多值可能性模型检测器的设计与实现

洪云端[1,2],李永明[1,2]
(1. 陕西师范大学 计算智能实验室,陕西 西安 710119;
2. 陕西师范大学 计算机科学学院,陕西 西安 710119)

摘 要:随着现代计算机软件和硬件的复杂性变大,模型检测作为一种形式化自动验证技术,与传统的检测技术相比有着一系列的优势,比如可以在系统实现之前对系统进行验证,可以提前发现问题,节约大量成本。传统的模型检测器大多是基于经典的模型检测技术实现的,而现实生活中存在大量的不确定信息,使用传统的模型检测无法解决这些问题。而多值模型检测理论的出现,结合多值计算树逻辑,构建多值可能性 Kripke 结构模型,可以很好地解决这些问题。为了实现模型检测自动化特性的最大优势,基于多值可能性定量模型检测的理论,设计了多值 Kripke 结构在计算机中的存储结构、计算模块等,实现了一个基于多值可能性测度的多值计算树逻辑的模型检测器 MvChecker,使得用户可以自动验证系统性质。

关键词:模型检测;多值可能性;Kripke 结构;自动验证;模型检测器
中图分类号:TP393 **文献标识码**:A **文章编号**:1673-629X(2019)05-0062-04
doi:10.3969/j.issn.1673-629X.2019.05.013

Design and Realization of Multi-valued Model Checker

HONG Yun-duan[1,2], LI Yong-ming[1,2]
(1. Computational Intelligence Laboratory, Shaanxi Normal University, Xi'an 710119, China;
2. School of Computer Science, Shaanxi Normal University, Xi'an 710119, China)

Abstract: With the increasing complexity of modern computer software and hardware, model detection, as a formalized automatic verification technology, has a series of advantages over traditional detection technology. For example, it can verify the system before the system implementation, detect problems in advance, and save a lot of costs. The traditional model checker is mostly based on the classic theory of model checking, but in real life there are a lot of uncertain information, so it cannot solve these problems. For this, we use multi-valued model theory, combined with multi-value computation tree logic to construct multi-value possibilistic Kripke model that can be an effective solution to these problems. In order to achieve the advantage of automatic characteristics of the model checking, based on the theory of multi-valued possibility quantitative model checking, we design the storage structure and calculation module of multi-value Kripke structure in computer, and implements a model detector MvChecker based on multi-valued possibility measure of multi-valued calculation tree logic, so that users can automatically verify the system property

Key words: model checking; multi-valued possibility; Kripke structure; automatic verification; model checker

0 引 言

模型检测[1-3]作为一种形式化验证工具,主要包含三个步骤:建立模型,描述系统性质,使用模型检测算法验证系统是否满足这些性质。系统模型通常使用布尔传递系统模型或者 Kripke 结构,系统性质则通过时序逻辑进行刻画,最终验证结果会给出一个布尔结果,满足性质给出是,不满足性质系统会给出否,并且给出反例。模型检测广泛应用于许多重要的系统,例如 SPIN[4],NuSMV[5]。

模型检测器受制于经典逻辑,有许多问题在经典逻辑上无法解决,例如现实生活当中的不确定性[6]。当处理这些信息的时候,传统的模型检测器便失去了作用。其中非经典多值逻辑提供了一种很好的解决方法来处理信息的不确定性和不完备性,例如文献[7]

收稿日期:2018-04-02 **修回日期**:2018-08-08 **网络出版时间**:2018-12-21
基金项目:国家自然科学基金(11671244)
作者简介:洪云端(1992-),男,硕士研究生,研究方向为模型检测、模糊系统分析;李永明,教授,博导,研究方向为非经典计算理论、计算智能、模糊系统分析、量子逻辑与量子计算、格上拓扑学。
网络出版地址:http://kns.cnki.net/kcms/detail/61.1450.TP.20181221.1445.008.html

提出的用三值逻辑去解释模型检测的结果,文献[8]使用四值逻辑对门级电路的抽象模型进行推理验证。

文献[9]提出基于多值可能性测度的多值计算树逻辑的模型检测,解决了在系统中存在“是”和“否”之间状态的逻辑问题,其采用两种方式:第一,将系统中状态解释为多值的;第二,将系统中状态之间的转移关系也设置为多值的。

基于文献[10],文中设计了自动化验证的多值模型检测器。首先,对构建的多值模型 Kripke 结构,从计算机存储的方面进行数据结构上的设计,并且分析了时间复杂度,尽量用最优化的结构对状态和转移函数进行存储,使模型检测器能够计算更多的状态;其次,设计了模型检测器的框架,从输入的模型以及性质,如何转换模型和性质,如何对性质进行计算,到完成模型检测后如何输出检测结果,并对时间复杂度进行了分析。

1　多值可能性测度下的模型检测

1.1　多值可能性 Kripke 结构

定义 1[11]:设 $(l, \leqslant)$ 是偏序集,若 l 中任意两个元素都存在上确界以及下确界,则称 $(l, \leqslant)$ 是格,为了方便,这样的格称为偏序格。

定义 2[12]:一个多值 Kripke 结构是一个五元组 $M=(S,P,I,\mathrm{AP},L)$,其中:

S 是一个可数、非空的状态集合;

$P:S\times S\to l$ 表示多值传递关系;

$I:S\to l$ 是初始分布函数;

AP 是原子命题的集合;

$L:S\times \mathrm{AP}\to l$ 是一个标签函数。

注:多值传递函数 $P:S\times S\to l$ 可以通过 l 格值表示,将这个矩阵表示为 $\boldsymbol{P}$。矩阵 $\boldsymbol{P}$ 传递闭包,表示为 $\boldsymbol{P}^+$。矩阵 $\boldsymbol{P}$ 的自反和传递闭包用 $\boldsymbol{P}^*$ 表示,$\boldsymbol{P}^*=\boldsymbol{P}^0\vee\boldsymbol{P}^+$。$\mathrm{Paths}(M)$ 表示 M 中所有路径的集合。

定义 3[13]:一个多值 Kripke 结构 M,其函数 $\mathrm{Po}^M:\mathrm{Paths}(M)\to l$ 定义如下:

$$\mathrm{Po}^M(\pi)=I(s_0)\wedge\bigwedge_{i=0}^{\infty}P(s_i,s_{i+1})$$

对任意 $\pi=s_0s_1\cdots\in\mathrm{Paths}(M)$,有如下定义函数:

$\mathrm{Po}^M:2^{\mathrm{Paths}(M)}\to l$

Po^M 称为在 $\Omega=2^{\mathrm{Paths}(M)}$ 上的广义可能性测度。

定义 4[14]:对于一个多值 Kripke 结构 M,定义一个函数 $r_P:S\to l$,表示在 M 上从初始状态 s 出发的最大可能性路径:

$$r_P(s)=\vee\{P(s,s_1)\wedge P(s_1,s_2)\wedge\cdots\mid s_1,s_2,\cdots\in S\}$$

其中,计算 $r_P(s)$ 的方法依据定理 1。

定理 1[15]:一个有限多值 Kripke 结构 M,M 中的一个状态 s,有:

$$r_P(s)=\vee\{P^+(s,t)\wedge P^+(t,t)\mid t\in S\}$$

使用矩阵表示为:

$$r_P(s)=\boldsymbol{P}^+\circ\boldsymbol{D}$$

其中,符号“$\circ$”表示 sup-inf 合成运算。

1.2　基于多值可能性测度的多值计算树逻辑

引入一种新的定量时序逻辑对性质进行描述,即可能性测度上的多值计算树逻辑。

定义 5[16](MvCTL 语构):多值 CTL(简称 MvCTL)关于原子命题 AP 的状态公式:

$$\Phi::r\mid\alpha\mid\Phi_1\wedge\Phi_2\mid\neg\Phi\mid\mathrm{Po}(\varphi)$$

其中,$\alpha\in\mathrm{AP}$,$r\in l$,φ 是 MvCTL 路径公式,如下:

$$\varphi::=\mathrm{O}\Phi\mid\Phi_1\cup\Phi_2\mid\Phi_1\cup^{\leqslant n}\Phi_2\mid\Box\Phi$$

其中 Φ,Φ_1 和 Φ_2 是状态公式并且 $n\in N$。

定义 6(MvCTL 语义):多值 Kriple 结构 $M=(S,P,I,\mathrm{AP},L)$,$a\in\mathrm{AP}$ 是原子命题,$s\in S$ 表示状态,Φ,Ψ 是 MvCTL 状态公式,φ 是 MvCTL 路径公式。对公式 Φ,其语义是格值模糊集 $\|\Phi\|:S\to l$,对任意 $s\in S$,$r\in l$,其定义如下:

$$\|r\|(s)=r$$

$$\|a\|(s)=L(s,a)$$

$$\|\Phi\wedge\Psi\|(s)=\|\Phi\|(s)\wedge\|\Psi\|(s)$$

$$\|\neg\Phi\|(s)=\neg\|\Phi\|(s)$$

$$\|\mathrm{Po}(\varphi)\|(s)=\mathrm{Po}(s\mid=\varphi)$$

对于路径公式 φ,其语义是一个格值模糊集 $\|\varphi\|:\mathrm{Paths}(M)\to l$,其定义如下:

$$\|\mathrm{O}\Phi\|(\pi)=\|\Phi\|(\pi[1])$$

$$\|\Phi\cup\Psi\|(\pi)=\vee_{j\geqslant0}\wedge_{k<j}\|\Phi\|(\pi[k])\wedge\|\Psi\|(\pi[j])\|;$$

$$\Phi\cup^{\leqslant n}\Psi\|(\pi)=\vee_{j\leqslant n}\wedge_{k<j}\|\Phi\|(\pi[k])\wedge\|\Psi\|(\pi[j]);$$

$$\|\Box\Phi\|(\pi)=\bigwedge_{i=0}^{\infty}\|\Phi\|(\pi([i]))$$

$$\mathrm{Po}(s\mid=\varphi)=\bigvee_{\pi\in\mathrm{Paths}(s)}\mathrm{Po}^M(\pi)\wedge\|\varphi\|(\pi)$$

路径公式 $\Diamond\Phi$(最终的)有如下语义:

$$\|\Diamond\Phi\|(\pi)=\bigvee_{j=0}^{\infty}\|\Phi\|(\pi[j])$$

2　多值可能性模型检测算法

MvCTL 模型检测的问题可以描述如下:

给一个有限广义的多值 Kripke 结构 M,状态 s 和一个 MvCTL 状态公式 Φ,计算 $\|\Phi\|(s)$ 的值。其计算采用递归的方法。

给出模型检测的算法[17]如下:

输入:GPKS M 和 MvCTL 公式 Φ

输出：s 满足公式 φ 的可能性，即 $\| \Phi \| (s)$

1：Procedure MvCTLCheck (Φ)

2：Case Φ

3：r return $(r)_{s \in S}$

4：a return $(L(s,a))_{s \in S}$

5：$\neg \Phi$ return $(\neg \| \Phi \| (s))_{s \in S}$

6：$\Phi_1 \wedge \Phi_2$ return $(\neg \| \Phi_1 \| (s) \wedge \| \Phi_2 \| (s))_{s \in S}$

7：Po($O\Phi$) return $P \circ D_\psi \circ r_p$

8：Po($\Phi \cup^{\leqslant n} \Psi$) return $\bigvee_{i=0}^{n} (D_\Phi \circ P)^i \circ D_\psi \circ r_p$

9：Po($\Phi_1 \cup \Phi_2$) return $(D_\Phi \circ P)^* \circ D_\Psi \circ r_p$

10：Po($\Diamond \Phi$) return $P^* \circ D_\Phi \circ r_p$

11：Po($\Box \Phi$) return Fixpoint($(1)_{s \in S}, f_\Phi$)

End Case

End Procedure

3 模型检测器的设计与实现

3.1 模型检测器的架构设计

模型检测的基本运行原理如图 1 所示，主要分为三个部分：系统建模、性质的形式化描述、模型检测算法。所以模型检测器的结构设计也主要依据这三个部分来进行。

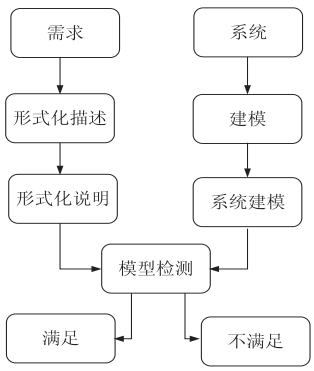

图 1 模型检测流程

模型检测器分为三层。第一层为模型构建层，主要完成对多值 Kripke 结构中的数据进行建模，用计算机中的数据结构进行存取；第二层为计算层，主要根据用户输入的性质进行计算；第三层为性质验证层，主要对计算出的结果进行验证，得出用户输入的性质是否满足系统的结果。模型检测器的系统架构如图 2 所示。

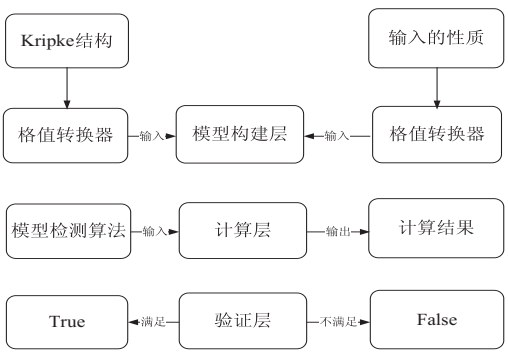

图 2 模型检测器架构

3.2 格值转换器设计

与经典的模型检测器的输入不同时，由于采用了格值的建模方式，因此在验证之前，设计一个格值转换器，主要方式是将格值映射到[0,1]区间上，例如三值逻辑，则将[T,M,F]映射为[0,0.5,1]的三值状态进行计算。

3.3 模型构建层设计

需要保存 Kripke 结构的节点信息以及传递信息，使用数据结构中的图[18]结构。由于有向图中有两种存储方式，一种是邻接矩阵，一种是邻接表，其相关的时间复杂度分别为 $O(n^2)$ 和 $O(E+V)$（E 为边数，V 为节点数），所以采用时间复杂度更低的邻接表的方式进行存储。

根据用户输入的 Kripke 结构，分别将节点和边刻画为 Node 类和 Edge 类，其中 Node 保存了节点的状态信息，Edge 类保存了传递信息。其实现为从用户的输入当中生成所有的节点，然后依次遍历每个节点信息，判断节点与节点之间是否有边，如有，则将边的信息添加到邻接表中，代码如下：

```
/ * * * * * * * * 省略部分源码 * * * * * * * * * * /
public void addEdge(Node v,Node w){
if(hasEdge(v,w)){
Return;//如有边,则返回
}
g. get(v. id). add(w);//g 为 list 实例
}
```

3.4 计算层设计

这一层主要设计的问题是模型检测算法的输入和计算函数的定义。将模型检测算法封装在一个函数当中，接收用户输入的性质，然后进行判断，判断完成后，需要遍历之前存储的邻接表，拿出状态和传递的数据，最后调用计算函数进行计算，将输出的结果进行保存。判断函数的部分代码如下：

```
//返回每一性质计算所需要的公式类
publicFomule[ ] choiceFomule(String[ ] s){
//循环遍历用户的每一个输入
for(int i=0;i<s. length;i++){
String s=s[i];
//match 函数用于找到匹配的公式
Fomule[i] fomule=match(s);
}
returnfomule;  //返回公式类
}
```

匹配公式之后，根据模型检测算法进行计算，需要定义圈乘“。”运算，交运算“ $\wedge$ ”，并运算“ $\vee$ ”的计算函数，圈乘函数代码如下：

```
public double[n][n]calculateCircle(
double[n][n]a,double[n][n] b){
```

```
double[n][n] c=new double[n][n];
//遍历 a 矩阵的行,遍历 b 矩阵的列
for(int i=0;i<n;i++){
for(int j=0;j<i;j++){
if(a[i][j]<b[j][i])     //取小
c[i][j]=a[i][j];
c[i][j]=b[j][i];
}
}
return c;//返回计算结果
}
```

对于并运算函数和交运算函数,由于和圈乘运算类似,只是改变了取大和取小操作,这里省略了代码。由于采用了双层循环的遍历,可以看出每个函数的时间复杂度为 $O(n^2)$ 。

3.5　验证层设计

这部分主要完成函数逻辑的调用,并计算出结果。由于在计算层封装了各种算法公式,在验证层,需要拿出计算层给出的结果,加以判断之后,输出是或者否。将传入计算层封装的结果类 ResultData,调用函数进行判断,返回一个 boolean 数组,最后遍历这个数组可得到计算结果。

4　模型检测器的验证

4.1　模型检测器时间测试实验

对设计出来的模型检测器做一个时间性能的测试。结果如表 1 所示,其中 n 表示状态的个数,true 表示正确完成了计算,false 表示计算失败,time 表示运行时间。

表 1　运行时间测试结果

n	Time/s	结果
1 000	1.0	true
5 000	9.6	true
10 000	27.2	true
100 000	2 842.8	true

模型检测器在状态数小于 10 000 的情况下,时间性能还不错,当大于 100 000 时,由于时间复杂度是指数级增长,时间性能上就差了一些。

4.2　恒温器性质验证实验

有一个三值恒温器模型,如图 3 所示。

接下来验证如下几个性质:

性质 1:系统是否可以从任何状态转移到 IDLE1 状态;

性质 2:当温度低于某个值时加热器是否可以启动;

性质 3:在什么情况下系统是否可以关闭;

性质 4:是否只有在空调关闭的状态下才能加热;

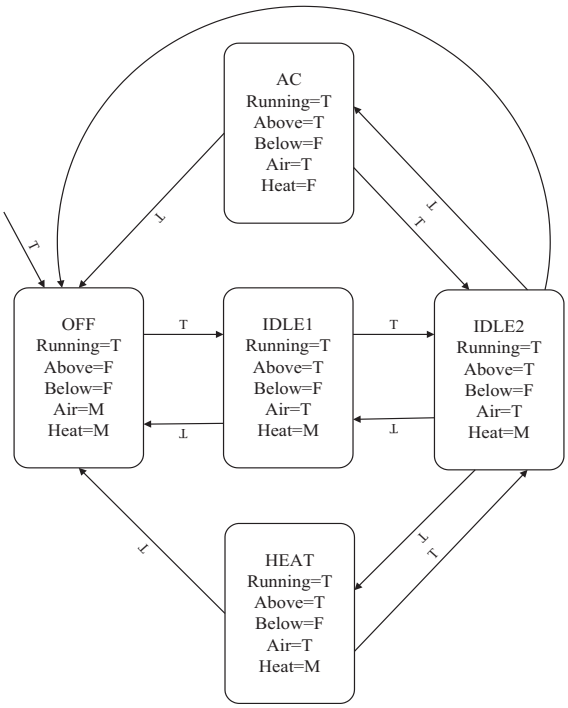

图 3　恒温器模型

性质 5:当温度高于某个温度时加热器能否启动。

将这些性质转化为 MvCTL 公式后,输入的结果如表 2 所示。

表 2　性质验证结果

性质	MvCTL 公式	结果
性质 1	Po(Po(*OIDLE*1))	(T,T,M,T,F)
性质 2	Po(¬ Heat ∪ Below)	(T,T,T,M,T)
性质 3	Po(□ Po(◇¬ Runing)))	(T,T,T,T,T)
性质 4	Po(□ (¬ AC → Heat))	(F,F,F,T,T)
性质 5	Po(□ (Above → ¬ Heat))	(T,T,T,M,T)

通过表 2 可以得到一些结论,比如系统可以从状态 OFF 转移到状态 IDLE1 的可能性为 T,转移到 IDLE2 的可能性为 M,转移到 AC 状态的可能性为 T。

5　结束语

根据基于多值可能性测度的多值可能性计算树逻辑的模型检测理论,设计能够自动化验证的多值模型检测器 MvChecker。根据模型检测流程,设计了三层模型检测器的框架,对每一层架构进行了详细设计,并使用 Java 语言进行了实现,用户可以自己构建系统和需要验证的性质,输入 MvChecker 进行系统的验证。

参考文献:

[1]　BAIER C,KATOEN J P. Principles of model checking[M]. Cambridge:MIT Press,2008.

(下转第 69 页)

[3] 袁　轶,王新房. 一种基于方差的文本特征选择算法[J]. 计算机工程,2012,38(12):155-157.
[4] KO Y,SEO J. Automatic text categorization by unsupervised learning[C]//Proceedings of 18th conference on computational linguistics. Saarbrücken, Germany: ACM, 2000: 453-459.
[5] BIDI N,ELBERRICHI Z. Feature selection for text classification using genetic algorithms[C]//International conference on modelling, identification and control. Algiers, Algeria: IEEE,2017:806-810.
[6] 林艳峰. 中文文本分类特征选择方法的研究与实现[D]. 西安:西安电子科技大学,2014.
[7] 李军怀,付静飞,蒋文杰,等. 基于 MRMR 的文本分类特征选择方法[J]. 计算机科学,2016,43(10):225-228.
[8] 郑　伟. 文本分类特征选取技术研究[D]. 呼和浩特:内蒙古大学,2008.
[9] QIU Liqing, ZHAO Ruyi, ZHOU Gang, et al. An extensive empirical study of feature selection for text categorization [C]//IEEE/ACIS international conference on computer and information science. Portland, OR, USA: IEEE, 2008: 312-315.
[10] 林少波. 中文文本分类特征提取方法的研究与实现[D]. 重庆:重庆大学,2011.
[11] 康岚兰,董丹丹. 一种改进的互信息特征选择方法[J]. 电脑知识与技术,2009,5(35):9889-9890.
[12] 卢新国,林亚平,陈治平. 一种改进的互信息特征选取预处理算法[J]. 湖南大学学报:自然科学版,2005,32(1):104-107.
[13] 蒋　健. 文本分类中特征提取和特征加权方法研究[D]. 重庆:重庆大学,2010.
[14] DING Xiaoming, TANG Yan. Improved mutual information method for text feature selection[C]//International conference on computer science & education. Colombo, Sri Lanka: IEEE,2013:163-166.
[15] JIANG Xiaoyu, JIN Shui. An improved mutual information-based feature selection algorithm for text classification[C]//5th international conference on intelligent human-machine systems and cybernetics. Hangzhou, China: IEEE, 2013: 126-129.

(上接第 65 页)

[2] BIRKHOFF G. Lattice theory[M]. 3rd ed. [s. l.]:[s. n.], 1973.
[3] LI Yongming. Quantitative model checking of linear-time properties based on generalized possibility measures[J]. Fuzzy Sets and Systems,2015,320:17-39.
[4] HOLZMANNN G J. The SPIN model checker: prinmer and reference manual[M]. Reading: Addison-Wesley,2004.
[5] CIMATTI A, CLARKE E, GIUNCHIGLIA F, et al. NuSMV: a new symbolic model checker[J]. International Journal on Software Tools for Technology Transfer,2000,2(4):410-425.
[6] LI Yongming, MA Zhanyou. Quantitative computation tree logic model checking based on generalized possibility measures[J]. IEEE Transactions on Fuzzy Systems,2015,23(6):2034-2047.
[7] 陈云霁,马　麟,沈海华,等. 龙芯 2 号微处理器浮点除法功能部件的形式验证[J]. 计算机研究与发展,2006,43(10):1835-1841.
[8] 张　良,易江芳,佟　冬,等. 使用局部建模的微处理器测试程序自动生成方法[J]. 电子学报,2011,39(7):1639-1644.
[9] 马占有,李永明. 基于决策过程的广义可能性计算树逻辑模型检测[J]. 中国科学:信息科学,2016,46(11):1591-1607.
[10] 邓楠轶. 基于广义可能性测度的模型检测器 GPoCheck 的设计与实现[D]. 西安:陕西师范大学,2015.
[11] 范艳焕,李永明,潘海玉. 不确定型模糊 Kripke 结构的计算树逻辑模型检测[J]. 电子学报,2018,46(1):152-159.
[12] LI Yongming, LI Yali, MA Zhanyou. Computation tree logic model checking based on possibility measures[J]. Fuzzy Sets and Systems,2015,262:44-59.
[13] 周从华,邢支虎,刘志锋,等. 马尔可夫决策过程的限界模型检测[J]. 计算机学报,2013,36(12):2587-2600.
[14] 林惠民,张文辉. 模型检测:理论、方法与应用[J]. 电子学报,2002,36(12A):1907-1912.
[15] 刘　阳,李宣东,马　艳,等. 随机模型检测研究[J]. 计算机学报,2015,38(11):2145-2162.
[16] 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.
[17] 李永明,李　平. 模糊计算理论[M]. 北京:科学出版社,2016.
[18] 严蔚敏,吴伟民. 数据结构[M]. 北京:清华大学出版社,2012.