基于切片技术的复杂模型验证方法

纪明宇 于明霞 么一诺 毕曦文
(东北林业大学信息与计算机工程学院 黑龙江 哈尔滨 150040)

摘 要 针对模型检测中的状态空间爆炸问题,提出一种基于带资源消耗的复杂概率迁移系统模型的性质验证方法。使用 PRCTL 进行模型语义逻辑表示,对迁移步数约束采用基于栈的深度遍历算法完成检验,针对冗余的迁移和状态实施二次约减,完成复杂的概率迁移系统的模型简化;针对简化后的迁移图进行概率矩阵迭代运算以判定概率条件,同时完成资源条件的判定,并进行相应的实例分析。该模型验证方法能够在一定程度上缓解复杂系统带来的状态冗余和计算繁琐问题,为性质验证工作提供了新的思路。

关键词 复杂概率系统 模型检测 概率矩阵 状态约减 反例路径

中图分类号 TP301.2 **文献标志码** A DOI:10.3969/j.issn.1000-386x.2020.09.031

COMPLEX MODEL VERIFICATION METHOD BASED ON SLICING TECHNIQUE

Ji Mingyu Yu Mingxia Yao Yinuo Bi Xiwen
(*School of Information and Computer Engineering, Northeast Forestry University, Harbin 150040, Heilongjiang, China*)

Abstract Aiming at the problem of state space explosion in model detection, this paper proposes a method for property verifications based on a complex probability transition system model with resource consumption. It used PRCTL to represent the semantic logic of the model and adopted a stack-based depth traversal algorithm to complete the verification of migration step constraints. The model of complex probability transition system was simplified by quadratic reduction for redundant migration and state. Then, the probability matrix iterative operation was carried out for the simplified migration graph to determine the probability conditions, and the resource conditions were determined at the same time. At last, related case analysis was conducted in this paper. The system verification method can alleviate the state redundancy and complex computation problem caused by the complex system to a certain extent. And it provides new ideas for the property verification work.

Keywords Complex probability system Model checking Probability matrix State reduction Counter-example path

0 引 言

随着科技发展的日新月异,计算机软硬件系统趋于复杂化和抽象化使得其优劣性质验证成为极为重要的内容。模型检测作为一种能够自动验证并提出反例路径的形式化验证技术,已被广泛应用于系统验证。其基本思想是利用模型描述语言对已有系统进行建模,之后对该模型进行既定性质验证,若满足性质则进行下一项验证,否则提出反例路径。常用的形式化方法主要有线性时序逻辑(LTL)和计算树逻辑(CTL)等[1]。典型的模型检测工具有 NuSMV[2]、SPIN[3]等。互模拟的引入使得 LTL 模型检测能够在验证混合自动机时具有可判定性[4]。广义 CTL 逻辑的讨论能够引发对模型检测的更广泛应用的思考[5]。目前在模型检测领域,验证技术已越来越成熟,其使用的手段如基

收稿日期:2019-06-03。国家自然科学基金项目(61806049);中央高校基本科研业务费专项资金项目(2572015CB32)。
纪明宇,副教授,主研领域:形式化验证,自然语言处理,机器学习。**于明霞**,本科生。**么一诺**,本科生。**毕曦文**,本科生。

于时序逻辑的规范化语言、自动机和不动点逻辑已为模型检测领域的发展作出极大贡献[6]。

状态空间爆炸问题始终是模型检测技术中需要解决的一大难点。在建立模型时所需考虑的状态数量往往呈指数增长,最终难以控制模型的大小以致增大系统验证的复杂度,解决该问题的思想包括状态空间压缩[7-8]、存储空间压缩的 Hash Compact 方法[9]、并行和分布式计算[10]、随机化和启发式搜索算法[11]等。近年来逐渐得到发展的限界模型检测和切片技术等多种约减方法能够在一定程度上解决有限马尔可夫链的状态爆炸问题。如:基于线性方程组求解的限界模型检测算法在反例较短的情况下能够快速完成检测过程并避免空间爆炸[12];基于懒惰切片提出的状态空间搜索方法,在截断状态基础上进行多次细化迭代,有效避免基于 CEGAR 的切片方法导致的重复计算[13];基于状态约减的信息攻防图生成算法在实现目标网络中主机覆盖的基础上描述了控制状态空间组合爆炸问题的过程[14];冗余路径的点约减方法探究了有效简化和优化 USV 在线路径的相关问题[15]等。

然而,上述性质验证和约减方法很少涉及对带资源消耗的复杂迁移系统进行性质验证。本文将在已有研究工作的基础上,结合改进的图的深度遍历算法和概率矩阵的迭代运算,提出针对复杂模型的性质检验方法。首先利用栈的概念对系统进行初始的步数约束检验,对系统完成约减以避免冗余状态和迁移的影响,再计算得出约减后的状态间概率和资源消耗反例路径,从而完成满足带资源消耗约束的直到性质验证过程。

1　背　景

迁移系统在复杂系统的模型性质验证领域,常用来作为模型来描述系统行为。

定义 1　迁移系统(TS)。迁移系统是基本的有向图,其中点表示状态,边模仿迁移,即状态变化。状态描述了某个确定时间的系统行为的信息。迁移系统 TS 可以表示为元组 $T=(S,Act,\rightarrow,AP,L)$,其中:$S$ 是状态集合;Act 是动作集合;$\rightarrow$是迁移关系;AP 是原子命题集合;标签函数为 $L:S\rightarrow 2^{AP}$。

如果 S、Act、AP 有限,那么 TS 被称为有限的迁移系统。由于基本的迁移系统模型在随机性和确定性的表达能力上存在不足,所以出现了连续型和离散型的不同马尔可夫模型用以描述待验证的复杂随机系统[16]。两种模型分别具备描述在离散时间和连续任意时间的状态迁移的能力,本文针对具有离散时间特征的系统模型展开研究。

定义 2　离散时间马尔可夫链(DTMC)。离散时间马尔可夫链可以表示为元组 $M=(S,P,L,AP,v)$,S 是状态的非空集合;$P:S\times S\rightarrow[0,1]$是迁移概率函数,且对所有状态 s:$\sum_{s'\in S}P(s,s')=1$;$L:S\rightarrow 2^{AP}$为状态标记函数;AP 表示原子命题的有限集;$v:S\rightarrow[0,1]$是初始分布,且 $\sum_{s\in S}v(s)=1$。

例 1　DTMC 举例。

图 1 为包含 9 个状态的双骰子投掷模型对应的不带资源消耗的离散时间马尔可夫模型。该模型描述了以下情形:投掷结果骰子数字和为 7 或 11 时为获胜,系统迁移到达 won 状态,骰子数字和在{2,3,12}中时到达 last 状态,骰子数字和为其他结果时则记为“points”,并继续投掷直到和为 7 时达到 last 状态或和为“points”时达到 won 状态。图 1 模型忽略了各状态所满足的原子命题。

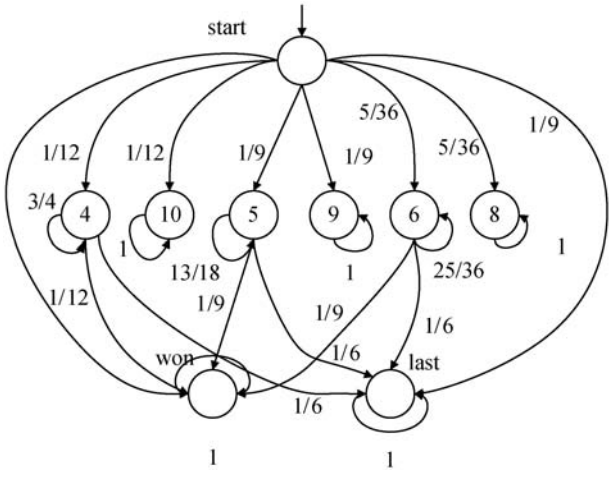

图 1　DTMC 举例

本文以 DTMC 为基础,针对带迁移资源消耗的离散时间马尔可夫回报模型(DTMRC)展开研究。

定义 3　离散时间马尔可夫回报模型(DTMRC)。DTMRC 为五元组 $M=(S,P,L,AP,N,v)$,其中:S、P、L、AP、v 在定义 2 中已有说明;$N:S\times S\rightarrow R_{\geqslant 0}$为迁移回报矩阵结构($R$ 为迁移回报集合),描述了从初始到各状态的迁移资源;$P_{s_1,n,s_2}=0.6$ 表示从 S1 迁移到 S2,发生迁移时所消耗的资源为 n 的概率为 0.6。

例 2　DTMRC 举例。

图 2 为带有资源消耗的马尔可夫模型,其包含 8 个状态以及 12 个迁移过程。S0 为初始状态,包含原子命题{a,c}。对于状态 S0 来说,在满足原子命题 a 和 c 的条件下,下一步可迁移状态集为{S1,S2,U},发

生迁移的概率分别为 0.5、0.3 和 0.2，消耗的资源分别为 10、10 和 15。

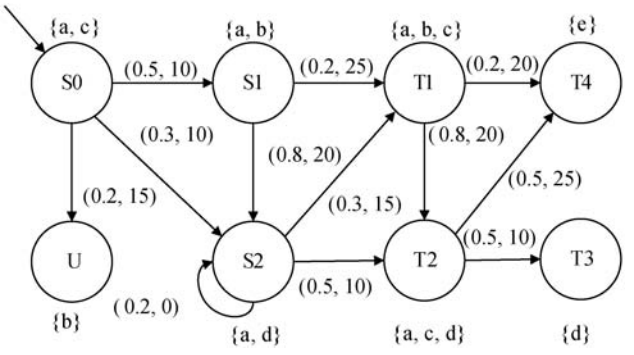

图 2 DTMRC 举例

DTMC 模型一般可通过概率计算树逻辑（PCTL）来描述验证的性质。

定义 4 概率计算树逻辑（PCTL）。基于原子命题集合 AP 的 PCTL 状态公式根据以下语法被形式化：$\phi ::= \text{true} \mid a \mid \phi_1 \wedge \phi_2 \mid P_J(\varphi)$。其中：$a \in AP$，$\varphi$ 是路径公式；$J \subseteq [0,1]$ 是有理数界的中间值；$P_J(\varphi)$ 表示当前状态满足公式 φ 的概率值为 J。

PCTL 路径公式可以根据以下语法被形式化：$\varphi ::= O\varphi \mid \phi_1 \cup \phi_2 \mid \phi_1 \cup^{\leqslant n} \phi_2$，其中 ϕ、ϕ_1、ϕ_2 是状态公式，$O\phi$ 表示最终进入满足状态公式 ϕ，$n \in \boldsymbol{N}$。

由于 PCTL 模型检测的语法并不能描述迁移系统中资源消耗情况，所以对于 DTMRC 模型的性质，一些学者提出了改进的概率回报计算树逻辑 PRCTL[17]，它与 PCTL 的区别在于路径公式的语义表达方式有所不同。

PRCTL 的路径公式表示为：

$$\varphi ::= \phi \mid \varphi_1 \wedge \varphi_2 \mid \neg\ \varphi \mid \varphi_1 \cup_r^n \varphi_2$$

式中：$n = [n_1, n_2] \subseteq R_{\geqslant 0}$ 和 $r = [r_1, r_2] \subseteq R_{\geqslant 0}$ 分别代表迁移步数和迁移资源消耗约束，U 代表直到算子。

例如 $P_{\leqslant p}(\mathrm{a} \cup_{\leqslant t}^{\leqslant n} \mathrm{e})$ 的带有资源与步数条件的直到概率公式，其表示为 n 步之内从满足原子命题 a 的状态迁移到满足原子命题 e 的状态的概率和小于等于 p 且资源消耗不超过 t（p、t 分别表示任意大于等于 0 的实数），且要求除最终状态外，其余状态皆需满足原子命题 a。

例如 $P_{\leqslant 0.2}(\mathrm{a} \cup_{\leqslant 100}^{\leqslant 3} \mathrm{e})$ 表示三步之内，从满足原子命题 a 的状态迁移至满足原子命题 e 的状态，该路径的概率和小于等于 0.2 且路径资源消耗和不超 100。

2 验证过程

针对 DTMRC 模型下形式为 $P_{\leqslant p}(\mathrm{a} \cup_{\leqslant t}^{\leqslant n} \mathrm{e})$ 包含步数条件、概率条件和资源条件的直到性质验证过程，说明如下：

对于待验证步数约束条件的 DTMRC 模型，可以将其看作带有冗余的状态和迁移的复杂系统。（1）利用改进的图的深度遍历算法，得出符合步数的路径集。针对进行模型检测时反复出现的状态爆炸问题，通过基于切片技术的约减方法来进行系统的缩减。首先将超过步数约束的迁移完全省去，实现一次约减；再检查剩余的状态是否包含在满足直到条件的路径中，若不包含则将这样的状态同时省去，实现二次约减，得到一个较为简单的迁移系统。（2）对于新的迁移系统构造各状态间的迁移概率矩阵，进而通过概率迭代运算，计算出 n 步内 a∪e 的概率和。

定义 5 迁移系统概率矩阵迭代计算公式为：

$$\boldsymbol{X}^{(i+1)} = \boldsymbol{A}\boldsymbol{X}^{(i)} + \boldsymbol{b} \quad 0 \leqslant i \leqslant n$$

$$\boldsymbol{X}^{(0)} = 0 \qquad i = 0$$

式中：$\boldsymbol{A}$ 表示为迁移系统的各状态间初始概率矩阵；$\boldsymbol{b}$ 表示为系统状态到结束状态的一步可达概率矩阵；$\boldsymbol{X}^{(i)}$ 表示为第 i 步的迭代后的概率矩阵。

例 3 n 步可达概率计算举例。

以图 1 中的 DTMRC 模型为例，计算从状态 start 到状态 won 的 n 步内可达概率：

首先得出初始概率矩阵：

$$\boldsymbol{A} = \frac{1}{36} \times \begin{array}{c|cccccccc} & \text{star} & 4 & 5 & 6 & 8 & 9 & 10 \\ \text{star} & 0 & 3 & 4 & 5 & 5 & 4 & 3 \\ 4 & 0 & 27 & 0 & 0 & 0 & 0 & 0 \\ 5 & 0 & 0 & 26 & 0 & 0 & 0 & 0 \\ 6 & 0 & 0 & 0 & 25 & 0 & 0 & 0 \\ 8 & 0 & 0 & 0 & 0 & 25 & 0 & 0 \\ 9 & 0 & 0 & 0 & 0 & 0 & 26 & 0 \\ 10 & 0 & 0 & 0 & 0 & 0 & 0 & 27 \end{array}$$

系统状态到结束状态 won 的一步可达概率矩阵：

$$\boldsymbol{X}^1 = \boldsymbol{b} = \frac{1}{36} \times \begin{array}{c|c} & \text{won} \\ \text{star} & 8 \\ 4 & 3 \\ 5 & 4 \\ 6 & 5 \\ 8 & 5 \\ 9 & 4 \\ 10 & 3 \end{array}$$

然后对一步可达矩阵进行迭代，求解出两步可达矩阵为：

$$\boldsymbol{X}^2=\frac{1}{36}\times\begin{array}{c|cccccc}
 & \text{star} & 4 & 5 & 6 & 8 & 9 & 10\\
\text{star} & 0 & 3 & 4 & 5 & 5 & 4 & 3\\
4 & 0 & 27 & 0 & 0 & 0 & 0 & 0\\
5 & 0 & 0 & 26 & 0 & 0 & 0 & 0\\
6 & 0 & 0 & 0 & 25 & 0 & 0 & 0\\
8 & 0 & 0 & 0 & 0 & 25 & 0 & 0\\
9 & 0 & 0 & 0 & 0 & 0 & 26 & 0\\
10 & 0 & 0 & 0 & 0 & 0 & 0 & 27
\end{array}\times$$

$$\frac{1}{36}\times\begin{matrix}8\\3\\4\\5\\4\\3\end{matrix}\times 5+\frac{1}{36}\times\begin{matrix}8\\3\\4\\5\\4\\3\end{matrix}\times 5=\frac{1}{36^2}\times\begin{array}{c|c}
 & \text{won}\\
\text{star} & 388\\
4 & 189\\
5 & 248\\
6 & 305\\
8 & 305\\
9 & 248\\
10 & 189
\end{array}$$

则 Pstar 到 Won 两步之内可达 $\frac{388}{36^2}$。

三步可达矩阵为:

$$\boldsymbol{X}^3=\frac{1}{36^3}\times\begin{array}{c|c}
 & \text{won}\\
\text{star} & 16\ 536\\
4 & 8\ 991\\
5 & 11\ 632\\
6 & 14\ 105\\
8 & 14\ 105\\
9 & 11\ 632\\
10 & 8\ 991
\end{array}$$

则 Pstar 到 Won 两步之内可达 $\frac{16\ 536}{36^3}$。以此类推:Pstar 到 Won 两步之内可达 $\frac{661\ 300}{36^4}$,Pstar 到 Won 两步之内可达 $\frac{1\ 653\ 255\ 047\ 766}{36^5}$。

最后,基于资源消耗反例路径的思想,根据每个迁移上所带的资源数量,对已满足概率条件和步数条件的迁移路径,通过遍历计算出各路径的资源消耗以寻找反例集。若某路径集的资源消耗和超过约束值,表示其为一个反例。

3　算法描述

本节对上节涉及的 2 个算法进行详细说明:

算法 1　基于图的路径寻找算法 getPaths

输入:DTMRC 模型 M 和步数约束 m, 当前的起始节点 cNode, 当前起始节点的上一节点 pNode,最初的起始节点 sNode,中间节点需要满足的原子命题 a,最终节点需要满足的原子命题 e。

输出:DTMRC 模型 M 中所有满足步数约束的路径集。

```
getPaths(M,m,cNode,pNode,sNode,a,e){
1   nNode <- null;
2   n <- stack length;          /*找到路径上的节点数*/
3   sNode ⊨ a;
4   if(cNode != null and pNode != null and cNode == pNode)
                                /*出现环路*/
5   return false;
6   if(cNode != null){
7     i <-0;                    /*起始节点入栈*/
8     if(cNode ⊨ e)
9     {if(n <= m+1)
10    showAndSavePath;
                  /*满足步数的路径转储并打印输出*/ }
11   else{If(cNode ⊨ a){
12    nNode <- cNode. getRelationNodes
 /*从与 cNode 有连接关系的节点集中得到一个节点作为
                        下一次递归寻路时的起始节点*/
13    while(nNode != null and nNode ⊨ a){
14    if(pNode != null and (nNode == sNode or nNode ==
pNode or isNodeInStack))
   /*产生环路,重新寻找与 cNode 有连接关系的节点*/
15    {i++;
16      if(i >= cNode. getRelationNodes size)
17        nNode <- null;
18    else nNode <- cNode. getRelationNodes of i;
19    continue; }
20    if(getPaths(M,m,nNode,cNode,sNode,a,e))
                                /*递归调用*/
21    stack. pop();          /*找到路径弹出栈顶节点*/
22    i++;
23    if(i >= cNode. getRelationNodes size)
24    nNode <- null;
25    else nNode <- cNode. getRelationNodes of I;
}}
26    stack pop;
27    return false; }}}
 /*以 cNode 为起始节点到终点的路径已经全部找到*/
28    else return false;}                /*算法结束*/
```

算法 2　面向资源消耗的反例路径寻找算法 getCE

输入:PR[n]为存储满足步数约束和概率约束的路径及其消耗资源数的结构体数组,R 为资源消耗限制数。结构体为:

```
structlujing{
  String P;          //把路径上的状态节点看成字符串
  int r;}                          //路径消耗的资源数
```

输出:反例路径集及其消耗的资源和。

```
getCE(PR[n],R){
1   unsigned int sta = 1;
```

```
2  inti,i1,i2,j,m=0,A[1000]={-1};
3  for(j=1;j<=(int)pow(2,n+1)-1;j++)
4  {for(i=0;i<n+1;i++)
5   {if((j&sta)!=0)
/*一条路径与一个二进制位相对应,二进制位不为0对应
                                    的路径存在*/
6    {m=m+PR[i].r;
7      A[i1]=i;  /*记录路径在结构体数组的位置*/
8      i1++;}
9    sta=sta<<1;          /*sta对应的二进制加一*/}
10 if(m>=R)
11 {i1=0;
12  while(i1<n+1)
13  {if(A[i1]!=-1)
14    {i2=A[i1];
15      printf("%s",PR[i2].P);}
16      i1++;}
17  printf("%d",m);}
18 m=0;i1=0;
19 A[1000]={-1};
20 sta=1;}
21 return 0;}                          /*算法结束*/
```

算法 1 的时间复杂度主要与待验证公式中的步数约束 m 有关,而算法 2 的时间复杂度主要与 PR[n]结构体数组的大小有关。算法 1 为算法 2 提供了满足步数约束的反例集。

4 实例分析

下面通过约减图 2 中的 DTMRC 模型来说明验证过程,所要验证的性质为 $P_{\leqslant 0.2}(a \cup_{\leqslant 100}^{\leqslant 3} e)$。

首先利用算法 1 进行步数条件的检验。得出满足步数约束的路径集 $\{S_0S_2T_1T_4, S_0S_2T_2T_4, S_0S_1T_1T_4\}$。一步、两步和三步遍历过程如图 3、图 4 和图 5 所示。然后进行冗余状态和迁移的二次约减,图 6 为满足步数约束的迁移图,图 7 为完成三步约减后的迁移系统。

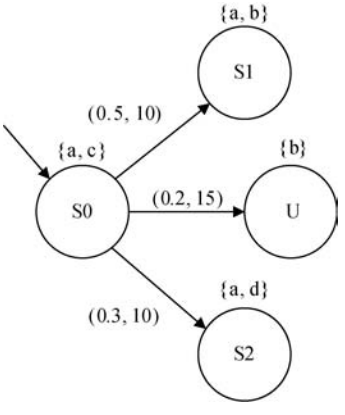

图 3 一步遍历过程

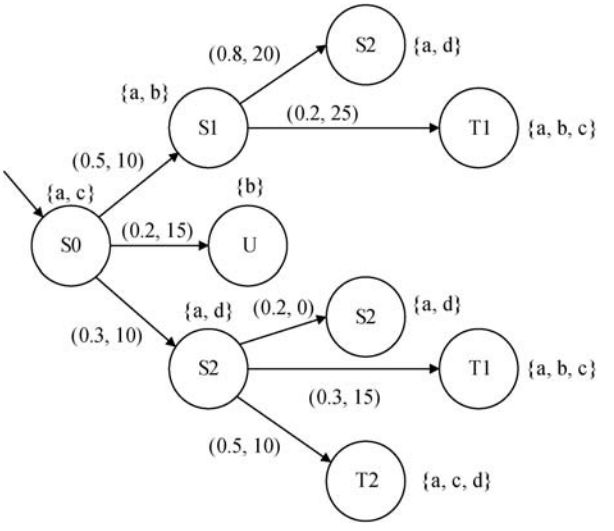

图 4 二步遍历过程

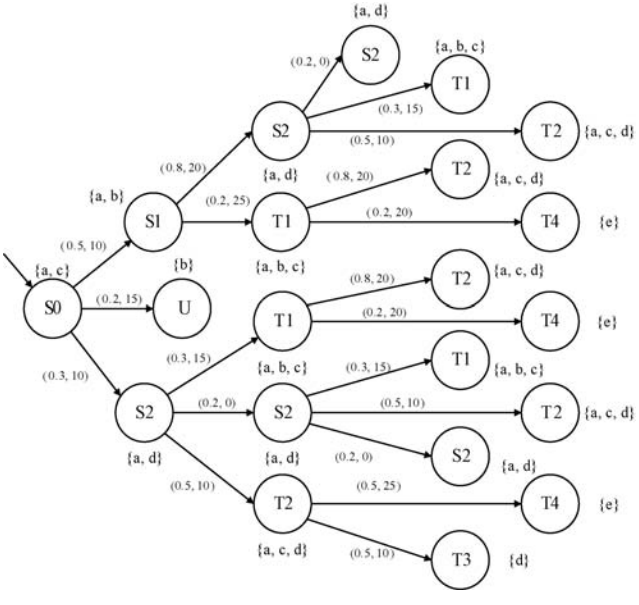

图 5 三步遍历过程

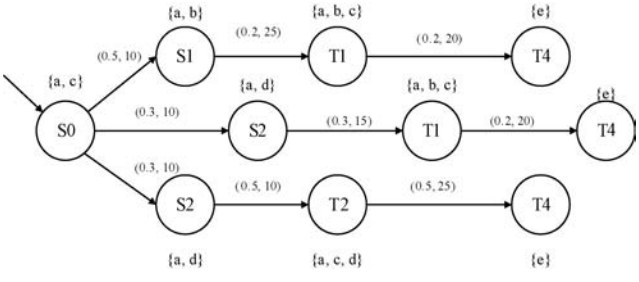

图 6 满足约束的迁移图

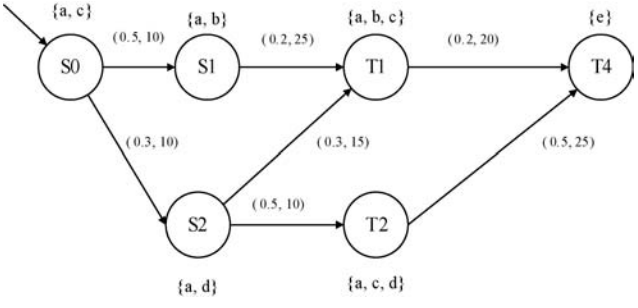

图 7 三步约减后的迁移系统

针对新的迁移系统进行矩阵运算,求出从状态 S_0 到 T_4 的 3 步之内可达且满足 a∪e 的路径概率。经计算可得 S_0 到 T_4 的 3 步可达概率为 0.113,由此可知路径集满足概率小于等于 0.2 的条件。最后,利用算法 2 对图 7 所示新的迁移系统进行资源消耗路径集求解,把符合步数条件和概率条件的路径集{$S_0S_2T_1T_4$,$S_0S_2T_2T_4$,$S_0S_1T_1T_4$}输入,求得所消耗的路径资源和为 145。此时不满足小于等于 140 的资源约束条件,则该路径集为一个反例集合。

本文采用实验对相应的方法进行了有效性验证,实验硬件环境为 Intel(R) Core(TM) i5-7200U CPU @ 2.50 GHz 2.70 GHz,内存为 8 GB。软件环境包括操作系统为 64 位的 Windows 7,编程软件用 MyEclipse 10.7。本文提出的方法可一定程度用于随机分布式算法、安全协议分析等多个领域,用于以数学分析的角度进行可达性分析验证。

5　结　语

本文对带迁移资源消耗的复杂概率模型分步进行了可达性检验、概率检验和资源消耗检验,最终证明了基于带资源消耗的复杂概率迁移系统的性质验证方法的合理性和有效性。分析表明其能够在一定程度上缓解状态和迁移冗余的问题。未来的主要工作是面向更为庞大的迁移系统和更为复杂的性质检验,在状态缩减和路径压缩等方法上展开进一步的研究。

参　考　文　献

[1] 彭飞,张涛,王金双,等. 计算机系统形式化验证中的模型检测方法综述[J]. 军事通信技术,2016,37(2):38 – 42,76.

[2] 何洋,洪玫,祁琳莹,等. 基于模型检测工具 NuSMV 的功能测试用例生成方法[J]. 计算机应用,2015(S2):155 – 159.

[3] 李翠翠. 基于 SPIN 模型检测的电子商务协议分析与验证[D]. 上海:华东理工大学,2012.

[4] Krishna S N, Trivedi A. Hybrid automata for formal modeling and verification of cyber-physical systems[J]. Journal of the Indian Institute of Science, 2013, 93(3):419 – 440.

[5] 梁常建,李永明. 广义可能性计算树逻辑的模型检测问题[J]. 电子学报,2017,45(11):2641 – 2648.

[6] 魏欣,石玉峰,徐丙凤,等. 软件模型检测中的抽象模型研究综述[J]. 计算机研究与发展,2015,52(7):1580 – 1603.

[7] 侯刚,周宽久,勇嘉伟,等. 模型检测中状态爆炸问题研究综述[J]. 计算机科学,2013,40(S1):77 – 86,111.

[8] 王荟雯. 基于 T-不变量的 Petri 网状态空间压缩算法[D]. 大连:大连理工大学,2015.

[9] Wang Y, Liu X L, Zhang D C, et al. Bit selection via walks on graph for hash-based nearest neighbor search[J]. Neurocomputing, 2016, 213:137 – 146.

[10] 徐光侠. 分布式实时系统的软件故障注入及可靠性评测方法研究[D]. 重庆:重庆大学,2011.

[11] Zhou K J, Wang X L, Hou G, et al. Software reliability test based on markov usage model[J]. Journal of Software,2012, 7(9): 2061 – 2068.

[12] 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.

[13] 黄宏涛. 基于懒惰切片的模型检测技术研究[D]. 哈尔滨:哈尔滨工程大学,2012.

[14] 张恒巍,余定坤,寇广,等. 基于状态约减的信息攻防图生成算法[J]. 火力与指挥控制,2016,41(8):64 – 69.

[15] 温乃峰,刘冠群,于海洋,等. 水面无人艇冗余路径点约减方法研究——基于最优路径代价估计[J]. 大连民族大学学报,2017,19(5):461 – 465.

[16] Baier C, Katoen J P. Principles of model checking[M]. The MIT Press, 2008.

[17] Ji M Y, Wu D, Li Y M, et al. Counterexample generation for conditional probability in probabilistic model checking [J]. Journal of Computers, 2013, 8(12):3272 – 3279.

(上接第 151 页)

[27] Gilani S M, Zare M, Raeisi E. Locating a new drainage well by optimization of a back propagation model[J]. Mine water and the environment,2019,38(2):342 – 352.

[28] Ronneberger O, Fischer P, Brox T. U-Net: convolutional networks for biomedical image segmentation[EB]. arXiv:1505.04597,2015.

[29] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[EB]. arXiv:1512.03385,2016.

[30] Huang X, Liu M Y, Belongie S, et al. Multimodal unsupervised image-to-image translation [EB]. arXiv: 1804.04732,2018.

[31] Xu Q T, Huang G, Yuan Y, et al. An empirical study on evaluation metrics of generative adversarial networks[EB]. arXiv:1806.07755,2018.

[32] Heusel M, Ramsauer H, Unterthiner T, et al. GANs trained by a two time-scale update rule converge to a nash equilibrium [EB]. arXiv:1706.08500,2017.

[33] Che T, Li Y R, Jacob A P, et al. Mode regularized generative adversarial networks[EB]. arXiv:1612.02136,2016.

[34] Gretton A, Borgwardt K, Rasch M J, et al. A kernel method for the two-sampleproblem[EB]. arXiv:0805.2368,2008.