DOI:10.13471/j.cnki.acta.snus.2015.04.009

一种优化的可能性测度计算树逻辑检测模型*

陈燕升[1,2]，张赞波[3]，吴忠坤[3]，任江涛[4]
（1. 广东轻工职业技术学院环境工程系，广东 广州 510300；
2. 昆山科技大学资讯管理系，台湾 台南 71003；
3. 广东轻工职业技术学院计算机工程系，广东 广州 510300；
4. 中山大学软件学院，广东 广州 510275）

摘 要：可能性测度计算树逻辑模型检测验证中存在诸多问题，例如低性能效率和高时间复杂度。针对上述问题，基于传统的模型检测标记算法，为满足高复杂性、大规模的公式标记检测，设计并实现了 I-PM_ CTL 算法。其基本步骤如下：第一步，先利用相关可能性测度对逻辑树公式进行计算，预处理标识公共子表达式的唯一性；第二步，在充分确保模型检测空间平衡状态下设定公共子表达式与可能性测度计算树逻辑模型状态；第三步，实施验证，为可能性测度计算树逻辑公式以极大概率一次性实现验证提供了保证。经过模拟实验发现，这一种方法一方面在很大程度上减小了相关时间复杂度，另一方面还使验证性能有所提升。

关键词：可能性测度；模型检测；公共子表达式；计算树逻辑
中图分类号：TP301.2 **文献标志码**：A **文章编号**：0529－6579（2015）04－0049－06

The Model of Checking Method Based on Improved Computation Tree Logic Possibility Measure

CHEN Yansheng[1,2], *ZHANG Zanbo*[3], *WU Zhongkun*[3], *REN Jiangtao*[4]
（1. Department of Environmental Engineering, Guangdong Industry Technical College, Guangzhou 510300, China;
2. Kun Shan University, Department of Information Management, Tainan 71003, China;
3. Department of Computer Engineering, Guangdong Industry Technical College, Guangzhou 510300, China;
4. School of Software, Sun Yat-sen University, Guangzhou 510275, China）

Abstract: To settle various problems in computation tree logic possibility measure model validation process, such as high time complexity and low efficiency performance. The I-PM_ CTL algorithm is proposed. The algorithm is based on the traditional model checking mark algorithm, so that it is adaptable to the need of mark detection of large-scale, highly complex formulas. The steps of the algorithm are as follows. Fristly, the logic tree formulas are calculated using relevant possibility measure, which is a preprocessing step to identify the uniqueness of the common sub-expressions. Secondly, it specifies the state of common sub-expressions and I-PM_ CTL model, while maintaining the equilibrium of model checking space. Finally, this algorithm implements the verification, ensuring the I-PM_ CTL formulas verification to be completed in one time with high probability. The results of simulation experiments show that the I-

* **收稿日期**：2014－11－23
基金项目：国家自然科学基金资助项目（11471342）；广东省教育科研“十二五”规划 2012 年度研究资助项目（2012JK089）；广东省高职教育教改资助项目（201401034）；校级自然科学基金资助项目（KJ201311/KJ2014）
作者简介：陈燕升（1979 年生），男；**研究方向**：系统自动化测试方法、网络技术；E-mail：yschenchina@qq.com

PM_ CTL algorithm not only effectively reduces time complexity, but also improves the verification performance.

Key words: possibility measure; model checking; common sub-expression; computation tree logic

作为一种智能化、自动化的软硬件装置验证方法，模型检测具有诸多方面的优势，例如具有较高的适用性、准确性与高效性等，正是由于其具有上述的优势，使其在软硬件集成设计等诸多方面深受欢迎，并且获得有关研发单位的大力推广。

现阶段，大量相关专家对模式检测这一个课题展开探讨，取得了一系列进展，Kattenbelt 等[1]在研究过程中阐明基于可能性测度计算树逻辑（PoCTL[2-4]）的检测标记算法，这个算法可准确的检测其有限状态并发系统是否与可能性测度的计算树逻辑公式规范相符，同时还描述了其符合程度，为进一步探讨并发系统的智能化检测原理奠定了基础，但是 PoCTL 标记算法仍然具有不足之处，例如其具有相对较高的时间复杂度，大规模进行使用的难度非常大；还有赵林等[5]在研究过程中，细致的探讨了可能性测度的计算树逻辑公式的复杂性，但是其标记算法具有不足之处，例如验证效率不高，同样无法得到普及推广；有国外学者 Alur[6]在赵林等[5]探讨的前提下，设计得到了优化型的算法，这一个算法主要是通过一种预先模型评估机制来描述公共子表达式，在很大程度上减小了冗余验证的任务量，最终能够获得较高的验证效率，但是这一种算法中所使用的预评估机制需要非常高的额外支出，尽管其能够提高验证效率，然而却未提高模型检测的总体性价比，鉴于这一个方面的原因，这一种方法也很难获得普及推广。此外，还有周从华等[7]、Baier 等[8]、杨晋吉等[9]在研究过程中进一步探讨了“状态空间爆炸”问题。

通过分析当前的大部分模型检测理论结果不难看出，重点是探讨了软硬件系统的唯一性这一问题，即一次模型检测只是可以进行一次验证。但是，在当前软硬件技术急剧前进的新时期，其规模和区域范围不断提高，过去的唯一性检测技术无法满足现实需要。对验证区域范围和规模应用较大情况，当前的模型检测主要是通过“各个击破”的方式进行处理，然而这一种验证方式存在着一定的不足之处，也就是检测效率不高，无法充分满足验证机制的性能需求。综上所述，可能性测度计算树逻辑模型检测验证中存在诸多问题，例如低性能效率和高时间复杂度。针对上述问题，本文在这里主要是基于过去的可能性测度的计算树逻辑模型检测标记算法，阐明了优化的算法，也就是 I-PM_ CTL 算法。这一个算法的基本步骤如下，第一步，先利用相关可能性测度对树逻辑公式进行计算，预处理标识公共子表达式的唯一性，第二步，在充分确保模型检测空间平衡状态下假设公共子表达式与可能性测度计算树逻辑模型状态，第三步，实施验证，为可能性测度计算树逻辑公式以极大概率一次性实现验证提供了保证。利用模拟实验发现，这一种算法一方面在很大程度上减小了相关时间复杂度，另一方面还使验证性能有所提升。

1 相关理论

前提是模型检测机制中单一系统结构模型具有若干特征，在此基础上，我们描述了软硬件系统结构模型，此处 S 是指检测状态集合；M 是指迁移系统；C 指某一可能性测度的计算树逻辑公式集合，同时还存在下列关系：$C = \{\varphi_1,\varphi_2,\cdots,\varphi_m\}(m \in \mathbf{Z}^*)$。由此看出，检测机制即对 M 是不是与 C 中某一个可能性测度的计算树逻辑公式模型相符进行验证，具体来说，也就是 $M,s| = \varphi_i(\varphi_i \in C,s \in S)$。一般来说，其可能性测度的计算树逻辑模型检测标记算法仍然没有验证每一个公共子表达式，鉴于这一个方面的原因，接下来我们主要探讨模型检测机制的关键特征与主要问题。

1.1 标记算法

标记算法是验证计算树逻辑公式（CTL）的传统处理机制[10]，其基本原理是选择某一满足要求的关联组合 $\{\neg, \wedge, \perp, AF, EU, EX\}$，第一步，通过其关联组合对等形式化描述代替某一个可能性测度的计算树逻辑公式 φ_i；第二步，依次检测 φ_i 编序的子表达式的符合性，用满足要求的子表达式来标记迁移系统 M 的有关状态信息，直至检测完 φ_i 编序的子表达式为止；第三步，处理标记的 M 状态集合，确定初始状态是不是与某个可能性测度的计算树逻辑公式 φ_i 的相关条件[11-13]相符。接下来本文将详细进行阐述：

说明 1 用 ψ 代替一个可能性测度的计算树逻辑公式 φ_i 的子表达式，在这里 $\psi \subset \varphi_i$。

说明 2 φ_i 按照结构文法形成一种特定语法树逻辑，叶子结点为 φ_i 的子表达式，其它的都属于关联组合 $\{\neg, \wedge, \perp, AF, EU, EX\}$ 里面的某一种。

说明 3　设多个可能性测度的计算树逻辑公式 φ_i 具有相同的子表达式 ψ，在这种情况下，则 ψ 属于 φ_i 的公共子表达式。

说明 4　设 $\{\neg, \wedge, \perp, AF, EU, EX\}^n$ 为关联组合 $\{\neg, \wedge, \perp, AF, EU, EX\}$ 的 n 元关联词。

说明 5　设 ψ_1, ψ_2 和 φ_i 为已知可能性测度的计算树逻辑公式，假设满足下面的条件，也就是：

$$\varphi_i = \begin{cases} op_u(\psi_1), op_u \in \{\neg, \wedge, \perp, AF, EU, EX\}^1; \\ \psi_1 op_b \psi_2, op_b \in \{\neg, \wedge, \perp, AF, EU, EX\}^2 \end{cases}$$

在这种情况下，则 ψ_1, ψ_2 属于 φ_i 的直接子表达式。

1.2　公共子表达式冗余性验证问题

推论 1　预定可能性测度的计算树逻辑公式 ψ_i 和 φ_i，在这里 ψ_i 为 φ_i 的子表达式，在这种情况下有 $\psi_i \subset \varphi_i$，则验证结果 $M,s| = \varphi_i(\varphi_i \in C, s \in S)$ 一定要符合 $M,s| = \psi_i(\psi_i \in C, s \in S)$。

具体可以通过下面的步骤进行证明：因 $\psi_i \subset \varphi_i$，则可以做出以下表述：$\varphi_i = \theta_1 op_1(\theta_2 op_2, \cdots \theta_i op_i(\psi)\cdots)$，同时 $\theta_k, op_k (0 < k \leqslant i)$ 两者分别为可能性测度的计算树关联词和逻辑公式。设存在满足 $\delta = \theta_i op_i(\psi)$ 的式子，在这种情况下，则 $\delta \subset \varphi$，并且符合说明 5 有关条件。

通过上面的证明结果不难看出，预先验证 $M, s| = \delta_i(\delta_i \in C, s \in S)$ 一定要符合 $M,s| = \psi_i(\psi_i \in C, s \in S)$。

假定 $SAT(\psi)$ 为处理 $M,s| = \psi_i(\psi_i \in C, s \in S)$ 的操作方法，得到的输出值为满足 ψ 的集合，这样可能性测度的计算树逻辑公式标记算法应当自说明 5 中的直接公共子表达式进行验证，则 δ_i 与 ψ 存在一定的关联，具体如下所示：

1) $\delta = \neg \psi, SAT(\delta) = S - SAT(\psi)$；

2) $\delta = \psi \wedge \psi_1, SAT(\delta) = SAT(\psi) \cap SAT(\psi_1)$；

3) $\delta = \psi \vee \psi_1, SAT(\delta) = SAT(\psi) \cup SAT(\psi_1)$；

4) $\delta = \psi \rightarrow \psi_1$，

$SAT(\delta) = SAT(\neg \psi \vee \psi_1) = SAT(\neg \psi) \cup SAT(\psi_1) = (S - SAT(\psi)) \cup SAT(\psi_1)$

5) $\delta = AX\psi$，

$SAT(\delta) = SAT(\neg EX \neg \psi) = \neg SAT_{EX}(\neg \psi) = S - SAT_{EX}(S - SAT(\psi))$

6) $\delta = EX\psi, SAT(\delta) = SAT_{EX}(\psi)$；

7) $\delta = A[\psi U \psi_1]$，

$SAT(\delta) = SAT(\neg (E[\neg \psi_1 U (\neg \psi \wedge \neg \psi_1)] \vee EG \neg \psi_1)) = (S - SAT_{EU}(\neg \psi_1, (\neg \psi \wedge \neg \psi_1))) \vee (S - SAT_{AF}(\psi_1))$

8) $\delta = E[\psi U \psi_1], SAT(\delta) = SAT_{EU}(\psi, \psi_1)$；

9) $\delta = EF\psi, SAT(\delta) = SAT_{EU}(T, \psi)$；

10) $\delta = EG\psi, SAT(\delta) = SAT(\neg AF \neg \psi) = S - SAT_{AF}(\neg \psi)$；

11) $\delta = AF\psi, SAT(\delta) = SAT_{AF}(\psi)$；

12) $\delta = AG\psi$，

$SAT(\delta) = SAT(\neg EF \neg \psi) = S - SAT(EF \neg \psi) = S - SAT_{EU}(T, \neg \psi)$；

通过上面的前 4 个式子发现：获得 $SAT(\psi)$ 之后才可以得到 $SAT(\delta)$ 的输出结果。根据可能性测度的计算树逻辑公式的形式化等价机制[14]，后 8 个式子已由关联组合 $\{\neg, \wedge, \perp, AF, EU, EX\}$ 形式化表示，通过上面的处理机制还能够得知，获得 $SAT(\psi)$ 之后才可以得到 $SAT(\delta)$ 的输出结果。从而获得结果：要是 $\psi \subset \delta$，在这种情况下，则如果想验证 $M,s| = \delta_i(\delta_i \in C, s \in S)$，则必须先对 $M, s| = \psi_i(\psi_i \in C, s \in S)$ 进行验证。

推理 1　预定可能性测度的计算树逻辑公式 φ_1 和 φ_2，假如 $\varphi_1 \cap \varphi_2 = \{\psi_i\}$，$\psi_i$ 为 φ_1 和 φ_2 的公共子表达式，则模型检测标记算法验证过程对 $M, s| = \varphi_1(\varphi_1 \in C, s \in S)$ 与 $M,s| = \varphi_2(\varphi_2 \in C, s \in S)$ 重复验证了 $M,s| = \psi_i(\psi_i \in C, s \in S)$。

推论 2　预定可能性测度的计算树逻辑公式 $\varphi_i = \psi_1 op \psi_2 (op \in \{\neg, \wedge, \perp, AF, EU, EX\}^2)$，假如 $\delta \subset \psi_1, \delta \subset \psi_2$，在这种情况下，则模型检测标记算法验证 $M,s| = \varphi(\varphi \in C, s \in S)$ 时必然冗余验证了 $M,s| = \delta(\delta \in C, s \in S)$。

经由上文的分析看出，当前的可能性测度的计算树逻辑公式在模型检测标记算法验证机制中仍然存在或多或少的不足之处，特别是公共子表达式冗余验证这一个问题。

2　I-PM_ CTL 算法

2.1　公式预处理机制

针对上文中提出的当前使用的模型检测验证时具有公共子表达式冗余验证的弊端，利用公式对标识公式集中的公共子表达式进行预处理，验证时将模型状态和公共子表达式两者绑定，科学合理描述公共子表达式，防止了重复验证。所以，为充分确保可能性测度的计算树逻辑公式的公共子表达式模型检测验证中只处理一次，我们通过语法树逻辑结构的扩展机制表达可能性测度的计算树逻辑公式[15-16]。

假定可能性测度的计算树逻辑公式的标准语法树逻辑结构为 $T < \mathrm{root_T} >$，在这种情况下，则其扩

展语法树逻辑结构 $S < \text{root}_S >$ 的产生步骤如下所示：

（i）初始化 $S < \text{root}_S >$，在这里，也就是 $\text{root}_S = \text{null}$；

（ii）对 $T < \text{root}_T >$ 实施后序遍历处理，将某一结点 N 输出，当这一个结点 N 为空时，在这种情况下，那么就继续进行步骤（v），否则，将继续进行步骤（iii）；

（iii）要是步骤（ii）输出的 N 是唯一的，同时满足条件 $N \not\subset S$，那么就会新形成结点 M；否则，要是 N 为唯一的，同时满足条件 $N \subset S$，在这种情况下，则继续进行步骤（ii）；除此之外，要是 N 为非唯一的，在这种情况下，则 N 属于关联词，接着进行步骤（iv）；

（iv）要是步骤（ii）输出的 N 为中间结点，还满足条件 $N \subset S$，并且根结点为 N 时和 S 的子树匹配，在这种情况下，则 N 为这个语法分析树的公共子表达式；否则，会新形成结点 M，同时自 T 追溯至 S 的子结点，利用关联词加以标识，继续进行步骤（ii）；

（v）要是 $\text{root}_S = M$，在这种情况下，则输出 S 的根结点信息，也就是 root_S。

下面将列举示例可能性测度的计算树逻辑公式，具体如下所示：

$$A[AX\neg\, pUE[EX(p \wedge q)U\neg\, p]] \quad (1)$$

$$EF[(AX\neg\, p) \wedge EX(p \wedge q)] \quad (2)$$

这样能够得出，上面两个式子公共子表达式构建的结点集合为 $\{p, q, \neg\, p, p \wedge q, EX(p \wedge q), AX\neg\, p\}$，在这里 p 和 $p \wedge q$ 分别为 $\neg\, p$ 和 $EX(p \wedge q)$ 的直接子表达式。

可能性测度的计算树逻辑公式的公共子表达式存在于任何子公式里面，怎样完成对多次存在的公共子表达式的一次检测，防止出现冗余检测已经发展成新型语法树逻辑结构的构造重点。我们在这里主要是利用预处理机制来实现唯一性标识，对多次存在冗余的可能性测度的计算树逻辑公式语法树逻辑结构进行组合，确保能够统一检测其冗余公共子表达式。

见图 1，其是式（1）和式（2）两者进行组合所形成的扩展性语法树逻辑结构。

2.2 I-PM_ CTL 模型检测机制

I-PM_ CTL 算法的输入参数主要是由两个方面组成，具体来说，也就是 S 的扩展性语法树逻辑结构与 M，在这里，就一切可能性测度的计算树逻辑公式 φ_i 来说，利用这一个算法处理机制后输出值

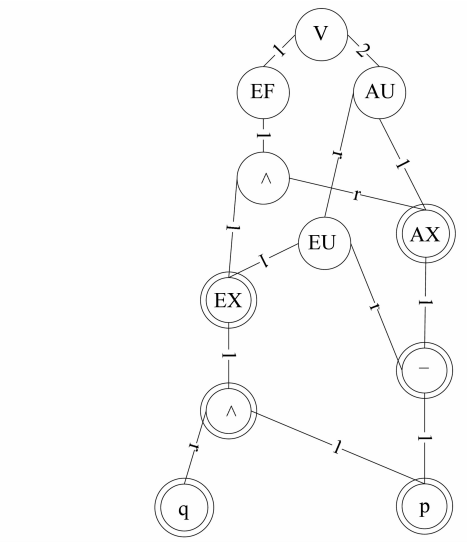

图 1 公式（1）与（2）组合扩展性语法树逻辑结构

Fig. 1 Formula (1) and (2) combination of scalability syntax tree logical structure

为 $M, s \mid = \varphi_i (\varphi_i \in C, s \in S)$ 的验证结果。和过去的算法存在差异，这一种改进的算法在开展模型检测验证以前新增了预处理机制，确定其可能性测度的计算树逻辑公式的子表达式直接与否、公共与否，为各可能性测度的计算树逻辑公式的子表达式实施唯一性标识，要是验证了公共子表达式，在这种情况下，那么后续验证时将不会重新对其实施验证操作。为使其性能效率有所增加，我们所提出的算法检测了各个可能性测度的计算树逻辑公式以后，将会把非公共子表达式的唯一性标识删除。

假定 φ 为 φ 的子表达式，在这种情况下，那么满足 $\varphi \subset \varphi$，同时 ψ 为 φ 的直接子表达式，要是预处理了全部的 ψ，对其直接子表达式实施了唯一性标识，则我们提出的优化算法检测 φ 具体步骤如下：

（i）判断 φ 是不是实施了唯一性标识，如果是，那么则继续步骤（ii），如果不是，那么则继续步骤（iii）；

（ii）判断 φ 是不是实施 $M, s \mid = \varphi_i (\varphi_i \in C, s \in S)$ 验证，如果是，那么则继续步骤（v），如果不是，那么则继续步骤（iii）；

（iii）根据关联词和预处理机制中的唯一性标识信息，通过我们设计的优化算法搜索满足要求的 φ 状态，接着进行步骤（iv）；

（iv）要是 φ 为可能性测度的计算树逻辑公式的公共子表达式，在 M 中实时对应其 φ 的状态信息，对扩展性语法树逻辑结构中 φ 唯一性标识实施

检测操作，然后进行步骤（v）；

（v）要是 φ 和 φ 两者等价，在这种情况下，则将其非公共子表达式的唯一性标识删除，否则，将 φ 唯一性标识状态信息输出。

在这里，我们具体描述一下 I-PM_ CTL 算法，接下来我们将说明其伪代码检测机制：

Input：Kripke * module，Tree * ctl_ tree，Formular * f；

// 系统模型 M，PoCTL 公式扩展性语法树逻辑结构，待验证的子表达式 f；

Output：Kripke * module；// 标识 f 与状态之后的系统模型

1）global Formular * g [ ] = ExtractFromTree (ctl_ tree)；

// 从扩展性语法树逻辑结构产生验证的子表达式集合 g，子表达式 $f \in g$

3　实验分析

3.1　准备工作

为验证我们设计的优化算法的优点，同时为了对其性能结果和应用范围进行测试，将通过 VC 与 OpenGL 平台等相关工具开展模拟相关标记算法。利用 IDA Pro 工具静态处理与分析 $M = (S, \rightarrow, L)$ 系统模型，检验其对可能性测度的计算树逻辑算法形成的操作程序，得到状态集合 S = {blk1，blk2，...，blkn}，在这里预处理机制唯一性标识集合主要包括 M 模型调用参数，通过 L = {emt，syscall1，syscall2，...，syscallm} 来进行描述，此处 emt 则表示没有 M 模型调用参数能力。除此之外，可能性测度的计算树逻辑公式验证特征基本上利用 M 模型调用的时序关系得到有关信息，下面各种序号对应的可能性测度的计算树逻辑公式集合不同。

（i）EF （ （callset = GetModuleFileName） &EF (callset = WriteFile))；

（ii）EF （((callset = FindFileFirst) |(callset = FindFileFirstEx)) &

EF （callset = FindNextFile))

（iii）EF （callset = GetSystemDirectory→

(EF(callset = CopyFile)|EF(callset = MoveFile)))

3.2　模拟实验

做好前面的准备工作后，接着利用模拟实验测试各种可能性测度的计算树逻辑公式集合，具体情况见表 1。

表 1　各种不同实验测试结果

Table 1　different kinds of test results

程序（exe）	状态量	迁移量	I-PM_ CTL 公式量	公共子表达式拥有量	处理时间周期/s	传统标记算法 LA	I-PM_ CTL 算法
Exl	523	2 204	1	0	0. 128	0. 778	0. 379
Exl	523	2 204	3	37. 4%	0. 344	2. 103	1. 072
Exl	523	2 204	5	71. 8%	0. 472	3. 983	1. 891
Exl	523	2 204	7	13. 3%	0. 619	5. 549	3. 735
Exl	523	2 204	9	52. 1%	0. 823	7. 349	5. 922
Exl	523	2 204	10	3. 9%	0. 986	8. 836	7. 138

通过表 1 看出，就同一可执行程序 Exl 来说，各个数量的可能性测度的计算树逻辑公式集合能够获得的实验结果存在着一定的差异。通过分析可以看出，当 M 一样的时候，假定可能性测度的计算树逻辑公式的公共子表达式拥有量较低时，过去的算法性能效率和处理时间周期相对较低[1-2]，相反地，假定公共子表达式拥有量较高时，其模型检测验证机制性能与公共子表达式的符合性不断增大。除此之外，能够看出，与传统标记算法相比较来说，本文涉及的优化算法使用的处理时间周期相对较短，同时具有相对较高的性能效率。为了在同一系统模型 M 与可能性测度的计算树逻辑公式集合的前提下分析对比两种算法的模型检测验证机制性能，我们利用多组程序系统模型展开研究，确保 ≤10 M 字节，同时都从电脑系统目录下的 exe 文件而来的。具体见图 2 和图 3。

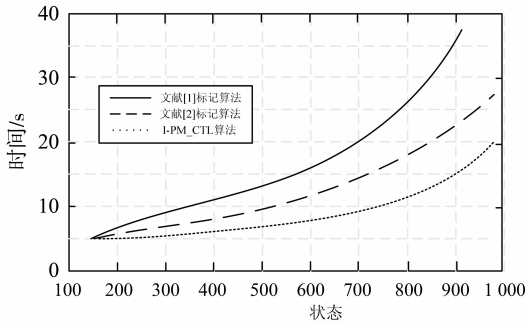

图 2 程序状态与模型检测时间周期对比图

Fig. 2 The program state and model checking time comparison chart

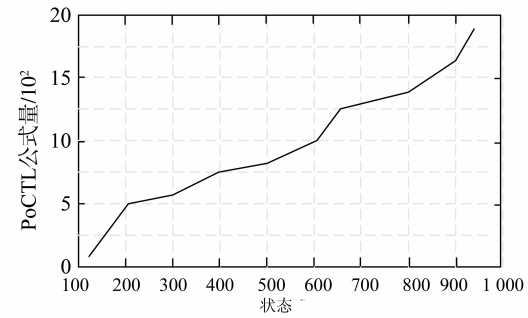

图 3 I-PM_ CTL 算法性能效率变化图

Fig. 3 I-PM_ CTL algorithm performance variation chart

通过图 2 可以看出，系统模型的可执行程序状态量愈大，I-PM_ CTL 模型检测算法的性能效率愈好，与 Kattenbelt 等[1]和邓辉等[2]提出的算法进行比对来说，前者具有非常明显的性能优势。通过图 3 我们能够看出，I-PM_ CTL 模型检测算法伴随可能性测度的计算树逻辑公式数量逐渐提高，其性能效率同样在不断增加，两者之间存在正相关性，通过实验可知，在高复杂性、大规模的可能性测度的计算树逻辑公式检测处理的条件下，我们设计的改进算法依旧可以通过相对偏高的性能效率实现相关验证工作，因此我们所设计的优化算法能够在大规模 CTL 公式检验中应用。

4 结 语

为妥善应对可能性测度的计算树逻辑模型检测验证中的低性能效率和高时间复杂度等方面的不足之处。我们设计出一种优化的算法，也就是 I-PM_ CTL 模型检测算法。其基本步骤如下：第一步，先利用相关可能性测度对树逻辑公式进行计算，预处理标识公共子表达式的唯一性；第二步，在充分确保模型检测空间平衡状态下假设公共子表达式与可能性测度计算树逻辑模型状态；第三步，实施验证，检验可能性测度计算树逻辑公式是否以极大概率一次性实现验证。利用模拟实验发现，这一种算法一方面在很大程度上减小了相关时间复杂度，另一方面还使验证性能有所提升。通过分析看出，我们所设计出的优化算法能够在高复杂性、大规模环境中应用。

参考文献：

[1] KATTENBELT M, KWIATKOWSKA M, NORMAN G, et al. A game-based abstraction refinement framework for Markov decision processes [J]. Formal Methods in System Design, 2010, 36(3): 246 – 280.

[2] 邓辉,薛艳,李亚利,等. 基于可能性测度的计算树逻辑 CTL＊与可能性互模拟[J]. 计算机科学, 2012, 39(10): 258 – 263.

[3] 盛景军, 王 晴, 侯立峰, 等. 基于 Pareto 适应度的混合遗传算法在多式联运问题中的应用[J]. 西南师范大学学报:自然科学版, 2012, 37(9): 43 – 47.

[4] 王仕平, 蒋 玲, 熊 江, 等. 一种基于序列数的关联规则挖掘算法[J]. 西南大学学报:自然科学版, 2011, 33(3): 122 – 127.

[5] 赵林,吴尽昭. 基于吴方法的多值模型检验[J]. 系统科学与数学, 2008, 28(8): 1020 – 1029.

[6] ALUR R. Model checking: from tools to theory [J]. Lecture Notes in Computer Science, 2008, 5000: 89 – 106.

[7] 周从华, 刘志锋, 王昌达. 概率计算树逻辑的限界模型检测[J]. 软件学报, 2012, 23(7): 1656 – 1668.

[8] BAIER C, KATOEN J P. Principles of model checking [M]. Cambridge: MIT Press, 2008: 745 – 907.

[9] 杨晋吉, 苏开乐, 骆翔宇,等. 有界模型检测的优化[J]. 软件学报, 2009, 20(8): 2005 – 2014.

[10] 陈志远, 黄少滨, 白玉,等. 模型检测中的 CTL 形式化描述模板[J]. 哈尔滨工程大学学报, 2013, 34(4): 483 – 487.

[11] DOVIER A, QUINTARELLI E. Applying model checking to solve queries on semistructured data [J]. Computer Languages: Systems and Structures, 2009, 35: 143 – 172.

[12] BARBUTI R, LEVI F, MILAZZO P, et al. Probabilistic model checking of biological systems with uncertain kinetic rattes [J]. Theoretical Computer Science, 2012, 419: 2 – 16.

[13] LI L J, LI Y M. Model-checking of linear-time properities in possibilistic kripke structure [C]// Proceedings of the QL&SC 2012, World Scientific, 2012: 287 – 294.

[14] 孙志安, 裴晓黎, 宋听. 软件可靠性工程[M], 北京: 北京航空航天大学出版社, 2009: 1 – 9.

[15] HARRISON J. Theorem proving for verification [C]// Proceedings of the 20th International Conference on Computer Aided Verification. Princeton, USA, 2008: 11 – 18.

[16] HOLZMANN G. The SPIN model checker: primer and reference manual [M]. Upper Saddle River: Prentice Hall, 2011: 326 – 346.