigma^*$, 称为 $\mathcal{A}$ 的特征集.

引理 2[7]. 设自动机 $\mathcal{A}$ 和 $\mathcal{H}$ 的状态数分别为 m,n, $m\geqslant n$, W 是 $\mathcal{H}$ 的特征集, P 是 $\mathcal{H}$ 的一个状态转移字符串的集合, 记 $Z=\Sigma^{[m-n]}\cdot W$, 则 $\mathcal{H}$ 和 $\mathcal{A}$ 等价, 当且仅当 $s_{0,\mathcal{H}}\equiv_{P\cdot Z}s_{0,\mathcal{A}}$.

W-method 等价查询算法是基于引理 2 等价条件的判别, 通过构建字符串集合 $P\cdot Z$ 进行枚举查询, 判断输出是否相等, 以寻找反例.

定义 12. 给定自动机 $\mathcal{A}$, $W_i=\cup_{s\in S,s\neq s_i}sep(s,s_i)$, $s_i\in S$, 称为状态 s_i 的特征集合, 记 $\mathcal{W}=\cup_{s_i\in S}\{W_i\}$, 并记 $u\otimes=u\cdot W_j$, 其中 $u\in\Sigma^*,\delta(s_0,u)=s_j$.

引理 3[8]. 设自动机 $\mathcal{A}$ 和 $\mathcal{H}$ 的状态数分别为 m,n, $m \geqslant n$, W 是 $\mathcal{H}$ 的特征集, W_i 是 $\mathcal{H}$ 中状态 s_i 的特征集, $\mathcal{W}=\cup_{s_i \in S_{\mathcal{H}}}\{W_i\}$, P 是 $\mathcal{H}$ 个状态转移字符串的集合, Q 为 $\mathcal{H}$ 的一个访问字符串的集合, 记 $R=P-Q$, $R\cdot\Sigma^{[m-n]}\otimes\mathcal{W}=\cup_{u\in R\cdot\Sigma^{[m-n]}}u\otimes\mathcal{W}$, 则 $\mathcal{H}$ 和 $\mathcal{A}$ 等价, 当且仅当 $s_{0,\mathcal{H}}\equiv_{Q\cdot Z}s_{0,\mathcal{A}}$ 且 $s_{0,\mathcal{H}}\equiv_{R\cdot\Sigma^{[m-n]}\otimes\mathcal{W}}s_{0,\mathcal{A}}$.

Wp-method 等价查询算法是基于引理 3 等价条件的判别, 通过构建字符串集合 $Q\cdot Z$ 和 $R\cdot\Sigma^{[m-n]}\otimes\mathcal{W}$ 进行枚举查询, 判断输出是否相等, 以寻找反例. 根据反例可以在学习阶段进一步改进抽象 $\mathcal{R}=\langle\mathcal{U},\kappa\rangle$, 得到新的假设 $\mathcal{H}$.

定理 1[24]. 反例分离. 给定 $\mathcal{A}$ 和 $\mathcal{H}$, 反例 $w\in\Sigma^*$, 则 w 可以分解成:

$$u\cdot a\cdot v,\ u,v\in\Sigma^*,\ a\in\Sigma,\ \text{s.t. } \lambda_{\mathcal{A}}(\lfloor u\rfloor_{\mathcal{H}}\cdot v)\neq\lambda_{\mathcal{A}}(\lfloor u\rfloor_{\mathcal{H}}a\cdot v).$$

以图 2(a) 的自动机为例, 其中双线圆圈为接受状态, 主要分析前两次学习阶段和等价查询阶段的过程, 如图 2(b)、图 2(c) 所示, 第 1 个反例是 b, 构成了对应的假设, 此时对应的集合分别为: $W_0=W_1=\{b\}$, $Q=\{\varepsilon,b\}$, $P=\{\varepsilon,b,a,ba,bb\}$, 由于 $W_0=W_1$, 所以此时 W-method 和 Wp-method 构成的集合是相同的; 第 2 个反例是 ab, 反例分离可写为 $\lambda_{\mathcal{A}}(\lfloor\varepsilon a\rfloor_{\mathcal{H}}\cdot b)\neq\lambda_{\mathcal{A}}(\lfloor\varepsilon\rfloor_{\mathcal{H}}a\cdot b)$, 形成第 2 个假设, 此时有: $W_0=\{b\},W_1=\{\varepsilon\},W_2=\{\varepsilon,b\}$, $W=\{\varepsilon,b\},Q=\{\varepsilon,b,a\}$, $P=\{\varepsilon,b,a,ba,bb,aa,ab\}$, $R=\{ba,bb,aa,ab\}$, 两种方法所形成的等价测试集合存在差异, 当 $m=n$ 时, W-method 集合元素个数为 11, Wp-method 为 10, 串 abb 是 W-method 独有的. 随着学习的深入, 两种算法的差异将会更大.

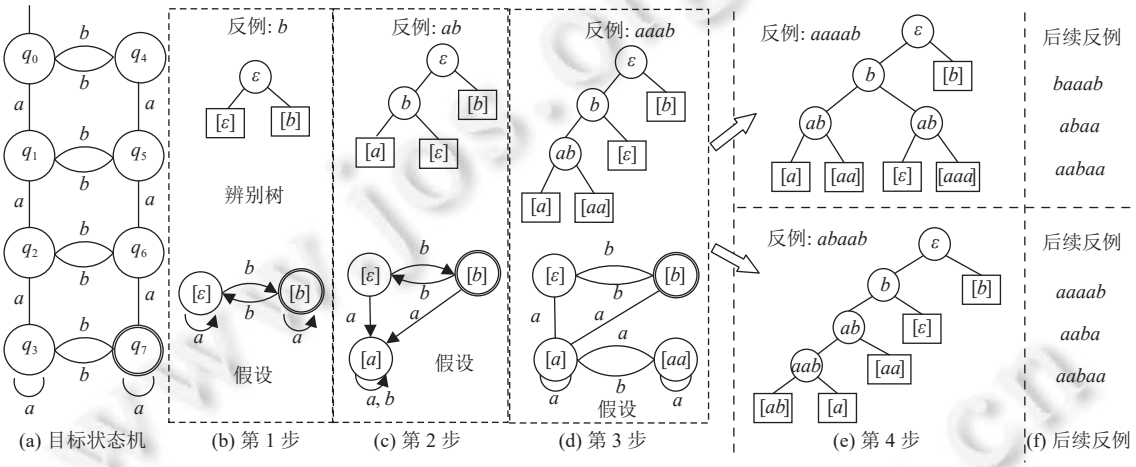

图 2 不同反例对学习过程的影响

2.3 数据结构与学习算法

当前实现自动机学习算法的数据结构主要为观测表和辨别树. 观测表包含 3 个组成部分: 一是有限的前缀闭合的字符串集合 $\mathcal{U}$, 二是有限的后缀闭合的字符串集合 $\mathcal{V}$, 三是映射 $\lambda:((\mathcal{U}\cup\mathcal{U}\cdot\Sigma)\cdot\mathcal{V})\to\{0,1\}$. 观测表可写成 $(\mathcal{U},\mathcal{V},\lambda)$, 映射 λ 是输出函数. 辨别树是一棵有向二叉树, 内部节点是辨别字符串, 叶节点是访问字符串 (代表状态). 常用的学习策略如下所示.

(1) 基于观测表

经典 L*算法: 将反例 w 的所有前缀都加入集合 $\mathcal{U}$.

Shahbaz 算法[15]: 反例分离为 $w=uv$, 其中 $u\in\mathcal{U}\cdot\Sigma$, 将 v 的所有后缀加入集合 $\mathcal{V}$.

Suffix1by1[41]: 将反例 w 的后缀按照长度递增的原则依次加入 $\mathcal{V}$, 直至观测表不再闭合.

(2) 基于辨别树

DT[16]: 依据后缀分离算法, 将 $\lfloor u\rfloor_{\mathcal{H}}a$ 加入集合 $\mathcal{U}$.

TTT 算法: 考虑到长反例的情况, 将 v 加入集合 $\mathcal{V}$, 若 v 不满足语义后缀闭合, 则将其所在的子树标记为临时子树, 对其进行进一步分析.

同时, 为便于后文的表达, 在此解释独特查询数和独特符号数的含义. 首先, 将一个串发送至 SUT 称为一次查

询, 其中串的长度记为符号数, 而在学习过程中不同的环节发送的串可能是相同的, 例如在学习阶段可能需要查询$a\cdot b$, 在等价查询阶段也需要查询$a\cdot b$, 为了避免在实际交互中发送重复的串, 程序实现时采用了缓存机制, 即对已经查询的串进行记录, 在下次查询时将先从缓存中进行读取, 若该串不存在, 才会发起实际的查询, 本文将不重复的串作为统计对象, 得到对应的独特查询数和独特符号数, 为便于表达, 后文中使用的查询数和符号数均特指独特查询数和独特符号数.

3 方 法

主动自动机学习可分为初始化和改进阶段, 其中初始化阶段是形成一个最初的假设, 改进阶段是依据等价查询产生的反例, 基于学习算法对假设进行更新, 其中反例的质量对于整个算法的影响较大. 如上文所述, Wp-method等价查询分为两个阶段: 第 1 阶段为集合覆盖测试$Q\cdot\Sigma^{[m-n]}\cdot W$, 第 2 阶段为状态转换覆盖测试$R\cdot\Sigma^{[m-n]}\otimes\mathcal{W}$, 而基于 MAT 框架的等价查询算法均是遇到第 1 个反例即终止, 依据该反例对假设进行改进, 因此测试用例构建的顺序是值得分析的.

3.1 反例分析

由于基于观测表的策略本质上是将等价查询的一部分用例转移至学习过程, 在学习过程中对一些反例进行了处理, 本质上是在学习过程中进行了一部分的启发式等价查询, 因此本节主要针对基于辨别树的算法进行分析. 基于辨别树的特点为学习过程中形成的$\mathcal{R}=\langle\mathcal{U},\kappa\rangle$满足$\mathcal{U}=Q$.

以图 2 中的自动机为例, 说明不同构造顺序对产生反例的影响, 图 2(d) 为经过反例*aaab*后形成的辨别树和假设, 图 2(e) 为通过不同方法构造的测试用例导致的反例, 上半部分为 Wp-method 方法形成的反例*aaaab*和辨别树, 下半部分为调整顺序后形成的反例*abaab*和辨别树, 可观察到不同的反例造成了完全不同的辨别树, 且会影响后续的学习过程, 如图 2(f) 中的后续反例, 最后形成的查询数分别为: 52、46, 所对应的符号数分别为: 224、190.

影响学习过程中查询数量和符号数量的重要集合为$\mathcal{U}$和W, Türker 等人[42]针对W研究, 提出基于广度优先搜索算法探索最小特征集合, 相对于原始实现有较好的提升效果. 本文重点探讨集合$\mathcal{U}$的影响, 也即辨别树中的叶节点, 认为反例形成的分解使得集合$\mathcal{U}$中所有元素长度总和越小越好, 给出以下定义.

定义 13. 给定不同学习过程中的$\mathcal{R}=\langle\mathcal{U},\kappa\rangle$和$\mathcal{R}'=\langle\mathcal{U}',\kappa'\rangle$, 有$ind(\mathcal{R})=ind(\mathcal{R}')$, 若$len(\mathcal{U})<len(\mathcal{U}')$, 也即集合$\mathcal{U}$中所有元素长度总和小于$\mathcal{U}'$的所有元素长度总和, 则称$\mathcal{R}$优于$\mathcal{R}'$.

该定义, 可称例子中的第 2 个反例优于第 1 个反例.

在此定义的基础上, 由于每个假设确定后Q,R,W都是确定的, 而$\Sigma^{[m-n]}$是造成用例数剧增的核心原因, 而$m-n$数值的不确定性使得等价查询的过程中应遵循由小至大的原则, 据此有以下原则.

原则 1. 反例构造依据$\Sigma^{[m-n]}$的由小至大进行构造. 基于该原则, 可得以下定理.

定理 2. 基于$Q\cdot\Sigma^{[m-n]}\cdot W$, 按照$\Sigma^{[m-n]}$长度递增的顺序测试, 若$w\in Q\cdot\Sigma^{l}\cdot W$是反例, $l\in\{0,1,\ldots,m-n\}$, 则其后缀分解固定为$w=u\cdot a\cdot v$, 其中$u\in Q$, $ua\notin Q$.

证明: 由于按照$\Sigma^{[m-n]}$长度递增的顺序测试, 若$w\in Q\cdot\Sigma^{l}\cdot W$是反例, 则有假设$\mathcal{A}$和$\mathcal{H}$关于$Q\cdot\Sigma^{[l-1]}\cdot W$等价的, 不妨设$w=u\cdot a\cdot v$, 其中$u\in Q, w\in W$.

通过反证法证明上述定理, 反例分解可以分为两种情形, 第 1 种情形是$w=\hat{u}\cdot\hat{a}\cdot\hat{v}$, 其中$\hat{u}\subsetneq_{\text{pref}}u$, $\lambda_{\mathcal{A}}(\lfloor\hat{u}\hat{a}\rfloor_{\mathcal{H}}\cdot\hat{v})\neq\lambda_{\mathcal{A}}(\lfloor\hat{u}\rfloor_{\mathcal{H}}\hat{a}\cdot\hat{v})$, 而由于$\hat{u},\hat{u}\hat{a}\in Q$, $\lambda_{\mathcal{A}}(\lfloor\hat{u}\hat{a}\rfloor_{\mathcal{H}}\cdot\hat{v})=\lambda_{\mathcal{A}}(\lfloor\hat{u}\rfloor_{\mathcal{H}}\hat{a}\cdot\hat{v})=\lambda_{\mathcal{A}}(\hat{u}\hat{a}\cdot\hat{v})$, 因此矛盾, 第 1 种情形不成立.

第 2 种情形是$w=\hat{u}\cdot\hat{a}\cdot\hat{v}$, 其中$u\subsetneq_{\text{pref}}\hat{u}$, 则$\hat{v}\subsetneq_{\text{suff}}a^{l-1}\cdot w$也即$\hat{v}\in a^{[l-2]}\cdot w$, $\lambda_{\mathcal{H}}(\lfloor\hat{u}\hat{a}\rfloor_{\mathcal{H}}\cdot\hat{v})=\lambda_{\mathcal{H}}(\lfloor\hat{u}\rfloor_{\mathcal{H}}\hat{a}\cdot\hat{v})$是恒成立的; 由于$\mathcal{A}$和$\mathcal{H}$关于$Q\cdot\Sigma^{[l-1]}\cdot W$等价, 因此有$\lambda_{\mathcal{H}}(\lfloor\hat{u}\hat{a}\rfloor_{\mathcal{H}}\cdot\hat{v})=\lambda_{\mathcal{A}}(\lfloor\hat{u}\hat{a}\rfloor_{\mathcal{H}}\cdot\hat{v})$, $\lambda_{\mathcal{H}}(\lfloor\hat{u}\rfloor_{\mathcal{H}}\cdot\hat{a}\hat{v})=\lambda_{\mathcal{A}}(\lfloor\hat{u}\rfloor_{\mathcal{H}}\cdot\hat{a}\hat{v})$, 也即$\lambda_{\mathcal{A}}(\lfloor\hat{u}\hat{a}\rfloor_{\mathcal{H}}\cdot\hat{v})=\lambda_{\mathcal{A}}(\lfloor\hat{u}\rfloor_{\mathcal{H}}\hat{a}\cdot\hat{v})$, 与反例分解矛盾.

综上所述, 定理 2 得证.

定理 3. 依据集合$Q\cdot\Sigma^{[m-n]}\cdot W$和$R\cdot\Sigma^{[m-n]}\otimes\mathcal{W}$, 按照$\Sigma^{[m-n]}$长度递增的顺序测试, 若$w\in R\cdot\Sigma^{l}\otimes\mathcal{W}$是反例, $l\in\{0,1,\ldots,m-n\}$, 则其后缀分解固定为$w=\hat{u}\cdot\hat{a}\cdot\hat{v}$, 其中$\hat{u}\in Q$, $\hat{u}\cdot\hat{a}\in R$.

证明: 由于按照$\Sigma^{[m-n]}$长度递增的顺序测试, 若$w \in R\cdot\Sigma^{l}\otimes\mathcal{W}$是反例, 则有假设$\mathcal{H}$和$\mathcal{A}$关于$Q\cdot\Sigma^{[l-1]}\cdot W$等价, 不妨设$w=ua\cdot a^{l}\cdot w$, 其中$ua\in R, w\in W$.

同样通过反证法证明上述定理, 反例分解可以分为两种情形, 第 1 种情形是$w=\hat{u}\cdot\hat{a}\cdot\hat{v}$, 其中$\hat{u}\subsetneq_{\text{pref}}u$, $\lambda_{\mathcal{A}}(\lfloor\hat{u}\hat{a}\rfloor_{\mathcal{H}}\cdot\hat{v})\neq\lambda_{\mathcal{A}}(\lfloor\hat{u}\rfloor_{\mathcal{H}}\hat{a}\cdot\hat{v})$, 而由于$\hat{u},\hat{u}\hat{a}\in Q$, $\lambda_{\mathcal{A}}(\lfloor\hat{u}\hat{a}\rfloor_{\mathcal{H}}\cdot\hat{v})=\lambda_{\mathcal{A}}(\hat{u}\hat{a}\cdot\hat{v})$, $\lambda_{\mathcal{A}}(\lfloor\hat{u}\rfloor_{\mathcal{H}}\hat{a}\cdot\hat{v})=\lambda_{\mathcal{A}}(\hat{u}\hat{a}\cdot\hat{v})$, 因此矛盾, 第 1 种情形不成立.

第 2 种情形是$w=\hat{u}\cdot\hat{a}\cdot\hat{v}$, 其中$u\subsetneq_{\text{pref}}\hat{u}$, $\hat{v}\subsetneq_{\text{suff}}a^{l-1}\cdot w$, 也即$\hat{v}\in a^{[l-1]}\cdot w$, $\lambda_{\mathcal{H}}(\lfloor\hat{u}\hat{a}\rfloor_{\mathcal{H}}\cdot\hat{v})=\lambda_{\mathcal{H}}(\lfloor\hat{u}\rfloor_{\mathcal{H}}\hat{a}\cdot\hat{v})$是恒成立的; 由于$\mathcal{A}$和$\mathcal{H}$关于$Q\cdot\Sigma^{[l-1]}\cdot W$等价, 因此有$\lambda_{\mathcal{H}}(\lfloor\hat{u}\hat{a}\rfloor_{\mathcal{H}}\cdot\hat{v})=\lambda_{\mathcal{A}}(\lfloor\hat{u}\hat{a}\rfloor_{\mathcal{H}}\cdot\hat{v})$, $\lambda_{\mathcal{H}}(\lfloor\hat{u}\rfloor_{\mathcal{H}}\cdot\hat{a}\hat{v})=\lambda_{\mathcal{A}}(\lfloor\hat{u}\rfloor_{\mathcal{H}}\cdot\hat{a}\hat{v})$, 也即$\lambda_{\mathcal{A}}(\lfloor\hat{u}\hat{a}\rfloor_{\mathcal{H}}\cdot\hat{v})=\lambda_{\mathcal{A}}(\lfloor\hat{u}\rfloor_{\mathcal{H}}\hat{a}\cdot\hat{v})$, 与反例分解矛盾.

综上所述, 定理 3 得证.

在上述定理的基础上, 制定下述原则.

原则 2. 在保证反例分解确定为$w=\hat{u}\cdot\hat{a}\cdot\hat{v}$, $\hat{u}\in Q$的前提下, 在学习过程中, $\hat{v}$的长度越长越好, 因为后缀的不闭合可能会使得该反例能形成多个新的状态, 从而减少等价查询的次数.

3.2 策略制定与分析

依据上述原则探讨 Wp-method 的构造顺序, Wp-method 算法的构造顺序为$Q\cdot\Sigma^{[m-n]}\cdot W+R\cdot\Sigma^{[m-n]}\otimes\mathcal{W}$, 其中$\Sigma^{[m-n]}$部分是依据长度由小至大构造. 而实际实现过程中存在多种顺序:

(1) $R\cdot\Sigma^{[m-n]}\otimes\mathcal{W}+Q\cdot\Sigma^{[m-n]}\cdot W$;

(2) $Q\cdot W+R\otimes\mathcal{W}+Q\cdot\Sigma^{1}\cdot W+R\cdot\Sigma^{1}\otimes\mathcal{W}+\ldots+Q\cdot\Sigma^{[m-n]}\cdot W+R\cdot\Sigma^{[m-n]}\otimes\mathcal{W}$;

(3) $R\otimes\mathcal{W}+Q\cdot W+R\cdot\Sigma^{1}\otimes\mathcal{W}+Q\cdot\Sigma^{1}\cdot W+\ldots+R\cdot\Sigma^{[m-n]}\otimes\mathcal{W}+Q\cdot\Sigma^{[m-n]}\cdot W$.

以上 3 种方法会使得反例出现的顺序不同, 不同的反例使得学习过程中构造的假设完全不同. 为便于理解, 后文中将学习阶段的算法称为策略, 3 种构造顺序分别称为方法 (1)、方法 (2)、方法 (3).

如图 2 中的两个反例*aaaab*和*abaab*, 其中$aaaab\in Q\cdot\Sigma^{1}\cdot W$, $abaab\in R\cdot\Sigma^{1}\otimes\mathcal{W}$, 而在分解时$aaaab=aa\cdot a\cdot ab$, 形成的新状态为[*aaa*], $abaab=a\cdot b\cdot aab$, 形成的新状态为[*ab*], 依据定义 13 我们认为$R\cdot\Sigma^{1}\otimes\mathcal{W}$应该先于$Q\cdot\Sigma^{1}\cdot W$测试.

Wp-method 算法与方法 (1) 将Q与R集合分成两个阶段, 第 2 阶段如 Wp-method 算法的$R\cdot\Sigma^{[m-n]}\otimes\mathcal{W}$在实际查询过程中的前期阶段无法体现, 只能在后期$m-n$较小时有所体现, 因此会较大地丢失第 2 阶段的反例. 方法 (2) 与方法 (3) 则是更加符合原则 1, 依据$\Sigma^{[m-n]}$的长度由小至大, 尽量同时利用$Q\cdot W$和$R\otimes\mathcal{W}$中测试用例可能存在的反例. 依据定理 2 和定理 3 可知, 其反例分解都是确定的, 因此方法 (3) 相对于方法 (2) 的优势在于在$m-n$的长度相同时, $R\otimes\mathcal{W}$所形成的新后缀更长, 依据原则 2, 则更有优势, 本文选择方法 (3) 为最终改进算法.

同时, 与 random Wp-method 相比, 依据原则 2, 随机化算法可能需要更少的等价查询轮次, 但其存在的问题在于由于其未能遵循原则 1, 因此其反例存在不确定性, 进而使得长反例的优势只能在 TTT 算法上有更好的效果.

4 实验分析

基于上述针对反例的分析和构造顺序的设计, 本节通过实验验证改进算法的有效性和原则的合理性.

实验环境: Ubuntu 16.04+Learnli b0.17.0+automatalib 0.11.0, 其中 LearnLib 0.17.0+automatalib 0.11.0 均是最新的开发版本.

实验对象: 随机生成的 DFA (自动机)、Mealy 机以及文献 [23] 中的 DFA 对象 peterson2、sched5. 为了便于阐述, 生成的 DFA 使用状态数-字母表大小进行表述, 如 100-20 表示目标 DFA 为状态数为 100, 字母表大小为 20 的 DFA, 分别生成了 40-20、60-20、80-20、100-20、200-20、300-20、100-15、100-25 共 8 类 DFA 各 100 个. Mealy 机与 DFA 的区别在于输出域不局限于{0,1}, 生成的 Mealy 机使用状态数-输入字母表-输出字母表进行表述, 本文生成了 100-5-5、100-20-5、200-5-5、200-10-10、200-20-5、300-20-5 共 6 类 Mealy 机各 100 个. 特别地, 除探讨接收状态比例差别的实验 2, 其他实验中自动机的接受状态比例都为 50%.

对比算法: 学习策略为前文中提到的 4 种策略: L*策略, suffix1by1 策略, DT 策略, TTT 策略, 等价算法分别采用了 Wp-method 算法、Increment 算法[21]、Random Wp-method 算法、改进的 3 种方法 (实验效果图中的 final 方法对应方法 (3)). 其中对应参数分别设置为: Wp-method 及其改进方法的 lookahead 为 2, 目标状态数设定为实际 DFA 的状态数; Increment 算法的参数最大深度设置为 10; Random Wp-method 算法设置的最小和最大长度分别为 0 和 10.

实验计划: 共进行 4 组实验, 以查询数和符号数为评价指标, 分别以随机生成的 DFA、Mealy、peterson2、sched5 为对象对比改进方法与已有算法 (实验 1、实验 2、实验 3), 以随机生成的 DFA 为对象对比 3 种改进方法 (实验 4).

4.1 实验结果

(1) 实验 1: 与已有算法的对比, 以随机生成的 DFA 为对象

图 3 是对象为 100 个随机生成的 100-20 的 DFA, 算法为 4 种学习策略与 4 种等价查询算法两两组合得到的查询数、符号数的对比图, 图例中标识为 final 的算法即为方法 (3), 4 种算法的图例填充分别为横线, 叉线, 点, 斜杠, 图 3(a)、图 3(b) 分别是 100 个实例在不同算法组合下查询数和符号数的平均值, 对于查询数, 可观察到在 TTT 策略下 random Wp 算法的查询数是最低的, 明显优于其他 3 种算法. 从符号数的角度, 可看出方法 (3) 在基于辨别树结构的学习策略 DT 和 TTT 上效果是最优的, 相比于 Wp-method 有明显的提升; 在 TTT 策略下方法 (3) 和 random Wp-method、Increment 算法的平均值相近, 而由于随机算法产生反例的长度具有不确定性, 在具有长反例处理能力的 TTT 策略上效果较好, 而在 DT 策略上效果较方法 (3) 略差. (注: 由于 random Wp-method 策略使得部分实例的符号数较大, 为了更好地显示效果, 去除了图 3(b) 中 L*算法下的异常值)

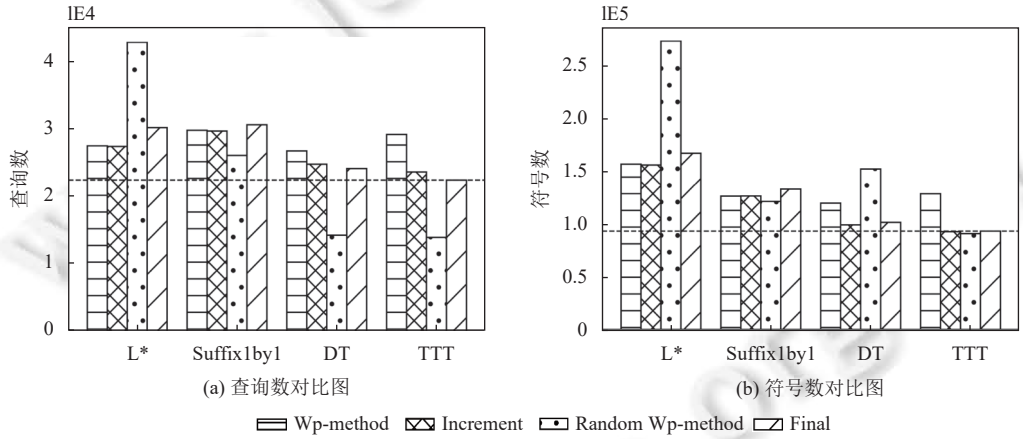

图 3 100 个 DFA (100-20) 实例学习的查询数和符号数对比图

以 100-20 规模的 DFA 为例, 学习策略为 TTT 策略, 对比记录学习过程中的访问字符串, 计算学习过程中访问字符串的长度总和, 将在 Wp-method 算法和方法 (3) 下的访问字符串长度总和分别作为横纵坐标, 绘制出具有 100 个点的散点图, 如图 4 所示, 其中黑色虚线是两种方法所得到的访问字符串长度相等的分界线, 可观察到绝大部分点都在分界线的下方, 也即方法 (3) 分离出的访问字符串总长度显著降低. 为更加直观地观察过程中访问字符串的差别, 表 2 为某个 DFA 学习过程中记录的访问字符串, 发现方法 (3) 得到的访问字符串基本上是依据长度递增的, 如第 7–12 次, 方法 (3) 新增的访问字符串长度均为 1, 而 Wp-method 算法所形成的访问字符串有一部分长度为 2, 而最后形成的访问字符串基本一致, 但过程中的长度差异将会使得查询数和符号数有所区别.

图 5 展示了 8 类 DFA 在几种算法组合下的平均值比较图, 从查询数的角度观察, random Wp-method 算法是在 DT、TTT 策略上是有较好效果的, 从符号数的角度分析, 方法 (3) 是具有与 random Wp-method 和 increment 算法相近的效果的, 究其原因, 是由于 random Wp-method 算法通过在设定的最小值和最大值中随机构造并选取, 因

此会使得随机算法会出现较长的反例, 而只有 TTT 策略具有较好的长反例处理效果, 因此其在 TTT 策略下查询数降低, 而符号数具有相近的效果; 反而在其他学习策略上效果不好.

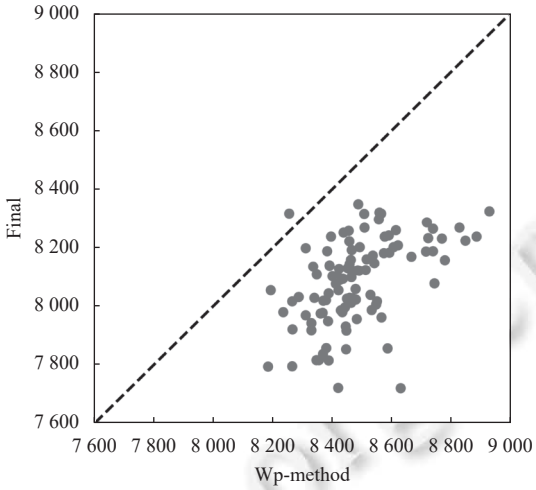

图 4 100 个 DFA (100-20) 实例学习的访问字符串长度对比图

表 2 第 n 次假设某个 DFA 实例学习过程中访问字符串的比较表

n	Wp-method	Final
7	5, 7, ε, 8, 0 0, 0, 1, 2	ε, 0, 1, 2, 3, 5, 7
12	5, 0 13, 7, ε, 9, 17, 5 0, 0 0, 0, 1, 2	ε, 0, 1, 2, 3, 5, 7, 8, 9, 12, 14
18	3, 5, 0 13, 6, 7, ε, 8, 9, 14, 16, 17, 5 0, 6 7, 0 0, 0 6, 0, 1, 2	ε, 0, 1, 2, 3, 4, 5, 7, 8, 9, 12, 14, 15, 16, 17, 0 0, 0 1, 0 2

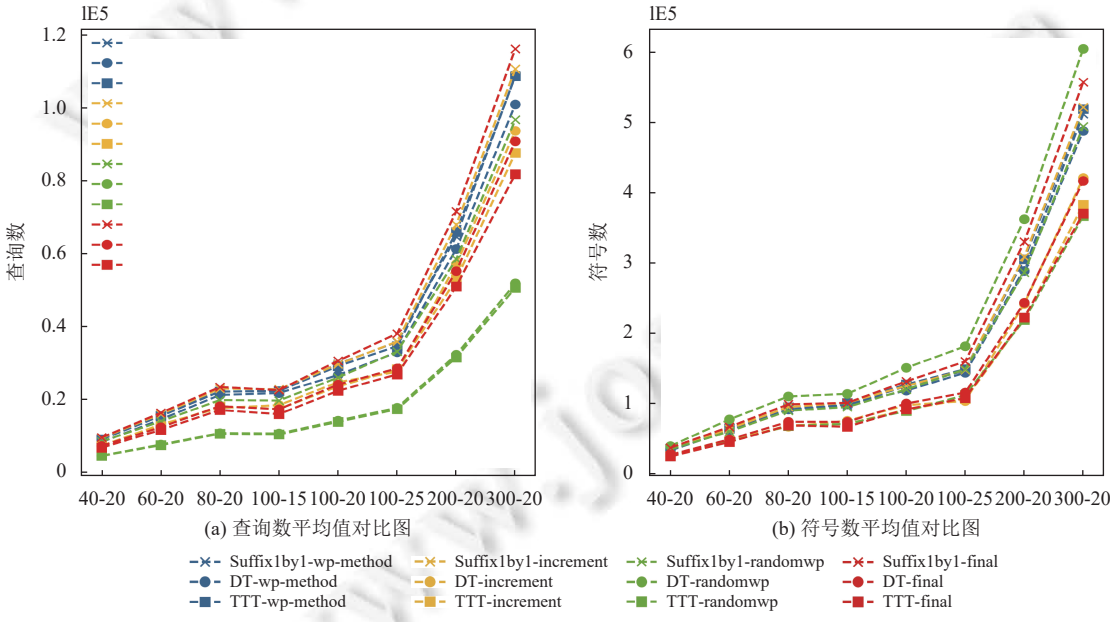

图 5 8 类 DFA 实例学习的查询数和符号数平均值比较图

表 3 是 8 类 DFA 对象基于 TTT 策略的查询数和符号数降低的平均值, 在 TTT 策略上, 方法 (3) 相较于 Wp-method 算法减少了 24% 的查询次数, 29% 的符号数, 在 DT 算法上, 分别降低了 15% 和 22%.

表 3 基于 TTT 算法的 8 类 DFA 实例学习的查询数和符号数

查询数/符号数	指标	40-20	60-20	80-20	100-15	100-20	100-25	200-20	300-20	Avg
查询数	Wp	9089.8	15505.3	22209.4	22364.0	29187.3	29187.3	65879.2	108634.6	—
	Final	6815.2	11686.4	17134.2	16015.5	22404.4	22404.4	51020.4	81774.3	—
	降低率	0.25	0.25	0.23	0.28	0.23	0.23	0.23	0.25	0.24
符号数	Wp	36795.6	65546.2	95958.1	101846.9	128177.7	128177.7	305442.0	518620.6	—
	Final	25978.5	46391.6	69973.2	68143.4	92786	92786	223149.2	370123.6	—
	降低率	0.29	0.29	0.27	0.33	0.28	0.28	0.27	0.29	0.29

(2) 实验 2: 与已有算法的对比, 以实例 peterson2、sched5 为对象

表 4 和表 5 分别为针对实例 peterson2、sched5, 不同算法组合的表现, 方法 (3) 在查询数和符号数上都明显优于其他算法, 原因主要在于 peterson2、sched5 的接受状态较多, 从而使得学习过程中反例较长. 为了研究接受状态比例对于效果的影响, 以 100-20 规模分别随机生成 100 个接受状态比例为 0.1、0.2、0.3、0.4、0.5 的 DFA, 实验结果如图 6 所示, 展示了方法 (3) 相较于 Wp-method 算法降低的查询数和符号数的比例, 其中负数为增加, 黄色虚线为比例为 0 的分界线, 黄色虚线以上说明改进方法更好, 以下说明 Wp-method 算法更好, 因此可看出随着接受状态和不接受状态数量逐渐均等, 方法 (3) 的效果有所下降, 比例超过 0.3 之后, 基于观测表的两种策略在改进方法下的效果比 Wp-method 算法差, 但在基于辨别树的学习算法上仍有一定的降低效果, 而上文的实例 peterson2、sched5 是非常特殊的 DFA, 其不接受状态只有 1 个, 且多数状态转移都指向不接受状态, 使得在采用等价查询算法找寻反例的代价大大提升, 此时遵循原则 1 的策略将会降低等价查询的代价, 因此方法 (2) 在该对象时具有和方法 (3) 同样的改进效果.

表 4 针对实例 peterson2 不同算法的表现

Peterson2	LStar		Suffix1by1		DT		TTT	
	queries	symbols	queries	symbols	queries	symbols	queries	symbols
Wp-method	168121	1628654	201023	1696415	198855	1672414	177973	1791083
Increment	192293	2141103	192937	1637858	170542	1442686	275389	3064911
random Wp-method	77949	865880	86400	980485	145174	1817030	287100	3696686
方法(2)	33664	371697	27885	302651	27915	268949	32114	329126
方法(3)	24316	284117	25726	311381	33657	336041	33655	335999

表 5 针对实例 sched5 不同算法的表现

Sched5	LStar		Suffix1by1		DT		TTT	
	queries	symbols	queries	symbols	queries	symbols	queries	symbols
Wp-method	284983	4294783	288351	4344016	154532	1971230	156478	2005147
Increment	373885	6036871	341181	5415321	343666	5425509	1044918	15401900
random Wp-method	120446	1972440	195160	3605779	982561	20388282	1218838	25864168
方法(2)	121926	2088719	130299	2229760	85470	1381297	86947	1409608
方法(3)	93123	1543678	92235	1530931	55177	877153	59595	952259

(3) 实验 3: 与已有算法的对比, 以随机生成的 Mealy 机为对象

探讨改进算法对于 Mealy 机的扩展性, 本文以 Mealy 机为对象进行了实验验证, 图 7 的实验对象为 100 个随机生成的 200-20-5 的 Mealy 机, 可观察到对于 Mealy 机, 方法 (3) 也具有一定的效果, 在符号数上有所降低. 本文分别对 6 种规模的 Mealy 机: 100-5-5、100-20-5、200-5-5、200-10-10、200-20-5、300-20-5 进行了测试, 具体结果如表 6 所示, 符号数平均降低了约 7%. 而 Increment 和 random Wp-method 算法在 Mealy 机上效果较差, 可能是由于 Mealy 机具有更多样化的输出, 通过一些启发式或随机性的方法去构造反例的方法具有不稳定性.

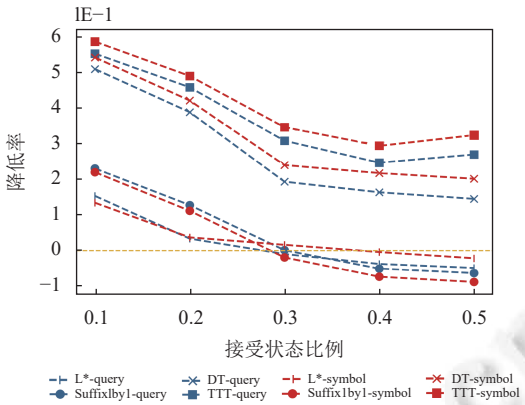

图 6 针对不同比例接受状态的自动机算法降低的符号数比例

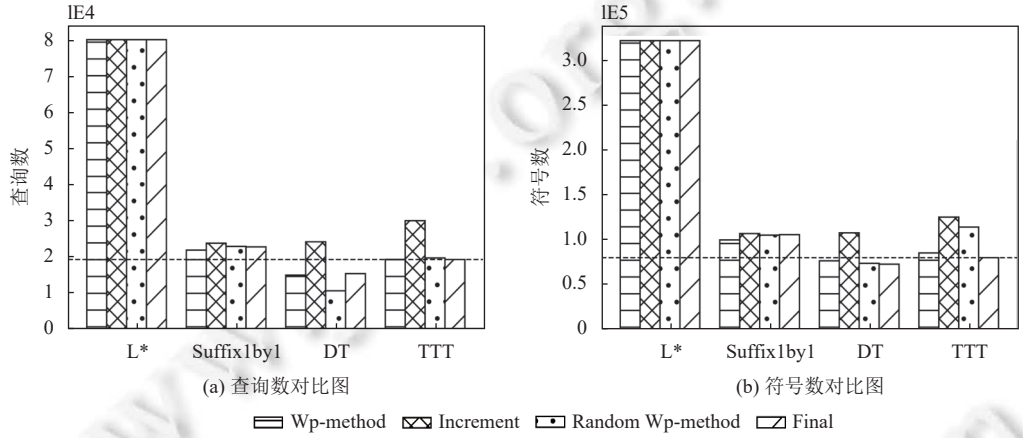

图 7 100 个 Mealy 实例 (200-20-5) 不同算法组合的效果比较图

表 6 基于 TTT 算法的 8 类 Mealy 实例学习的查询数和符号数

查询数/符号数	指标	100-5-5	100-20-5	200-5-5	200-10-10	200-20-5	300-20-5	Avg
查询数	Wp	1 918.1	8 748.2	4 192.3	7 423.6	19 143.9	30 106.7	—
	Final	1 899.5	8 716.17	4 147.4	7 393.1	19 106.1	30 040.9	—
	降低率	0.01	0.0	0.01	0.0	0.0	0.0	0.00
符号数	Wp	10 798.6	36 373.9	26 056.7	35 637.1	84 929.8	138 335.1	—
	Final	9 714.3	33 475.3	23 829.7	34 206.1	79 673.7	130 999.8	—
	降低率	0.10	0.08	0.09	0.04	0.06	0.05	0.07

(4) 实验 4: 3 种改进方法的对比

图 8 为基于改进思路实现的 3 种方法和 Wp-method 算法的效果对比, 其中方法 (1) 的测试顺序为 $R\cdot\Sigma^{[m-n]}\otimes\mathcal{W}+Q\cdot\Sigma^{[m-n]}\cdot W$, 方法 (2) 为 $Q\cdot W+R\otimes\mathcal{W}+Q\cdot\Sigma^{1}\cdot W+R\cdot\Sigma^{1}\otimes\mathcal{W}+\ldots+Q\cdot\Sigma^{[m-n]}\cdot W+R\cdot\Sigma^{[m-n]}\otimes\mathcal{W}$, 图 8 为 4 种学习策略与 4 种等价查询方法在 100 个 DFA 实例下的查询数和符号数平均值, 可看出在查询数目和符号数目上方法 (3) 在 DT 和 TTT 策略上都是最优的, 且从全局来看, TTT+方法 (3) 是最优选择, 进一步验证了第 3 节中的分析.

4.2 讨 论

综合上述针对不同对象的实验结果, 可观察到如下现象.

(1) 针对 DFA, 在 DT 和 TTT 策略上, 方法 (3) 的效果与 random Wp、Increment 算法相近, 相较于 random Wp 更为稳定, 优于 Wp-method 和方法 (1)、方法 (2);

(2) 针对特定对象 peterson2、sched5, 方法 (2)、方法 (3) 的效果优于 Wp-method、random Wp、Increment 算法;
(3) 针对不同接受状态比例的 DFA, 随着接受状态和不接受状态数量逐渐均等, 方法 (3) 的效果逐渐下降;
(4) 针对对象 Mealy, 在 DT 和 TTT 策略上, 方法 (3) 具有最好的效果.

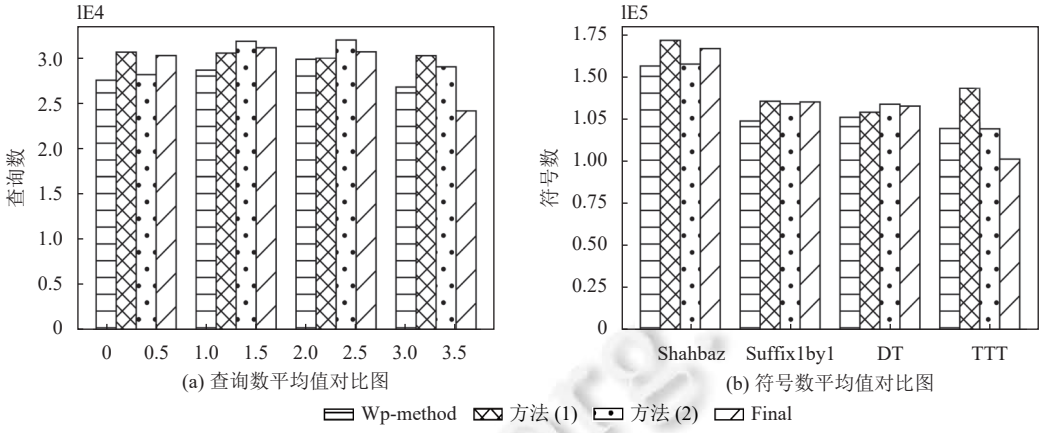

图 8 100 个 DFA 实例 (100-20) 不同算法组合的查询数和符号数平均值比较图

进一步分析算法的本质, Random W-method、Random Wp-method 算法、故障检测算法[43]本质上都是通过随机算法在 $P\cdot\Sigma^{(m-n)}\cdot W$ 或 $Q\cdot\Sigma^{[m-n]}\cdot W+R\cdot\Sigma^{[m-n]}\otimes BW$ 集合中选取测试用例进行测试, 与本文提出的算法调整构造顺序本质上是相同的, 区别在于策略的选取, 实验结果表明了改进的 Wp-method 算法能近似达到 random Wp-method 算法的效果, 在部分对象上甚至优于随机算法, 且具有更好的稳定性.

如 Isberner 在文献 [24] 中讨论, 基于辨别树的策略通常只能基于反例拆分一个等价类, 使得等价查询的数量急剧增加, 当使用成员查询对等价查询算法进行模拟时, 会使得查询次数大大增加; 而基于观测表的策略本质上是将等价查询的一部分用例转移至学习过程, 在学习过程中对一些反例进行了处理, 本质上是在学习过程中进行了一部分的启发式等价查询, 从而使得等价查询的次数减少了. 本文基于多种等价查询算法, 以独特的查询和符号数为评价指标, 以多种规格的 DFA 和 Mealy 为实验对象, 观察到基于辨别树的策略普遍优于基于观察表的策略, 说明了针对等价查询算法进行启发式的构造效果优于观察表隐式的启发式策略. 同时通过四组实验的对比, 也验证了所提出的 2 个原则和改进算法的合理性.

5 总结与展望

本文主要研究等价查询算法中反例对学习过程的影响, 定义了学习过程中假设的比较方法, 并基于基础理论和预先实验提出两个构造原则, 在原则上对 Wp-method 算法在实现过程中的测试顺序进行分析, 提出一种新的测试顺序. 而后以随机生成的 DFA 和先前研究论文中的实例作为实验对象, 基于 LearnLib 实现了改进方法, 如在 TTT 算法上, 改进算法相较于 Wp-method 算法减少了 24% 的独特查询次数, 29% 的独特符号数; 在 DT 算法上, 分别降低了 15% 和 22%; 相较于 Random Wp-method 和 Increment 算法, 对于 DFA 具有相近的效果, 对于 Mealy 具有更优的效果, 验证了原则和算法的有效性.

从实验结果观察, 随机算法虽然有着不确定性, 但在 TTT 算法上有较好的效果, 后续可以探讨随机算法的进一步优化, 随机算法的构造过程是 3 个组成部分的随机选择, 探讨学习过程中构成的假设在哪些状态、哪些后缀处更容易产生反例是值得分析的; 同时, 本文针对 Mealy 机只是在实验上验证了方法的扩展性, 后续仍可从理论上分析其合理性.

References:

[1] Ali S, Sun HL, Zhao YW. Model learning: A survey of foundations, tools and applications. Frontiers of Computer Science, 2021, 15(5): 155210. [doi: 10.1007/s11704-019-9212-z]

[2] Chalupar G, Peherstorfer S, Poll E, de Ruiter J, de Ruiter JEJ. Automated reverse engineering using LEGO. In: Proc. of the 8th USENIX

Workship on Offensive Technologies. San Diego: USENIX, 2014. 1–10.
[3] de Ruiter J, Poll E. Protocol state fuzzing of TLS implementations. In: Proc. of the 24th USENIX Conf. on Security Symp. Washington, DC: USENIX Association, 2015. 193–206.
[4] Aslam K, Cleophas L, Schiffelers R, van den Brand M. Interface protocol inference to aid understanding legacy software components. Software and Systems Modeling, 2020, 19(6): 1519–1540. [doi: 10.1007/s10270-020-00809-2]
[5] Gold EM. Complexity of automaton identification from given data. Information & Control, 1978, 37(3): 302–320. [doi: 10.1016/S0019-9958(78)90562-4]
[6] Angluin D. Learning regular sets from queries and counterexamples. Information & Computation, 1987, 75(2): 87–106. [doi: 10.1016/0890-5401(87)90052-6]
[7] Chow TS. Testing software design modeled by finite-state machines. IEEE Trans. on Software Engineering, 1978, SE-4(3): 178–187. [doi: 10.1109/TSE.1978.231496]
[8] Fujiwara S, Bochmann GV, Khendek F, Amalou M, Ghedamsi A. Test selection based on finite state models. IEEE Trans. on Software Engineering, 1991, 17(6): 591–603. [doi: 10.1109/32.87284]
[9] Daniel LA, Poll E, de Ruiter J. Inferring OpenVPN state machines using protocol state fuzzing. In: Proc. of the 2018 IEEE European Symp. on Security and Privacy Workshops (EuroS&PW). London: IEEE, 2018. 11–19. [doi: 10.1109/EuroSPW.2018.00009]
[10] Guo JX, Gu CX, Chen X, Wei FS. Model learning and model checking of IPSec implementations for internet of things. IEEE Access, 2019, 7: 171322–171332. [doi: 10.1109/ACCESS.2019.2956062]
[11] Fiterău-Broştean P, Jonsson B, Merget R, de Ruiter J, Sagonas K, Somorovsky J. Analysis of DTLS implementations using protocol state fuzzing. In: Proc. of the 29th USENIX Security Symp. Boston: USENIX Association, 2020. 2523–2540.
[12] Moon SJ, Helt J, Yuan YF, Bieri Y, Banerjee S, Sekar V, Wu WF, Yannakakis M, Zhang Y. Alembic: Automated model inference for stateful network functions. In: Proc. of the 16th USENIX Conf. on Networked Systems Design and Implementation. Boston: USENIX Association, 2019. 699–718.
[13] Aarts FD. Tomte: Bridging the gap between active learning and real-world systems [Ph.D. Thesis]. Nijmegen: Radboud Universiteit, 2014.
[14] Aarts F, Jonsson B, Uijen J, Vaandrager F. Generating models of infinite-state communication protocols using regular inference with abstraction. Formal Methods in System Design, 2015, 46(1): 1–41. [doi: 10.1007/s10703-014-0216-x]
[15] Shahbaz M, Groz R. Inferring mealy machines. In: Cavalcanti A, Dams DR, eds. Proc. of the 2009 Int'l Symp. on Formal Methods. Eindhoven: Springer, 2009. 207–222. [doi: 10.1007/978-3-642-05089-3_14]
[16] Kearns MJ, Vazirani UV. An Introduction to Computational Learning Theory. Cambridge: MIT Press, 1994.
[17] Sabnani KK, Dahbura AT. A protocol testing generation procedure. Computer Networks and ISDN Systems, 1988, 15(5): 285–297.
[18] Cui L, Zhang JB, Gong B, Wu LY. Approach for reduction test Suite of Wp method based on set covering problem. Journal of Beijing University of Technology, 2016, 42(9): 1332–1337 (in Chinese with English abstract). [doi: 10.11936/bjutxb2015120046]
[19] Yang N, Aslam K, Schiffelers R, Lensink L, Hendriks D, Cleophas L, Serebrenik A. Improving model inference in industry by combining active and passive learning. In: Proc. of the 26th IEEE Int'l Conf. on Software Analysis, Evolution and Reengineering (SANER). Hangzhou: IEEE, 2019. 253–263. [doi: 10.1109/SANER.2019.8668007]
[20] Rivest RL, Schapire RE. Inference of finite automata using homing sequences. Information & Computation, 1993, 103(2): 299–347. [doi: 10.1006/inco.1993.1021]
[21] Howar FM. Active learning of interface programs [Ph.D. Thesis]. Dortmund: TU Dortmund University, 2012. [doi: 10.17877/DE290R-4817]
[22] Isberner M, Steffen B. An abstract framework for counterexample analysis in active automata learning. In: Proc. of the 12th Int'l Conf. on Grammatical Inference. Kyoto: PMLR, 2014. 79–93.
[23] Isberner M, Howar F, Steffen B. The TTT algorithm: A redundancy-free approach to active automata learning. In: Bonakdarpour B, Smolka SA, eds. Runtime Verification. RV 2014. Lecture Notes in Computer Science, vol. 8734. Toronto: Springer, 2014. 307–322. [doi: 10.1007/978-3-319-11164-3_26]
[24] Isberner M. Foundations of active automata learning: An algorithmic perspective [Ph.D. Thesis]. Dortmund: TU Dortmund University, 2015. [doi: 10.17877/DE290R-16359]
[25] Cassel S, Howar F, Jonsson B, Merten M, Steffen B. A succinct canonical register automaton model. Journal of Logical and Algebraic Methods in Programming, 2015, 84(1): 54–66. [doi: 10.1016/j.jlamp.2014.07.004]
[26] Howar F, Steffen B, Jonsson B, Cassel S. Inferring canonical register automata. In: Kuncak V, Rybalchenko A, eds. Proc. of the Int'l Workshop on Verification, Model Checking, and Abstract Interpretation. Philadelphia: Springer, 2012. 251–266. [doi: 10.1007/978-3-642-27940-9_17]
[27] Argyros G, Stais I, Jana S, Keromytis AD, Kiayias A. SFADiff: Automated evasion attacks and fingerprinting using black-box differential automata learning. In: Proc. of the 2016 ACM SIGSAC Conf. on Computer and Communications Security. Vienna: ACM,

2016. 1690–1701. [doi: 10.1145/2976749.2978383]
[28] Raffelt H, Steffen B. LearnLib: A library for automata learning and experimentation. In: Baresi L, Heckel R, eds. Proc. of the 2006 Int'l Conf. on Fundamental Approaches to Software Engineering. Vienna: Springer, 2006. 377–380. [doi: 10.1007/11693017_28]
[29] Cassel S, Howar F, Jonsson B, Steffen B. Software Engineering and Formal Methods. Learning extended finite state machines. In: Giannakopoulou D, Salaün G, eds. Proc. of the Int'l Conf. on Software Engineering and Formal Methods. Grenoble: Springer, 2014. 250–264. [doi: 10.1007/978-3-319-10431-7_18]
[30] Aarts F, Heidarian F, Kuppens H, Olsen P, Vaandrager F. Automata learning through counterexample guided abstraction refinement. In: Giannakopoulou D, Méry D, eds. Proc. of the Int'l Symp. on Formal Methods. Paris: Springer, 2012. 10–27. [doi: 10.1007/978-3-642-32759-9_4]
[31] Cho CY, Basić D, Shin ECR, Song D. Inference and analysis of formal models of botnet command and control protocols. In: Proc. of the 17th ACM Conf. on Computer and Communications Security. Chicago: Association for Computing Machinery, 2010. 426–439. [doi: 10.1145/1866307.1866355]
[32] Rasool A, Alpár G, de Ruiter J. State machine inference of QUIC. arXiv:1903.04384, 2019.
[33] Fiterău-Broştean P, Janssen R, Vaandrager F. Combining model learning and model checking to analyze TCP implementations. In: Chaudhuri S, Farzan A, eds. Proc. of the 2016 Int'l Conf. on Computer Aided Verification. Toronto: Springer, 2016. 454–471. [doi: 10.1007/978-3-319-41540-6_25]
[34] Verleg P. Inferring SSH state machines using protocol state fuzzing [MS. Thesis]. Nijmegen: Radboud University, 2016.
[35] Shen YZ, Gu CX, Chen X, Zhang XL, Lu ZY. Vulnerability analysis of OpenVPN system based on model learning. Ruan Jian Xue Bao/Journal of Software, 2019, 30(12): 3750–3764 (in Chinese with English abstract) http://www.jos.org.cn/1000-9825/5612.htm. [doi: 10.13328/j.cnki.jos.005612]
[36] Shen YZ. Research on key technology of security protocols vulnerability analysis based on model learning [MS. Thesis]. Zhengzhou: Information Engineering University, 2018 (in Chinese with English abstract).
[37] Wang C, Wu LF, Hong Z, Zheng CH, Zhuang HL. Domain-specific algorithm of protocol state machine active inference. Computer Science, 2015, 42(12): 233–239 (in Chinese with English abstract).
[38] Fiterău-Broştean P, Lenaerts T, Poll E, de Ruiter J, Vaandrager F, Verleg P. Model learning and model checking of SSH implementations. In: Proc. of the 24th ACM SIGSOFT Int'l SPIN Symp. on Model Checking of Software. Santa Barbara: Association for Computing Machinery, 2017. 142–151. [doi: 10.1145/3092282.3092289]
[39] Howar F, Jonsson B, Vaandrager F. Combining black-box and white-box techniques for learning register automata. In: Steffen B, Woeginger G, eds. Computing and Software Science. Lecture Notes in Computer Science, vol. 10000. Springer, 2019. 563–588. [doi: 10.1007/978-3-319-91908-9_26]
[40] Smetsers R, Moerman J, Janssen M, Verwer S. Complementing model learning with mutation-based fuzzing. arXiv:1611.02429, 2016.
[41] Irfan MN, Oriat C, Groz R. Angluin style finite state machine inference with non-optimal counterexamples. In: Proc. of the 1st Int'l Workshop on Model Inference in Testing. Trento: Association for Computing Machinery, 2010. 11–19. [doi: 10.1145/1868044.1868046]
[42] Türker UC, Hierons RM, Jourdan GV. Minimizing characterizing sets. Science of Computer Programming, 2021, 208: 102645. [doi: 10.1016/j.scico.2021.102645]
[43] Aichernig BK, Tappler M. Efficient active automata learning via mutation testing. Journal of Automated Reasoning, 2019, 63(4): 1103–1134. [doi: 10.1007/s10817-018-9486-0]

附中文参考文献:

[18] 崔玲, 张建标, 公备, 吴丽影. 基于集合覆盖的Wp方法测试集约简方法. 北京工业大学学报, 2016, 42(9): 1332–1337. [doi: 10.11936/bjutxb2015120046]
[35] 申莹珠, 顾纯祥, 陈熹, 张协力, 卢政宇. 基于模型学习的OpenVPN系统脆弱性分析. 软件学报, 2019, 30(12): 3750–3764. http://www.jos.org.cn/1000-9825/5612.htm [doi: 10.13328/j.cnki.jos.005612]
[36] 申莹珠. 基于模型学习的安全协议脆弱性分析关键技术研究 [硕士学位论文]. 郑州: 战略支援部队信息工程大学, 2018.
[37] 王辰, 吴礼发, 洪征, 郑成辉, 庄洪林. 一种基于域知识的协议状态机主动推断算法. 计算机科学, 2015, 42(12): 233–239.

潘雁(1995－), 男, 博士生, 主要研究领域为软件逆向, 协议脆弱性分析.

祝跃飞(1962－), 男, 博士, 教授, 博士生导师, 主要研究领域为网络安全, 密码学.