软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2020,31(8):2388−2403 [doi: 10.13328/j.cnki.jos.005959]

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

基本并行进程活性的限界模型检测*

谭锦豪, 李国强

(上海交通大学 软件学院,上海 200240)

通讯作者: 李国强, E-mail: li.g@sjtu.edu.cn

摘 要: 基本并行进程是一个用于描述和分析并发程序的模型,是 Petri 网的一个重要子类.EG 逻辑是一种在 Hennessy-Milner Logic 的基础上增加 EG 算子的分支时间逻辑,其中的 AF 算子表示从当前的状态出发性质最终会被满足,因此 EG 逻辑是能够表达活性的逻辑.然而,基于基本并行进程的 EG 逻辑的模型检测问题是不可判定的.由此,提出了基本并行进程上 EG 逻辑的限界模型检测方法.首先给出了基本并行进程上 EG 逻辑的限界语义,然后采用基于约束的方法,将基本并行进程上 EG 逻辑的限界模型检测问题转化为线性整数算术公式的可满足性问题,最后利用 SMT 求解器进行求解.

关键词: 基本并行进程;限界模型检测;活性;线性整数算术;SMT 求解

中图法分类号: TP301

中文引用格式: 谭锦豪,李国强.基本并行进程活性的限界模型检测.软件学报,2020,31(8):2388−2403. http://www.jos.org.cn/1000-9825/5959.htm

英文引用格式: Tan JH, Li GQ. Bounded model checking liveness on basic parallel processes. Ruan Jian Xue Bao/Journal of Software, 2020,31(8):2388−2403 (in Chinese). http://www.jos.org.cn/1000-9825/5959.htm

Bounded Model Checking Liveness on Basic Parallel Processes

TAN Jin-Hao, LI Guo-Qiang

(School of Software, Shanghai Jiao Tong University, Shanghai 200240, China)

Abstract: Basic parallel process (BPP) is a model for describing and analyzing concurrent programs, which is an important subclass of Petri nets. The logic EG is a branching time logic obtained by extending Hennessy-Milner Logic with the EG operator, in which the AF operator means that a certain property will be satisfied eventually from the current state. Hence, the logic EG is a logic that can express liveness. However, model checking the logic EG over BPP is undecidable. This study proposes an algorithm for the problem of bounded model checking EG over BPP. First, a bounded semantics of EG over BPP is proposed. Then, according to the constraint-based approach, a reduction from the problem of bounded model checking EG over BPP to the satisfiability problem of linear integer arithmetic formulas is given. Finally, the linear integer arithmetic formulas are solved by SMT solvers.

Key words: basic parallel process; bounded model checking; liveness; linear integer arithmetic; SMT solving

随着大规模复杂计算机系统的广泛使用,并发程序得到了越来越多的应用.编写一段正确高效的并发程序十分困难,而验证一段并发程序则更加困难.近年来,许多学者提出了针对并发程序的验证技术[1−5].其中,Petri 网(Petri net)[6]是一种简明且自然的用于描述和分析并发程序的模型,它可以模拟具有无穷进程的程序.具体地说,我们可以使用 Petri 网的库所 p(place)表示程序的位置 l(location),用 p 的令牌(token)数表示当前运行至 l 的进程数量.然而,传统的并发程序验证使用 Petri 网建模,会使验证的复杂性太高.例如,在并发程序的验证中,活性

* 基金项目: 国家自然科学基金(61872232, 61732013)
Foundation item: National Natural Science Foundation of China (61872232, 61732013)
本文由“面向新兴系统的形式化建模与验证方法”专题特约编辑陈振邦副教授、冯新宇教授、刘志明教授推荐.
收稿时间: 2019-08-27; 修改时间: 2019-11-02, 2019-12-30; 采用时间: 2020-02-07; jos 在线出版时间: 2020-04-18

(liveness)是一类值得关注的性质,其表示“好事情最终总会发生”,例如程序最终会终止、如果事件 X 发生则事件 Y 最终也会发生等等.如果我们用递归时序逻辑(recursive temporal logic)[7]表达活性,那么异步程序的活性问题被归约为 Petri 网的可达性(reachability)问题上[8].然而 Petri 网的可达性问题最近被证明是 Tower 难的[9],因此不可能存在高效的验证工具.我们发现,一些异步并发程序可以通过使用 Petri 网的一个子类——基本并行进程(basic parallel process,简称 BPP)[10]进行建模,使得对并发程序验证的复杂性大为降低.例如,BPP 的可达性问题是 NP 完备的[11],因此我们可以对一类并发程序的活性问题进行验证.BPP 是一个由 Søren Christensen 提出的针对并发的无穷状态模型,在 BPP 中,一个进程被建模为一个符号,程序的状态被建模为符号的并行组合,而规则表示着进程的迁移方式.一个符号能够通过迁移产生更多的符号,因此 BPP 生成的是一个无穷状态系统.由于 BPP 的迁移语义是异步的,因此 BPP 可被视为非交互式 Petri 网(communication-free Petri net),即 Petri 网的一个子类.事实上,BPP 的进程符号 X 对应着 Petri 网的库所 p,一个 BPP 表达式α对应着 Petri 网的标记 M(marking),而α中 X 出现的次数对应着库所 p 内令牌的数目.

我们用一个程序的例子说明如何使用 BPP 为并发程序建模.表 1 程序中的 cobegin 和 coend 表示创建多个进程,其中,主程序在等待请求,如果没有收到请求,则创建两个进程用于打印信息和等待;否则,同样创建两个进程,一个进程根据请求的内容向文件写进相关信息,而另一个进程用于继续检测请求.我们可以用符号集为 $\{S,T,P,W\}$的 BPP 来表示该程序,其包含以下规则.

$$S \xrightarrow{v} T,$$
$$T \xrightarrow{u} PT,$$
$$T \xrightarrow{u} WS.$$

其中, $S \xrightarrow{v} T$ 表示程序由 *detect*()进入 *wait*(),而 $T \xrightarrow{u} PT$ 和 $T \xrightarrow{u} WS$ 分别对应程序根据请求数目的不同行为.符号 T 有两种迁移方式,分别对应着请求数量为 0 和不为 0 的情况.若没有收到请求,则迁移到含有 PT 的状态;否则,迁移到包含进程 WS 的状态.

Table 1 An example of program

表 1 一个程序的例子

```
1  main(){                          16 }
2  begin                            17
3     detect()                      18 async_parallel_wait(){
4  end                              19 cobegin
5  }                                20     p1: print("no request")
6                                   21     p2: wait()
7  detect(){                        22 coend
8  begin                            23 }
9     wait()                        24
10    if request.num==0 do          25 async_parallel_write(){
11       async_parallel_wait()      26 cobegin
12    else do                       27     p1: writeFile("record.txt",request.info)
13       async_parallel_write()     28     p2: detect()
14    endif                         29 coend
15 end                              30 }
```

并发程序验证的一个重要的手段是对并发模型进行模型检测.一个模型检测问题是指在一个模型下,给定一个模型中的状态 s 以及一个逻辑公式φ,要求检测状态 s 是否满足公式φ,其形式化的表示为 $s\models\varphi$.若给定的模型是 BPP,则状态空间为 BPP 表达式构成的集合.逻辑公式φ则表示待验证的性质.根据需要验证的性质,我们选取合适的逻辑进行模型检测.对于分支时间时序逻辑,我们常用算子 AF 来表示活性.逻辑公式 $AF\varphi$表达的是从当前的状态出发,性质φ最终会被满足.在本文,我们研究的逻辑是 EG 逻辑[12].EG 逻辑由 Esparza 提出,是一种在 Hennessy-Milner Logic(HML)的基础上增加 EG 算子的分支时间时序逻辑,其包含了常见的命题逻辑公式以及时序逻辑公式 $E\langle a\rangle\varphi$和 $EG\varphi$,其中,$E\langle a\rangle\varphi$表示的是存在通过动作 a 可以到达的满足性质φ的状态;而 EG 是 AF 的对偶算子,即 AF 可以用$\neg EG\neg$表示.因此,EG 逻辑是一种能够表达活性的逻辑.例如,$s\models AF(X\geqslant 1)$表示从 BPP 状态 s 出发,最终会到达一个进程 X 数量至少为 1 的状态.

在文献[12]中,Esparza 证明了基于 BPP 的 EG 逻辑的模型检测问题是不可判定的.因此在标准语义下,我们无法对 BPP 的活性进行模型检测.值得注意的是,一个 BPP 和一个作为初始状态的 BPP 表达式生成的标签迁移系统可能会含有无穷长但不包含循环的路径,这导致了 BPP 状态的进程符号数量会不断增加.然而在现实中,大部分的程序都会在有穷步内终止,因此状态的进程数量不会出现趋于无穷的情况,而进程的迁移行为也会在一定步数内停止.

基于上述观察,本文给出了基于 BPP 的 EG 逻辑的限界模型检测方法,将 BPP 上 EG 逻辑的限界模型检测问题转化为线性整数算术公式的可满足性问题.本文的思路如下.

- 我们首先给出基于 BPP 的 EG 逻辑的 k 步限界语义 $s\models_k\varphi$.
- 接着,根据 BPP(V,Δ)的模型结构、初始的 BPP 状态 s 以及表示特定性质的 EG 逻辑公式φ,给出对应的基于线性整数算术公式的刻画.具体地说,定义了原子约束、迁移约束和路径约束等相应的约束,并根据 k 步限界语义的定义给出了生成线性整数算术公式的算法.由于 BPP 从某个状态出发,其路径可能有无限条,因此我们并不能使用主流的具有门限定理的限界模型检测编码来给 BPP 上的 EG 进行编码.同时,由于路径长度是作为参数进行输入,并且 BPP 具有以上所说的特性而并非如 Kripke 结构的有穷模型,因此在对公式 $EG\varphi$进行 SMT 编码时,不含有检测 loop 的相应约束条件.
- 最后,将编码的线性整数算术公式作为 SMT 求解器的输入进行求解.若求解的结果为公式可满足,则说明性质在 k 步内成立;若不可满足,则说明性质在 k 步内不成立.

例如,对于表 1 程序对应的 BPP,如果我们希望检测,在 2 步之内,从只包含 *detect*()的进程的状态出发,是否存在一条路径,对于该路径上的每一个状态,若写文件的进程数至少为 1 则用于 *detect*()的进程数也至少为 1,那么我们可以用 BPP 状态 $s=S$,步长 $k=2$,EG 逻辑公式$\varphi=EG(W\geqslant 1\rightarrow S\geqslant 1)$作为输入,构造出对应的线性整数算术公式.用 SMT 求解器进行求解,得到的结果是可满足,因此待检测性质成立.

本文第 1 节介绍用到的一些基础知识,包括用到的一些基础符号、标签迁移系统、基本并行进程、EG 逻辑和线性整数算术的严格定义以及对 SMT 求解技术的简介.第 2 节对 BPP 上 EG 逻辑的模型检测问题的不可判定性进行分析,并给出了 EG 逻辑的限界语义.第 3 节给出了 BPP 上 EG 逻辑限界模型检测问题的基于线性整数算术公式的刻画.第 4 节展示工具的实现及实验结果的分析.第 5 节介绍基本并行进程研究的一些相关工作.第 6 节是对全文的总结和未来工作的展望.

1 预备知识

令 $Var=\{X,Y,Z,\ldots\}$为一个可数符号集,$Act=\{a,b,c,\ldots\}$为一个可数动作集.

令 S 为一集合,$\otimes$为一个二元运算 $S\times S\rightarrow S$.如果以下 3 个条件成立.

1) 封闭性:若 $a\in S$ 且 $b\in S$ 则 $a\otimes b\in S$.
2) 结合律:对任意 $a,b,c\in S$,$(a\otimes b)\otimes c=a\otimes(b\otimes c)$成立.
3) 存在单位元:存在 $e\in S$,使得对任意 $a\in S$,$a\otimes e=e\otimes a$ 成立.

则称($S,\otimes$)是一个幺半群(monoid).一般我们也直接称 S 为一个幺半群.令 C 为一集合,则由 C 中元素构成的一个有穷序列称为 C 上的一个字(word).由集合 C 生成的自由幺半群(free monoid)指的是幺半群(C^*,*concat*),其中,C^*是由 C 上的字组成的集合,二元运算 *concat* 是字的串接(concatenation)操作.空字ε为自由幺半群的单位元.若我们规定串接运算满足交换律,即对任意字 $x,y\in C^*$,$xy=yx$ 成立,则由集合 C 生成的自由幺半群称为自由交换幺半群(free communicative monoid),记作 $C^{\oplus}$.

1.1 标签迁移系统

定义 1(标签迁移系统). 一个标签迁移系统(labelled transition system)可以用一个四元组($S,A,\rightarrow,r$)表示,其中,S 是状态的集合,A 是动作(action)的集合,$\rightarrow\subseteq S\times A\times S$ 表示迁移关系,r 表示初始状态.一般我们将关系集$\rightarrow$中的元素(s,a,t)写作 $s\xrightarrow{a}t$,表示系统中状态间通过动作的迁移.标签迁移系统的一个路径(path)指的是一个无穷序列 $s_0,s_1,s_2,\ldots$,使得 $s_0\xrightarrow{a_0}s_1\xrightarrow{a_1}s_2\ldots$,或者是一个有穷序列 $s_0,s_1,\ldots,s_n$,使得 $s_0\xrightarrow{a_0}\ldots\xrightarrow{a_{n-1}}s_n$ 且 s_n 没有后继

状态.在本文,若不加说明,则π表示一条路径,$\pi(i)$(其中,$i\geqslant 0$)表示路径的第 i 个状态.

1.2 Petri网

由于 BPP 是 Petri 网的一个子类,因此我们首先引入 Petri 网的定义.

定义 2(Petri 网). 一个网 N 是一个三元组(S,T,W),其中,S 和 T 是不相交的有穷集合,$W:(S\times T)\cup(T\times S)\to\mathbb{N}$是权值函数.集合 S 和 T 中的元素分别称为库所和迁移.网 N 的一个标记是一个映射 $M:S\to\mathbb{N}$.一个 Petri 网是一个二元组(N,M_0),其中,N 是一个网,M_0 是 N 的一个标记.

定义 3(非交互式 Petri 网). 给定一个网 $N=(S,T,W)$.对于 $x\in S\cup T$,记$^\bullet x$ 为集合$\{y|W(y,x)>0\}$,$x^\bullet$为集合$\{y|W(x,y)>0\}$.如果对于任意 $t\in T$,$|^\bullet t|=1$,且对于任意 $s\in S,t\in T,W(s,t)\leqslant 1$,则称网 N 为非交互式网.给定一个 Petri 网(N,M_0),如果 N 为非交互式网,则称该 Petri 网为非交互式 Petri 网.

1.3 基本并行进程

在原始的定义中,一个 BPP 表达式由进程变量通过动作前缀(action prefix)、选择(choice)和结合(merge)运算构成,这些运算分别表示进程的迁移、进程对迁移的选择和进程的并行组合.BPP 则定义为一族递归等式$\{X_i=E_i|1\leqslant i\leqslant n\}$,其中,$X_i$ 为进程变量,E_i 是只包含变量$\{X_1,\ldots,X_n\}$的 BPP 表达式.在本文,我们采用如下更为简洁的等价定义[13],这种定义实际上是将 BPP 视为交换上下文无关语法(commutative context-free grammar).

定义 4(基本并行进程). 一个基本并行进程 BPP 是一个二元组(V,Δ),其中 $V\subseteq Var$ 是一个有穷符号集,Δ是一个由规则组成的有穷集合.Δ中的规则的形式为

$$X\xrightarrow{a}\alpha,$$

其中,$X\in V,a\in Act,\alpha\in V^{\oplus}$.其中,$V^{\oplus}$是由 V 生成的自由交换幺半群.在本文,我们称 $V^{\oplus}$中的元素为 BPP 表达式,用小写希腊字母α,β,γ等表示.

由一个 BPP(V,Δ)和一个 BPP 表达式α,我们可以定义标签迁移系统$(V^{\oplus},Act,\to,\alpha)$,即状态空间为 $V^{\oplus}$,初始状态为α.其中,迁移关系$\to$由以下规则生成.

$$\frac{X\xrightarrow{a}\alpha\in\Delta}{\beta X\gamma\xrightarrow{a}\beta\alpha\gamma}.$$

我们记$\to^*$为一步迁移关系$\{\xrightarrow{a}\}_{a\in Act}$的自反传递闭包(reflexive transitive closure),则$\alpha\to^*\beta$表示状态α经过若干(可能为 0)次迁移后到达状态β.

因为 $V^{\oplus}$是由 V 生成的自由交换幺半群,因此 BPP 表达式具有模交换性(modulo commutativity),即表达式中的符号元素可以交换位置.例如,XYZ,XZY,YXZ,YZX,ZXY 和 ZYX 这 6 个表达式都被视为同一个表达式.从直观上看,BPP 表达式由符号并行地组合而成,每个符号都能根据相应的迁移规则各自独立地完成迁移.

BPP 限定了每次迁移由单个进程触发,其对应非交互式 Petri 网.给定一个 BPP(V,Δ)和一个 BPP 表达式β,其可以转换为一个与其同构的非交互式 Petri 网.我们用$\alpha(X)$表示符号 X 在 BPP 表达式α中出现的次数.符号集 V 中的每个符号转换为 Petri 网的一个库所,即符号集转换为 Petri 网的库所集.对于Δ中的每条规则 $X\xrightarrow{a}\alpha$,动作 a 转换为 Petri 网的一个迁移,并添加一条从 X 到 a 的权值为 1 的边;而对于每个α中的符号 Y,添加一条从 a 到 Y 的权值为$\alpha(Y)$的边.非交互式 Petri 网中的标记 M_0 则定义为 $M_0(X)=\beta(X)$.图 1 展示了从 BPP$(\{X,Y\},\Delta)$转换到对应的非交互式网的例子.

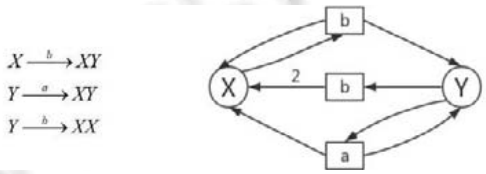

Fig.1 A BPP and the corresponding communication-free Petri net

图 1 一个 BPP 及与其对应的非交互式 Petri 网

如果初始状态β为 X,则对应的非交互式 Petri 网中的标记 $M_0:\{X,Y\}\to\mathbb{N}$定义为:$M_0(X)=1,M_0(Y)=0$.

1.4 EG逻辑

定义 5(EG 逻辑的语法). EG 逻辑的语法如下.

$$\varphi::=\text{true}|\neg\varphi|\varphi\wedge\varphi|E\langle a\rangle\varphi|EG\varphi,$$

其中 $a\in Act$.

定义 6(EG 逻辑的语义). 令 s 为状态空间 $V^{\oplus}$中的一个状态(即一个 BPP 表达式),φ是一个 EG 逻辑公式,则基于 BPP 的 EG 逻辑的语义可以如下归纳定义.

- $s\vDash$true 永远成立;
- $s\vDash\neg\varphi$当且仅当 $s\vDash\varphi$不成立;
- $s\vDash\varphi_1\wedge\varphi_2$ 当且仅当 $s\vDash\varphi_1$ 且 $s\vDash\varphi_2$;
- $s\vDash E\langle a\rangle\varphi$当且仅当存在状态 t,使得 $s\xrightarrow{a}t$ 且 $t\vDash\varphi$;
- $s\vDash EG\varphi$当且仅当存在路径π,使得$\pi(0)=s$ 且$\forall i\geqslant 0,\pi(i)\vDash\varphi$.

我们定义如下对偶算子.

- $\varphi_1\vee\varphi_2=\neg(\neg\varphi_1\wedge\neg\varphi_2)$;
- $A\langle a\rangle=\neg E\langle a\rangle\neg$;
- $AF=\neg EG\neg$.

由定义,我们可以得到这 3 个算子的语义.

- $s\vDash\varphi_1\vee\varphi_2$ 当且仅当 $s\vDash\varphi_1$ 或 $s\vDash\varphi_2$;
- $s\vDash A\langle a\rangle\varphi$当且仅当对于任意状态 t,如果 $s\xrightarrow{a}t$,则 $t\vDash\varphi$;
- $s\vDash AF\varphi$当且仅当对于任意路径π,如果$\pi(0)=s$,则$\exists i\geqslant 0,\pi(i)\vDash\varphi$.

基于 BPP 的 EG 逻辑的模型检测问题为:给定 BPP(V,Δ),BPP 表达式α以及 EG 逻辑公式φ,判断$\alpha\vDash\varphi$是否成立.由文献[12]我们可以得到以下结论.

定理 1. 基于 BPP 的 EG 逻辑的模型检测问题是不可判定的.

1.5 线性整数算术

线性整数算术(linear integer arithmetic)是一种一阶理论,线性整数算术公式的语法可以如下表示.

定义 7(线性整数算术的语法).

$$\psi::=a_0+a_1x_1+\ldots+a_nx_n\lhd|\neg\psi|\psi\wedge\psi|\exists x.\psi,$$

其中,$a_0,a_1,\ldots,a_n\in\mathbb{Z},\lhd\in\{>,=\}$.

1.6 SMT求解

SAT 问题是计算机科学和人工智能领域中的一个重要问题.然而在 SAT 求解中,求解的对象是表达能力相对较弱的命题逻辑公式,在试图求解特定背景领域的问题时,其难以表达各种约束.可满足性模理论(SMT)[14]是SAT的扩展,指的是在特定背景理论下,判定一阶逻辑公式的可满足性问题.目前的SMT求解器能处理的理论主要包括未解释函数(uninterpreted function)、整数演算(integer arithmetic)、实数演算(real arithmetic)、数组(array)以及位向量(bit vector).直观上,一个 SMT 公式可以被视为其命题变元代表着理论公式的逻辑公式,因此具备比 SAT 公式更强的表达能力.SMT 求解在模型检测、静态分析、线性规划和调度以及测试用例生成等领域中发挥着重要作用.由于 SMT 求解技术具有广阔应用前景,因此目前,其在工业界和学术界都得到了不少的重视.许多研究团队相继开发了 SMT 求解器,目前主要的 SMT 求解器包括 Yices[15]、Z3[16]、CVC4[17]和 MathSAT[18]等.其中,微软公司开发的SMT求解器Z3是目前综合求解能力最强的SMT求解器,因此在本文作为我们的求解工具.

2 EG 逻辑的模型检测及 k 步限界语义

本节主要对基于基本并行进程的 EG 逻辑模型检测问题的不可判定定理进行分析,并基于此提出了 EG 逻辑在 BPP 模型上的 k 步限界语义.

2.1 不可判定性分析

文献[12]实际上证明了 VBPP,一个比 BPP 更弱的模型,其针对 EG 逻辑的模型检测问题是不可判定的,因此对于 BPP 而言也是不可判定的.VBPP 是限定了规则集Δ中不能同时存在不同的规则 $X \xrightarrow{a} \alpha$ 和 $X \xrightarrow{b} \beta$ 的 BPP,即每个进程最多只能选择一种迁移方式.证明的思路是,将不可判定的计数器机停机问题[19]归约到该模型检测问题上.

一个计数器机 M 可以用一个三元组$(\{q_0,\ldots,q_{n+1}\},\{c_1,\ldots,c_m\},\{t_0,\ldots,t_n\})$表示,其中,$c_i$ 表示计数器;q_i 表示状态,而 q_0 是初始态,q_{n+1} 是终止态;t_i 表示 q_i 的转移规则.状态分为两类.

- 类型 1 状态的转移规则的形式是

$$c_j:=c_j+1;\ \text{goto}\ q_k;$$

- 而类型 2 状态的转移规则的形式是

$$\text{if}\ c_j=0\ \text{then goto}\ q_k\ \text{else}\ (c_j:=c_j-1;\ \text{goto}\ q_l).$$

M 的一个配置是一个元组$(q_i,j_1,\ldots,j_m)$,其中,$j_1,\ldots,j_m$ 是计数器 $c_1,\ldots,c_m$ 记录的数值.我们说 M 停机是指:如果从初始配置$(q_0,0,\ldots,0)$出发,M 能够根据转移规则到达终止配置,即状态为 q_{n+1} 的配置.计数器机的计算是确定性的,因为每个状态最多只有一条转移规则.给定一个计数器机 M,通过归约可以构造出一个 VBPP,VBPP 表达式 m(作为初始状态)以及 EG 逻辑公式 *Halt* 使得 M 停机当且仅当 m 满足 *Halt*.

标签迁移系统的一个趟(run)是一条从初始状态出发的路径.与计数器机不同,VBPP/BPP 的计算是非确定性的,一个状态可以迁移到多个不同的状态,因此不是所有的趟都会正确地模拟计数器机 M 的计算.我们称 VBPP/BPP 的一个趟是诚实的(honest),是指这个趟正确地模拟了 M 的计算;否则称其是不诚实的.通过 EG 逻辑公式的表达力,我们可以区分诚实和不诚实的趟.构造 *Halt* 的要点是先构造公式φ,使得φ满足两个性质:(1) 存在一个趟,使得这个趟上的所有状态都满足φ;(2) 一个趟上的所有状态都满足φ,当且仅当这个趟是诚实的.实际上,φ表示了 M 的计算过程中所需要满足的约束.接着,公式 *Halt* 定义为

$$Halt=AF(\neg\varphi\vee EN(halt)),$$

其中,$EN(a)=E\langle a\rangle\text{true}$ 表示 a 被触发(enable).在构造 VBPP 的时候,终止状态 q_{n+1} 被建模为进程 Q_{n+1},相应的迁移规则为 $Q_{n+1} \xrightarrow{halt} Q_{n+1}$.

由此,我们可以得知 M 停机当且仅当 m 满足 *Halt*.如果 m 满足 *Halt*,那么每个状态都满足φ的趟肯定存在一个状态满足 $EN(halt)$,且由上述两个性质可知,这样的趟肯定存在且诚实,因此 M 停机.反之,假设 M 停机.对于一个趟,如果它是诚实的,则因为 M 停机,这个趟会包含一个满足 $EN(halt)$的状态;如果是不诚实的,则由性质(2)可知,这个趟上会由一个状态不满足φ,即满足$\neg\varphi$.因此 m 满足 $AF(\neg\varphi\vee EN(halt))$.

直观上,计数器机 M 中类型 1 状态的转移规则对应着 VBPP/BPP 的迁移序列 $X\rightarrow^*\alpha$,其中,$\alpha(X)>1$;而类型 2 状态的转移规则对应着迁移序列 $X\rightarrow^*\varepsilon$.第 1 种迁移序列会使得 VBPP/BPP 中状态的符号逐渐增加,从而产生无穷长且不带循环的趟.这种特性对应着 M 的一种不停机的情况:计数器的数值不断增加而终止配置永远不会达到.我们通过下面的例子进行说明.假设 M 为

$$(\{q_0,q_1,q_2\},\{c_1,c_2\},\{t_0,t_1\}),$$

其中,t_0 为 $c_1:=c_1+1;\ \text{goto}\ q_1$,$t_1$ 为 $c_2:=c_2+1;\ \text{goto}\ q_0$,则从初始配置出发,$c_1,c_2$ 中的数值持续增加但 M 不停机.构造的 VBPP 具有以下规则(进程符号用||分割).

$$C_j \xrightarrow{dec_j} \varepsilon,\ \text{for } j=1,2,$$
$$Q_0 \xrightarrow{out_0} Q_1 \| C_1,$$
$$Q_1 \xrightarrow{out_1} Q_0 \| C_2,$$
$$Q_2 \xrightarrow{halt} Q_2.$$

初始配置表示为 Q_0,同时也作为初始状态 m.不难发现,存在以下诚实的趟模拟了 M 的不停机行为

$$Q_0 \xrightarrow{out_0} Q_1 \| C_1 \xrightarrow{out_1} Q_0 \| C_1 \| C_2 \xrightarrow{out_0} Q_1 \| C_1^2 \| C_2 \xrightarrow{out_1} Q_0 \| C_1^2 \| C_2^2 \xrightarrow{out_0} \ldots.$$

因此,从 m 出发存在上述路径,由于它诚实地模拟了 M,因此该路径上所有状态都满足 φ.同时,因为它始终没有进入终止状态 Q_2,所以动作 halt 没有被触发,即路径上每一个状态都不满足 $EN(halt)$.根据 EG 逻辑的语义,我们可以得到 $m\vDash EG(\varphi\wedge\neg EN(halt))$,即 $\neg Halt$.在给定的逻辑为 EG 逻辑的前提下,BPP 上的模型检测问题不可判定的原因在于其会产生长度无穷但不含循环的路径.基于这个观察,我们下面将路径限制在 k 步内进行模型检测.

2.2 k步限界语义

在 EG 逻辑的定义中,原子命题被限定为 true.为了表达进程的一些基础性质,我们对原子命题的形式进行扩展.具体地说,假设给定的 BPP 的有穷符号集 $V=\{X_1,\ldots,X_n\}$,我们规定原子命题为如下形式.

$$A\cdot M\geqslant B.$$

其中,A 是一个 $m\times n$ 维的整数矩阵,M 是由 BPP 中进程符号组成的向量$(X_1,\ldots,X_n)^T$,B 是一个 m 维整数向量.例如,下面的公式.

$$AF(X+Y=3\rightarrow E\langle a\rangle(Z>1))$$

表达的是:“如果一个状态中的进程 X 和 Y 的数量之和为 3,那么该状态存在一个经过动作 a 的后继状态,其进程 Z 的个数大于 1”这个性质在未来的某个时刻终将会被满足.

一个 BPP 表达式α可以很自然地表示为一个多重集合(multiset).例如,$X_1X_2X_2X_3$ 可以用集合$\{X_1,X_2,X_2,X_3\}$表示.在本文中,我们将α表示为 n 维向量.严格地说,定义 Parikh 映射(Parikh mapping)$P:V^{\oplus}\rightarrow N^n$ 为

$$P(\alpha)=(\alpha(X_1),\ldots,\alpha(X_n))^T,$$

其中,$\alpha(X)$表示 X 在中α出现的次数.注意:表示α的 n 维向量中的每个分量都必须满足非负的要求,因为一个进程符号在 BPP 表达式中出现的次数不能为负数.在下文,我们用 Parikh 映射来表示一个 BPP 状态.例如,若 $V=\{X_1,X_2,X_3\}$,$\alpha=X_1X_2X_2$,则α可以用$(1,2,0)^T$表示.对于两个 n 维向量 $\boldsymbol{x}=(x_1,\ldots,x_n)^T$,$\boldsymbol{y}=(y_1,\ldots,y_n)^T$,定义 $\boldsymbol{x}\leqslant\boldsymbol{y}$ 当且仅当

$$x_1\leqslant y_1,\ldots,x_n\leqslant y_n.$$

现在我们可以如下定义 EG 逻辑的 k 步限界语义.

定义 8(EG 逻辑的 k 步限界语义). 假设给定的 BPP 的有穷符号集 $V=\{X_1,\ldots,X_n\}$.令 M 为由给定的 BPP 中进程符号组成的向量$(X_1,\ldots,X_n)^T$,s 为一个 BPP 表达式,φ为一个 EG 逻辑公式,$k\geqslant 0$.k 步限界语义 $s\vDash_k\varphi$归纳定义如下.

- $s\vDash_k A\cdot M\geqslant B$ 当且仅当 $A\cdot s\geqslant B$;
- $s\vDash_k\neg\varphi$当且仅当 $s\vDash_k\varphi$不成立;
- $s\vDash_k\varphi_1\wedge\varphi_2$ 当且仅当 $s\vDash_k\varphi_1$ 且 $s\vDash_k\varphi_2$;
- $s\vDash_k E\langle a\rangle\varphi$当且仅当 $k\geqslant 1$,且存在状态 t,使得 $s\xrightarrow{a}t$ 且 $t\vDash_k\varphi$;
- $s\vDash_k EG\varphi$当且仅当存在路径π,使得$\pi(0)=s$ 且$\forall 0\leqslant i\leqslant k,\pi(i)\vDash_k\varphi$.

上述的 k 步限界语义表达了在从 s 出发的 k 步内的状态空间中φ在 s 上满足.注意:对于 $s\vDash_k E\langle a\rangle\varphi$,由于 $E\langle a\rangle\varphi$ 在 s 上成立必须经过一次迁移,因此需要限制步数至少为 1,即 $k\geqslant 1$.

3 线性整数算术公式刻画

本节主要根据上文给出的EG逻辑的 k 步限界语义,构造对应的线性整数算术公式.我们首先定义一些基本约束来表示 BPP 的状态迁移等行为,接着给出构造线性整数算术公式的算法并证明其正确性.在本节,我们假设

给定的 BPP 为($V=\{X_1,\ldots,X_n\},\Delta$).

3.1 基本约束

首先定义一些约束,包括原子约束、迁移约束和路径约束.

定义 9(**原子约束**). 给定 $m\times n$ 维的整数矩阵 $A=\{a_{ij}\}_{m\times n}$,n 维向量 $s=(s_1,\ldots,s_n)^T$,m 维整数向量 $b=(b_1,\ldots,b_m)^T$,原子约束如下定义.

$$\wedge_{i=1}^{m}\left(\sum_{j=1}^{n}a_{ij}\cdot s_j\geqslant b_i\right).$$

原子约束表示了原子命题 $A\cdot M\geqslant B$ 的语义 $A\cdot s\geqslant B$,即对于矩阵 A 的每个行向量 A_i,$A_i\cdot s\geqslant b_i$ 都必须成立.

接着我们定义迁移约束,即 $s\xrightarrow{a}t$ 对应的约束条件.我们首先定义一些基础符号和映射.对于每个规则 $r\in\Delta$,我们用 ${}^\bullet r$ 表示箭头左边的符号,$r^\bullet$ 表示箭头右边的 BPP 表达式,r^{act} 表示迁移动作.例如若 r 为 $X\xrightarrow{a}\alpha$,则 ${}^\bullet r=X$,$r^\bullet=\alpha$,$r^{act}=a$.

定义映射 $P^-:V\times V^{\oplus}\to\mathbb{N}^n$ 如下.

$$P^-(X,\alpha)=(c^-(X,\alpha,X_1),\ldots,c^-(X,\alpha,X_n))^T,$$

其中,

$$c^-(X',\alpha,X)=\begin{cases}\alpha(X)-1, & \text{if } X=X'\\ \alpha(X), & \text{otherwise}\end{cases}.$$

映射 P^- 与 Parikh 映射 P 类似,区别在于 $P^-(X,\alpha)$ 中 X 对应的整数值会比其在 α 中出现的次数 $\alpha(X)$ 少 1.例如对于符号集只包含符号 X_1,X_2,X_3 的 BPP,$P(X_1X_2X_2X_3X_3)=(1,2,2)^T$,$P^-(X_2,X_1X_2X_2X_3X_3)=(1,1,2)^T$.

给定 $r\in\Delta$,n 维向量 $s=(s_1,\ldots,s_n)^T$ 和 $t=(t_1,\ldots,t_n)^T$.定义约束 $T^-(s,t,r)$ 如下.

$$T^-(s,t,r)=\wedge_{i=1}^{n}(s_i+P^-({}^\bullet r,r^\bullet)_i=t_i).$$

约束 T^- 实际上刻画了 s 和 t 中各个进程符号数量的关系,表示 s 通过迁移规则 r 到达 t.例如假设符号集 $V=\{X_1,X_2,X_3\}$,r 为 $X_1\xrightarrow{a}X_2$,则 ${}^\bullet r=X_1$,$r^\bullet=X_2$,$P^-({}^\bullet r,r^\bullet)=P^-(X_1,X_2)=(-1,1,0)^T$,此时,$T^-(s,t,r)=(s_1-1=t_1)\wedge(s_2+1=t_2)\wedge(s_3=t_3)$.注意,约束 $T^-(s,t,r)$ 没有对 t 进行非负的限定,因此 t 可能不是一个合法的 BPP 表达式.通过约束 T^-,我们现在可以定义迁移约束来表示 $s\xrightarrow{a}t$.

定义 10(**迁移约束**). 给定两个 n 维向量 $s=(s_1,\ldots,s_n)^T$ 和 $t=(t_1,\ldots,t_n)^T$,迁移动作 $a\in Act$,则 s 通过动作 a 到达 t 可以通过以下约束 $T(s,t,a)$ 表示.

$$T(s,t,a)=\vee_{r\in\Delta}(r^{act}=a\wedge s({}^\bullet r)\geqslant 1\wedge T^-(s,t,r)).$$

注意:在迁移约束中,条件 $s({}^\bullet r)\geqslant 1$ 是迁移被触发的必要条件,其能够保证 BPP 的基本要求,即 BPP 表达式中每个进程符号出现的次数不能为负数.例如,对于($V=\{X,Y,Z\},\Delta$),其中,Δ 由下列规则组成.

$$X\xrightarrow{a}Y,$$

$$Y\xrightarrow{b}XZZ.$$

令 $p_1=(2,0,0)^T\xrightarrow{a}(1,1,0)^T\xrightarrow{b}(2,0,2)^T$,$p_2=(2,0,0)^T\xrightarrow{b}(3,-1,2)^T\xrightarrow{a}(2,0,2)^T$.$p_1$ 和 p_2 都是从 XX 出发的长度 $k=2$ 且最终到达 $XXZZ$ 的序列,但由于 $(3,-1,2)^T$ 中 Y 对应的整数值为 -1,因此该状态是不合法的.而 $s({}^\bullet r)\geqslant 1$ 确保了 $(2,0,0)^T\xrightarrow{b}(3,-1,2)^T$ 不会发生,因为对于规则 $r:Y\xrightarrow{b}XZZ$,${}^\bullet r=Y$,而 Y 在 XX 出现的次数为 0.

我们如下定义路径约束,即表示向量构成路径的约束.

定义 11(**路径约束**). 令 $k\geqslant 0$,$u(0),u(1),\ldots,u(k)$ 为 $k+1$ 个 n 维向量,则这些向量构成一条路径可以通过以下约束 $Path(u(0),\ldots,u(k))$ 表示.

$$Path(u(0),\ldots,u(k))=\wedge_{j=1}^{k}[\vee_{r\in\Delta}(u(j-1)({}^\bullet r)\geqslant 1\wedge T^-(u(j-1),u(j),r))].$$

路径约束 $Path(u(0),\ldots,u(k))$ 表示的是:对于每个 $1\leqslant j\leqslant k$,$u(j-1)$ 可以根据规则集 Δ 中的一条迁移规则 r 到达 $u(j)$.与迁移约束中的条件 $s({}^\bullet r)\geqslant 1$ 类似,路径约束中必须包含条件 $u(j-1)({}^\bullet r)\geqslant 1$.

3.2 构造线性整数算术公式

令 $k\geqslant 0$,s 是一个 n 维向量,φ是一个 EG 逻辑公式.构造线性整数算术公式的算法 $Trans(\varphi,s,k)$,如图 2 所示.

算法 1 接收一个 EG 逻辑公式φ、一个 n 维向量 s 以及一个非负整数 k 作为输入,而输出是一个线性整数算术公式.该递归算法根据 EG 逻辑公式φ的结构进行递归.若φ为原子公式,则生成原子约束.若φ的最外层算子是命题逻辑算子,则根据否定和合取的语义进行构造.若φ形如 $E\langle a\rangle\varphi_1$ 或 $EG\varphi_1$,则通过迁移约束和路径约束结合限界语义生成相应的线性整数算术公式.注意,算法返回的线性整数算术公式是一个闭合公式(closed formula).

算法 1. $Trans(\varphi,s,k)$.
输入:EG 逻辑公式φ,n 维向量 $s=(s_1,\ldots,s_n)^T$,非负整数 k.
输出:线性整数算术公式ψ.

```
1  begin
2      case φ=A·M⩾B do
3          ψ := ∧_{i=1}^{m} ( Σ_{j=1}^{n} a_ij · s_j ⩾ b_i )
4      case φ=¬φ1 do
5          ψ:=¬Trans(φ1,s,k)
6      case φ=φ1∧φ2 do
7          ψ:=Trans(φ1,s,k)∧Trans(φ2,s,k)
8      case φ=E⟨a⟩φ1 do
9          ψ:=k⩾1∧∃t1…∃tn.(T(s,(t1,…,tn),a)∧Trans(φ1,(t1,…,tn),k))
10     case φ=EGφ1 do
11         θ := Path((u(0)1,...,u(0)n),...,(u(k)1,...,u(k)n)) ∧ ∧_{i=1}^{n} u(0)_i = s_i ∧ ∧_{j=0}^{k} Trans(φ1,(u(j)1,...,u(j)n),k)
12         ψ:=∃u(0)1…∃u(0)n…∃u(k)1…∃u(k)n.θ
13     retrun ψ
14 end
```

Fig.2 Algorithm for generating corresponding LIA formulas
图 2 生成对应的线性整数算术公式的算法

下面我们证明:如果输入的向量 s 中所有分量都满足大于 0 的条件,且同时算法生成的线性整数算术公式ψ成立,则ψ中任意变量 x 都满足非负条件,即 $x\geqslant 0$.为了证明这个结论,我们首先证明以下引理.

引理 1. 给定 $r\in\Delta$,n 维向量 $s=(s_1,\ldots,s_n)^T$ 和 $t=(t_1,\ldots,t_n)^T$,如果 $\wedge_{i=1}^{n}s_i\geqslant 0$,$s(^\bullet r)\geqslant 1$ 和 $T^-(s,t,r)$同时成立,则 $\wedge_{i=1}^{n}t_i\geqslant 0$ 成立.

证明:因为 $T^-(s,t,r)$成立,故对于任意 $1\leqslant i\leqslant n$,由映射 P^-的定义,可得 $t_i=s_i+P^-(^\bullet r,r^\bullet)_i=s_i+c^-(^\bullet r,r^\bullet,X_i)$.注意:对于任意 $r\in\Delta$,$r^\bullet(X_i)\geqslant 0$.我们分以下两种情况讨论.

- 如果 $X_i\neq{}^\bullet r$,则 $t_i=s_i+r^\bullet(X_i)\geqslant s_i\geqslant 0$;
- 如果 $X_i={}^\bullet r$,则 $t_i=s_i+r^\bullet(X_i)-1=r^\bullet(X_i)+s(X_i)-1=r^\bullet(X_i)+s(^\bullet r)-1\geqslant r^\bullet(X_i)\geqslant 0$.

综上,$\wedge_{i=1}^{n}t_i\geqslant 0$ 成立. □

借助引理 1,我们可以证明以下定理.

定理 2. 给定 EG 逻辑公式φ、n 维向量 $s=(s_1,\ldots,s_n)^T$ 和非负整数 k.令$\pi=E\langle a\rangle\varphi$或 $EG\varphi$.若 $\wedge_{i=1}^{n}s_i\geqslant 0\wedge Trans(\pi,s,k)$成立,则对于任意在构造线性整数公式 $Trans(\pi,s,k)$的过程中新增的变量 x 都满足非负条件,即 $x\geqslant 0$.

证明:我们分别对$\pi=E\langle a\rangle\varphi$和$\pi=EG\varphi$两种情况讨论.

- 假设$\pi=E\langle a\rangle\varphi$.在 $Trans(E\langle a\rangle\varphi,s,k)$中,新增的变量为 $t_1,\ldots,t_n$,因为 $T(s,(t_1,\ldots,t_n),a)$成立,由引理 1 可知:对于任意 $1\leqslant i\leqslant n$,$t_i\geqslant 0$ 为真;
- 假设$\pi=EG\varphi_1$.在 $Trans(EG\varphi,s,k)$中,新增的变量为 $u(0)_1,\ldots,u(0)_n,\ldots,u(k)_1,\ldots,u(k)_n$.我们对步长 k 进行归纳证明.当 $k=0$ 时,由于 $\wedge_{i=1}^{n}u(0)_i=s_i$ 和 $\wedge_{i=1}^{n}s_i\geqslant 0$ 成立,因此变量 $u(0)_1,\ldots,u(0)_n$ 非负.对于 $1\leqslant j\leqslant k$,由于存在 $r\in\Delta$使得 $u(j-1)(^\bullet r)\geqslant 1\wedge T^-(u(j-1),u(j),r)$为真,且由归纳假设,变量 $u(j-1)_1,\ldots,u(j-1)_n$ 非负,因此由引理 1 可知,变量 $u(j)_1,\ldots,u(j)_n$ 非负.综上,变量 $u(0)_1,\ldots,u(0)_n,\ldots,u(k)_1,\ldots,u(k)_n$ 都为非负.

根据算法 1,在构造线性整数算术公式 $Trans(\varphi,s,k)$时,只有遇到 $E\langle a\rangle$算子和 EG 算子时才会增加新的变量.因此由定理 2 我们可知:只要输入的 n 维向量 s 是一个合法的 BPP 状态,即向量中的每个分量非负,且同时算法 1 生成的线性整数算术公式 $Trans(\varphi,s,k)$成立,则基于算法 1 的编码不会出现到达不合法的 BPP 状态的情况. □

3.3 算法的正确性

下面我们对 EG 逻辑公式进行结构归纳证明算法 1 的正确性.

定理 3(算法的正确性). 给定 EG 逻辑公式φ,n 维向量 $s=(s_1,\ldots,s_n)^T$ 和非负整数 k,则算法 1 是正确的,即下列命题成立.

- 终止性:算法会终止.
- 可靠性:如果 $s\models_k\varphi$成立,则 $Trans(\varphi,s,k)$可满足.
- 完备性:如果 $Trans(\varphi,s,k)$可满足,则 $s\models_k\varphi$成立.

证明:由于算法 1 是根据 EG 逻辑公式φ的结构进行递归构造线性整数算术公式的,因此可知算法 1 是终止的.对于可靠性和完备性,我们如下对φ进行结构归纳证明.

- 假设φ是原子命题公式,即$\varphi=A\cdot M\geqslant B$,则根据原子约束 $\wedge_{i=1}^{m}\left(\sum_{j=1}^{n}a_{ij}\cdot s_j\geqslant b_i\right)$以及 $s\models_k A\cdot M\geqslant B$ 当且仅当 $A\cdot s\geqslant B$,显然可得可靠性和完备性成立.
- 假设$\varphi=\neg\varphi_1$.对于可靠性,假设 $Trans(\neg\varphi_1,s,k)$不可满足,则根据算法 1,$\neg Trans(\varphi_1,s,k)$不可满足,因此 $Trans(\varphi_1,s,k)$正确.由归纳假设,$s\models_k\varphi_1$ 成立,故 $s\models_k\neg\varphi_1$ 不成立.反之,对于完备性,假设 $Trans(\neg\varphi_1,s,k)$正确,则根据算法 1,$Trans(\varphi_1,s,k)$不可满足,由归纳假设,$s\models_k\varphi_1$ 不成立,故 $s\models_k\neg\varphi_1$ 成立.
- 假设$\varphi=\varphi_1\wedge\varphi_2$.对于可靠性,假设 $Trans(\varphi_1\wedge\varphi_2,s,k)$不可满足,则根据算法 1 的构造,$Trans(\varphi_1,s,k)\wedge Trans(\varphi_2,s,k)$不可满足,因此 $Trans(\varphi_1,s,k)$不可满足或 $Trans(\varphi_2,s,k)$不可满足.由归纳假设,$s\models_k\varphi_1$ 或 $s\models_k\varphi_2$ 不成立,因此 $s\models_k\varphi_1\wedge\varphi_2$ 不成立.反之,对于完备性,假设 $Trans(\varphi_1\wedge\varphi_2,s,k)$正确,则根据算法 1,$\neg Trans(\varphi_1,s,k)$不可满足且$\neg Trans(\varphi_2,s,k)$不可满足.由归纳假设,$s\models_k\neg\varphi_1$ 且 $s\models_k\neg\varphi_2$ 不成立,因此 $s\models_k\varphi_1\wedge\varphi_2$ 成立.
- 假设$\varphi=E\langle a\rangle\varphi_1$.对于可靠性,假设 $Trans(E\langle a\rangle\varphi_1,s,k)$可满足,则 $k=0$ 或者对于任意 $t_1,\ldots,t_n$,如果 $T(s,(t_1,\ldots,t_n),a)$,则$\neg Trans(E\langle a\rangle\varphi_1,(t_1,\ldots,t_n),k)$.若 $k=0$,则显然 $s\models_k A\langle a\rangle\neg\varphi_1$ 成立.若后者,假设 $T(s,(t_1,\ldots,t_n),a)$正确,则$\neg Trans(\varphi_1,(t_1,\ldots,t_n),k)$正确,因此 $Trans(\varphi_1,(t_1,\ldots,t_n),k)$不可满足.由归纳假设,$(t_1,\ldots,t_n)\models_k\varphi_1$ 不成立,因此 $s\models_k A\langle a\rangle\neg\varphi_1$ 成立.综上,$s\models_k E\langle a\rangle\varphi_1$ 不成立.反之,对于完备性,假设 $Trans(E\langle a\rangle\varphi_1,s,k)$正确,则 $k\geqslant 1$ 且存在 $t_1,\ldots,t_n$,使得 $T(s,(t_1,\ldots,t_n),a)\wedge Trans(\varphi_1,(t_1,\ldots,t_n),k)$,因此$\neg Trans(E\langle a\rangle\varphi_1,(t_1,\ldots,t_n),k)$不可满足.由归纳假设,$(t_1,\ldots,t_n)\models_k\neg\varphi_1$ 不成立,因此$(t_1,\ldots,t_n)\models_k\varphi_1$ 成立,所以 $s\models_k E\langle a\rangle\varphi_1$ 成立.
- 假设$\varphi=EG\varphi_1$.对于可靠性,假设 $Trans(EG\varphi_1,s,k)$不可满足,则对于任意 $u(0)_1,\ldots,u(0)_n,\ldots,u(k)_1,\ldots,u(k)_n$,如果 $\alpha=path((u(0)_1,\ldots,u(0)_n),\ldots,(u(k)_1,\ldots,u(k)_n))\wedge\wedge_{i=1}^{n}u(0)_i=s_i$,则 $\neg\wedge_{j=0}^{k}Trans(\varphi_1,(u(j)_1,\ldots,u(j)_n),k)$.假设α正确,则存在 $0\leqslant j\leqslant k$,使得$\neg Trans(\varphi_1,u(j),k)$正确,因此 $Trans(\varphi_1,u(j),k)$不可满足.由归纳假设,$u(j)\models_k\varphi_1$ 不成立,因此 $s\models_k AF\neg\varphi_1$ 成立,即 $s\models_k EG\varphi_1$ 不成立.反之,对于完备性,假设 $Trans(EG\varphi_1,s,k)$正确,则存在 $u(0)_1,\ldots,u(0)_n,\ldots,u(k)_1,\ldots,u(k)_n$,使得 $\alpha\wedge\wedge_{j=0}^{k}Trans(\varphi_1,(u(j)_1,\ldots,u(j)_n),k)$.因此对于任意 $0\leqslant j\leqslant k$,$\neg Trans(\varphi_1,u(j),k)$不可满足.由归纳假设,$u(j)\models_k\neg\varphi_1$ 不成立,即 $u(j)\models_k\varphi_1$ 成立,所以 $s\models_k EG\varphi_1$ 成立. □

3.4 线性整数算术公式构造示例

为了更加直观地认识上述算法,下面我们举例说明如何构建一个线性整数算术公式.

我们考虑 BPP(V,Δ),其中,$V=\{X_1,X_2,X_3\}$,Δ包含以下规则.

$$r_1: X_1\xrightarrow{a}X_2X_3,$$
$$r_2: X_2\xrightarrow{a}X_1X_2,$$
$$r_3: X_3\xrightarrow{b}X_1.$$

我们令初始状态 $s=X_1$,令 $k=2$,则相应的标签迁移系统如图 3 所示.

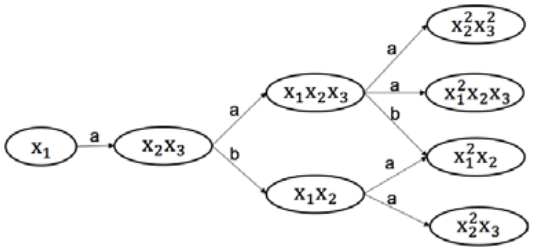

Fig.3 Corresponding labelled transition system

图 3 相应的标签迁移系统

待检测的 EG 逻辑公式 φ 为 $EG(E\langle a\rangle(X_2+X_3\geqslant 2))$,则 $Trans(\varphi,s,2)$ 为

$$\exists u(0)_1\exists u(0)_2\exists u(0)_3\exists u(1)_1\exists u(1)_2\exists u(1)_3\exists u(2)_1\exists u(2)_2\exists u(2)_3.\theta,$$

其中,

$$\theta := Path((u(0)_1,u(0)_2,u(0)_3),(u(1)_1,u(1)_2,u(1)_3),(u(2)_1,u(2)_2,u(2)_3))\wedge(u(0)_1=1\wedge u(0)_2=0\wedge u(0)_3=0)\wedge$$
$$\wedge_{j=0}^{2} Trans(E\langle a\rangle(X_2+X_3\geqslant 2),(u(j)_1,u(j)_2,u(j)_3),2).$$

我们首先展开 θ 的第 1 个合取支即路径约束.根据映射 P^- 的定义,可知:

$$P^-(^\bullet r_1,r_1^\bullet)=(-1,1,1)^T,P^-(^\bullet r_2,r_2^\bullet)=(1,0,0)^T,P^-(^\bullet r_3,r_3^\bullet)=(1,0,-1)^T.$$

因此,路径约束中的每个合取支(j=1,2)为

$$\psi_{trans}(j) := [u(j-1)_1\geqslant 1\wedge(u(j-1)_1-1=u(j)_1\wedge u(j-1)_2+1=u(j)_2\wedge u(j-1)_3+1=u(j)_3)]\vee$$
$$[u(j-1)_2\geqslant 1\wedge(u(j-1)_1+1=u(j)_1\wedge u(j-1)_2+0=u(j)_2\wedge u(j-1)_3+0=u(j)_3)]\vee$$
$$[u(j-1)_3\geqslant 1\wedge(u(j-1)_1+1=u(j)_1\wedge u(j-1)_2+0=u(j)_2\wedge u(j-1)_3-1=u(j)_3)].$$

所以路径约束为 $\psi_{trans}(1)\wedge\psi_{trans}(2)$.

接着,我们展开 θ 的第 3 个合取支.对于 j=0,1,2,考虑 $Trans(E\langle a\rangle(X_2+X_3\geqslant 2),(u(j)_1,u(j)_2,u(j)_3),2)$,该约束对应着子公式 $E\langle a\rangle(X_2+X_3\geqslant 2)$ 在 $(u(j)_1,u(j)_2,u(j)_3)$ 上 2 步内成立的情形,展开为

$$\psi_{sub}(j) := 2\geqslant 1\wedge\exists t_1\exists t_2\exists t_3.(([a=a\wedge u(j)_1\geqslant 1\wedge(u(j)_1-1=t_1\wedge u(j)_2+1=t_2\wedge u(j)_3+1=t_3)]\vee$$
$$[a=a\wedge u(j)_2\geqslant 1\wedge(u(j)_1+1=t_1\wedge u(j)_2+0=t_2\wedge u(j)_3+0=t_3)]\vee$$
$$[b=a\wedge u(j)_3\geqslant 1\wedge(u(j)_1+1=t_1\wedge u(j)_2+0=t_2\wedge u(j)_3-1=t_3)])\wedge t_2+t_3\geqslant 2).$$

因此,第 3 个合取支为 $\psi_{sub}(0)\wedge\psi_{sub}(1)\wedge\psi_{sub}(2)$.注意:对于上述公式中的迁移动作,我们在实际中可以用整数进行编码.

综上,根据算法生成的公式 $Trans(\varphi,s,2)$ 为

$$\psi := \exists u(0)_1\exists u(0)_2\exists u(0)_3\exists u(1)_1\exists u(1)_2\exists u(1)_3\exists u(2)_1\exists u(2)_2\exists u(2)_3$$
$$(\psi_{trans}(1)\wedge\psi_{trans}(2)\wedge(u(0)_1=1\wedge u(0)_2=0\wedge u(0)_3=0)\wedge\psi_{sub}(0)\wedge\psi_{sub}(1)\wedge\psi_{sub}(2)).$$

4 工具实现及实验结果

4.1 技术架构

基于线性整数算术公式刻画,我们采用基于约束的方法实现了工具,其架构如图 4 所示.

工具接受 3 个输入,分别是 BPP 模型(V,Δ)、待检测的 EG 逻辑公式 φ 以及步长 k.工具从这些输入获取 BPP 中符号集 V 和规则集 Δ 的信息,并根据 EG 逻辑公式 φ 的结构以及步长 k 的大小,生成原子约束、迁移约束和路径约束等约束条件,进而构造出相应的线性整数算术公式,保存为约束文件作为 SMT 求解器 Z3 的输入.如果 Z3 的求解结果为 sat,则说明在限界语义下 EG 逻辑公式 φ 在 k 步内满足;若求解的返回结果为 unsat,则说明在 k 步内不满足.

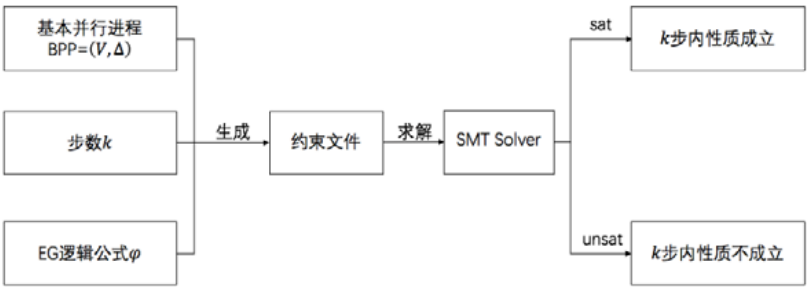

Fig.4 Architecture of the tool

图 4 工具架构

4.2 实验结果

我们在实验部分首先讨论第 3.4 节中的 BPP 的例子(如图 3 所示),该 BPP 包含 3 个进程符号以及 3 条规则.我们对以下 4 个 EG 逻辑公式在限制步长 k 分别为 1,5,10,20,50 和 100 的条件下进行限界模型检测.

$$\varphi_1 = X_1 + X_2 \geqslant 1 \wedge X_3 \geqslant 0,$$
$$\varphi_2 = EG(E\langle a\rangle(X_2 + X_3 \geqslant 2)),$$
$$\varphi_3 = EG(X_1 + X_2 \geqslant 2 \rightarrow E\langle a\rangle(X_1 \geqslant 2 \wedge X_3 \geqslant 1)),$$
$$\varphi_4 = EG(AF(X_1 + X_2 \geqslant 2)).$$

实验结果见表 2,实验结果的单位为 s.

Table 2 Experimental results

表 2 实验结果

	1	5	10	20	50	100
φ_1	0.01	0.01	0.01	0.01	0.01	0.01
φ_2	0.02	0.03	0.05	0.08	0.18	0.34
φ_3	0.03	0.05	0.07	0.12	0.27	0.54
φ_4	0.05	0.09	0.27	0.98	6.22	52.81

从表 2 可以看出,对于φ_1,步长 k 不影响求解时间.这是因为φ_1是两个原子命题公式的合取式,根据算法 1,对于原子命题公式,作为参数的 k 不参与线性整数算术公式的构造,因此求解φ_1的时间总体趋于一致.而对于φ_2,φ_3,φ_4,求解时间总体上随着步长 k 的增大而增加.公式φ_3的嵌套深度比φ_2的和φ_4的均要小,然而求解φ_3的时间只比求解φ_2的稍多,而求解φ_4的时间相对于求解φ_2,φ_3的时间较多,这在步长 k 较大,如 k=50,100 时尤为明显.这是因为限制求解速度的原因在于变量的数量.对于 EG 算子,根据算法 1,生成的线性整数算术公式需要引入$(k+1)\cdot n$个变量,其中,n 为 BPP 中进程符号的数量.而φ_4实际上等价于公式 $EG(\neg EG\neg(X_1+X_2\geqslant 2))$,因此对于$\varphi_4$,生成的线性整数算术公式中含有的变量数目为$(k+1)^2\cdot n^2$,因此在 k 较大时,求解器在较长时间后返回结果.

因为已有的程序验证基准案例并不能证明其全部可以 BPP 可解,所以这些案例并不能直接使用.同时,现有的与 Petri 网相关的验证工具都是针对传统 Petri 网,它们所使用的标准测试集也不再适用.因此,我们随机生成了测试用例,对不同规模的 BPP 在多个步长下进行实验.BPP 的规模主要体现在其进程符号的数量以及迁移规则的数目.我们实验的对象是进程符号数分别是 20,50 和 100 的 BPP.对于具有相同进程符号数的 BPP,我们将迁移规则数分别设置为 10,20 和 30 进行实验.对于 EG 逻辑公式,我们选取了上述的φ_2,φ_3,φ_4.表 3~表 5 分别对应 k 分别为 15,20,25 的实验结果,单位为 s.

Table 3 Experimental results (k=15)

表 3 实验结果(k=15)

	φ_2		φ_3		φ_4	
20	10	0.692	10	0.841	10	9.029
	20	1.346	20	1.631	20	12.689
	30	1.927	30	2.128	30	17.362
50	10	1.597	10	2.146	10	20.954
	20	3.102	20	3.588	20	29.190
	30	4.718	30	4.957	30	51.181
100	10	3.450	10	4.285	10	36.718
	20	6.092	20	6.848	20	64.620
	30	8.826	30	10.503	30	99.130

Table 4 Experimental results (k=20)

表 4 实验结果(k=20)

	φ_2		φ_3		φ_4	
20	10	0.915	10	1.137	10	11.429
	20	1.810	20	1.917	20	22.122
	30	2.664	30	2.978	30	33.313
50	10	2.135	10	2.641	10	23.680
	20	4.124	20	4.645	20	50.214
	30	6.476	30	6.803	30	91.353
100	10	4.183	10	5.572	10	59.651
	20	9.047	20	9.963	20	100.159
	30	12.443	30	13.297	30	171.353

Table 5 Experimental results (k=25)

表 5 实验结果(k=25)

	φ_2		φ_3		φ_4	
20	10	1.158	10	1.545	10	21.584
	20	2.285	20	2.645	20	30.469
	30	3.668	30	4.126	30	44.819
50	10	2.854	10	3.419	10	42.042
	20	5.429	20	5.750	20	88.374
	30	8.528	30	8.657	30	111.447
100	10	5.206	10	6.252	10	88.267
	20	12.821	20	11.074	20	181.874
	30	17.927	30	18.893	30	226.909

从实验结果可以看出,求解时间随着进程符号数量和迁移规则数目的增加以及 k 的增大而增加.进程符号数量影响了线性整数算术公式中引入的量词和变量的数目.此外,对于一个 BPP 状态 s,因为我们要保证经过一次迁移后得到的状态 t,s 和 t 中每一个进程符号的数量关系要与相应的规则一致,因此 BPP 进程符号数的增多会导致对应约束数量的增加,从而影响求解时间.在迁移约束和路径约束中,由于要在 BPP 规则集中寻找使得迁移和路径成立的规则,因此规则集的大小影响了迁移约束和路径约束中析取式的公式大小.而步长 k 决定了最外层算子为 EG 的公式对应的线性整数算术公式的路径约束中合取式的大小,即子约束的数目,同时,k 也对引入的变量数目产生影响.

5 相关工作

在基于 BPP 和 Petri 网的模型检测问题上,对于线性时间逻辑,Petri 网上的线性时间 mu-演算的模型检测问题是可判定的[20].然而,对于一些表达力比较弱的逻辑,如原子命题为 $EN(a)$且只包含 Fφ作为模态算子的线性时间逻辑,在 VBPP 上的模型检测却是不可判定的[12].对于分支时间逻辑,除了上文提及的不可判定性结果,对于 EF 逻辑,即在 Hennessy-Milner Logic 的基础上增加 EF 算子的逻辑,Esparza 还在文献[12]中证明了以下结论:

(1) 基于 Petri 网的模型检测问题是不可判定的,然而对于 BPP,即使给定的 BPP 是有穷状态的,该问题是 PSPACE 难的;(2) 在 BPP 上,对于长度固定的公式(fixed formula),模型检测问题是 $\sum_d^p$ 难的,其中,d 是模态算子的嵌套深度.在这个结论的基础上,Mayr 证明了若给定的模型为 BPP,则 EF 逻辑的模型检测问题实际上分别是 PSPACE 完备和 $\sum_d^p$ 完备的[21].EGF 是 CTL*的一个算子,其表达的是:“存在一条路径,使得某个特定的性质被满足无穷次”.在正则模型检测(regular model checking)中,EGF 代表了活性和反复可达性(recurrent reachability).在 EF 逻辑的基础上加上 EGF 算子,我们可以得到 EGF 逻辑.文献[22]证明了 BPP 上对 EGF 逻辑的模型检测问题是可判定的.而 Fu 证明了该问题的复杂度是 PSPACE 完备且对于长度固定的公式是 $\sum_d^p$ 完备的[13],也就是说,增加 EGF 算子并没有提高模型检测的复杂度.证明的关键思路是使用一种称为半片段(semisegment)的结构,而该结构也被用于处理基于 Petri 网模型检测的状态爆炸问题[23].

在安全性问题上,Petri 网的可达性问题是可判定的[24],且最早被证明是 EXPSPACE 难的[25],然而该问题的复杂度仍然是学术界的一个开问题.最近,文献[26]证明了该问题的一个新的 Ackermannian 上界,而文献[9]证明了该问题的一个新的 Tower 难下界.Petri 网的可覆盖性问题是 EXPSPACE 完备的[27,28],对于 Petri 网的可覆盖性,目前检测工具主要分为两类:一类基于完备算法,对状态空间进行搜索,判断可达的状态集是否覆盖了指定状态,这类工具包括 BFC[1],IIC[29]和 MIST;而另一类工具如 Petrinizer[30]将给定的 Petri 网的结构和状态以及可覆盖性条件转换为约束,然后交给求解器进行求解.然而,Petrinizer 在理论上是不完备的,其无法判定不安全的测试用例.以上这些工具由于 Petri 网的可覆盖性问题复杂度过高,因此都无法处理规模较大的测试用例.而 BPP 的可达性以及可覆盖性问题是 NP 完备问题[11].针对 BPP,文献[31]实现了工具 CFPCV,在 Petri 网可覆盖性检测的安全性方法的基础上,额外增加了基于 BPP 的子网可标记方法,并以此来验证状态是否可达,保证了算法在理论上是完备的.

在限界模型检测上,模型检测问题一般被规约为 SAT 问题.该方法最先被运用到检测 LTL 性质上[32],接着在 ACTL 性质检测上也具有应用[33].但是这两类研究只关注时序逻辑的存在片段(existential fragment),其限界语义不能适用于如本文提及的活性 $AF\varphi$等表示全称(universal)性质的公式.文献[34]就此问题提出了 CTL 的限界语义,并将 CTL 公式编码为 QBF 公式.以上研究讨论的模型都是有穷状态系统,对于无穷状态系统,文献[35]提出了将性质分别用 LTL_{-X} 和 $ECTL_{-X}$ 表示,并在分布式时间 Petri 网上进行限界模型检测的方法,该方法仍是将模型检测问题编码为命题逻辑公式.不同于直接处理时序逻辑公式的方法,文献[36]首先将时序逻辑公式转换为对应的ω自动机,其接受条件为安全性或活性等性质条件,然后将ω自动机编码为 Presburger 算术,而该方法适用于无穷状态系统.

6 总结及未来的工作

本文关注基本并行进程上 EG 逻辑的限界模型检测问题,并将该问题转化为线性整数算术公式的可满足性问题.首先,我们根据基于基本并行进程的限定步长为 k 的 EG 逻辑限界语义,以及基本并行进程的模型结构和表示特定性质的 EG 逻辑公式,定义相应的约束,并给出了对应的基于线性整数算术公式的刻画,然后将线性整数算术公式作为 SMT 求解器的输入进行求解.

我们未来的工作主要包括两个方面:一是对并发程序的一些性质如动态更新(dynamic updating)进行分析,使用基本并行进程对并发程序进行建模并验证这些性质;二是尝试寻找描述并发程序的其他模型,并比较和分析其在安全性和活性的验证上与基本并行进程的异同.

References:

[1] Kaiser A, Kroening D, Wahl T. Dynamic cutoff detection in parameterizedconcurrent programs. In: Proc. of the 22nd Int'l Conf. on Computer Aided Verification, Vol.6174. Berlin: Springer-Verlag, 2010. 645–659.

[2] D'Osualdo E, Kochems J, Ong CHL. Automatic verification of Erlang-styleconcurrency. In: Proc. of the 20th Int'l Symp. on Static Analysis, Vol.7935. Berlin: Springer-Verlag, 2013. 454–476.

[3] Kaiser A, Kroening D, Wahl T. Efficient coverability analysis by proof minimization. In: Proc. of the 23rd Int'l Conf. on Concurrency Theory, Vol.7454. Berlin: Springer-Verlag, 2012. 500−515.

[4] Bouajjani A, Emmi M. Bounded phase analysis of message-passing programs. In: Proc. of the 18th Int'l Conf. on Tools and Algorithms for the Construction and Analysis of Systems, Vol.7214. Berlin: Springer-Verlag, 2012. 451−465.

[5] Ganty P, Majumdar R. Algorithmic verification of asynchronous programs. ACM Trans. on Programming Languages and Systems, 2012,34(1):Article No.6.

[6] Petri CA. Communication with automata [Ph.D. Thesis]. Technische Universitat Darmstadt, 1962.

[7] Vardi MY. Verification of concurrent programs: The automata-theoretic framework. Annals of Pure and Applied Logic, 1991,51: 79−98.

[8] Ganty P, Majumdar R, Rybalchenko A. Verifying liveness for asynchronous programs. In: Proc. of the 36th Symp. on Principles of Programming Languages. New York: ACM, 2009. 102−113.

[9] Czerwinski W, Lasota S, Lazic R, Leroux J, Mazowiecki F. The reachability problem for Petri nets is not elementary. In: Proc. of the 51st Annual Symp. on Theory of Computing. New York: ACM, 2019. 24−33.

[10] Christensen S. Decidability and decomposition in process algebras [Ph.D. Thesis]. Edinburgh: The University of Edinburgh, 1993.

[11] Esparza J. Petri nets, commutative context-free grammars, and basic parallel processes. Fundamenta Informaticae, 1997,31:13−25.

[12] Esparza J. Decidability of model checking for infinite-state concurrent systems. Acta Informatica, 1997,34(2):85−107.

[13] Fu H. Model checking EGF on basic parallel processes. In: Proc. of the 9th Int'l Symp. on Automated Technology for Verification and Analysis. Berlin: Springer-Verlag, 2011. 120−134.

[14] Nieuwenhuis R, Oliveras A, Tinelli C. Solving SAT and SAT modulo theories: From an abstract Davis-Putnam-Logemann-Loveland procedure to DPLL (T). Journal of the ACM, 2006,53(6):937−977.

[15] Dutertre B. Yices 2.2. In: Proc. of the Int'l Conf. on Computer Aided Verification, Vol.8559. Berlin: Springer-Verlag, 2014. 737−744.

[16] De Moura L, Bjorner N. Z3: An efficient SMT solver. In: Proc. of the 14th Int'l Conf. on Tools and Algorithms for the Construction and Analysis of Systems. Berlin: Springer-Verlag, 2008. 337−340.

[17] Barrett C, Conway CL, Deters M, Hadarean L, Jovanović D, King T, Reynolds A, Tinelli C. CVC4. In: Proc. of the 23rd Int'l Conf. on Computer Aided Verification. Berlin: Springer-Verlag, 2011. 171−177.

[18] Cimatti A, Griggio A, Schaafsma BJ, Sebastiani R. The MathSAT5 SMT solver. In: Proc. of the 19th Int'l Conf. on Tools and Algorithms for the Construction and Analysis of Systems. Berlin: Springer-Verlag, 2013. 93−107.

[19] Minsky M. Computation: Finite and Infinite Machines. Prentice Hall, 1967.

[20] Esparza J. On the decidability of model checking for several mu-calculi and Petri nets. In: Proc. of the Trees in Algebra and Programming (CAAP'94). Berlin: Springer-Verlag, 1994. 115−129.

[21] Mayr R. Weak bisimulation and model checking for basic parallel processes. In: Proc. of the 16th Conf. on Foundations of Software Technology and Theoretical Computer Science, Vol.1180. Berlin: Springer-Verlag, 1996. 88−99.

[22] To AW, Libkin L. Recurrent reachability analysis in regular model checking. In: Proc. of the 15th Int'l Conf. on Logic for Programming, Artificial Intelligence, and Reasoning, Vol.5330. Berlin: Springer-Verlag, 2008. 198−213.

[23] Strehl K, Thiele L. Interval diagram techniques for symbolic model checking of Petri nets. In: Proc. of the 1999 Design, Automation and Test in Europe. Berlin: Springer-Verlag, 1999. 756−757.

[24] Mayr E. An algorithm for the general Petri net reachability problem. In: Proc. of the 13th Annual Symp. on Theory of Computing. New York: ACM, 1981. 238−246.

[25] Lipton R. The reachability problem requires exponential-space hard. Technical Report, 62, Department of Computer Science, Yale University, 1976.

[26] Leroux J, Schmitz S. Reachability in vector addition systems is primitive-recursive in fixed dimension. In: Proc. of the 34th Annual Symp. on Logic in Computer Science. IEEE, 2019. 1−13.

[27] Karp RM, Miller RE. Parallel program schemata. Journal of Computer and System Sciences, 1969,3(2):147−195.

[28] Rackoff C. The covering and boundedness problems for vector addition systems. Theoretical Computer Science, 1978,6(2): 223−231.

[29] Kloos J, Majumdar R, Niksic F, *et al*. Incremental, inductive coverability. In: Proc. of the Int'l Conf. on Computer Aided Verification. Berlin: Springer-Verlag, 2013. 158−173.

[30] Esparza J, Ledesma-Garza R, Majumdar R, *et al*. An SMT-based approach to coverability analysis. In: Proc. of the Int'l Conf. on Computer Aided Verification. Berlin: Springer-Verlag, 2014. 603−619.

[31] Ding RJ, Li GQ. Efficient implementation of coverability verification on communication-free Petri net. Ruan JianXue Bao/Journal of Software, 2019,30(7):1939−1952 (in Chinese with English abstract). http.//www.jos.org.cn/1000-9825/5750.htm [doi: 10.13328/j.cnki.jos.005750]

[32] Biere A, Cimatti A, Clarke EM, Zhu YS. Symbolic model checking without BDDs. In: Proc. of the Tools and Algorithms for Construction and Analysis of Systems, Vol.1579. Berlin: Springer-Verlag, 1999. 193−207.

[33] Penczek W, Wozna B, Zbrzezny A. Bounded model checking for the universal fragment of CTL. Fundamenta Informaticae, 2002, 51(1-2):135−156.

[34] Zhang WH. Bounded semantics. Theoretical Computer Science, 2015,564:1−29.

[35] Meski A, Pólrola A, Penczek W, Wozna-Szczesniak B, Zbrzezny A. Bounded model checking approaches for verification of distributed time Petri nets. In: Proc. of the Int'l Workshop on Petri Nets and Software Engineering, Vol.723. CEUR-WS.org, 2011. 72−91.

[36] Schüle T, Schneider K. Bounded model checking of infinite state systems. Formal Methods in System Design, 2007,30(1):51−81.

附中文参考文献:

[31] 丁如江,李国强.非交互式 Petri 网可覆盖性验证的高效实现.软件学报,2019,30(7):1939−1952. http.//www.jos.org.cn/1000-9825/5750.htm [doi: 10.13328/j.cnki.jos.005750]

谭锦豪(1996－),男,硕士生,CCF 学生会员,主要研究领域为形式化验证,程序语言理论.

李国强(1979－),男,博士,副教授,CCF 高级会员,主要研究领域为形式化方法,程序语言理论,知识表示与推理.