CN 43-1258/TP
ISSN 1007-130X

计算机工程与科学
Computer Engineering & Science

第 37 卷第 10 期 2015 年 10 月
Vol. 37, No. 10, Oct. 2015

文章编号:1007-130X(2015)10-1884-06

一种基于扩展不完全 Kripke 结构的三值逻辑模型检测方法*

刘　姣,雷丽晖

(陕西师范大学计算机科学学院,陕西 西安 710119)

摘　要:多值模型检测是解决形式化验证中状态爆炸问题的一种重要方法,三值模型检测是多值模型检测的基础,其中如何检验不确定状态的真值是一难点。针对不确定状态检验,提出了一种模型检测方法,首先对不完全 Kripke 结构 PKS 进行了扩展,然后在扩展后的模型上给出了检测不确定状态真值的方法,最后给出了基于扩展不完全 Kripke 结构的三值逻辑模型检测算法。与已有的三值逻辑模型检测算法相比,该算法降低了算法复杂度,完善了对于不确定或不一致信息的处理,从而增强了三值逻辑模型检测的实用性。

关键词:三值逻辑;模型检测;扩展的不完全 Kripke 结构

中图分类号:TP301.6　　**文献标志码**:A

doi:10.3969/j.issn.1007-130X.2015.10.014

A three-valued logic model checking approach based on extensional partial Kripke structure

LIU Jiao, LEI Li-hui

(School of Computer Science, Shaanxi Normal University, Xi'an 710119, China)

Abstract: Multi-valued model checking is an important method to solve the state explosion problem in formal verification, and its basis is the three-valued logic model checking. The challenge is how to obtain the value of uncertain states. We first propose a method to extend the partial Kripke structure (PKS), then present an approach for for obtaining the values of uncertain states based on the extended PKS, and finally design a three-valued logic model checking algorithm. Compared with the existing three-valued model checking algorithms, our algorithm reduces the complexity. Moreover, the proposed algorithm can improve the processing of uncertain or inconsistent information, and enhance the practicality of the three-valued logic model checking.

Key words: three-valued logic; model checking; extensional partial Kripke structure

1　引言

模型检测[1]是可用于验证系统的一种形式化方法,因其需要穷尽搜索系统的所有状态,所以使用模型检测技术来验证系统的过程需要占用大量的计算和存储资源。随着软硬件的快速发展,待验证的系统规模越来越庞大,使得系统验证过程中出现状态爆炸的几率大大增加。

为了有效避免状态爆炸,国内外研究者给出了许多方法,其中一类方法是对系统状态进行抽象合并,以简化模型结构,减少系统状态数,但是随之而来的是,系统状态在合并过程中会引入不确定信息或不一致信息,如图 1 所示。

图 1 中的状态 S_0 与 S_1 合并以减少系统状态数,但合并过程导致变量 A 成为了不确定的,使得系统不可避免地涉及到不确定信息的处理。为处理这些不确定信息或不一致信息,多值模型验证应

* 收稿日期:2015-07-10;修回日期:2015-09-03
基金项目:国家自然科学基金资助项目(61003061,11271237)
通信地址:710119 陕西省西安市陕西师范大学计算机科学学院
Address: School of Computer Science, Shaanxi Normal University, Xi'an 710119, Shaanxi, P. R. China

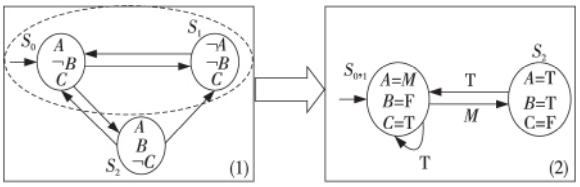

Figure 1　Process of state merging

图 1　状态合并过程

运而生。本文提出一种基于三值逻辑的模型检测方法,可作为多值模型检测基础。

在现有的多值逻辑模型检测方法中,一种方法是将多值逻辑转化为经典逻辑进行处理,如 Chechik M[2~5]和 Bolc L[6]等将已有三值问题分解成若干经典逻辑问题进行处理;而另一种方法是基于三值逻辑本身完成模型检测,如 Li Yong-ming 等[7,8]通过人为设置状态不确定性的可接受度,采用模糊模型检测计算出系统满足性质的可接受度。而郭建等[9]通过算法在模型检测过程中对公式为真和未知的情况作出检查,再由此推断出公式为假的情况,这样可以及早发现未知公式,了解哪些状态需要信息细化,以便检验未知公式的真值,但其算法中并未涉及状态信息的细化,无法检测未知公式的真值,三值模型检测算法不完整。

本文提出了一种三值逻辑模型检测过程,并给出三值逻辑模型检测算法。与已有的三值逻辑模型检测算法相比,该算法降低了算法的复杂度,完善了对于不确定或者不一致信息的处理,从而增强了三值逻辑模型检测的可应用性。

2　三值逻辑模型

三值逻辑模型可以在经典逻辑模型的基础上,通过对经典逻辑模型中部分状态进行合并或抽取得到,从而减少系统状态数量,以避免系统检测中出现状态爆炸。值得注意的是,在状态迁移过程中状态变量的数量是不变的,所以当相邻的一个或多个状态进行合并时,只需考虑状态变量值存在不一致性的情况,即状态变量具有多个不同的值,记此类变量的值为 M,生成三值逻辑模型。例如,图 1(1)中,由于存在搜索路径$[S_0S_2]^w$、路径$[S_0S_1]^w$、路径$[S_0S_2S_1]^w$等,搜索过程中部分路径重合,此时为减少搜索路径,减少时间和空间上的消耗,可对部分状态进行合并,简化状态模型。

由于三值逻辑模型存在不一致信息,当某一个或者多个不确定变量对多个性质检测产生影响,多值模型检测返回 M 时,需要增加信息,重新设置变量,这样会使检测过程非常复杂,因此建立三值逻辑模型时应遵循以下原则:

(1)尽量减少不确定的状态变量数;

(2)状态合并遵循三值命题逻辑,本文应用 Kleene 最强三值逻辑命题;

(3)三值系统模型要与原系统模型相一致。

如图 1(2)所示,若状态 S_0 与 S_1 合并,则只有 A 不确定;若 S_0 与 S_2 合并,则存在 A、B 两个不确定变量,所以应将 S_0 与 S_1 合并,其搜索路径为$[(S_{0,1})^wS_2]^w$,较图 1(1)明显减少。

三值模型检验是针对带有不确定和不一致信息的系统而建立起来的一套形式验证理论,是经典模型检验的扩展。下面给出了一种三值逻辑模型检测的基本过程,如图 2 所示。

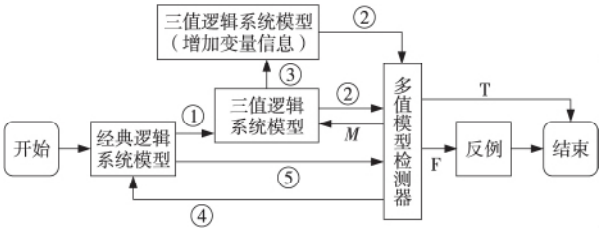

Figure 2　Process of 3-valued model checking validation

图 2　三值逻辑模型验证过程

①经典逻辑模型抽象为三值逻辑模型;

②在系统模型中,除要检测的状态变量外,其他变量都设置为 M(因不同的待验证系统性质与不同的系统变量相关,而系统模型却与所有系统变量相关,这样做的目的是为减少验证过程中不必要的计算),利用多值模型检测器检测状态是否满足系统所需的性质 P;若模型检测器返回 T 或 F,检测结束;

③如果检测器返回 M,返回原三值逻辑模型,增加信息,重新设置变量,再进行检测;

④三值逻辑模型的所有确定信息都被检测后,检测器仍返回 M,则返回到经典逻辑模型;

⑤多值模型检测器对经典模型进行验证,验证未知性质的真值,检测结束。

3　三值逻辑模型检测实现

模型检测是在描述系统的状态迁移系统 K 上验证时序逻辑公式 p 描述的系统性质是否可满足的一个过程。经典模型检测使用 Kripke 结构来对系统建模,而三值模型检验通常使用不完全 Kripke 结构对系统建模。本文对原有三值逻辑模型结

构(不完全 Kripke 结构)进行扩展,采用扩展的不完全 Kripke 结构 EPKS(Extend Partial Kripke Structure)来处理系统模型中出现的未知状态。

3.1 扩展不完全 Kripke 结构

定义 1[2] 不完全 Kripke 结构 P 是一个五元组(S,S_0,AP,R,L),其中:

(1)S:有限状态集合;

(2)$S_0\subseteq S$,初始状态集合;

(3)AP:原子命题集合;

(4)$R\subseteq S\times S$,是三值逻辑迁移关系函数;

(5)L:标签函数,$S\times P\rightarrow\{T,F,M\}$,$M$ 表示未知。

定义 2 扩展的不完全 Kripke 结构 EM 是一个七元组(S,S_0,AP,R,L,V,S'),其中:

(1)S:有限状态集合。

(2)$S_0\subseteq S$,初始状态集合。

(3)AP:原子命题集合。

(4)$R\subseteq S\times S$,是三值逻辑迁移关系函数。

(5)L:标签函数,$S\times AP\rightarrow\{T,F,M\}$,$M$ 表示未知。

(6)V:状态 s 的有序变量集。

(7)S':记录状态集合的每一个状态的信息,是一个二元组(s'',s_0''),其中:

①$s''\in s$,即状态 s 由哪些状态合并而成;$|s|=1$,则该状态未曾进行合并;

②$s_0''\in s$,表示合并状态 s 子集中的初始状态。

例如,图 1 中针对的扩展的不完全 Kripke 结构,其中 $S=\{S_{0,1},S_2\}$,$S_0=\{S_{0,1}\}$,$V(S_{0,1})=\{B,C,A\}$,$V(S_2)=\{A,B,C\}$;$S_{0,1}$ 子集 $S'_{0,1}=\{s_0,s_1\}$,其中 $s_0\in s_0''$;S_2 子集 $S'_2=\{s_2\}$,其中 $s'_2\in s''_0$。

3.2 三值逻辑模型检测算法

分支时态逻辑 CTL(Computation Tree Logic)常用于表示软/硬件属性,它是在命题逻辑的基础上加一些时态算子和路径量词来构成的。为了简化算法过程,本文只考虑三值逻辑下部分 CTL 公式,文献[10]对 CTL 逻辑公式 φ 的语义进行了定义。

定义 3[10] 设 P 是非空有限的原子命题逻辑公式集合,在 EPKS 下,CTL 公式可以用下列的抽象语法表示,其中 $p\in P$:$\varphi::=p\mid\neg\varphi\mid\varphi_1\wedge\varphi_2\mid\neg\bigcirc\varphi\mid\Diamond\bigcirc\varphi\mid\Diamond(\varphi_1\cup\varphi_2)\mid\Box(\varphi_1\cup\varphi_2)$。

CTL 公式在扩展的不完全 Kripke 结构 EM 中的语义定义如下:

(1)$[(EM,s)\models p]=L(s,p)$;

(2)$(EM,s)\models\varphi$,即$(EM,s)\models\varphi\wedge(EM,s)=M$,记为:$[(EM,s)\models\varphi]=comp[(EM,s)\models\varphi]$;

(3)$[(EM,s)\models\varphi_1\wedge\varphi_2]=[(EM,s)\models\varphi_1\wedge(EM,s)\models\varphi_1]$;

(4)$[(EM,s)\models\Box\bigcirc\varphi]=T$,即 $\forall s\in S,s\rightarrow s'$ 时,$(EM,s')\models\varphi$;

$[(EM,s)\models\Box\bigcirc\varphi]=F$,即 $\exists s\in S,s\rightarrow s'$ 时,$(EM,s')\models\varphi$;

(5)$[(EM,s)\models\Diamond\bigcirc\varphi]=T$,即 $\exists s\in S,s\rightarrow s'$ 时,$[(EM,s')\models\varphi]$;

$[(EM,s)\models\Diamond\bigcirc\varphi]=F$,即 $\forall s\in S,s\rightarrow s'$ 时,$(EM,s')\models\varphi$;

(6)$[(EM,s)\models\Diamond(\varphi_1\cup\varphi_2)]=T$,即 $\exists s_0,s_1,\cdots,\exists i\geqslant 0$,$(EM,s_i)\models\varphi_2$,且 $\forall j,0\leqslant j<i\rightarrow[EM,s_j]\models\varphi_1$;

$[(EM,s)\models\Diamond(\varphi_1\cup\varphi_2)]=F$,即 $\forall s_0,s_1,\cdots,\exists i\geqslant 0,(EM,s_i)\models\varphi_2\vee((EM,s_i)\models\varphi_2\wedge\exists j,0\leqslant j<i\rightarrow[[EM,s_j]\models\varphi_1]=F)$;

(7)$[(EM,s)\models\Box(\varphi_1\cup\varphi_2)]=T$,即 $\exists s_0,s_1,\cdots,\exists i\geqslant 0$,$(EM,s_i)\models\varphi_2$,且 $\forall j,0\leqslant j<i\rightarrow[EM,s_j]\models\varphi_1$;

$[(EM,s)\models\Box(\varphi_1\cup\varphi_2)]=F$,即 $\exists s_0,s_1,\cdots,\exists i\geqslant 0,(EM,s_i)\models\varphi_2\vee((EM,s_i)\models\varphi_2\wedge\exists j,0\leqslant j<i\rightarrow[[EM,s_j]\models\varphi_1]=F)$。

本文三值逻辑模型检测算法是在郭建的算法基础之上提出的。首先将三值模型 EM 和经典模型 CM 同时输入模型验证器;然后,对于每一个状态,都设置三个集合:(1)在此状态下为真的公式集合 $lableT()$;(2)此状态下公式为未知的公式集合 $lableM()$;(3)此状态下公式为假的公式集合 $labelF()$。

公式 1 $p(p\in P)$。首先对已知状态变量进行搜索,当已知状态变量子集可以确定 $[(EM,s)\models p]=T$ 时,把 s 归入 $lableT(s)$;否则,当 $[(EM,s)\models p]=F$ 时,把 s 归入 $lableF(s)$;当 $[(EM,s)\models p]=M$ 时,把 s 归入 $lableM(p)$。则相应算法 1 的伪码如下:

```
1. CheckEM(s,p)
2. { 调用 EM 输入变量集 V;
3.   { 变量子集 V_i 合并; check V_i→(EM,s) ⊨ p
     {if([V_i→(EM,s) ⊨ p]=T)返回 labelT(p);
4.   Else if([V_i→(EM,s) ⊨ p]=F)返回 labelF(p);
5.   Else i=i+1; } }
6. If(|V_i|==|V|)返回 labelM(p);}
```

公式 2 $\neg\varphi$。当状态$\{s\mid\varphi\in labelF(s)\}$时,把

φ归入 $labelT()$;当状态$\{s|\varphi\in labelT(s)\}$时,把$\varphi$归入 $labelF()$;当$\{s|\varphi\in labelM(s)\}$时,采用经典逻辑对状态 s 包含的子状态模型进行检测。设置两个状态集合公式:(1)当子状态模型中存在一条路径使得$[[CM,s]\quad\varphi]=T$,则我们将 φ归入 $labelET()$;(2)当子状态模型中存在一条路径使得$[[CM,s]\quad\varphi]=F$,则将 φ归入 $labelEF()$。算法 2 伪码如下:

① $CheckM(s,\quad\varphi)$/*查询未知状态是否满足 φ*/
② {调用 EM 搜索 s 的子状态集 s',Stop EM;调用经典逻辑模型 CM,从 s'_0 开始搜索 CM;
③ 如果存在一条路径满足 φ返回 $labelEF(s)$;
④ 如果存在一条路径不满足 φ返回 $labelET(s)$;
⑤ Stop CM;}

公式 3　$\varphi_1\wedge\varphi_2$。算法 3 描述:

(1)若$[\varphi_1\in labelT(s)\wedge\varphi_2\in labelT(s)]\vee[\varphi_1\in labelET(s)\wedge\varphi_2\in labelET(s)]$,则把公式$\varphi_1\wedge\varphi_2$ 归入到 $labelT()$中;

(2)若$[\varphi_1\in labelF(s)\vee\varphi_2\in labelF(s)]\vee[\varphi_1\in labelEF(s)\wedge\varphi_1\notin labelET(s)]\vee[\varphi_2\in labelEF(s)\wedge\varphi_2\notin labelET(s)]$,则把公式 $\varphi_1\wedge\varphi_2$ 归入到 $labelF(s)$中。

公式 4　$\bigcirc\varphi$ 与$\Diamond\bigcirc\varphi$。根据$\bigcirc\varphi$ 与$\Diamond\bigcirc\varphi$ 的语义,可以推出$\bigcirc\varphi=\quad\Diamond\bigcirc\quad\varphi$,所以只要证明$\Diamond\bigcirc\varphi$ 即可。算法 4 描述:

(1)对$\forall s_i,\exists s_{i+1},(s_i,s_{i+1})\in R$ 且 $\varphi\in[labelT(s_{i+1})\vee labelET(s_{i+1})]$,则把公式$\Diamond\bigcirc\varphi$ 归入 $labelT(s)$中;

(2)对$\forall s_i,\exists s_{i+1},(s_i,s_{i+1})\in R$ 且 $\varphi\in[labelF(s_{i+1})\vee[\varphi\in labelEF(s_{i+1})\wedge\varphi\notin labelET(s)]]$,则把公式$\Diamond\bigcirc\varphi$ 归入 $labelF(s)$中。

公式 5　$\Diamond(\varphi_1\cup\varphi_2)$。若$[(EM,s)\quad\Diamond(\varphi_1\cup\varphi_2)]=T$,则$\exists Path(s_0),\exists s_i\quad\varphi_2\wedge\forall s_j\quad\varphi_1,0\leqslant j<i<n$。所以,此算法是求出满足$\Diamond(\varphi_1\cup\varphi_2)$的一条路径。算法 5 描述如下:

(1)首先,搜索所有使 φ_2 为真的状态集合 I,若不存 I,则将$\Diamond(\varphi_1\cup\varphi_2)$加入到 $labelF(s)$;若存在转到步骤(2)。

(2)对$\forall s_i,s_i\in I$,检测$\forall s_j\quad\varphi_1,0\leqslant j<i$,则将$\Diamond(\varphi_1\cup\varphi_2)$加入到 $labelT(s)$中,否则加入到 $labelF(s)$中。当 I 中的所有状态遍历完毕,就可以找到满足$\Diamond(\varphi_1\cup\varphi_2)$的路径。算法 5 伪码如下:

① $CheckE(\varphi_1,\varphi_2)$/* 查询是否存在一条路径满足 $\varphi_1\cup\varphi_2$*/
② {查询满足 φ_2 的状态集 I;
③ 如果 $I\neq\varnothing${查询 s_i 之前的所有状态 s_j 是否满足 $\varphi_1,s_i\in I$,$1\leqslant j<i-1$;
④ {如果 $\varphi_1\in labelT(s_j)\vee\varphi_1\in labelET(s_j)$
{Input $\Diamond(\varphi_1\cup\varphi_2)$ into $labelT()$; $j=j-1$;}
⑤ 否则 Input $\Diamond(\varphi_1\cup\varphi_2)$ into $labelF(s_j)$;}
⑥ 如果 $j=0$ Input $\Diamond(\varphi_1\cup\varphi_2)$ into $labelT()$;}
⑦ 否则 Input$\Diamond(\varphi_1\cup\varphi_2)$ into $labelF()$;}

公式 6　$\Box(\varphi_1\cup\varphi_2)$。若$[(EM,s)\quad\Box(\varphi_1\cup\varphi_2)]=T$,则$\forall Path(s_0),\exists s_i\quad\varphi_2\wedge\forall s_j\quad\varphi_1,0\leqslant j<i<n$。可以根据$\Diamond(\varphi_1\cup\varphi_2)$算法,求$\exists Path(s_0)$,$[(EM,s)\quad\Box(\varphi_1\cup\varphi_2)]=F$,则$\Box(\varphi_1\cup\varphi_2)$归入 $labelF()$;反之归入 $labelT()$。算法 6 描述如下:

(1)首先,对 CTL 树先序遍历,对$\forall Path(s_0),\exists s_i\quad\varphi_1\wedge\varphi_2$,则$\Box(\varphi_1\cup\varphi_2)$加入到 $labelF()$,搜索完毕;若$\forall s_j\quad\varphi_1\wedge s_i\quad\varphi_2$,此条路径标记为 true,搜索下一条路径。

(2)若 $s_i\quad\varphi_1$,则将 φ_1 在此状态下的真值入栈;继续搜索,直到后继节点中存在 $s\quad\varphi_2$,若该条路径中没有满足 φ_2 的节点,则将$\Box(\varphi_1\cup\varphi_2)$加入 $labelF()$;跳出程序。

(3)对 CTL 树的所有路径进行搜索,若存在一条路径标记为 false,停止搜索,将$\Box(\varphi_1\cup\varphi_2)$加入到 $labelF()$;反之加入 $labelT()$。其伪码如下:

① $PreCheckAL(A,Node)$/* 先序查询所有路径是否满足 A,其中 A 表示$\Box(\varphi_1\cup\varphi_2)$,$Node$ 表示节点 */
② {Boolen $flag$;/*标记*/
③ 如果 $Node$ 不为空{搜索左子树;
④ {如果 $\varphi_1\in labelF(s)\vee\varphi_1\in labelEF(s)$
⑤ {如果 $(\varphi_2\in labelF(s)\vee\varphi_2\in labelEF(s)$ {停止搜索; Input A into $labelF()$; 跳出程序;}
⑥ 否则 {$flag=1$; Input A into $labelT()$;}
⑦ 递归查询右子树;}
⑧ 否则{设置一个栈来存放公式 f 在此状态的真值;继续搜索;}}
⑨ 如果 $flag=0${Input A into $labelF()$; 停止搜索;跳出程序;}
⑩ 否则 Input A into $labelT()$;}

4　算法实例

本节使用三值模型算法对实例 1 和实例 2 的 CTL 逻辑公式进行验证。如图 3 所示,实例 1 中将二值逻辑结构中 S_1 和 S_4 进行合并,其初始状态为 S_0,其中:$L(S_0)=\{p,q,\quad r,\quad z\}$;$L(S_{1,4})=\{p,M,\quad r,z\}$;$L(S_2)=\{\quad p,q,r,z\}$;$L(S_3)=\{p,q,\quad r,z\}$;$L(S_5)=\{p,\quad q,r,z\}$。

在这里我们验证公式 $(p\wedge q)\wedge\Diamond(r\wedge z)$,

我们用公式 $s(f)$ 表示满足公式 f 的状态集合。

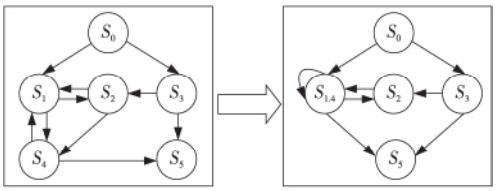

Figure 3 CM to EM 3-valued model checking process of case 1

图 3 实例 1 CM 到 EM 的三值模型检测过程

首先我们根据算法 1,对满足各性质的状态进行搜索,加入各性质集合中,如表 1 所示。

Table 1 Properties checking results of case 1

表 1 实例 1 性质检测结果

P	$labelT(P)$	$labelF(P)$	$labelM(P)$
p	$\{S_0,S_{1,4},S_3,S_5\}$	$\{S_2\}$	$\varnothing$
q	$\{S_0,S_2,S_3\}$	$\{S_5\}$	$\{S_{1,4}\}$
r	$\{S_2,S_5\}$	$\{S_0,S_{1,4},S_3\}$	$\varnothing$
z	$\{S_{1,4},S_2,S_3,S_5\}$	$\{S_0\}$	$\varnothing$

由表 1 可以得出,在检测性质 q 时,存在不可判定的状态 $S_{1,4}$,所以需要增加判定条件来确定 $S_{1,4}$ 是否满足性质 q,这时调用算法 2,根据 EM 模型找到 $S_{1,4}$ 在经典逻辑模型 CM 中包含的子状态集 $\{S_1,S_4\}$,其中 S_1 是局部模型的初始状态,在 CM 中找到该局部模型的初始状态,然后开始进行该局部状态搜索,假设找到了一条路径使得 $S_{1,4}$ 满足 q,同时也找到了一条路径不满足性质 q,则将 $S_{1,4}$ 归入到:$labelET(q)=\{S_0,S_{1,4},S_2,S_3\}$ 和 $labelEF(q)=\{S_{1,4},S_5\}$ 中;然后根据算法 3,可得出 $labelF(p\wedge q)=\{S_2,S_5\}$,即 $labelT[(p\wedge q)]=\{S_2,S_5\}$。

由于 $labelT[(p\wedge q)]$ 中不存在 S_0,则 $labelT[(p\wedge q)\wedge\Diamond(r\wedge z)]$ 中不存在 S_0,所以此公式在 EM 结构中真值为假,输出:$(p\wedge q)\wedge\Diamond(r\wedge z)=$F;搜索到此结束。

实例 2 中使用三值模型算法对进程互斥的 CTL 性质进行验证。如图 4 所示。

将二值逻辑结构中的多个状态进行合并,初始状态集为 $\{S_0\}$,其中性质集:

$m:y=0\wedge Crit1=$T; $n:y=0\wedge Crit2=$T; $o:Wait1=$T; $u:Wait2=$T; $w:y=1$。

有:$L(S_0)=\{m,n,o,u,w\}$; $L(S_{1,4})=\{m,n,o,M,w\}$; $L(S_{2,3})=\{m,n,o,M,w\}$; $L(S_5)=\{m,n,o,u,w\}$; $L(S_{6,7})=\{m,n,M,u,w\}$。

验证是否存在一条路径使资源最终被占用,时序逻辑为:$\Diamond(m\wedge n)\cup((o\wedge u\wedge w)\vee((o\wedge u\wedge w))$。首先由算法 1 对状态进行搜索,加入各性质集合中,如表 2 所示。

Table 2 Properties checking results of mutual exclusion

表 2 互斥进程性质检测结果

P	$labelT(P)$	$labelF(P)$	$labelM(P)$
m	$\{S_{2,3}\}$	$\{S_0,S_{1,4},S_5,S_{6,7}\}$	$\varnothing$
n	$\{S_{6,7}\}$	$\{S_0,S_{1,4},S_{2,3},S_5\}$	$\varnothing$
o	$\{S_{1,4}\}$	$\{S_0,S_{2,3},S_5\}$	$\{S_{6,7}\}$
u	$\{S_5\}$	$\{S_0,S_{6,7}\}$	$\{S_{1,4},S_{2,3}\}$
w	$\{S_0,S_{1,4},S_5\}$	$\{S_{2,3},S_{6,7}\}$	$\varnothing$

由表 2 可知,存在 $o(S_{6,7})=M$,根据算法 2 对 $S_{6,7}$ 进行扩展,找到 $S_{6,7}$ 在经典逻辑模型 CM 中的子状态集 $\{S_6,S_7\}$,然后从 S_6 开始局部搜索 CM,存在 $Path(S_6)$ 使得 $S_{6,7}$ 满足 r,则 $labelET(m)=\{S_{6,7}\}$;同理 $u(S_{1,4})=M$ 和 $u(S_{2,3})=M$,得 $labelET(u)=\{S_{1,4},S_{2,3}\}$。根据算法 4 搜索 $s((o\wedge u\wedge w)\vee((o\wedge u\wedge w))$,存在集合 $I=\{S_{2,3},S_{6,7}\}$;然后搜索 $S_{2,3}$、$S_{6,7}$ 前的所有状态是否满足 $s(m\wedge n)$,得 $labelT(m\wedge n)=\{S_0,S_{1,4},S_5\}$,所以验证结果为真,输出结果为:$\Diamond(m\wedge n)\cup((o\wedge u\wedge w)\vee((o\wedge u\wedge w))=$T;搜索结束。

本文使用 NuSMV 对以上两个实例经典逻辑模型的 CTL 公式进行验证,验证结果与三值逻辑算法验证结果一致。然后采用实例 2 将本文算法与郭建算法进行对比,其对比数据如表 3 所示。根据算法对比结果,本算法对所有未知性质计算并得出其真值;而采用郭建算法有六条性质未知,并且未提出扩展方法,算法不完整,故本文采用数学方法对算法复杂度进行分析。

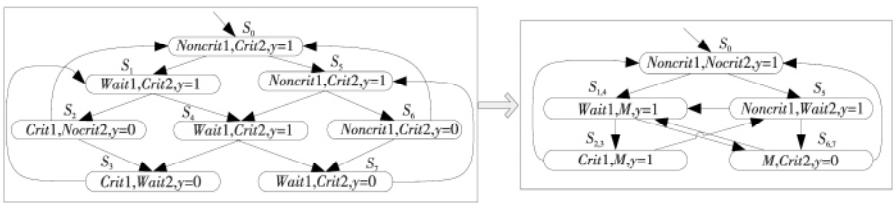

Figure 4 CM to EM 3-vallued model checking process of mutual exclusion

图 4 进程互斥的 CM 到 EM 的三值模型检测过程

郭建算法的时间复杂度为 $O[(|S|+|R|)\cdot|f|]$,同一逻辑公式 $O(|f|)$ 是相同的,所以我们只考虑 $O(|S|)+O(|R|)$ 的大小。本文 $O(|S|)$ 由两部分组成,当三值逻辑模型判断性质为 M 时要返回到二值逻辑进行判断,所以其时间复杂度为:$O(|S|)=O(|S'|+|s'|)$,其中 $O(|s'|)$ 为二值逻辑模型检测的时间复杂度,根据上文的合并准则:$|s'|<<|S'|$,而郭建的为 $O(|S|)=O(|S'|)$,其中 $|S|$ 是关于状态数 n 的关系式,有:

$$\lim_{n\to\infty}\left(\frac{|S'|+|s'|}{|S'|}\right)=1$$

所以,$O(|S|)$ 复杂度也相同。

Table 3　Results of algorithms comparison

表 3　算法对比结果

模型结构 Model Structure	输入状态数 Input State-Nums	性质数 P-Nums	验证结果 Verification-Results		
			T-Nums	F-Nums	M-Nums
EPKS	13	21	12	9	0
PKS	5	21	7	8	6

本文对状态变量进行了合并,对于 $A_1\ op(A_2\ op\ A_3\ op\cdots\ op\ A_n)$ 检测执行时,先考虑 A_1 真值,其他变量均为 Unknown 的情况,检测性质;如系统返回为 Unknown,信息不足,必须增加判定条件,如考虑 A_1 和 A_2 的组合,其他变量均为 Unknown,以此类推。如果这样的判定节点在流程图路径中出现两次,节点平均处理次数为 $(4+2^{2n})/2$,最坏情况下的关系时间复杂度为 $O(|R|)=2^{2n}$,而郭建算法在所有未知状态都无需扩展的情况下,即最佳情况下为 $O(|R|)=2^{2n}$,所以,本文三值逻辑模型检测算法的 $O(|R|)$ 较小。

综上,本文算法时间复杂度低于郭建的三值逻辑模型检测算法,而在空间复杂度上,由于本算法还需调用二值逻辑模型,其空间复杂度较大。所以,要求时间复杂度可以选择本算法,若要求空间复杂度则郭建的算法较优。

5　结束语

本文提出的三值逻辑模型检测是在郭建的三值逻辑模型检测算法和 Clarke E 的二值逻辑模型检测的基础上修改得到的。在郭建的三值逻辑的基础上,进行了状态变量合并搜索,并对使公式未知的状态进行了扩充,用 Clarke E 的算法对二值逻辑模型进行搜索,找出满足 f 的二值逻辑状态集,为此设置一个 $labelET()$;并且找出不满足 f 的状态集,设置 $labelEF()$、$labelET$ 和 $labelEF$ 都是用来记录未知状态的真值性。

在本算法中,因为对未知状态进行了扩充,所以未知公式的真值是确定的,此时 $labelT$、$labelF$、$labelET$、$labelEF$ 是同时进行的,在计算子公式为真时,我们不用考虑公式为未知的情况。相对于郭建的算法,本算法不仅更简单精确,而且对其无法处理的未知状态问题也提出了解决方法,是一种可行的方法,能够解决一些领域由于状态爆炸无法进行模型检测的问题。

参考文献:

[1] Clarke E, Grumberg O, Peled D. Model checking[M]. Cambridge, MA: The MIT Press, 1999.

[2] Chechik M, Devereux B, Easterbrook S. Implementing a multi-valued symbolic model checker[C]// Proc of Tools and Algorithms for the Construction and Analysis of Systems, 2001: 404-419.

[3] Chechik M, Devereux B, GurfinkelA. xChek: A multi-valued model checker[C]// Proc of the 14th International Conference on Computer-Aided Verification, 2002: 505-509.

[4] Chechik M, Devereux B, Easterbrook S, et al. Multi-valued symbolic model checking[J]. ACM Transactions on Software Engineering and Methodology, 2003, 12(4): 371-408.

[5] Gurfinkel A, Wei O, Chechik M. Model checking recursive programs with exact predicate abstraction [M]. Berlin: Springer, 2008.

[6] Bolc L, Borowik P. Many-valued logics[J]. Oxford: Clarendon Press, 1992.

[7] Li Yong-ming, Li Ya-li, Ma Zhan-you. Computer tree logic model checking based on possibility measures [J]. Fuzzy Sets and System, 2015, 262(5): 44-49.

[8] Pan Hai-yu, Li Yong-ming, Cao Yong-zhi, et al. Model checking fuzzy computation tree logic [J]. Fuzzy Sets and System, 2015, 262(5): 60-77.

[9] Guo Jian, Han Jun-gang. Model checking using partial Kripke structure with three-valued temporal logic [J]. Computer Science, 2006, 33(3): 263-278. (in Chinese)

[10] Bruns G, Godefroid P. Model checking partial state spaces with 3-valued temporal logics[M]// Computer Aided Verification. Berlin: Springer, 1999: 274-287.

附中文参考文献:

[9] 郭建,韩俊刚. 基于不完全 Kripke 结构三值逻辑的模型检验[J]. 计算机科学, 2006, 33(3): 263-278.

作者简介:

刘姣(1990-),女,陕西咸阳人,硕士生,CCF 会员(50222G),研究方向为模型检测。**E-mail**: liujiao@snnu.edu.cn

LIU Jiao, born in 1990, MS candidate, CCF member(50222G), her research interest includes model checking.