计算机科学 COMPUTER SCIENCE http://www.jsjkx.com

DOI: 10.11896/jsjkx.xx

并发系统指定动作上的强等价检查及其最短反例生成

胥松杭[1] 王以松[1] 周欣[1,2] 冯仁艳[3] 张元睿[4]

1 公共大数据国家重点实验室，贵州大学计算机科学与技术学院，贵阳，550025
2 贵阳学院计算机科学学院 贵阳 550005
3 贵州财经大学信息学院 贵阳 550025
4 南京航空航天大学计算机科学与技术学院 南京 211106
（gs.skxu23@gzu.edu.cn）

摘 要 基于行为互摸拟的等价性检查是系统形式化验证和模型检测中的一个关键环节，忽略特定动作集 V 的强互摸拟（简称强 V-互摸拟）是检查系统规范（如 Hennessy-Milner Logic 公式）在特定动作集上等价的重要特征。本文提出一种计算强 V-互模拟的算法 StrongVB 和生成两个状态不强 V-互摸拟的最短反例算法 SCSVB，证明了算法的正确性并分析了算法的多项式时间复杂性；在 CADP 的 VLTS 数据集上充分的实验结果表明这两个算法都是有效的。由于强 V 互摸拟是强互摸拟的一般化，故这两个算法分别推广了传统的强互摸拟计算方法和强互摸拟的反例生成方法，这对计算在特定动作集上基于最弱条件修复模型提供了算法思路。

关键词： 等价性检查；强 V-互模拟；反例；忽略；泛化

中图分类号 TP301

Strong Equivalence Checking on Specified Actions and its Shortest Counterexample Generation for Concurrent Systems

Xu Songhang[1]，Wang Yisong[1]，Zhou Xin[1,2]， Feng Renyan[3] and Zhang Yuanrui[4]

1 State Key Laboratory of Public Big Data, College of Computer Science and Technology, Guizhou University, Guiyang, 550025, China
2 School of Computer Science, Guiyang University，Guiyang 550005, China
3 School of Information, Guizhou University of Finance and Economics，Guiyang 550025, China
4 School of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics，Nanjing 211106, China

Abstract Equivalence checking based on behavioral bisimulation is a crucial aspect of formal verification and model checking of systems. Strong bisimulation ignoring a specific set of actions V (strong V-bisimulation for short) is an important characteristic for checking the equivalence of system specifications (such as Hennessy-Milner Logic formulas) over a specific set of actions. This paper proposes the StrongVB algorithm for computing strong V-bisimulation, and the SCSVB algorithm for generating the shortest counterexample when two states are not strongly V-bisimilar. Moreover, we prove the correctness of the algorithms and analyze the time complexity. Extensive experimental results on the VLTS dataset from CADP demonstrate the effectiveness of the previous algorithms. Since strong V-bisimulation is a generalization of strong bisimulation, these algorithms extend the traditional methods for computing strong bisimulation and generating its counterexamples. Finally, this paper provides an algorithmic approach for calculating the repair model based on the weakest condition on a specific action set.

Keywords Equivalence checking，strong V-bisimulation，counterexamples，ignoring， generalization

到稿日期： 返修日期：
基金项目：国家自然科学基金项目资助（62376066，61976065）；贵州省科技计划项目（黔科合支撑[2022]一般-259）
This work was supported by the National Natural Science Foundation of China (62376066，61976065) and Guizhou Science and Technology Program ([2022]259).
通信作者：王以松（yswang@gzu.edu.cn）

1. 引言

随着软硬件规模与并发性提升，软件验证愈发困难。软硬件系统在持续演进中常常因为修补漏洞、添加功能并适配新环境，易引入冗余代码与隐藏缺陷，形成维护负担。更关键的是，系统状态数量随复杂度提升呈指数级增长，从而导致验证愈发困难（状态空间爆炸），这使得在有限资源下验证安全性、活性与无死锁等性质变得艰难。判定系统在行为语义上的等价性是形式化验证的关键环节[1,2]，并在模型检测领域占据重要地位[3]。通过比较不同的行为等价关系（如迹等价与强互模拟[4]等），等价性检查能够确认两个系统在行为层面的一致性，从而保障实现的正确性。该方法还能有效揭示系统实现是否与原始规范模型保持一致。

传统的等价性检查方法（如强互模拟、分支互模拟、弱互模拟等）通常采用显式状态范式：先自初始状态枚举全部可达状态与转移构造 LTS，再在整图上执行等价计算与判定，因此天然依赖对全局状态空间的遍历与维护[3]。其中，强互模拟作为基础、严格的行为等价性检查方法之一[4]，要求两个状态能够逐步以同名动作相互模拟，且每一步都转移到仍保持互模拟关系的后继状态[3]。这一判定通常需要在完整状态空间上进行全局计算，时间与空间开销显著。在该框架内，Blom 提出了基于强互模拟的规约算法，可在保持强互模拟等价的前提下规约 LTS，从而压缩状态与转移规模并降低后续验证成本[6]。

为降低与结论无关的开销，我们基于强互模拟提出并研究强 V-互模拟：通过引入“遗忘”的动作集 V，忽略掉不重要或不关心的动作（V 中的动作），仅在关键动作上进行严格匹配[5]。由此，检查过程无需对全部状态与转移进行显式遍历与维护，而是聚焦于与 V 无关的局部片段，显著减少无关转移与中间结构的处理量，从而大幅缩短等价性检查时间并提升检测效率，同时保持针对关键动作的语义判定精度。

本文基于 Blom 的互模拟规约框架，提出了基于强 V-互模拟的等价性检查算法 StrongVB，即一种基于强 V-互模拟的规约状态空间技术算法。该算法通过引入遗忘动作集 V 消除了大量对强互模拟检测结果无关的信息，有效缓解显式状态模型检测中的状态空间爆炸，并显著降低后续验证的时间与空间开销。

在上述 StrongVB 规约与检查框架之上，我们进一步关注反例这一验证闭环中的关键产物：反例不仅定位模型与规范的不一致，还给出可复现的错误路径，直接服务于缺陷定位与修复。为此，本文提出强 V-互模拟最短反例生成算法 SCSVB：在“遗忘”动作集 V 后，如果两个状态是不强 V-互模拟的，该算法可以在规约后的模型上寻找最短的反例路径。直观地说，SCSVB 只关注与结论相关的行为，少走弯路、快速定位差异点，从而更高效地给出可复现、可用于修复的最短反例。

本文同时分析了算法的正确性与复杂性，并在 CADP 的 VLTS 数据集上进行了广泛测试，实验结果表明，忽略掉部分动作后，该算法在计算强 V-互模拟上更高效，也能有效计算强 V-互模拟的最短反例。

第 2 章主要介绍了本文涉及到的并发系统的相关概念及定义；第 3 章给出算法 StrongVB 和算法 SCSVB 的设计与实现；第 4 章给出实验结果以及分析； 最后对本文工作进行总结并展望未来工作。

2. 预备知识

本节介绍标记转换系统（labelled transition system, LTS）、派生、强 V-互模拟（strong V-bisimulation）、划分（partition）、签名（signature）[4,5,9]、反例和最短反例的基本思想和概念。在通信并发系统（calculus of communicating systems，CCS）中，一个状态能够执行一个动作，并转换到其它状态。本文使用符号$\mathcal{F}$来表示该转换下可见转换动作名称的集合，其对应的对偶名称集合记作$\overline{\mathcal{F}} = \{\overline{f} \mid f \in \mathcal{F}\}$。所有可见动作名称的集合为$\mathcal{L} = \mathcal{F} \cup \overline{\mathcal{F}}$，$Act = \mathcal{L} \cup \{\tau\}$，其中$\tau$表示内部动作。$Act^* = \bigcup_{i\geq 0} Act^i$的元素称为动作序列(action sequence)。

定义 1.一个标记转换系统 LTS 是一个三元组(S, T, s_0)，其中：

1) S是非空状态的集合；
2) $T \subseteq S \times Act \times S$是状态转换的集合；$(s, \alpha, t) \in T$简记为$s \xrightarrow{\alpha} t$，表示系统从状态s可执行动作$\alpha$到达状态t；
3) s_0是初始状态；

定义 2.给定任意状态$P,Q \in \mathrm{S}$。如果$P \xrightarrow{\alpha} Q$，则称(α,Q)为P的一个直接派生(derivative)，其中α为P的一个可执行动作，Q为P的α-派生。我们用$P \xrightarrow{\alpha}$表示P可以执行动作α，用$P \not\xrightarrow{\alpha}$表示$P$无法执行动作α。我们将$P$的所有可执行动作的集合记作$ALL(P)$，定义如下：

$\alpha \in ALL(P)$当且仅当存在状态P'，使得$P \xrightarrow{\alpha} P'$。

令$A \subseteq Act$，$P \twoheadrightarrow^{A}$表示P可以执行集合A中所有动作，即$A \subseteq ALL(P)$。如果$P \xrightarrow{a_1} \cdots \xrightarrow{a_n} P'$，称$(a_1 \cdots a_n, P')$是$P$的一个直接序列派生，$a_1 \cdots a_n$为$P$的一个可执行动作序列，$Q$为$P$的$a_1 \cdots a_n$-派生。如果$P$有一个可执行动作序列$s = a_1 a_2 \cdots a_n, a_i \in Act, 1 \le i \le n$，则称$P \xrightarrow{s} Q$为存在一个从$P$到$Q$的转换，即$P \xrightarrow{a_1} \xrightarrow{a_2} \cdots \xrightarrow{a_n} Q$，在此，$Q$表示$P$的一个$s$-派生。当$n = 0$时，我们用$\epsilon$表示空动作序列，则$P \xrightarrow{\epsilon} P$，即$P$也是它本身的一个派生。

定义 3.设 V⊆ Act，$A = (S, T, s_0)$ 是一个标记转换系统，一个二元关系$\mathrm{R} \subseteq \mathrm{S} \times \mathrm{S}$是强 V-互模拟关系，当且仅当下面的条件成立：如果$(\mathrm{P}, \mathrm{Q}) \in \mathrm{R}$，那么对于所有$\alpha \in \mathrm{Act} - \mathrm{V}$：

1) 若$P \xrightarrow{\alpha} P'$，则存在某个状态$Q'$使得$Q \xrightarrow{\alpha} Q'$且$(P', Q') \in R$；
2) 若$Q \xrightarrow{\alpha} Q'$，则存在某个状态$P'$使得$P \xrightarrow{\alpha} P'$且$(P', Q') \in R$；

对于某个强 V-互模拟R，如果$(P, Q) \in R$，则称P和Q在A下是强 V-等价或强 V-互模拟的，记为$P \sim_{V,A} Q$。显然，当 V 是空集时，强 V-互摸拟退化为标准强互摸拟；此时记$P \sim_{A} Q$为$P \sim_{\emptyset,A} Q$。两个不同的转换系统(S_1, T_1, s_1)和(S_2, T_2, s_2)(S_1与S_2不相交)是强 V-互模拟的当且仅当它们的初始状态s_1和s_2在组合 LTS($\mathrm{S}_1 \cup \mathrm{S}_2, \mathrm{T}_1 \cup T_2, s_1$)中是强 V-互模拟的，其中组合 LTS 的初始状态为任意状态，即只关心两个 LTS 的初始状态在组合 LTS 下是否是 V-互模拟的。为方便起，我们记$ALL_V(P) = \{\alpha \in Act - V \mid \exists P' : P \xrightarrow{\alpha} P'\}$。

定义 4.令υ=$\{\mathrm{B}_1, \ldots, \mathrm{B}_\mathrm{k}\}$为有限状态集合S上的划分，当且仅当

$\cup_{B \in \upsilon} B = S$,且 $\mathrm{B_i} \cap \mathrm{B_j} (1 \le i \ne j \le k) = \emptyset$。

划分中的元素称为块(block)，若S的划分υ_2中的任意块B_2，在S的划分υ_1中都存在某个块B_1使得$B_2 \subseteq \mathrm{B}_1$，则称$\upsilon_2$是$\upsilon_1$的细化（refinement）。

定义 5.给定一个 LTS$(\mathrm{S}, \mathrm{T}, \mathrm{s}_0)$，υ是$S$上的一个划分。设$s \in S$且$B \in \upsilon$，记$s$在$\upsilon$上的签名为：

$$\sigma_\upsilon(s) = \{(\alpha, B) \mid s \xrightarrow{\alpha} s' \text{ 且 } s' \in B\}$$

如果划分υ中的每个块中的状态都具有相同的签名，则称该划分是稳定（stable）的，此时υ的块中每个状态之间是强 V-互模拟，且该划分被称为关系最粗划分(relational coarsest partition)[10]。典型的划分细化计算算法由 Kanellakis-Smolka[11]提出，给定一个初始划分，通过对转换关系的计算，将具有相同转换关系的状态置于同一块中。

定义 6.给定一个 LTS$(\mathrm{S}, \mathrm{T}, \mathrm{s}_0)$，$V \subseteq Act, P, Q \in S$，记$Pfin^+(Act - V) = \cup_{k \ge 1} [Act - V]^k$,其中$[V]^m = \{F \subseteq V \mid |F| = m\}$。我们称$(s, F) \in (Act - V)^* \times Pfin^+(Act - V)$是$P$和$Q$强 V-互模拟反例，当且仅当如下条件之一成立：

1) 存在满足$P \xrightarrow{s} P'$的P'且$F \subseteq ALLV(P')$，对任意满足$Q \xrightarrow{s} Q'$的Q'，都存在$\alpha \in F$使得$Q' \not\xrightarrow{\alpha}$，或者
2) 存在满足$Q \xrightarrow{s} Q'$的Q'且$F \subseteq ALLV(Q')$，对任意满足$P \xrightarrow{s} P'$的P'，都存在$\alpha \in F$使得$P' \not\xrightarrow{\alpha}$。

给定一个 LTS A = (S, T, s_0)，设$V \subseteq Act$，记$CountE_{V,A}(P, Q)$为状态P, Q在 A 中强 V-互模拟的反例集。

定义 7.设(s, F)是P和Q的强 V-互模拟反例，(s, F)的长度，记作$l(s, F)$，定义为$|s| + 1$，其中$|s|$为动作序列s中动作的个数。若对于P,Q的任何强 V-互模拟反例(s', F')，都有$l(s, F) \le l(s', F')$，则称(s, F)为P,Q的最短强 V-互模拟反例。

例 1. 给定一个 LTS $A = (S, T, s_0)$，其中 $S = \{O, P_1, P_2, Q_1, Q_2, Q_3\}, T = \{(P_1, a, P_2), (P_2, b, O), (P_2, c, O), (Q_1, a, Q_2), (Q_1, a, Q_3), (Q_2, b, O), (Q_3, c, O)\}, s_0 = P_1$。若$V = \emptyset$，则$(s, F) = (a, \{b, c\})$是$P_1$和$Q_1$强 V-互模拟反例。这是因为$P_1 \xrightarrow{\alpha} P_2$, $ALLV(P_2)$=$\{b, c\}$,而Q_1执行动作a后可以到达Q_2或Q_3，$Q_2 \not\xrightarrow{c}$，$Q_3 \not\xrightarrow{b}$，则我们有F=$\{b, c\} \subseteq ALLV(P_2)$。其长度为$l(s, F) = 2$。在这种情况下，$CountE_{\emptyset,A}(P, Q)$=$\{(a, \{b, c\})\}$,最短反例为$(a, \{b, c\})$。在 V 取其他值（包含$\{b\}$或$\{c\}$或$\{b, c\}$），均不生成反例，$P_1$和$Q_1$强 V-互模拟的。

3. 算法设计与实现

本节围绕三部分展开：首先介绍强 V−互模拟算法，给

出 V 处理与基于签名的判定流程；随后给出反例算法，用于生成按关键动作计数的最短反例以解释“不等价”；最后讨论算法正确性证明。

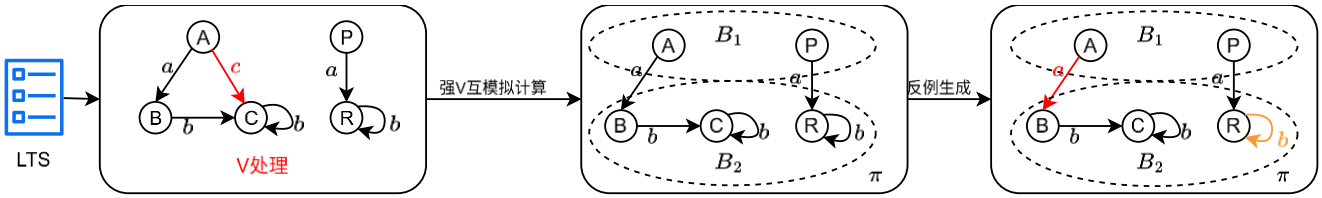

图 1 强 V-互模拟算法-反例示意图

Fig.1 Strong V-bisimulation algorithm-counterexample schematic diagram

如图 1 所示，该图给出强 V-互模拟反例的整体示意。S={A,B,C,P,R},T={(A,a,B),(A,c,C),(B,b,C),(C,b,C),(P,a,R),(R,b,R)}，s_0=A。设 V={c}，首先进行 V处理，将标号为 c 的转换从 T中删除（左图红色箭头）；随后在剩余转换上计算强 V-互模拟划分，要求同一块内的任意两个状态在每个不属于 V的动作上都能相互匹配其迁移，且到达的后继仍落在同一块（交换两状态同样成立），据此将满足条件者置于同一块（中图）。在该例中，A 可以执行动作a至 B，而 R仅能执行动作b至自身且不存在可与a匹配的转换，故 A与 R被划入不同块，不满足强 V-互模拟（右图）。

3.1 强 V-互模拟算法

3.1.1 V 处理

我们只关心“关键动作”，把不重要或不关注的动作收集为集合 V。所谓 V 处理，就是把 LTS 中所有标签属于 V 的转换整体“遗忘”（从转换集中删除，不再参与后续计算）；若 $V=\emptyset$，则不做任何改动。这样得到的模型仅保留与结论相关的行为，使后续等价性检查更聚焦、更高效。

形式化地，给定标记转换系统A=LTS(S,T,s_0)和动作集合V，其中 V$\subseteq Act$，记$A\setminus V$为 LTS(S,T',s_0)，其中$T'=T-\{(s,a,t)|(s,a,t)\in T$ 且 $a\in V\}$。V 处理即将 LTS A 中出现在 V中动作的状态转换删除，最终变为$A\setminus V$。V 处理的正确性由第 3.3 节的引理 1，引理 2 给出。

3.1.2 划分计算

划分计算首先将所有状态放入“不稳定集合”U。每一轮只处理U中的状态：为每个状态计算当前划分下的签名，并把签名相同的状态归类到同一块中，签名不同的则分到不同块。若某状态在本轮被重新分块，则需要将所有能到达它的前驱状态标记为不稳定状态，并放入下一轮待计算的动作集合中，以待重新计算划分，反复迭代，直到没有状态发生分裂为止，此时划分稳定，从而得到最粗的稳定划分。具体而言，给定A=LTS(S,T,s_0)，对任意 $s,s'\in S$，

$$\sigma[s]=\sigma[s']\Leftrightarrow mark[s]=mark[s'],$$

其中 $mark[s]$为状态 s的块号；若两个状态 σ不同，则必须将这两个状态分到不同块，即 $mark[s]\neq mark[s']$。在具体实现中，我们用哈希表 ρ将相同签名聚类到同一块，保证块内签名一致。每次迭代仅关注不稳定集合 U中的状态；一旦某状态在本轮由旧块号 $pmark[x]$变为新块号 $mark[x]$，则所有能经某动作 α到达该状态的前驱 y（即 $y\xrightarrow{\alpha}x$）被标记为不稳定并加入 $nextU$。当本轮 U处理完毕，将 $nextU$ 赋给 U并清空 $nextU$，重复上述过程，直至 U为空。下面给出具体算法，我们约定←为赋值操作。

算法 1 强 V-互模拟计算 StrongVB($A\setminus V$)

输入：$A\setminus V$;

输出：最终划分下的块数组$mark[x]$。

1. U ← S;
2. while U ≠ ∅
3. nextU ← ∅, ρ ← ∅;
4. for all x ∈ U do
5. σ[x] ← ∅;
6. for $(labelId,y)\in T[x]$ *do*
7. σ[x] ← σ[x] ∪ {$(labelId$, mark[y])};
8. end for
9. end for
10. remark ← {i | $\sum_{x\in U}$(mark[x] = i) = countB[i] };
11. for all x ∈ U do

12. pmark ← mark[x], $\sigma_x \leftarrow \sigma[x]$;
13. if $\sigma_x \in \rho$ then
14. mark[x] ← ρ;
15. elseif mark[x] ∈ remark then
16. remark ← remark − {mark[x]};
17. else
18. mark[x] ← blockN, blockN + +;
19. end if
20. $\rho \leftarrow \rho \cup \{(\sigma_x, \text{mark}[x])\}$;
21. if mark[x] ≠ pmark then
22. nextU ← nextU ∪ $\{y \mid y \in T^{-1}[x]\}$;
23. countB[pmark] − −, countB[mark[x]] + +;
24. end if
25. end for
26. $U \leftarrow nextU$;
27. end while
28. return mark;

其中$T[x]$分别表示状态x所有的转换信息，如$(labelId, y)$表示其转换动作标签$labelId$和目标状态y。S、T和T^{-1}分别表示标记转换系统$A \setminus V$的非空状态集S，状态转换的集合T以及T的逆T^{-1}，若T=(s, a, t)，那么T^{-1}=(t, a, s),输入$T[s]$则会返回(a, t)。remark负责将块号blockN重新分配，以便不稳定状态的计算块号，countB数组用于存储对应块号的状态数。算法的正确性在第 3.3 节的引理 3，引理 4 和定理 5 给出。

假设某 LTS 有 n 个状态，m 条状态转换。算法的时间复杂度取决于迭代阶段。在每次迭代中，算法计算每个状态的σ，这一步骤需要检查所有转出转换，其时间复杂度为O(m)。之后，算法根据σ更新每个状态的块，这一步骤的时间复杂度为O(n)。因此，单次迭代的时间复杂度为O(n + m)。最坏情况下，算法需要n次迭代，每次迭代需要引入新的块，直到达到最细划分。因此，算法的总时间复杂度为$O(nm + n^2)$。

3.1.3 划分计算

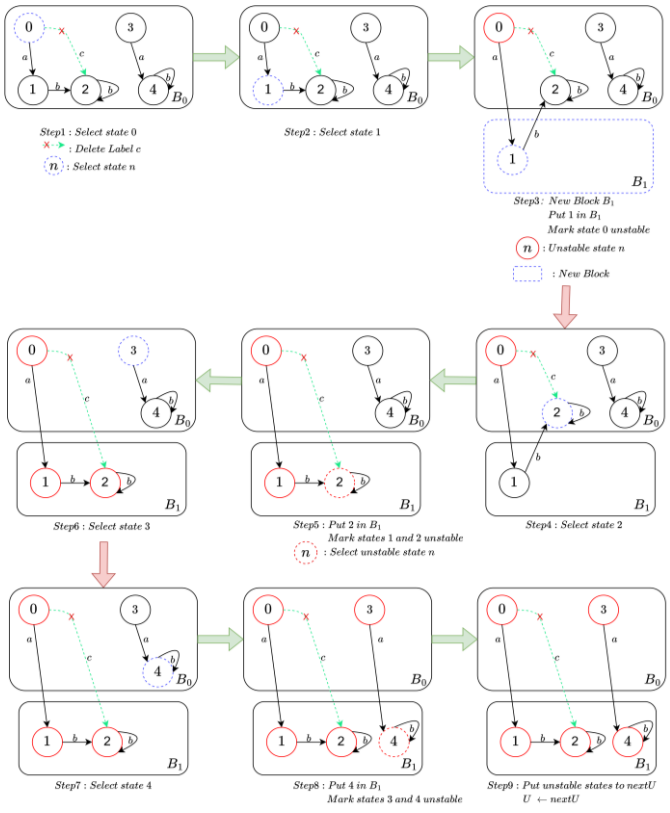

图 2 强 V-互模拟算法单次迭代的示例图

Fig.2 An example of a single iteration in the strong V-bisimulation algorithm

图 2 给出了图 1 实例在强 V-互模拟细化中的第一轮处理：初始将全部状态置于初始块B_0，并令不稳定集合$U \leftarrow S$。在本轮中，对任意取出的状态$s \in U$(按读取顺序)，先计算其当前划分下的签名$\sigma(s)$。随后据签名确定块号：若存在已登记的映射$\rho(\sigma(s)) = 0$,则$mark(s) \leftarrow 0$，否则新建块号1（若$\sigma(s)$在可复用池 $remark$中存在空闲号则复用，否则分配新号），并保存$\rho(\sigma(s)) = 1$、将s放入新块B_1中。若$mark(s) \neq pmark(s)$（即本轮发生分裂），则对任意前驱状态t与存在某动作a使得$(t,a,s) \in T$，将t标记为不稳定并加入$nextU$。

以图 2 为例：状态 0 的块号可在$remark$中复用，直接确认并移除该号； 状态 1 的$\sigma(1) = \{(b,0)\}$在ρ与$remark$中均不存在映射，因此分配新号 1、立块B_1，且因$pmark(1) \neq 1$,将所有满足$(t,a,1) \in T$的前驱t加入$nextU$； 状态2的$\sigma(2) = \{(b,0)\}$与状态1相同，ρ已存在映射到块号1，据此赋$mark(2) = 1$，并因$pmark(2) \neq 1$,将其全部前驱加入$nextU$；状态 3 的$\sigma(3) = \{(a,0)\}$在ρ中存在映射到块号0，且$mark(3) = pmark(3)$，本轮无动作；状态4的$\sigma(4) = \{(b,0)\}$与B_1中的特征一致，赋$mark(4) = 1$，并因$pmark(4) \neq 1$将其全部前驱加入$nextU$。当且仅当本轮对任意$s \in U$均已处理完毕，令$U \leftarrow nextU$并清空$nextU$。若此后$U = \emptyset$，状态 4 的$pmark$值不同，需要将状态 4 加入到B_1块中，则达到了“同名当且仅当同块”的最粗稳定划分，即$\forall s,s' \in S: \sigma[s] = \sigma[s'] \Leftrightarrow mark[s] = mark[s']$。

3.2 反例算法

3.2.1 最短反例算法

在最终划分中，同一块内的任意两个状态x,y必须满足：对所有需要匹配的动作只要 x能通过该动作到达某个后继x'，就存在y的某个后继y'，使得y也能通过同一动作到达y'，并且x'与y'落在同一块；将x与y互换后，上述要求同样成立。算法上等价为：在同一块内、对每个需要匹配的动作检查两点——执行的动作是否一致、到达的后继块号是否一致；若有不一致，则据此分裂该块，直至一致为止。具体检查条件可写为：

$$(action_x = action_y) \wedge (markto_y = markto_x),$$

其中，$action_x$，$action_y$为状态x,y在该标签上的转换动作，$markto_x$，$markto_y$为x,y执行该动作后到达的目标块的块号。

反例算法的核心基于定义 2.6。状态x,y不是强 V-互模拟的核心原因在于状态x,y或其对应的后继状态不能执行相同的动作。换句话说，当反例为(ϵ,α)时，表示状态x或y中只有一个可以执行动作α；当$1 \leq |s|$时，表示x或y的s-派生可以执行动作α，而另一状态的s-派生不能。

算法 2 最短反例生成算法 SCSVB(x,y,LTS)

输入：状态x,y，$A \setminus V$，最终划分下的块数组$mark$；

输出：True或反例。

1. if $mark[x] = mark[y]$ then
2. return True;
3. end if
4. $queue, visited \leftarrow (x,y), aPath[(x,y)] \leftarrow \epsilon$;
5. while ! $queue.isEmpty$ do
6. $(x',y') \leftarrow queue.pull$;
7. if ! $isActionEq(x',y')$ then
8. return $aPath[(x',y')] + dF(x',y')$;
9. end if
10. for all $(action_{x'}, targrts_{x'}) \in T[x']$ do
11. for all $(action_{y'}, targrts_{y'}) \in T[y']$ do
12. if $action_{x'} = action_{y'}$ then
13. if ! $visited.contains(targrts_{x'}, targrts_{y'})$ then
14. $queue, visited \leftarrow (targrts_{x'}, targrts_{y'})$;
15. $aPath[(targrts_{x'}, targrts_{y'})] \leftarrow aPath[(x',y')] + action_{x'}$;
16. end if
17. end if
18. end for
19. end for
20. end while

其中$isActionEq$函数计算两个状态各自的动作在去重后，是否相同，如相同返回 True，不同返回 False。dF函数输入两个状态，返回两个状态不同的动作。aPath为一个字符串数组，如aPath[(x, y)]返回对应状态x, y的字符串。最短反例算法通

过广度优先遍历输入状态对(x,y)以及后继的状态对，当出现$isActionEq$函数返回$False$时（即不能执行相同的动作），则输出反例。算法 2 的正确性在第 3.3 节的定理 6 给出。

我们规定n是状态的总数，m_{max}为 LTS 中单个状态的最大转换数。算法通过广度优先搜索遍历所有状态对，并对每个状态的转换进行嵌套遍历。因此每次处理状态对时需要检查 O($m_{max}{}^2$) 次转换。最坏情况下，队列中最多有 $O(n^2)$个状态对。因此最坏情况下最短反例的时间复杂度为$O(m_{max}{}^2 \cdot n^2)$。

3.2.2 SCSVB 算法举例

仅生成反例并不足以解释为何两个状态不满足强 V-互模拟；对反例的分析则能定位差异并指导最小化修改[22]，基于此，本文采用算法 2（SCSVB）持续计算最短反例(s,F)并分析其中的关键动作，以将系统修改控制在最小范围内。

以图 3（售卖机）所示的 LTS $A=(S,T,s_0)$为例。当 V={τ}（忽略内部动作）时，判定初态对(Ven,Nven)是否强 V-互模拟，SCSVB 返回最短反例$(s,F)=(10¥,withdraw)$。其语义为：存在路径 Ven$\xrightarrow{10¥}$Ven1 使得 withdraw∈ ALL_V(Ven1)，而对任意满足Nven$\xrightarrow{10¥}$Nven1的后继，Nven1 均不存在$withdraw$转移，故二者不强 V-互模拟。据此做最小增边 (Nven1,withdraw,withdraw.Nven) 后复检，可得 Ven $\sim_{V,A}$ Nven。

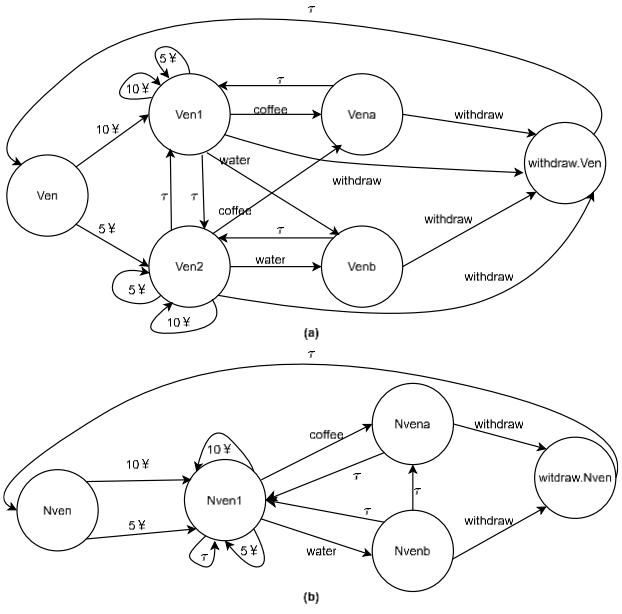

图 3.售卖机例子

Fig.3 Vending machine example

当 V=∅时，SCSVB 仍先给出(10¥,{withdraw})，在完成上述增边后再次检测，算法返回新的最短反例(s,F)=(10¥.water.τ,{10¥})：即存在 Ven1$\xrightarrow{10¥.water.\tau}$ Ven2 且$10¥ \in ALL_{\emptyset}(Ven2)$，而对任意 Nven$\xrightarrow{10¥.water.\tau}$Nvena 的后继，Nvena 均不能执行10¥。考虑到 Nvena表示“取完咖啡”的状态而 Ven2 仍允许再次投币（并在 {coffee,water,withdraw}中选择），据此做最小删边 (Nvenb,τ,Nvena)。第三次检测显示不再存在反例 $(s,F) \in \mathrm{Count}E_{V,A}$(Ven,Nven)，从而 Ven $\sim_{\emptyset,A}$ Nven成立。

3.3 算法正确性

下面的引理给出 V 处理的正确性，本文称$P \sim_{V,A} Q$为 P,Q 在标记转换系统 A 下为强 V-互模拟的，称$P \sim_{A\setminus V} Q$为 P,Q 在标记转换系统$A \setminus V$下为强互模拟[4]的。

引理 1[5]. 设$A=(S,T,s_0)$ 为一个 LTS，$V \subseteq Act$，且$P,Q \in S$。则 $P \sim_{V,A} Q$ 当且仅当 $P \sim_{A\setminus V} Q$。

引理 1 表明，若状态 P 和 Q 是强 V-互模拟的，当且仅当在动作集合限制 V 下的 LTS，状态 P 和 Q 是强互模拟的。

对于给定的 A=LTS(S,T,s_0)和 S 的一个初始划分υ_0，如果状态$P,Q \in S$在划分υ_0下存在强 V-互模拟关系，记为 $P \rightleftharpoons_{\upsilon_0} Q$，则 A 经过 V 处理后，若$P$和$Q$依然保持相同的互模拟关系，即$P \rightleftharpoons_{\upsilon_0 V} Q$，则 V 处理是可行的。为了方便，用 $P \nrightarrow$表示状态P没有任何状态转换（包括自身）。

引理 2.设$A=(S,T,s_0)$为一个 LTS，$V \subseteq Act$，υ_0为初始划分，且$P,Q \in S$。如果$P \rightleftharpoons_{\upsilon_0} Q$，则 $P \rightleftharpoons_{\upsilon_0 V} Q$。

证明:对于状态P的所有转换，有以下两种情况：

情况 1: $P \nrightarrow$, 且$P \rightleftharpoons_{\upsilon_0} Q$

$\Rightarrow$ $P \nrightarrow$ (V 处理后),

$\Rightarrow$ $Q \nrightarrow$ 或 $Q \twoheadrightarrow^{A}$ $(A \subseteq V)$

1) 如果 $Q \nrightarrow$,

$\Rightarrow \rightleftharpoons_{\upsilon_0}$ = ~(P和Q都不能执行任何动作),

$\Rightarrow \rightleftharpoons_{\upsilon_0 V}$ = ~(V 处理后, P和Q拥有相同的转换),

$\Rightarrow \rightleftharpoons_{\upsilon_0 V} = \rightleftharpoons_{\upsilon_0}$,

$\Rightarrow (P,Q) \in \rightleftharpoons_{\upsilon_0 V}$;

2) 如果 $Q \twoheadrightarrow^{A}$ $(A \subseteq V)$,

$\Rightarrow$ $Q \nrightarrow$ (V 处理后),

$\Rightarrow \rightleftharpoons_{\upsilon_0 V}$ = ~(P和Q都不能执行任何动作),

$\Rightarrow$ (P, Q) $\in \rightleftharpoons_{\upsilon_0 V}$;

情况 2: P $\twoheadrightarrow^{B}$, 且B ⊆ Act，考虑以下两种子情况:

1. 如果 B ⊆ V,

$\Rightarrow$ P $\nrightarrow$ (V 处理后),

⇒ Q ↛ 或 Q $\twoheadrightarrow^{B}$,

a) Q ↛,

⇒ Q ↛ (V 处理后),

⇒ $\rightleftharpoons_{\upsilon_0 V} = \sim$,

⇒ $(P,Q) \in \rightleftharpoons_{\upsilon_0 V}$;

b) Q $\twoheadrightarrow^{B}$,

⇒ Q ↛ (V 处理后),

⇒ $\rightleftharpoons_{\upsilon_0 V} = \sim$,

⇒ $(P,Q) \in \rightleftharpoons_{\upsilon_0 V}$;

2. 如果 $B \subseteq Act - V$,

⇒ P $\twoheadrightarrow^{B}$ (V 处理后),

⇒ Q $\twoheadrightarrow^{A}$且$B \subseteq A \subseteq (B \cup V)(P \rightleftharpoons_{\upsilon_0} Q)$,

⇒ Q $\twoheadrightarrow^{B}$(V 处理后),

⇒ $\rightleftharpoons_{\upsilon_0 V} = \sim$,

⇒ $(P,Q) \in \rightleftharpoons_{\upsilon_0 V}$;

3. 如果 $B \setminus V \neq \emptyset$,

⇒ P $\twoheadrightarrow^{B'}$且$B' = B - V$(V 处理后),

⇒ Q $\twoheadrightarrow^{A}$且$B' \subseteq A \subseteq (B' \cup V)(P \rightleftharpoons_{\upsilon_0} Q)$,

⇒ Q $\twoheadrightarrow^{B'}$(V 处理后),

⇒ $\rightleftharpoons_{\upsilon_0 V} = \sim$,

⇒ $(P,Q) \in \rightleftharpoons_{\upsilon_0 V}$;

⇒ P $\rightleftharpoons_{\upsilon_0 V}$ Q。

引理 3，引理 4 和定理 5 给出了算法 1 的正确性证明。为方便，记$mark^t[P]$，ρ_P^t，σ_P^t分别为状态P在第t次划分计算的块号，哈希表值和签名。

引理 3.设υ_0为初始划分，υ_t为由算法 1 第t次计算生成的划分。对于任意迭代次数$t \in N$，$P \sim_{\upsilon_0 V} Q$蕴含 $P \sim_{\upsilon_t V} Q$。

证明：基始: $P \sim_{\upsilon_0 V} Q$ 蕴含 $P \sim_{\upsilon_0 V} Q$, 这显然成立。

步骤: 假设 $P \sim_{\upsilon_t V} Q$成立。

⇒ $mark^t[P] = mark^t[Q](P \sim_{\upsilon_t V} Q)$。

⇒ 在划分υ_t中，由某个块生成一个新块 B_{t+1}（根据定义 2.4）。

⇒ $P \sim_{\upsilon_{t+1} V} Q$取决于$P$与$Q$是否有转换到达块$B_{t+1}$：

1. 若P和Q都到达块B_{t+1}，

⇒ $\{P,Q\} \in nextU$（根据算法 1 的步骤 20-22），

⇒ $\rho_P^t = \rho_Q^t(P \sim_{\upsilon_t V} Q)$，

⇒ $\rho_P^{t+1} = \rho_Q^{t+1}$（$P$和$Q$到达块$B_{t+1}$），

⇒ $mark^{t+1}[P] = mark^{t+1}[Q]$，

⇒ $P \sim_{\upsilon_{t+1} V} Q$。

2. 若P和Q都未到达块B_{t+1}，

⇒ $\{P,Q\} \notin nextU$（根据算法 1 的步骤 20-22），

⇒ $mark^t[P] = mark^t[Q] = mark^{t+1}[P] = mark^{t+1}[Q]$，

⇒ $P \sim_{\upsilon_{t+1} V} Q$。

3. 若存在$P' \in B_{t+1}$ 使得 $P \to P'$，但Q不满足，

⇒ 对于所有满足$Q \to Q'$的$Q' \in S$，有 $Q' \notin B_{t+1}$，

⇒ $mark^t[P'] \neq mark^t[Q']$，

⇒ $P' \sim_{\upsilon_t V} Q'$ ($P \sim_{\upsilon_t V} Q$），

⇒ $mark^t[P'] = mark^t[Q']$（矛盾）。

因此，在所有情况下, $P \sim_{\upsilon_{t+1} V} Q$均成立。

引理 4.给定一个$A \setminus V = LTS(S,T,s_0)$，算法 1 将在有限步数内终止。

证明:本文通过证明$blockN$是有界的且其变化是有限的来证明此引理。

有界性：对于任意状态$x \in S$，只有当 $\sigma_x \notin \rho$ 且$mark[x] \notin remark$时，有$blockN = blockN + 1$ (见算法 1 第 18 步)，在最坏情况下，$\rho_a \neq \rho_b$，对于任意状态，$a,b \in S$,所以$blockN = n + 1$，因此$blockN \leq n + 1$，这证明了$blockN$是有界的。

有限性：只有在算法 1 的第 14 和第 18 步中才有更改$mark$的操作，其中第 14 步确保对于任意状态 $a,b \in S$,$\sigma_a = \sigma_b \Leftrightarrow mark[a] = mark[b]$，从而减少了$mark$的变化。第 18 步确保如果 $\sigma_x \notin \rho$ 且 $mark[x] \notin remark$则生成新的块，并将新块号赋值给$mark[x]$。最终，所有状态x处于稳定块中，因此没有状态满足第 21 步。这使得$nextU = \emptyset$，在下一轮迭代中$U = \emptyset$，算法终止。

定理 5.给定一个$A \setminus V = LTS(S,T,s_0)$，$V \subseteq Act$，最终$mark[x]$所确定的关系$\sim_{V,A}$是$A$上的一个强 V-互模拟关系。

证明:令n为最后一次循环的索引，并且对于任意状态 $x,y \in S$,满足$mark^n[x] = mark^n[y]$,

⇒ $\sigma_x^n = \sigma_y^n(\sigma_x = \sigma_y \Leftrightarrow mark[x] = mark[y])$,

假设 $x \xrightarrow{\alpha} x'$，$y \xrightarrow{\alpha} y'$，其中 $\alpha \in Act - V$。若要证明$(x',y') \in \sim_{V,A}$，有：

$\{(labelId_\alpha, mark^{n-1}[x'])\} \in \sigma_x^n$(根据定义 2.5 和算法 1 的第 7 步)，

⇒ $\{(labelId_\alpha, mark^{n-1}[y'])\} \in \sigma_y^n$(根据定义 2.5 和算法 1 的第 7 步)，

⇒ $mark^{n-1}[x'] = mark^{n-1}[y'](\sigma_x^n = \sigma_y^n)$，

⇒ $mark^{n-1}[x'] = mark^n[x'] mark^{n-1}[y'] = mark^n[y']$ (n是$nextU = \emptyset$的最后一次循环)，

⇒ $mark^n[x'] = mark^n[y']$，

⇒ $(x',y') \in \sim_{A \setminus V}$(定义 2.5 和公式(3))，

⇒ $(x',y') \in \sim_{V,A}$(引理 1)。

下文的定理 6 给出了反例算法的正确性证明，本文称$CountE_{A \setminus V}(P,Q)$为状态$P,Q$在$A \setminus V$下的反例。

定理 6.设$A = (S,T,s_0)$ 是一个 LTS，$V \subseteq Act$，并且$P,Q \in S$。若 P,Q 不强 V-互模拟，则必然存在$CountE_{V,A}(P,Q) = CountE_{A \setminus V}(P,Q)$。

证明:(⇒)$CountE_{V,A}(P,Q)$

⇒存在一个反例$(s,\alpha) \in CountE_{V,A}(P,Q)$是在$A$下的强 V-

互模拟反例，

⇒对于$(s,\alpha)\in(Act-V)^*\times(Act-V)$有如下条件之一成立：

a) 存在满足$P\xrightarrow{s}P'\xrightarrow{\alpha}P''$的$P'$和$P''$，对任意满足$Q\xrightarrow{s}Q'$的$Q'$，$Q'\nrightarrow^{\alpha}$，或者

b) 存在满足$Q\xrightarrow{s}Q'\xrightarrow{\alpha}Q''$的$Q'$和$Q''$，对任意满足$P\xrightarrow{s}P'$的$P'$，$P'\nrightarrow^{\alpha}$。

⇒ (s,α)是P,Q在$A\setminus V$下的一个反例

⇒ $CountE_{A\setminus V}(P,Q)$。

(⇐)$CountE_{A\setminus V}(P,Q)$

⇒存在一个反例$(s,\alpha)\in CountE_{A\setminus V}(P,Q)$是在$A\setminus V$下的强互模拟反例，

⇒对于$(s,\alpha)\in(Act-V)^*\times(Act-V)$有如下条件之一成立：

a) 存在满足$P\xrightarrow{s}P'\xrightarrow{\alpha}P''$的$P'$和$P''$，对任意满足$Q\xrightarrow{s}Q'$的$Q'$，$Q'\nrightarrow^{\alpha}$，或者

b) 存在满足$Q\xrightarrow{s}Q'\xrightarrow{\alpha}Q''$的$Q'$和$Q''$，对任意满足$P\xrightarrow{s}P'$的$P'$，$P'\nrightarrow^{\alpha}$。

⇒ (s,α)是P,Q在A下的强 V-互模拟反例，

⇒ $CountE_{V,A}(P,Q)$。

4. 实验结果与分析

本文使用 CADP 的 VLTS 数据集[1]，该数据集使用 AUT 格式，这是一种描述 LTS 的简单文本文件格式。数据集中有死锁、活锁和确定性等，覆盖不同并发系统情况，且转换系统数据规模大。VLTS 数据集包含 40 个 LTS，其状态个数$n\in[289,33949609]$，转换个数$m\in[1224,165318222]$，动作$Act\subseteq\mathcal{F}\cup\tau$。本实验的环境是 Linux20.04、8 核 3.50GHz 的 CPU 和 16GB 内存，算法使用 JAVA 11.0.24 实现，代码及实验数据公开在 github 上[2]。

4.1 实验结果

本实验在不同的动作集合 V 下进行，本实验考虑 V=∅、V = {τ}、|V| = 1、|V| = 3和|V| = 5这三种情况因为动作随机选取，可能含有动作τ。首先对数据集中的所有数据的每种情况均进行了 10 次随机实验并统计平均计算时间。

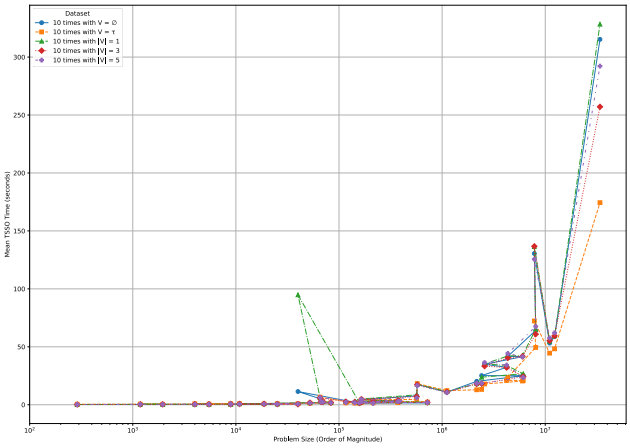

图 4.StrongVB 算法平均 CPU 时间(秒)

Fig.4 The mean CPU time(s) for StrongVB

在图 4 中，展示了 StrongVB 算法计算强 V-等价的问题规模(状态数量)与执行时间（单位：秒）的关系。结果显示，在问题规模小于10^5时，观察到当遗忘动作集V为空集（∅）或仅包含一个元素(|V| = 1)时，算法的运行时间出现了显著增长。这种增长与数据集中的数据 vasy40_60 的特性有关。因此本文在下节具体分析数据 vasy40_60 的特性。

由于部分数据的测试时间过长，如 vasy_11026 和 cwi_33949 等，其单个测试时间在 2 至 4 分钟左右。若对所有数据计算 1000 次取平均值，虽然其实验结果接近真实值，但需花费时间过长，因此本文仅展示表 1 的实验数据，每种动作集合 V 情况下数据均运行 1000 次，取平均值。

表 1 不同动作集合 V 下实验结果（部分），时间（T）单位为毫秒

Table 1 Experimental results (partial) under different action sets V, times (T) are in ms

数据	n	m	V={∅}			V={τ}			|V|=1			|V|=3			|V|=5		
			BN	T_{Pre}	T_{Alg}	BN	T_{Pre}	T_{Alg}	BN	T_{Pre}	T_{Alg}	BN	T_{Pre}	T_{Alg}	BN	T_{Pre}	T_{Alg}
vasy0_1	289	1224	9.0	31.8	16.3	9.0	31.9	16.4	5.0	31.2	14.5	1.0	30.0	10.2	1.0	30.0	10.1
cwi1_2	1952	2387	1132.0	41.4	46.4	11.0	39.0	17.1	1044.0	41.7	45.1	852.1	41.5	41.9	703.8	41.1	39.7
vasy1_4	1183	4464	28.0	47.4	27.5	4.0	46.8	20.7	23.8	47.6	25.4	12.1	46.6	21.8	2.8	45.2	16.5
cwi3_14	3996	14552	62.0	62.8	39.4	2.0	57.1	16.0	31.2	60.4	27.6	1.0	57.0	16.5	1.0	57.1	16.5

[1] https://cadp.inria.fr/resources/vlts/

[2] https://github.com/sokbutnew/StrongV-bisimulationandcounterexample/tree/main.

vasy5_9	5486	9676	145.0	64.8	40.4	133.0	64.3	36.5	140.6	65.0	40.4	130.2	64.6	39.3	118.6	64.7	38.8
vasy8_24	8879	24414	416.0	86.3	77.9	119.0	84.7	48.5	366.5	87.1	75.0	227.5	86.2	66.2	102.3	84.8	54.4
vasy8_38	8921	38424	219.0	143.8	75.7	193.0	143.3	69.4	216.4	146.6	74.7	211.4	147.2	73.6	205.1	149.2	72.1
vasy10_56	10849	56156	2112.0	127.0	238.8	28.0	126.7	92.7	946.4	131.3	190.0	186.7	132.1	112.8	33.4	129.9	70.5

如表 1 所示，本文对 5 种动作集合 V 的取值分别按照块数（BN）、V 处理时间（T_{Pre}）和算法时间（T_{Alg}）统计了数据（部分）的计算结果3。其中 n 指数据中的状态数，m 指的是数据中状态转换个数。块数的单位为个，V 处理以及算法的时间单位为毫秒（ms）。从表中可以观察到，随着|V|的增大，BN、T_{Pre}和T_{Alg}都在降低，这意味着如果对更多的动作信息进行遗忘，就会有越来越多的状态之间满足强 V-互模拟关系，甚至在极端情况下，所有状态都是强 V-互模拟的。

如 vasy0_1 数据，其共有两个转换标签“G !TRUE”和“G !FALSE”，当 V={∅}或 V={τ}时，V 处理效果相同，故而实验结果大致相同，最终划分有 9 个块，V 处理时间在 31.8ms 左右，算法时间在 16.3ms 左右；当|V|=1 时，最终划分中的块数为 5.0，这意味着当遗忘某个动作信息后，某些状态之间有了互模拟关系；当|V|=3 时，由于该数据一共仅有两个动作，故而在遗忘所有动作信息的情况下，所有状态都是强 V-互模拟的。因此，在满足遗忘信息条件的情况下，尽可能多地遗忘掉不相关的信息，可以使检测效率提高。当 V 取{τ}时，其实验效果可能比|V|=1 甚至|V|=3 好，如数据 cwi1_2、vasy8_38 等，这表示 V 的选取与动作在转换中出现的次数有很大关系。

4.2 vasy40_60 数据分析

vasy40_60 数据包含 40,006 个状态、60,007 个转换和 3 个动作标签。动作标签出现的次数较为均匀，动作标签τ出现在 20,003 个状态转换中，动作标签 1 和动作标签 2 各出现在 20,002 个状态转换中。接着考虑对于不同 V 的选取对算法运行时间（V 处理时间以及强 V-互模拟算法时间）和生成块的个数的影响。当 V 取∅时，共生成了 40,006 个块，这表明该 LTS 中任意两个状态都不是强互模拟的；当$V = \{1\}$或$V = \{2\}$时，划分中有 40,006 个块，这表示该 LTS 中任意两个状态都不满足强{1}互模拟或强{2}互模拟；然而，当$V = \{\tau\}$时，划分中仅有 6 个块，这表示有许多状态为强{τ}互模拟。表 2 给出了|V| = 1 下十次运行下最终块数、算法进入循环的次数（#It）与 V 处理时间和算法时间的详细信息。当 V = {2} 时，运行时间最大，超过 150 秒，而 V = {1}时运行时间相对较低，当 V = {τ}时，算法的运行时间相较于其他太小而忽略不记。

表 2 vasy40_60 数据分析，时间（T）单位为秒

Table 2 Analysis for vasy40_60, times (T) are in s

V	BN	#It	T_{Pre}	T_{Alg}
1	40006	40002	0.360	37.995
2	40006	40002	0.399	169.587
2	40006	40002	0.360	202.918
1	40006	40002	0.672	20.933
τ	6	2	0.377	0.177
2	40006	40002	0.359	158.294
2	40006	40002	0.473	153.319
1	40006	40002	0.517	26.077
τ	6	2	0.572	0.188
2	40006	40002	0.425	174.596

从表中可以发现，对于 V={1}，V={2}的情况，除了最终块数为 40,002，算法迭代的次数也为 40,002，这表示该算法的每次迭代都几乎仅有一个状态被确定所属块。这表明状态之间存在类似于线性的关系。一个状态被确定后，与之相连的状态才能在下一次迭代被确定。接下来，进一步分析 vasy40_60 数据的实际内容结构，vasy40_60 主要按τ、1 和 2 的顺序排列（在开头和结尾稍有变化）。总体上，转换主要由动作τ和 2 主导，而动作 1 偶尔出现状回归现象（例如，从状态 i 到状态$i-3$），如图 4 所示。

在图 4 中，Iteration1 和 Iteration 2 之后的三角形和钩形符号指示了每次迭代中被放在$NextU$中的状态。当$V = \emptyset$时，每次迭代中有大量状态在$NextU$中，导致较长的运行时间。相反，当$V = \tau$时，第一次迭代后状态数量迅速减少，$NextU$仅包含具有转出转换的状态。这导致下一次迭代中$NextU$为空，从而使得其运行时间最快。

[3] 实验结果由于篇幅限制，只展示部分，对于所有块数和算法时间，取 1000 次计算结果的平均值，按四舍五入取小数点后 1 位.

当$V = \{1\}$时，尽管在第一次迭代后状态数量迅速减少，但所有状态按顺序排列，导致算法在每次迭代中回溯到初始状态。该固定模式要求遍历所有状态，每次迭代仅处理一个状态，导致资源消耗显著，平均运行时间较高。

相反，当$V = \{2\}$时，每次迭代中$NextU$包含大量状态。由于状态之间的相互关联，每次迭代处理大量状态，在随后的迭代中$NextU$仅略微减少。例如，如果在第i次迭代中$NextU$的大小为n，则在第$i+1$次迭代中它减少为$n-1\ (1 < i < Vertex,\ 0 < n < Vertex)$。这导致运行时间最长。

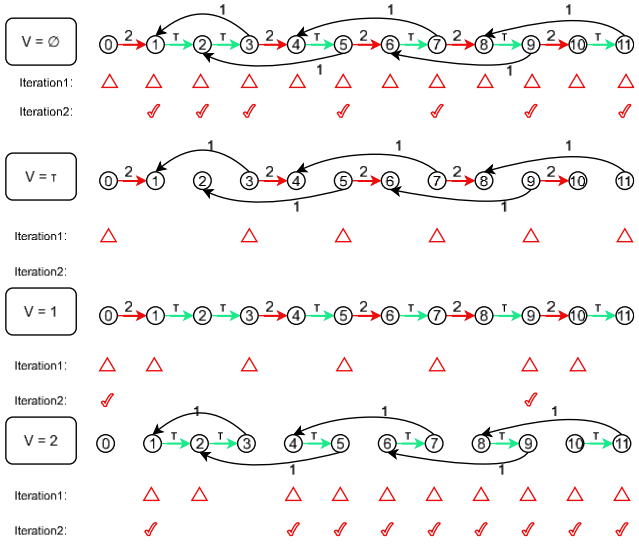

图 4.vasy40_60 数据详细分析（部分）

Fig.4 Detailed analysis of vasy40_60 benchmark(partial)

4.3 结果分析

强 V-互模拟算法的运行时间与动作集V的大小以及V中转换动作的出现频率密切相关。当$V = \emptyset$或$|V| = 1$时，算法往往需要执行更多迭代，特别是在较大的数据中，导致较长的运行时间。例如，在 vasy40_60 数据中，当$V = \emptyset$时，运行时间超过 150 秒，并伴随大量迭代次数。随着V的大小增加（如$|V| = 3$或$|V| = 5$），迭代次数明显减少，缩短了运行时间，提高了算法在大规模状态空间下的处理效率。

当$V = \{\tau\}$时，算法的表现效果与其在 LTS 中出现的次数以及出现的转换结果有关，因此，遗忘动作信息需要针对具体的 LTS 结构以及其在转换中出现的次数。尽可能的多遗忘动作信息，特别是在大型数据总运行时间显著降低。这表明，随着$|V|$的增加，计算时间显著减少，凸显了选择合适的V以实现最佳性能的重要性，特别是在处理具有庞大状态空间的系统时，选择合适的动作集合 V 尤其重要。

4.4 反例实验

表 3 展示了对数据集中 8 个数据分别随机取两个状态进行 SCSVB 算法计算，每个数据计算 1000 次取不满足强 V-互模拟结果的平均值，这里l_{SCE}指最短反例的平均长度，T_{SCE}指 SCSVB 算法运行的平均时间。可以发现，当|V|=3 或|V|=5 时，vasy0_1 和 cwi3_14 数据中l_{SCE}，T_{SCE}均为-符号，这是因为当|V|=3 或|V|=5 时，这两个数据中所以状态都是强 V-互模拟的，不存在反例。

表 3 反例结果（部分），时间单位（ms）

Table 3 Counterexample results(partial), times are in ms

数据	|V|=1		|V|=3		|V|=5	
	l_{SCE}	T_{SCE}	l_{SCE}	T_{SCE}	l_{SCE}	T_{SCE}
vasy0_1	2.35	17.81	-	-	-	-
cwi1_2	3.52	18.86	3.46	19.05	3.45	19.12
vasy1_4	1.14	19.08	1.16	19.28	1.23	19.05
cwi3_14	41.35	131.16	-	-	-	-
vasy5_9	1.02	18.33	1.03	18.33	1.02	18.24
vasy8_24	1.05	18.30	1.08	18.67	1.10	18.85
vasy8_38	1.02	16.60	1.02	16.65	1.02	16.76
vasy10_56	1.33	19.10	1.24	19.37	1.17	19.28

5 结论

等价性检查技术长期以来一直是软硬件系统形式化验证的重要方法之一[2]，Paker 和 Milner 提出的强互模拟概念[4]要求系统状态之间逐步以同名动作严格匹配并保持后继互模拟；随之而来的弱互模拟[14] 与分支互模拟[15] 为内部动作与分支结构提供了更宽松的等价标准。进一步贴近工程实践中“只关注关键动作”的需求， Zhou[5]首次提出强 V-互模拟的概念，其允许系统“遗忘”[16,17]动作，从定义层面增强了等价性检查的针对性与泛化能力。Blom 和 Orzan[2] 提出了分布式互模拟算法以在多处理器间分摊工作量、缓解状态空间爆炸[12,13]，但其进程间通信仍是可扩展性的关键瓶颈。

此外，基于强互模拟的等价性检测技术已被多个等价性检查工具纳入其中，CADP 在其 Bcg_min 工具中采用了 Paige-Tarjan 算法进行 LTS 最小化，有效减少了状态规约后的数量。同样，mCRL2 提供了 lps2lts 和 ltsmin 等工具，通过强互模拟进行 LTS 生成和最小化，从而能够高效分析大规模模型。SPIN[21]也利用互模拟技术检查等价性并简化系统行为，助力高效验证并发系统。

尽管强互模拟与分支互模拟已有广泛研究与工程实现，强 V-互模拟仍缺乏成熟实现与系统性的反例生成研究。本文在 Blom 分布式互模拟算法的框架上，给出了强 V-互模拟的工程化实现（StrongVB）及其配套的最短反例算法（SCSVB），并从正确性、终止性与最短性方面给出理论保证。在 CADP 的 VLTS 基准上，实验显示：在遗忘部分动作后，状态与边规模显著下降，等价判定与最短反例生成的时间与内存占用均明显优于强互模拟基线；同时，反例驱动的最小增删边能够有效消除差异，服务于并发系统面向关键动作的修复需求。

本文围绕指定动作集 V 提出并实现了强 V-互模拟的等价判定与最短反例生成方法。通过 V 处理把判定关系收敛到关键动作，StrongVB 在保证等价保持的同时有效缓解状态空间爆炸，SCSVB 提供可解释、可用于最小化修复的最短反例。理论与实验共同表明，该路线能够在并发系统中兼顾效率、可解释性与可操作性，为实用化的等价性检查与修复提供了新的支持。

未来，我们计划将 StrongVB/SCSVB 推广到基于分支互模拟的并发系统等价检查，进一步完善静默动作与分支对齐的处理；其次，结合最弱充分条件规范，将最短反例提炼为差异原因的逻辑解释，并据此生成可执行的修复建议，形成“性质提炼—反例解释—最小修复”的闭环；再次，在分布式环境中按 V 划分工作与通信边界以抑制跨进程通信；最后，扩充 VLTS/工业级基准并引入可重复的反例生成设置，为强 V-互模拟的实证研究提供更扎实的数据支持。

参考文献

[1] BURKART O, CAUCAL D, MOLLER F, et al. Verification on Infinite Structures[M/OL]//Handbook of Process Algebra. Elsevier, 2001: 545-623.

[2] GARAVEL H, LANG F. Equivalence Checking 40 Years After: A Review of Bisimulation Tools[M]//JANSEN N, STOELINGA M, VAN DEN BOS P. A Journey from Process Algebra via Timed Automata to Model Learning: Vol. 13560. Cham: Springer Nature Switzerland, 2022: 213-265.

[3] KEIREN J J. Advanced reduction techniques for model checking[A/OL]. Technische Universiteit Eindhoven, 2013.

[4] MILNER R. Communication and concurrency[M]. New York: Prentice Hall, 1989.

[5] ZHOU X, WANG Y, FENG R, et al. Strong Forgetting in Hennessy-Milner Logic[M]//CHIN W N, XU Z. Theoretical Aspects of Software Engineering: Vol. 14777. Cham: Springer Nature Switzerland, 2024: 465-472.

[6] BLOM S, ORZAN S. A Distributed Algorithm for Strong Bisimulation Reduction of State Spaces[J]. Electronic Notes in Theoretical Computer Science, 2002, 68(4): 523-538.

[7] FERNANDEZ J C, GARAVEL H, KERBRAT A, et al. CADP a protocol validation and verification toolbox[M]//ALUR R, HENZINGER T A. Computer Aided Verification: Vol. 1102. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996: 437-440.

[8] GROOTE J F, MATHIJSSEN A, VAN WEERDENBURG M, et al. From μCRL to mCRL2[J]. Electronic Notes in Theoretical Computer Science, 2006, 162: 191-196.

[9] BLOM S, ORZAN S. Distributed Branching Bisimulation Reduction of State Spaces[J/OL]. Electronic Notes in Theoretical Computer Science, 2003, 89(1): 99-113.

[10] KANELLAKIS P C, SMOLKA S A. CCS expressions, finite state processes, and three problems of equivalence[J]. Information and Computation, 1990, 86(1): 43-68.

[11] JEONG C, KIM Y, KIM H, et al. A Faster Parallel Implementation of The Kanellakis-Smolka Algorithm for Bisimilarity Checking[J]. 1998.

[12] CLARKE E M, HENZINGER T A, VEITH H, et al. Handbook of Model Checking[M]. Cham: Springer International Publishing, 2018.

[13] CLARKE E M, GRUMBERG O. Avoiding the state explosion problem in temporal logic model checking[C]//Proceedings of the sixth annual ACM Symposium on Principles of distributed computing - PODC ’87. Vancouver, British Columbia, Canada: ACM Press, 1987: 294-303.

[14] MILNER R. A Calculus of Communicating Systems: volume 92[M/OL]. Berlin, Heidelberg: Springer Berlin Heidelberg, 1980.

[15] VAN GLABBEEK R J. The linear time — Branching time spectrum II: The semantics of sequential systems with silent moves extended abstract[M/OL]//BEST E. CONCUR’93: Vol. 715. Berlin, Heidelberg: Springer Berlin Heidelberg, 1993: 66-81.

[16] FENG R, WANG Y, QIAN R, et al. Knowledge forgetting in propositional μ-calculus[J]. Annals of Mathematics and Artificial Intelligence, 2023, 91(1): 1-43.

[17] FANGZHEN LIN, REITER R. Forget It！[C]//Working Notes of AAAI Fall Symposium on Relevance. 1994: 154-159.

[18] WIJS A. GPU Accelerated Strong and Branching Bisimilarity Checking[M]//BAIER C, TINELLI C. Tools and Algorithms for the Construction and Analysis of Systems: Vol. 9035. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015: 368-383.

[19] PAIGE R, TARJAN R E. Three Partition Refinement Algorithms[J/OL]. SIAM Journal on Computing, 1987, 16(6): 973-989.

[20] MARTENS J, GROOTE J F, HAAK L B V D, et al. Linear parallel algorithms to compute strong and branching bisimilarity[J]. Software and Systems Modeling, 2023, 22(2): 521-545.

[21] HOLZMANN G J. The spin model checker: primer and reference manual[M]. 4th print. Boston, Mass. Munich: Addison-Wesley, 2008.

全部作者中文简介

胥松杭，2000 年生。硕士。主要研究方向为知识表示与推理、人工智能。

王以松，1975 年生。贵州大学人工智能研究院负责人、教授；Annals of Mathematics and Artificial Intelligence 副编辑。主要研究知识表示与推理、归纳学习、知识遗忘等.

周欣，1981 年生。特里尔大学(德国)硕士研究生、贵州大学博士研究生。德国特里尔数字人文研究中心研究员、贵阳学院教师。主要研究方向为软件工程、知识表示与推理、人工智能。

冯仁艳，1991 年生。冯仁艳，贵州财经大学，副教授。本科就读于北方工业大学计算机科学与技术学院，贵州大学硕士，贵州大学和阿姆斯特丹自由大学(Vrije University Amsterdam)联合培养博士，中国科学院软件研究所国家重点实验室、西南大学软件研究与创新中心访问学者，在国大外一流期刊和会议上发表数篇文章。研究兴趣:知识表示与推理，模型检测，遗忘理论，智能规划和回答集程序等。

张元睿，南京航空航天大学，2019 年毕业于华东师范大学软件工程专业，获得博士学位。2013-2014 年赴法国 INRIA 国家信息自动化研究所访问 1 年，2019-2023 年在西南大学从事博士后研究工作。目前，已在软件工程形式化方法领域的国内外期刊会议发表论文 20 余篇，包括：Science of Computer Programming 、 Formal Aspects of Computing 、 Journal of Logical and Algebraic Methods in Programming 、 Frontier of Computer Science、TASE、QRS 等。主持国家自然科学青年基金、重庆市自然科学博士后基金等项目资助，

Xu songhang, born in 2000. Master. His main research interests include knowledge representation and reasoning, artificial intelligence.

Wang yisong, born in 1975. Head and Professor at the artificial Intelligence Research Institute, Guizhou University. Annals of Mathematics and Artificial Intelligence Deputy Editor. His main research interests include knowledge representation and reasoning, Inductive learning, knowledge forgetting.

Zhou xin, born in 1981. PHD candidate. His main research interests include software engineering, knowledge representation and reasoning, artificial intelligence. （zhouxinj@hotmail.com）

Feng Renyan, born in 1991. PHD candidate. Her main research interests include knowledge representation and reasoning, artificial intelligence. （827220951@qq.com）

Zhang Yuanrui, born in 1990. PHD candidate. His main research interests include software engineering. (yuanruizhang@nuaa.edu.cn)