软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2015,26(8):1968−1982 [doi: 10.13328/j.cnki.jos.004689]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

一个命题投影时序逻辑符号模型检测器*

逄 涛[1,2], 段振华[1,2], 刘晓芳[1,2]

[1](西安电子科技大学 计算理论与技术研究所,陕西 西安 710071)

[2](综合业务网理论与关键技术国家重点实验室(西安电子科技大学),陕西 西安 710071)

通讯作者: 段振华, E-mail: zhduan@mail.xidian.edu.cn

摘 要: 现有模型检测工具的形式化规范语言,如计算树逻辑(computation tree logic,简称 CTL)和线性时序逻辑(linear temporal logic,简称 LTL)等的描述能力不足,无法验证ω正则性质.提出了一个命题投影时序逻辑(propositional projection temporal logic,简称 PPTL)符号模型检测工具——PLSMC(PPTL symbolic model checker)的设计与实现过程.该工具基于著名的符号模型检测系统 NuSMV,实现了 PPTL 的符号模型检测算法.PLSMC 的规范语言 PPTL 具有完全正则表达能力,这使得定性性质和定量性质均可被验证.此外,PLSMC 可以有效地缓解模型检测工具中容易发生的状态空间爆炸问题.最后,利用 PLSMC 对铁路公路交叉道口护栏控制系统的安全性质和周期性性质进行验证.实验结果表明,PPTL 符号模型检测工具扩充了 NuSMV 系统的验证能力,使得时间敏感、并发性和周期性等实时性质可以被描述和验证.

关键词: 符号模型检测;时序逻辑;模型检测器;嵌入式系统验证

中图法分类号: TP181

中文引用格式: 逄涛,段振华,刘晓芳.一个命题投影时序逻辑符号模型检测器.软件学报,2015,26(8):1968−1982. http://www.jos.org.cn/1000-9825/4689.htm

英文引用格式: Pang T, Duan ZH, Liu XF. Symbolic model checker for propositional projection temporal logic. Ruan Jian Xue Bao/Journal of Software, 2015,26(8):1968−1982 (in Chinese). http://www.jos.org.cn/1000-9825/4689.htm

Symbolic Model Checker for Propositional Projection Temporal Logic

PANG Tao[1,2], DUAN Zhen-Hua[1,2], LIU Xiao-Fang[1,2]

[1](Institute of Computing Theory and Technology, Xidian University, Xi'an 710071, China)

[2](State Key Laboratory of Integrated Services Networks (Xidian University), Xi'an 710071, China)

Abstract: The formal specification languages for existing model checking tools such as computation tree logic (CTL) and linear temporal logic (LTL) are not powerful enough to describe ω-regular properties, in that those properties cannot be verified with them. In this study, a design and implementation procedure of propositional projection temporal logic (PPTL) symbolic model checker (PLSMC) is developed by implementing the symbolic model checking algorithm for PPTL from author's previous work based on the acclaimed symbolic model checking system NuSMV. As PPTL has the expressive power of full-regular expressions, both qualitative and quantitative properties can be verified with PLSMC. Moreover, PLSMC is an effective model checking tool to tackle the state space explosion problem. Finally, safety and iterative properties of a railway and highway crossing guardrail control system are checked with PLSMC. Experimental results show that the presented symbolic model checker for PPTL extends the validation functionality of the NuSMV system such that state sensitive, concurrent and periodic properties can be specified and verified with PPTL.

Key words: symbolic model checking; temporal logic; model checker; embedded system verification

* 基金项目: 国家自然科学基金(61133001, 61202038, 61272117, 61272118, 61322202); 国家重点基础研究发展计划(973) (2010CB328102)

收稿时间: 2013-05-20; 修改时间: 2013-09-09; 定稿时间: 2014-07-01

模型检测[1]是一种验证并发系统性质的重要形式化方法,它将被验证系统的行为建模为有限状态机,如Kripke 结构[2]、状态迁移系统或者自动机;将系统期望的性质描述为时序逻辑公式.然后,穷举搜索有限状态机以检查性质是否满足,并在不满足时提供反例.相对于其他形式化方法,如定理证明[3]和等价性验证[4]等,模型检测具有自动化程度高、不需要用户干预、在验证失败后可以给出反例等优点.近 30 年来,提出了很多智能和优秀的模型检测工具,如 SPIN[5,6],NuSMV[7],NuSMV2[8],CHESS[9]和 BLAST[10]等,用于通信协议、软硬件系统和嵌入式系统等的验证.特别地,NuSMV 是由 Carnegie Mellon University 和 Istituto per la Ricerca Scientifica e Techolgica(IRST)联合开发的基于计算树逻辑(computation tree logic,简称 CTL)[11]和线性时序逻辑(linear temporal logic,简称 LTL)[12]的符号模型检测系统.它非常适合于硬件电路和嵌入式系统的验证,可以有效地缓解 SPIN 和 CHESS 等显式状态模型检测工具可能出现的组合状态空间爆炸问题,并与 BLAST 等基于谓词抽象方法的模型检测系统互为补充,因此受到了学术界和工业界的广泛认可.然而,上述模型检测工具的规范语言如 CTL 和 LTL 等的描述能力有限[13],不足以描述ω正则语言的性质,如"请求信号的振荡频率为 4MHz(即,每 250 个时钟周期发生一次振荡)"、"在收到请求信号的第 15 个~第 40 个时钟周期内授权信号有效"等,使得一些时间敏感、并发性和周期性等实时相关的性质不能被描述和验证.因此,扩展现有模型检测工具形式化规范语言的描述能力具有重要意义.

命题投影时序逻辑(propositional projection temporal logic,简称 PPTL)[14,15]引入了 projection 等新的操作符,且描述能力等价于完全正则语言[16],它很适合被用作模型检测的规范语言.性质"请求信号的振荡周期为 250 个时钟周期"和"在收到请求信号的第 15 个到第 40 个时钟周期内授权信号有效"可以方便地被描述为 PPTL 公式:

$$len(250)^{*}\,\mathrm{prj}\,\Box request \text{ 和 } request\rightarrow len(15);\Diamond grant\wedge len(25);\mathrm{true}.$$

文献[5]中提出了基于 SPIN 的命题投影时序逻辑模型检测方法,扩展了 SPIN 的功能,使得它能够支持 PPTL 公式的验证.它将以 Promela 语言描述的系统模型和以 Never-Claim 结构描述的 PPTL 公式分别转化为 Büchi 自动机,并将两个自动机求积,通过判断积自动机接受字是否为空的方法验证系统模型是否满足期望的性质.然而该工具基于显式状态空间描述,当被验证系统规模过大或者期望的性质较为复杂时,极易引发组合状态空间爆炸问题.此外,该工具仅能处理以有限模型的 PPTL 公式描述的性质,无法验证无限模型的 PPTL 公式.

针对上述问题,文献[17]提出了 PPTL 的符号模型检测方法.它使用规约有序二叉决策图(reduced ordered binary decision diagram,简称 ROBDD)[18]符号化描述被验证系统模型,借助于 ROBDD 的压缩存储和有效地状态空间操作机制应对状态空间爆炸问题;使用 PPTL 公式描述系统期望的性质,借助于 PPTL 的完全正则表达能力扩展模型检测形式化规范的描述能力.此外,该方法同时支持对于有限和无限模型 PPTL 公式的验证.

目前,已有基于 CTL 和 LTL 的符号模型检测工具,如 NuSMV[7]和 NuSMV2[8]等,但上述工具均不支持 PPTL 的符号模型检测方法.本文基于 NuSMV 系统和 PPTL 符号模型检测方法的特点,提出了 PPTL 符号模型检测工具 PLSMC 的设计和实现过程.该工具首先通过语言解析模块解析以 SMV[19]语言符号化描述的系统模型 $M=(S, I,R,L)$和以 PPTL 公式 φ 描述的系统期望性质,分别得到系统模型的语法树和 PPTL 性质公式的范式[14];其次,由编译模块对语法树执行扁平化、状态变量以及迁移关系编码等操作,得到符号化的系统模型;再次,依据 NuSMV 提供的符号化状态空间遍历机制和模型检测模块中实现的 PPTL 符号模型检测方法递归地计算状态结合 S 中满足$\neg\varphi$的状态集合 $Sat(\neg\varphi)$;最后,将 $Sat(\neg\varphi)$与初始状态集合 I 进行集合与操作,通过判断与操作结果是否为空集检测系统模型是否满足期望的性质,并在不满足时给出反例路径.与其他模型检测工具相比,PPTL 符号模型检测工具 PLSMC 的主要贡献如下:

(1) 该工具的性质规范语言 PPTL 具有完全正则表达能力,使得时间敏感性质、周期性质和并发性质等实时相关性质可以被方便地描述和验证,而其他工具的规范语言描述能力有限,无法完整地描述上述性质.

(2) 该工具基于著名的符号模型检测工具 NuSMV 和 PPTL 符号模型检测方法实现,可以有效地应对基于显式状态空间描述的模型检测工具(如 SPIN 和 CHESS 等)中容易出现的组合状态空间爆炸问题.

(3) 该工具可验证基于 SPIN 的 PPTL 模型检测工具无法验证的以无限模型 PPTL 公式描述的性质.

本文第 1 节简要介绍命题投影时序逻辑的基本概念,包括语法和语义等.第 2 节介绍命题投影时序逻辑符号模型检测方法的基本思想.第 3 节给出 PLSMC 工具的设计和实现过程.第 4 节通过对铁路公路交叉道口护栏控制系统的验证实例来展示 PLSMC 工具的正确性.第 5 节通过若干实验结果来展示 PLSMC 工具的运行效率.第 6 节是对本文的总结和对未来工作的展望.

1 命题投影时序逻辑

命题投影时序逻辑[14,15]是一种描述并发系统性质的有效形式化语言,它引入了新的投影时序操作符.设 *Prop* 是一个可数原子命题集合,PPTL 公式的语法定义如下:

$$\varphi ::= p \mid \bigcirc \varphi \mid \neg \varphi \mid \varphi_1 \vee \varphi_2 \mid (\varphi_1,...,\varphi_m) \text{ prj } \varphi.$$

其中,$p \in Prop$,$\varphi_i(1 \leqslant i \leqslant m)$和$\varphi$为一般的 PPTL 公式;“○”(next)和“prj”(projection)为时序操作符.如果一个 PPTL 公式包含时序操作符,则称为时序公式;否则,称为状态公式.

状态 s 定义为从 *Prop* 到集合 $B=\{\text{true,false}\}$的映射 $s:Prop \to B$.用 $s[p]$来表示命题 p 在 s 状态下的值,$s[p]=\text{true}$表示 p 在 s 状态下成立;反之,不成立.

区间$\sigma=\langle s_0,s_1,...,s_{|\sigma|}\rangle$是一条有限或者无限状态序列.对于有限区间,区间长度$|\sigma|$为状态数减 1;对于无限区间,$|\sigma|=\omega$,其中,状态 $s_{|\sigma|}$未定义.另外,$\sigma_1 \bullet \sigma_2$ 表示由区间σ_1 的结束状态与区间σ_2 的起始状态连接成一个新的区间.$\sigma_{(i..j)}$表示σ的子区间$\langle s_i,...,s_j\rangle$.

为定义投影操作符的语义,需要定义辅助操作符↓.设区间$\sigma=\langle s_0,s_1,...\rangle$,$r_1,...,r_h(h \geqslant 1)$是整数且 $0 \leqslant r_1 \leqslant ... \leqslant r_h \leqq |\sigma|$.其中,“≦”定义为$\leqslant -\{\omega,\omega\}$,表示≤中不包含$\omega=\omega$的情况.$\sigma$在序列 $r_1,...,r_h$ 的投影为

$$\sigma \downarrow (r_1,...,r_h) = \langle s_{t_1}, s_{t_2}, ..., s_{t_l} \rangle,$$

其中,$t_1,...,t_l$是删除 $r_1,...,r_h$ 中的重复元素所得的严格递增序列子序列,例如,$\langle s_0,s_1,s_2,s_3\rangle \downarrow (0,0,2,2,3)=\langle s_0,s_2,s_3\rangle$.

解释 $I=(\sigma,k,j)$是一个三元组,其中,σ表示一个区间,k 为整数,j 为整数或者ω,$0 \leqslant k \leqq j \leqslant |\sigma|$.$I \vDash \varphi$表示解释 $I=(\sigma,k,j)$满足公式φ.解释的归纳定义如下:

- $I \vDash p$ 当且仅当 $s[p]=\text{true}$;
- $I \vDash \neg\varphi$当且仅当 $I \nvDash \varphi$;
- $I \vDash \varphi_1 \vee \varphi_2$ 当且仅当 $I \vDash \varphi_1$ 或 $I \vDash \varphi_2$;
- $I \vDash \bigcirc\varphi$当且仅当 $k<j$ 且$(\sigma,k+1,j) \vDash \varphi$;
- $I \vDash (\varphi_1,...,\varphi_m)$ prj φ,如果存在整数序列 $k=r_0 \leqslant r_1 \leqslant ... \leqslant r_m \leqslant j$,使得$(\sigma,r_0,r_1) \vDash \varphi_1$,$(\sigma,r_{l-1},r_l) \vDash \varphi_l(1 \leqslant l \leqslant m)$,并且对于以下 2 种情况有$(\sigma',0,|\sigma'|) \vDash \varphi$:
  (1) $r_m<j$ 并且$\sigma'=\sigma\downarrow(r_0,...,r_m)_\bullet, \sigma_{(r_m+1...j)}$, 或者
  (2) $r_m=j$ 并且有$\sigma'=\sigma\downarrow(r_0,...,r_h)$,其中,$0 \leqslant h \leqslant m$.

如果$(\sigma,0,|\sigma|) \vDash \varphi$,那么公式$\varphi$在区间上$\sigma$是可满足的,记作$\sigma \vDash \varphi$;如果存在$\sigma$使得$\sigma \vDash \varphi$,则公式$\varphi$是可满足的;如果对于任意的$\sigma$,$\sigma \vDash \varphi$,则公式$\varphi$是有效的.常用的派生公式为

$$empty \equiv \neg\bigcirc\text{true}, \quad more \equiv \neg empty, \quad skip \equiv len(1), \quad \Diamond\varphi \equiv \text{true};\varphi, \quad \Box\varphi \equiv \neg\Diamond\neg\varphi$$

$$len(n) \equiv \bigcirc^n empty, \quad \bigcirc^0\varphi \equiv \varphi, \quad \bigcirc^n\varphi \equiv \bigcirc(\bigcirc^{n-1}\varphi), \quad \varphi_1;\varphi_2 \equiv (\varphi_1,\varphi_2) \text{ prj } empty.$$

直观来讲,上述 PPTL 公式在状态区间上的解释如下:

- $\bigcirc\varphi$表示φ在区间的下一状态成立.
- $(\varphi_1,...,\varphi_m)$ prj φ用两个不同的时间粒度刻画进程,如图 1 所示:细粒度的是由$\varphi_1,...,\varphi_m$顺序复合而成的局部序列;粗粒度的是并行执行φ的全局序列.即,φ与$\varphi_1,...,\varphi_m$ 在一个区间上并行执行,φ的执行区间由$\varphi_1,...,\varphi_m$各自执行区间的端点组成.
- *empty* 表示当前区间长度为 0.
- $\Diamond\varphi$表示φ将在区间将来某个状态被满足.

- □φ表示φ将在区间将来所有状态被满足.
- *len*(n)表示区间长度为 n.
- $\bigcirc^n\varphi$表示φ将在区间将来第 n(n>0)个被满足.
- (φ_1;φ_2)表示φ_1和φ_2被区间顺序满足.

定义 1(**范式**). 设$\varphi_p \subseteq Prop$ 为 PPTL 公式φ中出现的原子命题的集合,φ的范式为

$$(\varphi_e \wedge empty) \vee (\vee_{i=1}^{m} (\varphi_i \wedge \bigcirc \varphi_i')),$$

其中,φ_e和φ_i可以为 true,或者形为 $\dot{r}_1 \wedge ... \wedge \dot{r}_n$ 的合取式,$l=|\varphi_p|$,$0 \leqslant m \leqslant 2^l$,$0 \leqslant n \leqslant l$;$r_n$ 属于φ_p,且对于任意 $r \in \varphi_p$,$\dot{r}$ 表示 r 或者$\neg r$;φ_i'是一个一般的 PPTL 公式.

如图 2 所示,范式由 present 部分和 future 部分组成.present 部分是一个状态公式,如φ_e和φ_i;future 部分可以是 *empty* 或者是用"○"操作符修饰的一般 PPTL 公式,如○ φ_i'.其中,$\varphi_e \wedge empty$ 表示 present 部分φ_e在当前状态 s_0 成立,且 s_0 是满足公式$\varphi_e \wedge empty$ 的区间的最后一个状态;$\varphi_i \wedge \bigcirc \varphi_i'$ 表示 present 部分φ_i在区间σ的当前状态 s_0' 成立,而 future 部分在子区间上 $\langle s_1', s_2', s_3', s_4', s_5', ... \rangle$ 满足,且有$\sigma \vDash \varphi_i \wedge \bigcirc \varphi_i'$.

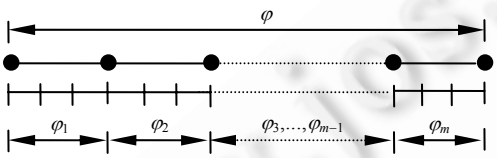


Fig.1 Semantics of formula (φ_1,φ_2,...,φ_m) prj φ

图 1 公式(φ_1,φ_2,...,φ_m) prj φ的语义

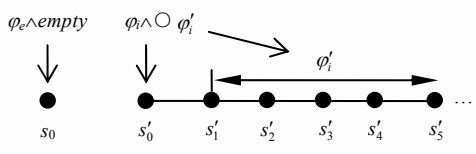


Fig.2 Normal form of PPTL

图 2 PPTL 的范式

PPTL 公式的范式刻画了满足该公式的状态集合的特征,已经证明,可以将任意 PPTL 公式等价地转化为其范式,转化算法的具体细节参见文献[20].

2 PPTL 符号模型检测算法

模型检测将被验证系统行为建模为有限状态机 M;将系统期望的性质描述为时序逻辑公式φ.通过检查 M 是否满足φ来验证系统模型是否满足期望的性质.由于系统模型大多基于显式状态空间描述,可验证系统的规模十分有限;在实际系统设计过程中,系统模型往往随系统并发组件的数目呈指数级增长.因此,模型检测受制于组合状态空间爆炸问题.此外,现有的模型检测方法的规范语言描述能力有限,使得周期性和并发性等实时相关性质无法被描述和验证.

针对上述问题,文献[17]中提出了命题投影时序逻辑的符号模型检测方法.由于符号模型检测提供基于 ROBDD 的状态空间压缩存储和操作机制,加之 PPTL 的表达能力等价于完全正则语言[16],因此该算法可有效地应对组合状态空间爆炸问题,并且提高模型检测规范语言的描述能力.

在 PPTL 符号模型检测方法中,集合的操作均由基于 ROBDD 的布尔操作实现,其基本思想如下:

(1) 使用 ROBDD 符号化描述以 Kripke 结构 $M=(S,I,R,L)$建模的待验证系统模型,用 PPTL 公式φ刻画系统期望的性质.

(2) 将性质公式φ等价地转换成其范式φ_{NF},并根据范式语义从φ的最简单子公式开始,递归地求出模型 M 中满足φ的子公式的状态集合,并逐步扩展,最终得到满足φ的状态集合 *Sat*(φ).

(3) 以状态集合 S 为全集对 *Sat*(φ)求补集,得到满足$\neg\varphi$的状态集合 *Sat*($\neg\varphi$),并判断 *Sat*($\neg\varphi$)和初始状态集合 $I \subseteq S$ 的交集 *Sat*($\neg\varphi$)$\cap I$ 是否为空:如果为空,则系统模型 M 满足性质φ;否则不满足,且从 *Sat*($\neg\varphi$)$\cap I$ 中的任意的状态出发都可以求出一条反例路径.

3 PLSMC 的设计与实现

NuSMV[7]是基于 CTL 和 LTL 的符号模型检测系统,在 NuSMV 系统中,待验证系统行为模型通过 SMV 输

入语言[19]描述,系统期望的性质采用经典的 CTL 或 LTL 公式来表达.NuSMV 通过内嵌的 SMV 输入语言解析模块解释系统的 SMV 模型,形成以 ROBDD 符号化描述的系统模型;并利用 CTL 或 LTL 解析模块求出性质公式所有的子公式.然后,从性质公式的最基本的子公式开始,基于符号化系统模型递归求出满足性质公式的初始状态集合.最后,对此状态集合求补集,获得不满足性质公式的初始状态集合.若该集合为空,则系统模型满足期望的性质,输出 true;否则,输出 false 并给出反例.然而,NuSMV 系统不支持对以 PPTL 公式描述的性质进行验证.本节基于 NuSMV 和 PPTL 符号模型检测方法的特点,设计和实现了 PPTL 符号模型检测工具 PLSMC.

3.1 PLSMC体系结构

PLSMC 工具的体系结构设计如图 3 所示.该工具主要由 6 个松散耦合的模块构成:用户接口模块(user interface)、解析模块(parser)、编译模块(compile)、性质公式存储模块(proposition)、模型检测模块(model checking)和 CUDD 模块.其中,Parser 模块内嵌将性质公式转化为良好形式(well-formed-formula)的 Wff 模块.

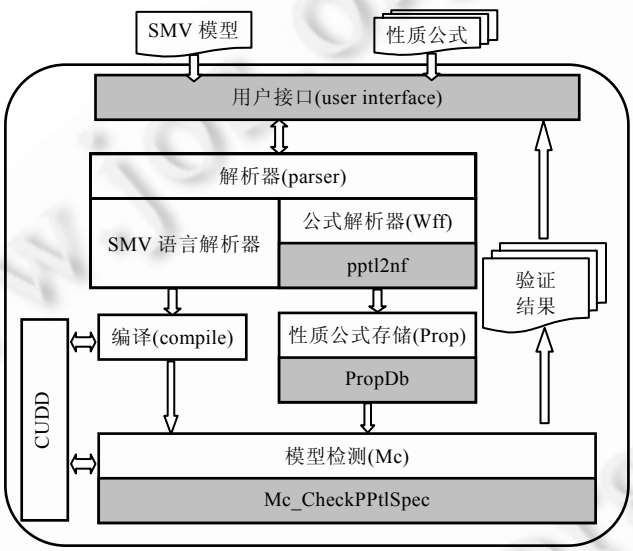


Fig.3 Architecture of PLSMC

图 3 PLSMC 体系结构

用户接口模块支持用户以命令行和图形界面两种方式输入待验证系统的SMV语言模型和以PPTL公式描述的系统性质,输入文件可以使用任意文本编辑器进行编辑.由于 NuSMV 系统不支持 PPTL 公式中常用的时序操作符,表 1 给出了 PLSMC 工具中扩展的 PPTL 时序操作符及其含义、优先级以及对应的 ASCII 码表示.

Table 1 Temporal operators available in PLSMC

表 1 PLSMC 支持的时序操作符

操作符	含义	优先级	ASCII 码	例子
¬	Negation	1	!	$!\varphi$
○	Next	2	()	$()\varphi$
◇	Sometimes	2	⟨⟩	$\langle\rangle\varphi$
□	Always	2	[]	$[]\varphi$
+	Chop-Plus	2	+	φ^+
*	Chop-Star	2	*	φ^*
∧	And	3	&&	$\varphi_1\&\&\varphi_2$
∨	Or	3	‖	$\varphi_1\|\|\varphi_2$
→	Imply	4	→	$\varphi_1\rightarrow\varphi_2$
↔	Iff	4	↔	$\varphi_1\leftrightarrow\varphi_2$
;	Chop	5	;	$\varphi_1;\ \varphi_2$
prj	Projection	5	prj	$(\varphi_1;\varphi_2)$ prj φ

Parser 模块基于 SMV 语言以及 PPTL 的词法和语法规则对用户输入的 SMV 系统模型和 PPTL 性质公式进行解析.如果系统模型或者性质公式的词法或语法有错误,则将错误类型和位置返回给用户界面,提示用户进行修改;如果没有错误,则将系统模型翻译成语法树,并调用 Wff 模块将 PPTL 公式等价地转化为其范式,然后将范式及所有子公式存储在 Prop 模块中.特别地,在 NuSMV 系统中,Wff 模块负责将以 CTL 和 LTL 描述的性质公式转化为良好形式,PLSMC 工具扩展了该模块的功能,集成了 PPTL 范式转化工具 pptl2nf[5,16,20].

Compile 模块从 Parser 模块获取系统模型的语法树,并依次对语法树执行扁平化(flatten hierarchy)、变量编码(encode variables)以及模型构建(build model)等操作,分别得到扁平化模型(flattened model)、代数决策图(algebraic decision diagram,简称 ADD)模型和 ROBDD 模型.然后,将 ROBDD 模型传递给 Mc 模块,作为 PPTL 符号模型检测过程的模型输入.

Prop 模块定义了 PropDb 数据结构,用于存储 Parser 模块解析得到的性质公式范式及其子公式.PLSMC 系统运行时,该模块根据 Mc 模块的需要从 PropDb 中取回性质公式,对性质公式的类型进行判断,然后将结果返回给 Mc 模块.经过扩展的 Prop 模块支持公式类型有 Prop_ctl,Prop_ltl,Prop_psl 以及 Prop_pptl 等.

Mc 模块为 PLSMC 工具的核心模块.该模块执行 PPTL 的符号模型检测过程.它从 Compile 模块获得符号化的系统模型,从 Prop 模块中获得系统性质公式;然后,根据 PropDb 返回的公式类型调用相应的操作对性质公式进行验证,并将验证结果返回给用户界面显示.如果性质不成立,则会输出反例提示.该模块为用户查找和定位系统模型中的错误和缺陷提供必要的帮助.

CUDD 模块提供高效的布尔函数操作.该模块基于美国科罗拉多大学研发的 CUDD 软件包[21]实现 Compile 模块中符号化系统模型的构建、Mc 模块中系统模型的遍历以及状态集上的集合操作等.

3.2 PLSMC实现

依据如图 3 所示的体系结构设计图,PLSMC 符号模型检测工具采用 C/C++语言开发,基于 Linux 平台运行.PLSMC 工具的模型检测过程如图 4 所示,其中,阴影部分表示基于文献[17]中的 PPTL 符号模型检测方法扩展并实现的函数或操作,“a::b”表示框架设计图 3 中 a 模块所包含的 b 函数或 b 操作.

(1) 将以 SMV 语言描述的待验证系统模型和系统期望的性质输入到 PLSMC 中.

(2) 符号化描述待验证系统模型:

1) Parser 模块调用 ReadModel 函数检查系统的 SMV 模型是否满足词法和语法规则:若满足,则输出系统模型的语法树(syntax tree);如果不满足,则将错误信息返回用户界面,提示用户进行修改;

2) Compile 模块调用 FlattenHierarchy 函数去除语法树中模块间的层级关系,输出扁平化系统模型;

3) Compile 模块调用 EncodeVariables 函数对扁平化系统模型中的状态变量进行编码,输出以 ADD 符号化描述的系统模型;

4) Compile 模块调用 BuildModel 函数将 ADD 系统模型中的初始状态和迁移关系采用 ROBDD 描述,输出以 ROBDD 符号化描述的系统模型.

(3) 计算期望性质公式的良好形式:

1) Parser 模块判断性质公式φ的类型:如果φ不是 PPTL 公式,则交由 NuSMV 系统按照 CTL 或 LTL 的符号模型检测流程进行处理;如果φ是 PPTL 公式,则交给 Wff 模块中集成的 pptl2nf 工具处理;

2) pptl2nf 工具计算出 PPTL 性质公式φ的范式φ_{NF}及所有子公式,并存储在 Prop 模块的 PropDb 中.

(4) 判断系统模型是否满足期望的性质:

1) Mc 模块调用 CheckPPtlSpec 函数从 PropDb 数据结构中提取性质公式φ的范式φ_{NF}及其子公式,由 Prop 模块判断公式类型,并将公式类型(设为 Prop_pptl)传递给 PropDb_verify_all_type 函数;

2) PropDb_verify_all_type 函数根据参数类型(设为 Prop_pptl),依次调用 Mc 模块的 eval_pptl_spec 和 eval_pptl_spec_recur 函数,从φ_{NF}的最简单子公式开始,逐步扩展求得 ROBDD 系统模型中满足φ的状态集合 $Sat(\varphi)$;

3) 对集合 $Sat(\varphi)$求补集,获得满足$\neg\varphi$的状态集合 $Sat(\neg\varphi)$.若该集合与初始状态集合 I 的交集

$Sat(\neg\varphi)\cap I$ 为空,则系统模型满足期望的性质,输出 true;否则,输出 false 并给出反例路径.

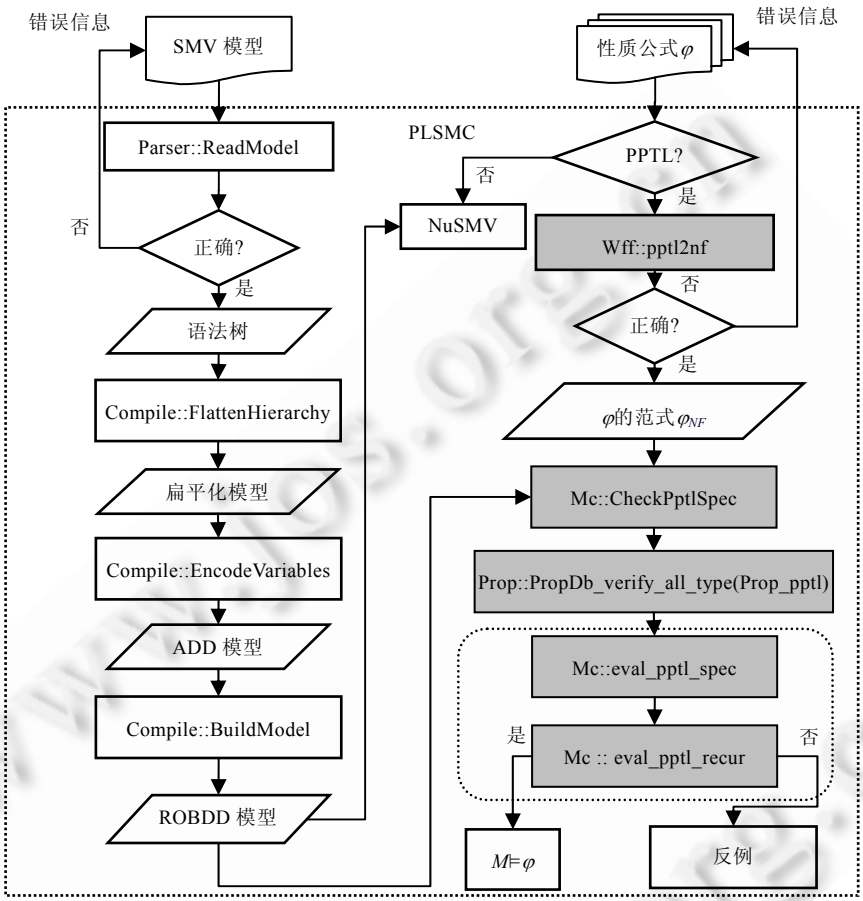


Fig.4 Flowchart of PLSMC

图 4 PLSMC 流程图

注意,如图 4 所示的流程图中核心部分为圆角矩形虚线框中的 eval_pptl_spec 和 eval_pptl_spec_recur 函数.这两个函数实现了文献[17]中提出的 PPTL 符号模型检测方法,使得 PLSMC 工具支持对以 PPTL 公式描述的期望性质进行验证.图中所示的符号化系统模型的构建、模型的遍历以及状态集上的集合操作均由 CUDD 模块提供的 ROBDD 操作函数实现.此外,由于步骤(2)和步骤(3)之间不存在数据共享,因此这两个步骤可以并行执行,从而加快工具的运行效率.

4 验证实例

4.1 铁路公路交叉道口护栏控制系统

在铁路与公路的平交道口上,为防止铁路段车辆与公路段车辆以及行人同时进入道口发生冲撞危险,一般通过升降护栏控制铁路和公路段的交通流量.如图 5 所示,基于可编程控制器(programmable logic controller,简称 PLC)的护栏控制系统(guardrail control system,简称 GCS),由 PLC 控制器、双侧护栏、护栏电机、传感器、警报器和红绿灯等部件组成.下面以火车自西向东方向运行通过交叉道口为例,解释 GCS 系统的运行过程:

(1) 系统处于空闲状态时,护栏打开,火车运行在道口之外,公路段绿灯和铁路段红灯处于点亮状态.公路段的汽车和行人允许通过道口,铁路段禁止通行.

(2) 火车沿自西向东方向运行到达预知传感器 1(灰色矩形)时,警报器蜂鸣提示道口中汽车和行人尽快离开道口,公路段绿灯和铁路段红灯此时仍处于点亮状态.

(3) 警报器蜂鸣持续 3 分钟后,公路段和铁路段黄灯点亮,PLC 控制器通过护栏电机控制护拦下降.

(4) 公路段和铁路段黄灯点亮持续 2 分钟后,护拦关闭,公路段红灯和铁路段绿灯点亮,公路段禁止通行,火车花费 1~3 分钟通过道口.

(5) 火车经过道口达到东侧预知传感器 2(灰色矩形)时,公路段和铁路段黄灯点亮,警报器关闭,PLC 控制器通过护栏电机控制护拦上升.

(6) 公路段和铁路段黄灯点亮持续 2 分钟后,公路段绿灯和铁路段红灯点亮,护拦处于打开状态,铁路段禁止通行,公路段的汽车和行人允许进入道口,系统转到空闲状态.

(7) 如果自东向西和自西向东方向均没有火车通过道口,那么系统将一直在空闲状态驻留.

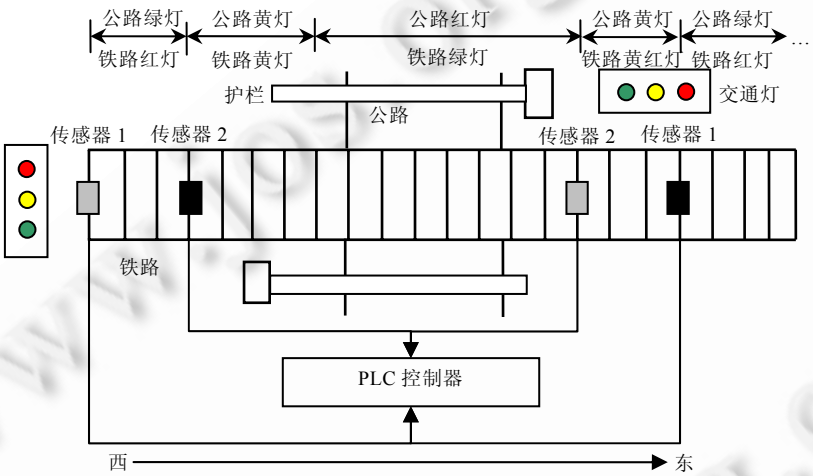


Fig.5 Railway and highway crossing guardrail control system

图 5 铁路、公路交叉道口护栏控制系统

铁路公路交叉道口的防护对提高铁路、公路的通过能力和保证汽车及行人安全具有重要的意义.本节将对上述 GCS 系统的实时安全性质和迭代性质进行分析和描述,并采用 PPTL 符号模型检测工具 PLSMC 对其进行验证;同时,检验该工具的设计和实现过程的正确性.

4.2 护栏控制系统建模

为了使用 PLSMC 工具对 GCS 系统进行验证,需要使用 SMV 语言对系统行为进行建模.SMV 语言的具体细节参见文献[19].下面分析并定义 GCS 系统的状态变量.

火车到达西侧预知传感器 1 时警报器蜂鸣,火车离开道口到达东侧预知传感器 2 时警报器关闭.因此,定义布尔类型的变量 *alert* 表示火车的到达与离开:

alert:*boolean*.

例如,*alert*=TRUE,表示火车到达交叉道口西侧预知传感器 1 时,警报器蜂鸣;*alert*=FALSE,表示火车离开交叉道口到达东侧预知传感器 2 时,警报器关闭.

公路和铁路段各有一组红、绿、黄三色交通灯,在车辆和行人通过道口时提供必要的提示.因此定义两个枚举类型的变量 *railway_light* 和 *highway_light* 表示各个交通灯当前所处的状态:

- *railway_light*:{*red*,*green*,*yellow*};
- *highway_light*:{*red*,*green*,*yellow*};

例如,*highway_light=yellow* 表示公路段的黄灯处于点亮状态.此外,依据 GCS 系统的功能描述,公路段红灯和铁

路段绿灯、公路段绿灯和铁路段红灯以及公路段黄灯和铁路段黄灯点亮后的持续时间相同,因此定义枚举型变量 *light_duration* 表示交通灯当前状态持续的时间:

light_duration:{*one*,*two*,*zero*,*stay*}.

例如,*railway_light=yellow* & *higway_light=yellow* & *light_duration=two* 表示公路段和铁路段的黄灯均持续点亮了 2 分钟.注意:由于公路段绿灯和铁路段红灯在系统空闲时处于恒亮状态,只有当控制系统进入工作状态时才需要记录公路段绿灯和铁路段红灯点亮的持续时间,因此,定义 *stay* 表示系统空闲时公路段绿灯和铁路段红灯恒亮;定义 *zero* 表示控制系统进入工作状态,并准备记录当前处于点亮状态的交通灯的持续时间.

双侧护栏在护栏电机的控制下有打开、关闭、上升和下降 4 个状态,因此定义枚举型变量 *guardrail_status* 表示护栏当前所处的状态:

guardrail_status:{*open*,*closed*,*lowering*,*raising*}.

例如,*guardrail=lowering* 表示护栏处在下降状态.

依据上述状态变量的定义,GCS 系统的初始状态或空闲状态可以用 ***init*** 关键字定义如下:

- ***init***(*alert*):=FALSE;
- ***init***(*railway_light*):=*red*;
- ***init***(*highway_light*):=*green*;
- ***init***(*light_duration*):=*stay*;
- ***init***(*guardrail_status*):=*open*.

表示系统处于空闲状态时,火车未进入交叉道口 *alert*=FALSE,护栏处于打开状态 *guardrail_status=open*,公路段绿灯和铁路段红灯处于恒亮状态 *light_duration=stay*.

通常情况下,如果没有特别说明,以状态迁移结构描述的系统模型中,所有状态迁移均在单位系统时间内完成.设计人员可以根据验证需求选择合适的时间粒度作为系统时间标准.在这里,规定 GCS 系统的标准时间为 1 分钟,那么状态迁移关系可由当前状态的状态变量取值和下一状态状态变量取值共同描述.例如,

- ***next***(*guardrail_status*):=***case***
- (*alert* & *railway_light=red* & *highway_light=green* & *light_duration=two* & *guardrail_status=open*)|…:*lowering*;
- (*alert* & *railway_light=green* & *highway_light=red* & *light_duration=zero* & *guardrail_status=closed*)|…:*raising*;
- (*alert* & *railway_light=yellow* & *highway_light=yellow* & *light_duration=one* & *guardrail_status=lowering*)|…:*closed*;
- (!*alert* & *railway_light=red* & *highway_light=green* & *light_duration=stay* & *guardrail_status=open*)|…:*open*;
- TRUE:*open*;***esac***;

其含义是:(1) 如果当前状态警报器蜂鸣,铁路段红灯和公路段绿灯持续点亮 2 分钟且护栏打开时,下一状态护栏将处于下降状态;(2) 如果当前状态警报器蜂鸣,铁路段绿灯和公路段红灯刚点亮且护栏处在关闭状态时,下一状态护栏将处于上升状态;(3) 如果当前状态警报器蜂鸣,铁路段和公路段黄灯持续点亮 1 分钟且护栏正在下降时,下一状态护栏处于关闭状态;(4) 如果当前状态警报器关闭,铁路段红灯和公路段绿灯恒亮且护栏打开时,下一状态护栏仍将处于打开状态.注意:在上述状态变量 *guardrail_status* 的状态迁移描述中,"|…"表示存在其他条件使得护栏下一状态处于打开、关闭、上升和下降这 4 个状态.限于篇幅,在此不再详述.

类似地,GCS 系统行为模型中的状态变量的迁移关系,如 *next*(*alert*),*next*(*railway_light*),*next*(*highway_light*)和 *next*(*light_duration*)等均可用状态变量 *alert*,*railway_light*,*highway_hight*,*light_duration* 和 *guardrail_status* 的取值变化及 SMV 语言提供的关键字如 *next*,*case* 和操作符&(逻辑与)、操作符|(逻辑或)等来描述.

4.3 GCS系统验证

GCS 系统的目的是保证火车进入道口前护栏已经关闭,行人和汽车离开道口,公路段红灯点亮禁止通行,铁路段绿灯点亮允许通行,而且在火车彻底离开道口前护栏不会打开.使得公路方向的汽车或行人和来自铁路方向的火车不会因为同时进入道口而发生冲撞危险,同时又不会因为护栏关闭时间过长而影响公路方向的交通流量.该系统的安全性可以描述为:

(1) 火车进入交叉道口前护栏已处于关闭状态,公路方向红灯和铁路方向绿灯点亮.

(2) 护栏关闭后最多 5 分钟就会打开.

此外,由于没有火车通过道口时 GCS 系统可以在空闲状态长时间驻留,该系统的迭代性质可以描述为:

(3) 若当前状态系统处于空闲状态,那么将来状态 GCS 系统可转入工作状态,也可在空闲状态继续驻留.

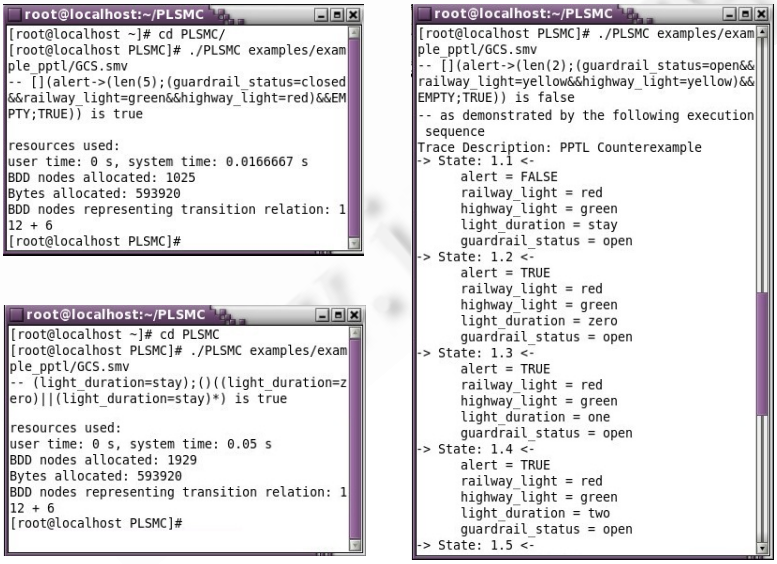
从以上分析可以看出,GCS 系统的安全性和迭代性分别为实时性质和循环性质,使用经典的 CTL 和 LTL 无法完整地描述.PPTL 是基于区间的时序逻辑,很适合描述时间敏感和周期性等实时性质;同时,PPTL 中的*(chop-star)操作符可以方便地描述迭代性质.因此,GCS 系统的安全性和迭代性可以描述为以下 PPTL 公式:

(1) $\square(alert\rightarrow(len(5);(guardrail\_status=closed\wedge railway\_light=green\wedge highway\_light=red))\wedge empty;\mathrm{TRUE}))$.
任意时刻,火车到达传感器 1,警报器蜂鸣 5 分钟后,护栏关闭,公路段红灯和铁路段绿灯点亮.

(2) $\square(alert\rightarrow(\diamond(\square(guardrail\_status=closed\wedge(len(1)\vee len(2)\vee len(3)\vee len(4)\vee len(5))));\mathrm{TRUE}))$.
任意时刻,警报器蜂鸣后护栏将会关闭,关闭持续时间不超过 5 分钟.

(3) $(light\_duration=stay);\bigcirc((light\_duration=zero)\vee(light\_duration=stay)^*)$.
当前状态系统空闲,下一状态系统将转入工作状态或者在空闲状态发生有限多次自身转移(self-transition).

这里仅通过性质(1)和性质(3)来验证 GCS 系统的安全性、迭代性以及 PLSMC 工具的正确性,其他性质的验证过程类似.如图 6(a)上半部分所示,将系统的 SMV 模型和以 SPEC 关键字修饰的 PPTL 公式输入到 PLSMC 工具后,系统提示没有错误,性质(1)成立.因此,GCS 系统满足“火车进入交叉道口前护栏已处于关闭状态,公路方向红灯和铁路方向绿灯点亮”这一安全性质.



(a) (b)

Fig.6 Verification result

图 6 验证结果

如果将性质(1)改为

□(*alert*→(*len*(2);(*guardrail_status=open*∧*railway_light=yellow*∧*highway_light=yellow*))∧*empty*;TRUE)),

即,在任意时刻火车到达预知传感器 1,警报器蜂鸣后 2 分钟护栏打开,公路段和铁路段黄灯同时点亮.系统提示有错误,并生成相应的反例路径.如图 6(b)给出的反例路径所示,在 state:1.2 时,火车到达预知传感器 1,警报器蜂鸣 *alert*=TRUE;2 分钟后,即,经过两个状态迁移到达 state:1.4 时,铁路段红灯与公路段绿灯点亮且持续了 2 分钟,护栏打开,而不是性质规范描述的护栏打开,公路段和铁路段黄灯点亮:

guardrail_status=open∧*railway_light=yellow*∧*highway_light=yellow*.

类似地,如图 6(a)下半部分所示,性质(3)成立.因此,GCS 系统满足性质"若当前状态系统处于空闲状态,那么将来状态 GCS 系统可以转入工作状态,也可在空闲状态继续驻留".

PLSMC 工具提供的验证结果显示,GCS 系统满足其安全性质和迭代性质,间接说明了该工具设计和实现过程的正确性.此外,在对其他系统进行功能验证时,可借助 PLSMC 工具提供的反例路径,对系统功能进行调整和修改.

5 实验结果

在 Fedora GNU/Linux 7.0(内核版本 2.6.21)平台+HP Z800 工作站(Intel Xeon 2.67GHZ 2.66GHZ/12G)上,通过卡内基梅隆大学及 Fondazione Bruno Kessle 组织提供的若干模型检测典型案例对 PLSMC 工具的性能进行了一系列实验评估,并与 NuSMV 2.5.4 版本进行了比较.实验主要关注以下两个方面的问题:

(1) 通过对可同时用 CTL/LTL 和 PPTL 公式描述的性质进行验证,比较 NuSMV 和 PLSMC 工具的效率;

(2) 通过对只能用 PPTL 公式描述的性质进行验证,展示 PLSMC 的效率和可处理问题的规模.

如果没有特别说明,实验结果中的运行时间开销均为 10 次模型检测过程中系统时间(system time)与用户时间(user time)之和的平均值;空间开销用可达状态数与公式长度之积来评估;"–"表示工具运行失败或者超时,"-coi"表示使用影响椎规约(cone of influence reduction)参数后系统运行的时间开销.

5.1 Gigamax缓存一致性协议检测实验结果

Gigamax 是一种分布式共享内存多处理器体系结构.在 Gigamax 体系结构中,处理器被划分为多个 cluster,每个 cluster 通过本地总线和总线探测机制维护其内部处理器缓存的一致性;同时,cluster 通过 UIC 接口及全局总线与其他 cluster 相连,UIC 接口在 cluster 之间充当总线探测器与总线主设备的作用,从而使所有 cluster 内部缓存保持一致.文献[19]中给出了 Gigamax 缓存一致性协议的 SMV 语言模型,其对应的 NuSMV 和 PLSMC 验证结果见表 2.

Table 2 Model checking results for Gigamax cache coherence protocol

表 2 Gigamax 缓存一致性协议模型检测结果

性质公式	可达状态/所有状态	运行时间(s)	
		NuSMV/NuSMV -coi	PLSMC/PLSMC -coi
AGEF(*p*0.*readable*)	3408/1.76319e+11	0.032/0.032	0.032/0.032
AG!(*p*0.*writable*&*p*1.*writable*&*p*2.*writable*)	3408/1.76319e+11	0.023/0.027	0.023/0.027
[]⟨⟩(*p*0.*readable*)	3408/1.76319e+11	–	0.098/0.101
[]([!](*p*0.*writable*&*p*1.*writable*&*p*2.*writable*))	3408/1.76319e+11	–	0.038/0.039

在表 2 中,CTL 公式 *AGEF*(*p*0.*readable*)表示从初始状态出发的所有路径中均存在一个状态使得处理器 *p*0 可以读取缓存内容;公式 *AG*!(*p*0.*writable*&*p*1.*writable*&*p*2.*writable*)表示在任何状态下均不允许处理器 *p*0,*p*1 和 *p*2 同时执行写缓存操作.上述公式在 NuSMV 工具验证的时间开销分别为 0.032s/0.032s 和 0.023s/0.027s.由于 PLSMC 工具是对 NuSMV 系统的扩展,因此二者在验证 CTL 性质时的时间消耗一致.特别地,上述性质也可用 PPTL 公式[]⟨⟩(*p*0.*readable*)和[]([!](*p*0.*writable*&*p*1.*writable*&*p*2.*writable*))来描述,其对应的时间开销分别为 0.098s/0.101s 和 0.038s/0.039s.

5.2

[22] ,

. 7 , Sender,Receiver,S2R R2S 4 .Sender

S2R ./*data*(*b*,*d*) Receiver, ,*b*∈{0,1} ,*d*

; Receiver S2R , R2S *ack*(*b*).Sender

, Receiver , *b* ;Receiver

, S2R . 3, ,CTL

AGAF (*sender.state=get*) PPTL []〈〉(*sender.state=get*)

, Sender *sender.state=get*.PLSMC NuSMV CTL

0.042s/0.018s; PPTL NuSMV , PLSMC

0.043s/0.018s. ,PPTL :

(*sender.state=get*)&&()(*sender.state=send*)→(〈〉(*sender.state=wait_for_ack*)$^+$;(*sender.state=get*))

Sender *sender.state=get* *sender.state=send* , *sender.state=wait_*

for_ack, (*sender.state=wait_for_ack*)$^+$, Receiver

, . , CTL

LTL , NuSMV , PLSMC 0.056s/0.028s.

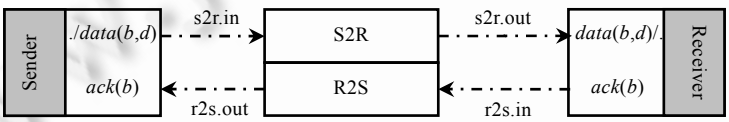


Fig.7 Alternating bit protocol

7

Table 3 Model checking results for alternating bit protocol

3

	/	(s)	
		NuSMV/NuSMV -coi	PLSMC/PLSMC -coi
AGAF(*sender.state=get*)	139776/6.0398e+08	0.042/0.018	0.042/0.018
[]〈〉(*sender.state=get*)	139776/6.0398e+08	–	0.043/0.018
(*sender.state=get*)&&()(*sender.state=send*)→ (〈〉(*sender.state=wait_for_ack*)$^+$;(*sender.state=get*))	139776/6.0398e+08	–	0.056/0.028

5.3 PCI

PCI , ,

. [22] 8 PCI

, 2 6 , 3 2 /1 1

3 /1 . (fixed priority)

(round robin).PCI 4.

4 ,CTL *AG*(*isa_bridge.req*→A(*isa_bridge.req*U*arb.grant*=0) *isa_bridge*

isa_bridge.req , , *arb.grant*=0.

,PPTL [](*isa_bridge.req*→(*isa_bridge.req*)*;*arb.grant*=0) *isa_bridge.req* ,

(*isa_bridge.req*)* .NuSMV PLSMC

CTL ; PPTL , -coi

. ,PPTL :

〈〉(*all_req*→(*arb.grant*=0;*arb.grant*=1;*arb.grant*=2;*arb.grant*=3;*arb.grant*=4;*arb.grant*=5) prj *arb.out*)

,

all_req≡*isa_bridge.req*&&*scsi_ctrl.req*&&*vga_ctrl.req*&&*slot*0.*req*&&*processors.req*&&*slot*1.*req*.

, 8 6 (

), :

arb.out≡*arb.grant*=0||*arb.grant*=1||*arb.grant*=2||*arb.grant*=3||*arb.grant*=4||*arb.grant*=5||*arb.grant=idle*

. , PPTL prj ,PLSMC

6.755s/1.094s.

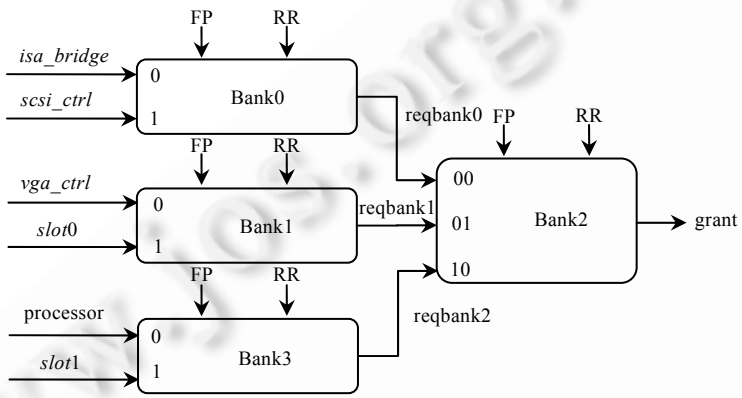


Fig.8 PCI bus protocol

8 PCI

Table 4 Model checking results for PCI bus protocol

4 PCI

	/	(s)	
		NuSMV/NuSMV -coi	PLSMC/PLSMC -coi
AG(*isa_bridge.req*→*A*(*isa_bridge.req*U*arb.grant*=0))	1.29267e+06/1.21193e+18	1.328/1.157	1.328/1.157
[](*isa_bridge.req*→(*isa_bridge.req*)*;*arb.grant*=0)	1.29267e+06/1.21193e+18	–	1.396/1.074
〈〉(*all_req*→(*arb.grant*=0;*arb.grant*=1;*arb.grant*=2; *arb.grant*=3;*arb.grant*=4;*arb.grant*=5) prj *arb.out*)	1.29267e+06/1.21193e+18	–	6.755/1.094

:

(1) PPTL PLSMC CTL/LTL NuSMV ,
CTL/LTL .

(2) PPTL , CTL PPTL
, PLSMC , NuSMV CTL
.

(3) PPTL , PLSMC NuSMV
. PLSMC , .

6

NuSMV PPTL , PPTL PLSMC
. ,
; , .

PPTL pptl2nf , .
,PLSMC , pptl2nf
, ; , PLSMC , ;
, PLSMC , PPTL ,
.

References:

[1] Clarke M, Grumberg O, Peled A. Model Checking. Cambridge: MIT Press, 2000. 35−49.

[2] Kripke A. Semantical analysis of modal logic I: Normal propositional calculi. Mathematische Logik und Grundlagen der Mathematik, 1963,9:67−96. [doi: 10.1002/malq.19630090502]

[3] Wang ZM, Chen YY, Wang ZF. Automated theorem prover for pointer logic. Ruan Jian Xue Bao/Journal of Software, 2009,20(8): 2037−2050 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/572.htm [doi: 10.3724/SP.J.1001.2009.00572]

[4] Yang Z, Ma GS, Zhang S. Equivalence verification of high-level datapaths based on polynomial symbolic algebra. Journal of Computer Research and Development, 2009,46(3):513−520 (in Chinese with English abstract).

[5] Tian C, Duan ZH. Model checking proposition projection temporal logic based on SPIN. In: Butler M, Hinchey M, Larrondo-Petrie MM, eds. Proc. of the Int'l Conf. on Formal Engineering Methods 2007. LNCS 4789, Berlin, Heidelberg: Springer-Verlag, 2007. 246−265. [doi: 10.1007/978-3-540-76650-6_15]

[6] Holzmann J. The model checker spin. IEEE Trans. on Software Engineering, 1997,23(5):279−295. [doi: 10.1109/32.588521]

[7] Cimatti A, Clarke M, Giunchiglia1 F, Roveri M. NuSMV: A new symbolic model verifier. In: Halbwachs N, Peled D, eds. Proc. of the Int'l Symp. on Computer Aided Verification. LNCS 1633, Berlin, Heildelberg: Springer-Verlag, 1999. 495−499. [doi: 10.1007/3-540-48683-6_44]

[8] Cimatti A, Clarke EM, Giunchiglia E, Giunchiglia F, Pistore M, Roveri M, Sebastiani R, Tacchella A. Nusmv 2: An opensource tool for symbolic model checking. In: Goos G, Hartmanis J, Leeuwen J, eds. Proc. of the 14th Int'l Conf. on Computer Aided Verification (CAV 2002). LNCS 2404, Heidelberg: Springer-Verlag, 2002. 359−364. [doi: 10.1007/3-540-45657-0_29]

[9] Musuvathi M, Qadeer S. CHESS: A systematic testing tool for concurrent software. Technical Report, MSR-TR-2007-149, Redmond: Microsoft Research, 2007. 2−11.

[10] Henzinger TA, Jhala R, Majumdar R, Sutre G. Lazy abstraction. In: Launchbury J, Mitchell JC, eds. Proc. of the Symp. on Principles of Programming Languages. Portland: ACM Press, 2002. 58−70. [doi: 10.1145/503272.503279]

[11] Ben-Ari M, Manna Z, Pnueli A. The temporal logic of branching time. Acta Informatica, 1983,20(1):207−226. [doi: 10.1007/BF01257083]

[12] Pnueli A. The temporal logic of programs. In: Young P, ed. Proc. of the 18th Annal IEEE Symp. on Foundations of Computer Science. Washington: IEEE Computer Society, 1977. 46−57. [doi: 10.1109/SFCS.1977.32]

[13] Wolper PL. Temporal logic can be more expressive. Information and Control, 1983,56(1-2):72−99. [doi: 10.1016/S0019-9958(83)80051-5]

[14] Duan ZH.Temporal Logic and Temporal Logic Programming. Beijing: Science Press, 2006. 9−104 (in Chinese).

[15] Duan ZH. An extended interval temporal logic and a framing technique for temporal logic programming [Ph.D. Thesis]. Newcastle: University of Newcastle Upon Tyne, 1996.

[16] Tian C, Duan ZH. Proposition projection temporal logic, Büchi automata and omega-expressions. In: Agrawal M, *et al*., eds. Proc. of the 5th Annual Conf. on Theory and Applications of Models of Computation. LNCS 4978, Berlin: Springer-Verlag, 2008. 47−58. [doi: 10.1007/978-3-540-79228-4_4]

[17] Pang T, Duan ZH, Tian C. Symbolic model checking for propositional projection temporal logic. In: Proc. of the 6th Int'l Symp. on Theoretical Aspects of Software Engineering. Piscataway: IEEE Computer Society, 2012. 9−16. [doi: 10.1109/TASE.2012.35]

[18] Bryant RE. Graph-Based algorithms for Boolean function manipulation. IEEE Trans. on Computers, 1986,35(8):687−691. [doi: 10.1109/TC.1986.1676819]

[19] McMillian L. Symbolic model checking, an approach to the state explosion problem [Ph.D. Thesis]. Boston: Carnegie Mellon University, 1993. 23−152.

[20] Duan ZH, Tian C, Zhang L. A decision procedure for propositional projection temporal logic with infinite models. Acta Informatica, 2008,45(1):43−78. [doi: 10.1007/s00236-007-0062-z]

[21] Somenzi F. CUDD: CU decision diagram package release 2.5.0. 2012. http://vlsi.colorado.edu/~fabio/CUDD/

[22] Fondazione Bruno Kessler. NuSMV examples: The collection. 2011. http://nusmv.fbk.eu/examples/examples.html

:

[3] , , . . ,2009,20(8):2037−2050. http://www.jos.org.cn/1000-9825/572.htm [doi: 10.3724/SP.J.1001.2009.00572]

[4] , , . . ,2009,46(3):513−520.

(1984), , , , , , .

(1990), , , .

(1948), , , , ,CCF , .