可能性测度下的LTL模型检测并行化研究

雷丽晖 王 静

(陕西师范大学计算机科学学院 西安 710062)

摘 要 分布式模型检测是一种缓解状态空间爆炸的有效途径,已有文献提出了定性的分布式模型验证算法,然而定量LTL验证算法并行化问题还未得到有效解决。对此,展开两个方面的工作:提出一种新的动态系统状态空间划分方法;在定性LTL分布式验证算法的基础上给出了定量模型检测并行化验证算法。首先,将系统模型转化为可能的Kripke结构并选取一个并发分量,依据状态之间的关系完成系统状态的分割,使得关系紧密的状态尽可能分布在同一个计算节点上;其次,调整划分结果以使得计算负载平衡;然后,将划分结果与其他并发分量的状态进行叉乘,以完成系统状态空间的划分;最后,将待检测性质用自动机表示,在两者的乘积上,利用扩展的基于嵌套DFS的分布式验证算法完成系统的定量验证。

关键词 可能的Kripke结构,状态空间划分,定量分布式模型检测

中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2018.04.010

Parallelization of LTL Model Checking Based on Possibility Measure

LEI Li-hui WANG Jing

(School of Computer Science, Shaanxi Normal University, Xi'an 710062, China)

Abstract Distributed model checking(DMC) is an effective solution for the state-explosion problem in model checking. The qualitative LTL distributed model verification algorithm has been proposed. However, the problem of parallelization quantitative LTL verification has not been solved effectively. In this paper, a new dynamic state partition method was presented and a quantitative DMC was proposed based on the qualitative LTL distributed verification algorithm. Firstly, the system model is transformed into a possible Kripke structure to choose a concurrent component, and its state space is divided by the relationships of states, which aims to allocate the related states to the same computing node. Secondly, the partition results are adjusted for the load balance of DMC system. Then, the partition results and the states of the rest components are done cross product to complete the state partition of the system. Finally, the properties are represented by nondeterministic finite automaton, and their cross products can complete the quantitative verfication of system based on the extended nested DFS distributed verification algorithm.

Keywords Possibilistic Kripke structure, State partition of system, Quantitative distributed model checking

1 引言

现实生活中,很多问题的答案是不确定的。在人们的日常交流中,言语表达往往包含一些不确定性,即具有一定的模糊性,不能够用确定或者否定的二分法来完全描述。它既不同于随机性,也不同于无知性。模糊测度是以测度理论为根本,把测度的概念应用于模糊性而给出的一个度量。本文重新确定了模糊集合的定义,将模糊测度应用在经典集合上,通过将测度运算(包括条件测度运算)引入模糊集合运算,提出了模糊集合的测度运算公式。

为处理具有不确定性的系统,提出一类定量的模型检测算法,包括概率模型检测[1]、多值模型检测[2]等。然而在概率系统中,概率测度是可以进行相加运算的,而对于一些复杂的实际应用系统而言,它们的性质并不一定满足可加性,因此概率模型检测并不能解决所有的具有非确定性和不一致性特点的系统验证问题。

1965年Zadeh提出了模糊集合的概念,其他学者在此基础上进行了相关研究。可能性测度[3]是模糊集理论的一个分支,是概率测度的推广,可以研究概率测度不能研究的非可加的情况。近年来,李永明教授及其团队对可能性测度下的模型检测进行了研究[4-7]。

模型检测[7]是基于对状态空间的穷举搜索,来验证系统

到稿日期:2017-05-19 返修日期:2017-07-21 本文受国家自然科学基金(11271237,11301321),中央高校基本科研业务费专项资金(GK201603086)资助。

雷丽晖(1976-),女,博士,副教授,主要研究方向为模型检测,E-mail:leilihui@snnu.edu.cn(通信作者);王 静(1990-),女,硕士生,主要研究方向为模型检测。

是否满足某个特定的系统属性。对于并发系统而言,当并发分量增加时,其状态数目会呈指数增长。因此,当系统的并发分量较多时,直接对其状态空间进行搜索是困难的,甚至不可行的,这就是模型检测中的状态爆炸问题。状态爆炸问题已成为模型检测方法应用于软件可信评价及验证的一个具有挑战性且无法回避的难题。

对于状态空间爆炸问题,在过去的几十年中,已有许多文献讨论了解决方案,大体上可分为 4 种基本方法:存储空间压缩[8]、并行和分布式计算[9]、状态空间压缩[10]、随机化和启发式搜索算法。其中,并行和分布式模型检测借助了多核及分布式技术来加速验证过程,极大地提高了验证的效率,是一种缓解状态空间爆炸的有效途径。该方法将一个需要强大计算能力才能解决的问题分成若干部分,然后对这些分量分别进行处理,最后再综合各个分量的处理结果来进行处理和分析,从而得到最终的验证结果。分布式模型检测为解决状态爆炸问题提供了一种可行的解决思路,特别是近年来以云计算为代表的互联网技术的快速发展,使得分布式模型检测成为未来模型检测发展的重要趋势。

综上所述,对于包含不确定信息的系统,可以使用基于可能性测度理论的模型检测来完成验证,但在验证过程中也会遇到状态爆炸[11]问题。为解决该问题,本文对可能性测度下的模型检测并行化进行了深入研究,并且解决了基于可能性测度的模型检测在并行化过程中出现的两个基本问题。

1)系统状态空间划分

目前,国内外学者提出了多种描述并发系统的模型。常见的系统模型可以分为两大类:1)基于状态转移图的模型(如 Kripke 结构、自动机和 Petre 网)[12];2)数学模型(如进程代数和微分方程)。第一类模型的状态空间可以分割成组,对"系统状态空间的划分"可转换为对"状态转移图的划分"问题,分布式模型验证算法可以在划分所得的子图上完成模型验证。本文将现有的解决"状态转移图的划分"问题的方法分为两类:1)静态划分;2)动态划分。

静态划分是指,系统状态转移图的存储与分割都在同一个计算节点上完成。一个系统状态转移图被视为一个有向图(图中的节点表示系统状态,边表示状态迁移),通过一些算法将图分割成几个部分。可利用一些经典的方法来解决该问题。一种简单的划分方法[13]是,对所有的系统状态进行编号,然后再利用哈希函数将系统状态均匀地分为几组。这种方法在使用前需要知道系统状态转移图被分割成的组数,缺点是划分系统状态时没有考虑状态之间的关系,会影响分布式验证算法的执行效率。

动态划分是指,系统状态转移图的存储与分割不在同一个计算节点上完成。文献[14]给出的方法是:先在并发系统 $p_1 \| p_2 \| \cdots \| p_n$ 的 n 个并发向量中选取一个 p_d(每个并发分量的状态存储于一台计算机上),对 p_d 的所有状态进行编号,再利用哈希函数将这些状态均匀分为 k 组 $S_{d1}, S_{d2}, \cdots, S_{dk}$,并将其分布到不同的计算机 $C_1, C_2, \cdots, C_k$ 上,最后在 C_i 上将 $S_{di}(1 \leqslant i \leqslant k)$ 与其他并发分量的状态 $S_t(1 \leqslant t \leqslant n, t \neq d)$ 进行叉乘,从而完成整个并发系统状态的划分,如图 1 所示。设并发系统包含 small,middle 和 big 3 个并发分量并将其分别存储在 3 台计算机上,small 被分割为 6 组 $small_i(1 \leqslant i \leqslant 6)$,在 6 台计算机上 $small_i(1 \leqslant i \leqslant 6)$ 分别与并发分量 big 和 middle 进行叉乘,从而得到该并发系统状态的划分结果。

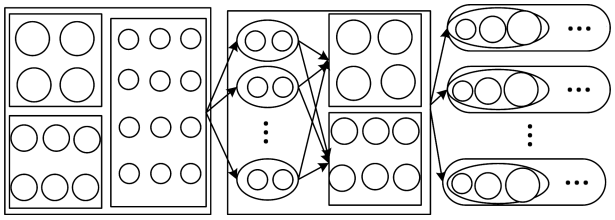

图 1 有 3 个并发分量的并发系统的状态空间的动态划分

Fig. 1 Dynamic division of state space of concurrent system with three concurrent components

前者需要得到系统所有的状态后再划分;而后者是先对一个并发分量进行划分,再获取系统状态的整体划分。显然,动态系统状态划分方法更合理。然而,上述动态划分方法在保留系统状态之间的关系(有利于提高分布式模型检测算法的执行效率)方面还有待加强。本文提出的新划分方法,即将动态划分与静态划分相结合,能保证分布式模型检测系统的负载平衡,并为提高分布式验证算法的执行效率提供良好的基础。

2)定量模型检测并行化验证

系统模型使用可能的 Kripke 结构进行描述,用户期望的性质使用非确定性有限自动机进行描述。在两者乘积之上,利用扩展的基于嵌套 DFS 的分布式 LTL 验证算法完成了系统的定性验证和定量验证。

目前,国内外基于可能性测度的 LTL 模型检测并行化的研究尚处于起步阶段。本文的贡献在于,为解决可能性测度下 LTL 模型检测中的状态爆炸问题提供了一种可行的方法。

2 系统状态划分

2.1 Kripke 结构划分

定义 1 一个 Kripke 结构 M 是一个五元组 $M=(S, I, R, AP, L)$。其中,S 是一个可数非空状态集合;I 是初始状态;$R \subset S \times S$ 是状态迁移关系;AP 是原子命题的集合;$L: \rightarrow 2^{AP}$ 是状态的标签函数。

本文提出的新划分方法将动态划分与静态划分相结合,如图 2 所示。并发系统中有 3 个分量 white,gray 和 black,先将 white 中的状态划分为 3 个子图,然后这 3 个子图分别在 3 台计算机上与 black,gray 叉乘。大致分为 3 个步骤:

1)将 n 个并发分量的 Kripke 模型分别存储于 n 台计算机上。

①分别将 n 个并发分量的 Kripke 模型转换为 n 个状态转移图;

②按如下的 Partition 算法分别对 n 个状态转移图进行划分;

③利用 Refinement 算法对 Partition 算法的结果进行调整。

2）在并发系统 $p_1 \parallel p_2 \parallel \cdots \parallel p_n$ 中选取一个分量 p_d（选择条件如下：首先选择划分时产生“跨界迁移”数量最少的分量；如果可选项不止一个，则在待选项中选择划分后各个子图状态数目相等的分量；如果可选项仍不止一个，则在待选项中选择状态数目最多的分量），将 p_d 划分得到的子图分布到不同计算机 $C_1, C_2, \cdots, C_k$ 上。

3）在 C_i 上将 $S_{di}(1<i<k)$ 与其他并发向量的状态叉乘，完成整个系统状态的划分。

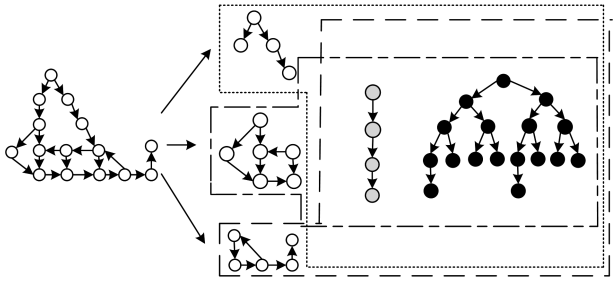

图 2　并发系统状态空间的动态划分示意图

Fig. 2　Dynamic partition of concurrent system state space

Partition 部分：1）将 Kripke 结构转化为无向图 $ST=(V, W_v, E, W_e)$，其中 V 是非空有限顶点集合，且每个顶点包含若干状态；$W_V(v)$ 表示 v 中状态之间迁移的个数；$E \subset V \times V$ 是边的集合；$W_E(e)$ 表示边 e 的条数。2）将所有顶点按照相邻节点数从少到多的顺序组成队列，从队列的第一个顶点开始，查找出 $W_V(p)+W_E(p \times q)+W_V(q)$ 最大的两个节点相匹配（其中 p, q 是顶点）。3）在队列中删除匹配在一起的两个顶点，并重新排序。重复第 2）步，直到顶点数目等于节点数。

Refinement 部分：调节各个区域之间的节点，其中需要考虑两个因素，即状态平衡和跨界数量最少。1）判定状态是否平衡。2）如果需调整，则选取包含状态最少的一个/若干子图，对与子图相关的跨界迁移进行分类（如对一个跨界迁移进行调整后系统中的跨界迁移总数减少，则认为是最佳调整方案；如对一个跨界迁移进行调整后系统中的跨界迁移总数不变，则认为是次佳调整方案；如对一个跨界迁移进行调整后系统中的跨界迁移总数增加，则认为是最后可用调整方案）。3）按照最佳、次佳、最后可用调整方案的优先次序选取跨界迁移进行调整。重复上述过程，直到满足状态平衡，结束调整工作。

2.2　可能的 Kripke 结构划分

定义 2[5]　可能的 Kripke 结构 $M=(S, P, I, AP, L)$ 是一个五元组。其中，S 为可数非空状态集合；$P: S \times S \to [0,1]$ 是可能性转移函数，满足对于每个状态 s，都有 $\bigvee\limits_{s' \in S} P(s, s')=1$；$I: S \to [0,1]$ 是初始可能性分配函数，满足对于每个状态 s，都有 $\bigvee\limits_{s' \in S} I(s)=1$；$AP$ 是一组原子命题集合；$L: S \to 2^{AP}$ 是标记函数，即在每个状态都有一个赋值（AP 的子集）。

可能的 Kripke 结构与 Kripke 结构唯一的不同在于状态迁移之间的可能性，因此在划分可能 Kripke 结构时应重点考虑怎样保证这些参数不丢失。本文在 2.1 节的基础上提出了如下划分方法。

1）向量 $A_i=(a_{i0}, a_{i1}, \cdots, a_{im})(0 \leqslant i \leqslant n)$ 描述状态 s_i 与所有状态之间的可能性转移函数 $P: S \times S \to [0,1]$。所有状态的迁移可能性用向量表示，并组成一个 $n \times n$ 的可能性矩阵存储在一台计算机上。

2）将可能 Kripke 结构转化为状态转移图，按照 2.1 节的方法进行状态划分。

3）状态图划分之后，将不同计算机 $C_1, C_2, \cdots, C_K$ 所划分状态的向量存储于各个计算机上。

例 1　划分如图 3 所示的可能的 Kripke 结构。

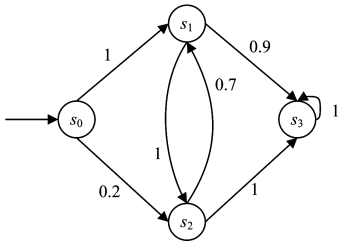

图 3　可能的 Kripke 结构

Fig. 3　Possibilistic Kripke structure

1）分别用向量 $A_0=(0,1,0.2,0)$，$A_1=(0,0,1,0.9)$，$A_2=(0,0.7,0,1)$，$A_3=(0,0,0,0)$ 表示所有状态之间的转移可能性，4 个向量组成一个 4×4 的矩阵 P 存储在一台计算机上。

$$P=\begin{pmatrix} 0 & 1 & 0.2 & 0 \\ 0 & 0 & 1 & 0.9 \\ 0 & 0.7 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

2）按照 2.1 节的划分方法，图 3 所示的可能 Kripke 结构的划分过程如图 4 所示。

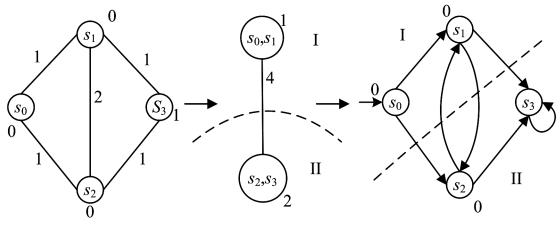

图 4　可能的 Kripke 结构的划分过程

Fig. 4　Partition process of possibilistic Kripke structure

3）状态图划分之后，状态 s_0 和 s_1 被划分在 I 上，状态 s_2 和 s_3 被划分在 II 上，因此将 $A_0=(0,1,0.2,0)$，$A_1=(0,0,1,0.9)$ 组成的 2×4 的矩阵 P_{I} 存储于 I 上，将 $A_2=(0,0.7,0,1)$，$A_3=(0,0,0,0)$ 组成的 2×4 的矩阵 P_{II} 存储于 II 上。

$$P_{\mathrm{I}}=\begin{pmatrix} 0 & 1 & 0.2 & 0 \\ 0 & 0 & 1 & 0.9 \end{pmatrix}, P_{\mathrm{II}}=\begin{pmatrix} 0 & 0.7 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

3　分布式验证算法

3.1　算法描述

定性验证算法是将模型对应的 Kripke 结构与性质对应的自动机叉乘之后，找出是否存在能从初始状态到达可接受状态且此可接受能自达的环路。而定量验证算法是在可能

Kripke 结构与自动机进行叉乘之后寻找环路，并计算出系统满足性质的可能性测度。

定义 3[7] 非确定有限自动机(NFA)$A=(Q,\Sigma,\delta,J,F)$是一个五元组，其中：

1)Q 是非空有限状态集合；

2)Σ 是有限的输入字符集；

3)δ:$Q\times\Sigma\times Q$ 为状态转移函数；

4)$J\subseteq Q$ 表示初始状态；

5)$F\subseteq Q$ 表示接受状态或终止状态。

定义 4[7] 令 $M=(S,P,I,AP,L)$是一个可能的 Kripke 结构，$A=(Q,\Sigma,\delta,J,F)$是非确定性有限自动机。乘积 $M\otimes A$ 是一个可能的 Kripke 结构，$M\otimes A=(S\times Q,P',I',L',AP')$，其中：

1)$AP'=S\times Q$。

2)$L'(s,q)=(s,q)$，对于任何$(s,q)\in S\times Q$。

3)$I'(s,q)=\begin{cases}I(s), & \text{如果 } q=\delta(q_0,L(s))\\ 0, & \text{否则}\end{cases}$。

4)$M\otimes A$ 的可能转移函数为：

$$P'((s,q),(s',q'))=\begin{cases}P(s,s'), & q'=\delta(q_0,L(s'))\\ 0, & \text{否则}\end{cases}$$

定义 5[4] 设 M 是可能的 Kripke 结构，$Paths(M)=\bigcup_{s\in S}Paths(s)$。映射 Po^M:$Paths(M)\to[0,1]$的定义如下：对任意路径 $\pi,\pi\in Paths(M),\pi=s_0s_1\cdots,Po^M(\pi)=I(s_0)\wedge\bigwedge_{i=0}^{\infty}P(s_i,s_{i+1})$，进而 $A\subseteq Paths(M)$，如果定义：$Po^M(A)=\vee\{Po(\pi)\mid\pi\in A\}$。

一般的定量 LTL 模型验证算法，是找出 $M\otimes A$ 中所有能从初始状态 $I(s,q)$到达可接受状态 $S\times F$ 且此可接受状态能够自达的环路。根据定义 5 中的两个公式计算出 M 满足 A 的可能性测度。

本文对传统的分布式验证进行扩展，根据定性分布式验证[15]的方法，给出可能性测度下的定量分布式验证算法。

传统的分布式 LTL 模型验证算法采用的是嵌套 DFS，文献[15]提出了一个动态的数据结构——依赖结构(Dependency Structure，DepS)和管理器进程(Manager Process)。Deps 是一个有向图，存储于各台计算机上，其顶点是 seed(种子：可接受状态)和有名称索引的 border state(边界状态：跨计算机的状态)，边是状态之间的可达关系。在构建 DepS 过程中引入 Remove 算法，将不是 seed 和 border state 的状态删除。管理器进程是对各个节点(计算机)发送过来的 DepS 中的状态执行嵌套 DFS，找出存在接受状态的环路。具体步骤如下：

1)利用第 2 节介绍的方法对系统状态空间进行动态划分，描述用户期望性质的自动机被视为系统的一个并发分量，将其分别与系统模型的划分结果进行叉乘。

2)在各个计算节点上进行深度优先搜索，以获取各自的依赖结构；每一个节点获取分配给自己的状态，并逐一检查该状态是否属于本地的状态集。如状态是本地的，则继续检查其后继状态；否则将该状态发送至其所有者。在整个搜索过程中，动态形成依赖结构。

①判定一个状态是否是开始状态、边界状态或种子；

②将非种子、非边界状态的其他状态从状态空间中删除；

③依据原有的网络拓扑结构将开始状态、边界状态和种子连接起来。

3)执行嵌套的深度优先搜索。

①管理进程获取所有计算节点的依赖结构，将其连接成一个 Kripke 结构；

②在该 Kripke 结构上检查是否存在着从开始状态到可接受状态的路径；

③根据此路径判定系统是否可以满足性质。

定量的分布式验证算法在定性验证算法的基础上，从以下两个方面进行扩展。

1)构建的 DepS 不仅有顶点和边，而且边上还保存着 $M\otimes A$ 中状态之间的可能性转移函数(见例 2)。在 Remove 过程中，不仅要判断状态是否是 seed 和 border state，还要比较所删状态与其前驱、后继状态之间的可能性转移函数的大小。

2)在管理器进程中执行嵌套 DFS 过程(见图 6)，找出能从初始状态到达可接受状态且此可接受能自达的环路。计算每条环路状态之间可能性转移函数的最小值，然后取所有环路可能性转移函数的最大值，得到系统满足性质的可能性测度。

例 2 图 5(a)表示 $M\otimes A$ 中的状态分布在 3 台计算机上，可接受状态是 s_4 和 s_{14}，边界状态分别是 s_5，s_7，s_8，s_{12}。图 5(b)表示各台计算机上的 DepS。

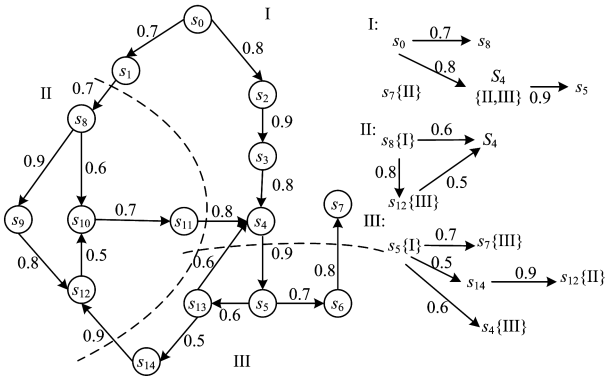

图 5 各台计算机上的 DepS

Fig. 5 DepS on each node

管理进程在图 6 上完成定量验证。

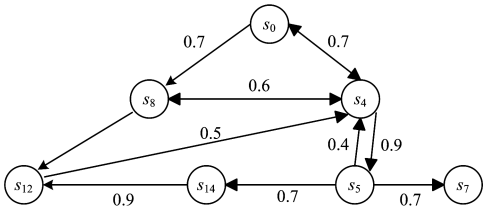

图 6 在管理进程上合成的 Kripke 结构

Fig. 6 Synthetic Kripke structure on management process

1）如没有路径从 s_0 出发到达 s_4 或 s_{14}，则系统满足性质的可能性为 0；

2）如有路径从 s_0 出发到达 s_4 或 s_{14}，则每条路径 π 的可能性 $Po(\pi)=\bigwedge_{i=0}^{\infty}P(s_i,s_{i+1})$，那么系统满足性质的可能性为 $Po^M(A)=\vee\{Po(\pi)\mid\pi\in A\}$。

在构建各台计算机上的 DepS 过程中，对 Remove 算法进行改进的基本思想如下：1）确定要删除的节点，找到前躯节点和后继节点，由转移函数矩阵得到具体的转移函数（递归进行）；2）计算过程中，每次比较 1）中的两个数值，进行取小运算（定义 5），更新依赖结构，并对数值进行记录；3）删除节点时，回溯过程得到的路径可能不止一条，对路径间的可能转移函数值进行取大运算（定义 5），更新依赖结构，并对数值进行记录；4）重复以上步骤，确定最终的依赖结构。

3.2　验证实例

可能的 Kripke 结构 $M=(S,P,I,AP,L)$，其中 $S=\{s_0,s_1,s_2\}$，$L(s_0)=\{a\}$，$L(s_1)=\{b\}$，$L(s_2)=\{a,b\}$，$I(s_0)=1$。可能转移函数 P 如图 7(a)所示。利用 LTL 性质公式构造出的自动机 $A=(Q,2^{AP},\delta,q_0,F)$，其中 $Q=\{q_0,q_2,q_3\}$，$AP=\{a,b\}$，$F=\{q_2\}$，转移函数如图 7(b)所示。本文将验证安全性 $\mathscr{L}=\Diamond B$ 和活性 $\mathscr{L}=\Box\Diamond B$，其中 $B=S\times F$。

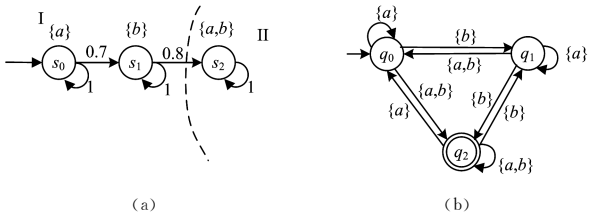

图 7　可能的 Kripke 与自动机

Fig. 7　Possible Kripke and automaton

根据第 2 节所提出的划分方法将可能结构的状态转移图划分为两个子图（见图 7(a)），分布在两台计算机 I 和 II。在各台计算机上的乘积关系 $M\otimes A=(S\times Q,P',I',AP',L')$ 和转移关系 Deps 如图 8 和图 9 所示。

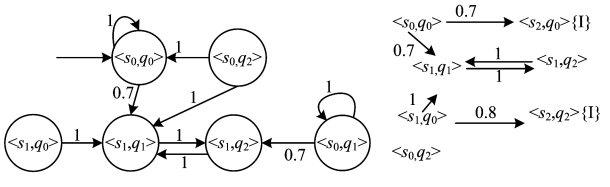

图 8　节点 I 上的 $M\otimes A$ 和 DepS

Fig. 8　$M\otimes A$ and DepS on node I

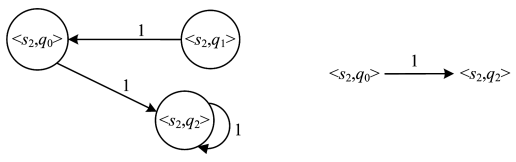

图 9　节点 II 上的 $M\otimes A$ 和 DepS

Fig. 9　$M\otimes A$ and DepS on node II

在这种情况下，管理器进程向所有节点询问空闲状态来获取有关其依赖关系结构的信息，并根据节点的依赖结构在管理进程上合成整体图，其结果如图 10 所示。

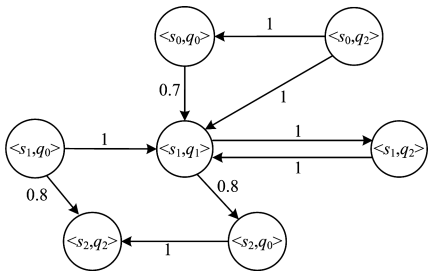

图 10　在管理器进程上合成的 Kripke 结构

Fig. 10　Synthetic Kripke structure on management process

通过在管理进程中对图 10 执行嵌套来完成定量验证。因 $B=S\times F$，故 $M\otimes A$ 中的可接受状态为：$\langle s_0,q_2\rangle$，$\langle s_1,q_2\rangle$，$\langle s_2,q_2\rangle$。

1）安全性

在管理器上可以搜索到两条从初始状态 $\langle s_0,q_0\rangle$ 到达可接受状态的路径。

①到达 $\langle s_1,q_2\rangle$ 的路径 π_1：$\langle s_0,q_0\rangle\xrightarrow{0.7}\langle s_1,q_1\rangle\xrightarrow{1}\langle s_1,q_2\rangle$；

②到达 $\langle s_2,q_2\rangle$ 的路径 π_2：$\langle s_0,q_0\rangle\xrightarrow{0.7}\langle s_1,q_1\rangle\xrightarrow{0.8}\langle s_2,q_0\rangle\xrightarrow{1}\langle s_2,q_2\rangle$。

因此，可以得出满足安全性的测度，即：$Po^M(s_0\models P_{\text{safe}})=Po^{M\otimes A}((\langle s_0,q_0\rangle\models\Diamond B)=0.7$。

2）活性

在管理器上可搜索到一条从初始状态 $\langle s_0,q_0\rangle$ 开始的包含接受状态 $\langle s_1,q_2\rangle$ 的环路。

$$\langle s_0,q_0\rangle\xrightarrow{0.7}\langle s_1,q_1\rangle\underset{1}{\overset{1}{\rightleftarrows}}\langle s_1,q_2\rangle$$

因此，可以得出系统满足活性的可能性测度为：

$Po^{M\otimes A}((\langle s_0,q_0\rangle\models\Box\Diamond B)=0.7$

结束语　本文在定性 LTL 分布式验证算法的基础上，提出了定量 LTL 分布式验证算法，主要在系统状态划分和分布式验证两个方面进行了研究。1）针对分布式模型检测的状态空间划分问题，提出新的动态状态空间划分方法，并且用新的数据结构来存储可能 Kripke 结构的可能性转移函数，保证了验证的完整性。2）将定性分布式验证算法进行扩展应用，提出了构造动态数据结构 DepS 的新方法，利用扩展的基于嵌套 DFS 的分布式验证算法完成了系统的定量验证。

今后的工作中需要进一步研究以下问题：管理器进程需要将收到的各个分区传送的依赖结构合并为一个可能的结构，然后在这个可能的结构上完成性质验证，如果系统状态很多，在划分系统状态空间时势必需要多分几组，导致产生的跨界迁移很多，边界状态也会随之增加，从而使得管理器进程中最后要维护和处理一个巨大的可能的 Kripke 结构，甚至单个计算节点无法完成该任务。因此，在今后的研究工作中，可以尝试利用分层处理的思想来进一步优化可能性模型检测的并行化。

（下转第 88 页）

参 考 文 献

[1] ALAVI Y,BEHZAD M,LESNIAK-FOSTER L M,et al. Total matchings and total coverings of graphs[J]. Journal of Graph Theory,1977,1(2):135-140.

[2] LAN J K,CHANG G J. On the mixed domination problem in graphs [J]. Theoretical Computer Science,2013,476(476):84-93.

[3] MANLOVE D F. On the algorithmic complexity of twelve covering and independence parameters of graphs[J]. Discrete Applied Mathematics,1999,91(1-3):155-175.

[4] ALAVI Y,LIU J,WANG J,et al. On total covers of graphs[J]. Discrete Mathematics,1992,100(1-3):229-233.

[5] ZHAO Y,KANG L,SOHN M Y. The algorithmic complexity of mixed domination in graphs [J]. Theoretical Computer Science,2011,412(22):2387-2392.

[6] GAREY M R,JOHNSON D S. Computers and Intractability:A Guide to the Theory of NP-Completeness[M]. W. H. Freeman and Company,1979.

[7] BERTOSSI A A. Dominating sets for split and bipartite graphs [J]. Information Processing Letters,1984,19(1):37-40.

[8] PAPADIMITRIOU C H,STEIGLITZ K. Combinatorial optimization:algorithms and complexity[M]. Prentice Hall,1998.

[9] JOHNSON D S. Approximation algorithms for combinatorial problems[J]. Journal of Computer and System Sciences,1982,9(3):256-278.

[10] FUJITO T,NAGAMOCHI H. A 2-approximation algorithm for the minimum weight edge dominating set problem [J]. Discrete Applied Mathematics,2002,118(3):199-207.

[11] NEMHAUSER G L,JR L E Trotter. Properties of vertex packing and independence system polyhedra[J]. Mathematical Programming,1974,6(1):48-61.

[12] YANNAKAKIS M,GAVRIL F. Edge Dominating Sets in Graphs[J]. SIAM Journal on Applied Mathematics,1980,38(3):364-372.

[13] NIEBERG T,HURINK J. A PTAS for the Minimum Dominating Set Problem in Unit Disk Graphs[M]// Approximation and Online Algorithms. Springer Berlin Heidelberg,2005:296-306.

[14] HEDETNIEMI S T,LASKAR R C. Bibliography on domination in graphs and some basic definitions of domination parameters [J]. Discrete Mathematics,1991,86(1):257-277.

[15] XIAO M,KLOKS T,POON S H. New parameterized algorithms for edge dominating set[J]. Theoretical Computer Science,2013,4(511):147-158.

[16] HATAMI P. An approximation algorithm for the total covering problem[J]. Discussiones Mathematicae Graph Theory,2007,27(3):553-558.

[17] RAZ R,SAFRA S. A sub-constant error-probability low-degree test,and a sub-constant error-probability PCP characterization of NP[C]// Proceedings of Twenty-Ninth ACM Symposium on Theory of Computing(STOC). ACM,1997:475-484.

(上接第 75 页)

参 考 文 献

[1] BAIER C,KATOEN J P. Principles of model checking[M]. Cam-bridge:The MIT Press,2008.

[2] LI Y M,DROSTE M,LEI L H. Model checking of linear-time properties in multi-valued system [J]. Information Sciences,2017,377:51-74.

[3] KHOUSSAINOV B,NERODE A. Automata Theory and its Applications[M]. Birkauser,2001.

[4] LI Y M,LI L J. Model checking of linear-time properties based on possibility measure [J]. IEEE Transaction on Fuzzy Systems,2013,21(5):842-854.

[5] LI L,LI J. Model-checking of linear-time properties in possibilistic Kripkestructure[C]// Quanitative Logic and Soft Computing-the QI&SC. 2012:287-294.

[6] LI Y M. Two methods for Model-checking ofLTL[J]. Journal of Shaanxi Normal University(Natural Science Edition),2014,42(4):22-25. (in Chinese)
李永明. 可能 LTL 模型检测的两种方法[J]. 陕西师范大学学报(自然科学版),2014,42(4):22-25.

[7] LI L J. Model checking of linear-time properti-es based on possibility measure [D]. Xi'an:Shaanxi Normal University,2012. (in Chinese)
李丽君. 基于可能性测度的 LTL 模型检测[D]. 西安:陕西师范大学,2012.

[8] EDMUND E,GRUMBERG O,PELED D. Model checking [M]. Cambridge:The MIT Press,1999.

[9] EVANGELISTA Y M,PRADAT-PEYRE J F. Memory Efficient State Space Storage in Explicit Software[C]// International Spin Workshop on Model Checking of Software. 2005:43-57.

[10] BARNAT J,BRIM L,ROCKAI P. Scalable multicore LTL model checking[M]// Model Checking Sofware. Springer Berlin Heidelberg,2007:187-203.

[11] HOU G,ZHOU K J,YONG J W,et al. Research overview of state explosion in the model checking[J]. Computer Science,2013,40(6A):77-86. (in Chinese)
候刚,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(6A):77-86.

[12] JIANG Y X,LIN C,XING X J. Model checking of Petri net based on linear logic[J]. Journal of System Simulation,2003,15(21):6-10. (in Chinese)
蒋屹新,林闯,刑栩嘉. 基于线性逻辑的 Petri 网的模型检测[J]. 系统仿真学报,2003,15(21):6-10.

[13] BOURAHLA M. Distributed CTL model checking[J]. IEEE Proceedings Software,2005,152(6):297.

[14] LERDA F,SISTO R. Distributed-memory model checking with spin[C]// International Spin Workshops on Theoretical & Practical Aspects of Spin Model Checking. Spinger-Verlag,1999:22-39.

[15] BARNAT J,BRIM L,STŘÍBRNÁ J. Distributed LTL model-checking in SPIN[C]// Proceedings of the 8th International SPIN Workshop on Model Checkingof Software. 2001:200-216.