软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2018,29(6):1635−1646 [doi: 10.13328/j.cnki.jos.005459]

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

APTL 公式的可满足性检查工具*

王海洋[1,2], 段振华[1,2], 田 聪[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](ISN 国家重点实验室(西安电子科技大学),陕西 西安 710071)
通信作者: 段振华,田聪, E-mail: {zhhduan,ctian}@mail.xidian.edu.cn

摘 要: 交替投影时序逻辑(alternating projection temporal logic,简称 APTL)公式简单易懂,表达能力强;不仅可以描述经典时序逻辑 LTL 可以描述的性质,而且可以描述与区间相关的顺序和循环性质以及开放系统和多智能体系统中与博弈相关的性质.在验证系统是否满足所给的 APTL 公式所描述的性质之前需要检查公式的可满足性.根据检查 APTL 公式的可满足性的方法,开发实现了工具 APTL2BCG.具体细节如下:首先,利用公式 P 的范式构造 P 的标记范式图(labeled normal form graph,简称 LNFG);然后,将 LNFG 转化为广义的基于并发博弈结构的交替 Büchi 自动机(generalized alternating Büchi automaton over concurrent game structure,简称 GBCG);最后,将 GBCG 转化为基于并发博弈结构的交替 Büchi 自动机(alternating Büchi automaton over concurrent game structure,简称 BCG)并且化为最简形式并检查公式 P 的可满足性.
关键词: 交替投影时序逻辑;范式;标记范式图;基于并发博弈结构的交替 Büchi 自动机;可满足性
中图法分类号: TP311

中文引用格式: 王海洋,段振华,田聪.APTL 公式的可满足性检查工具.软件学报,2018,29(6):1635−1646. http://www.jos.org.cn/1000-9825/5459.htm
英文引用格式: Wang HY, Duan ZH, Tian C. Tool for checking satisfiability of APTL formulas. Ruan Jian Xue Bao/Journal of Software, 2018,29(6):1635−1646 (in Chinese). http://www.jos.org.cn/1000-9825/5459.htm

Tool for Checking Satisfiability of APTL Formulas

WANG Hai-Yang[1,2], DUAN Zhen-Hua[1,2], TIAN Cong[1,2]

[1](Institute of Computing Theory and Technology, Xidian University, Xi’an 710071, China)
[2](State Key Laboratory of Integrated Service Networks (Xidian University), Xi’an 710071, China)

Abstract: Alternating projection temporal logic (APTL) has a concise syntax with strong expressive power. It is able to express not only properties specified in classical temporal logic LTL, but also interval related sequential and periodical properties, as well as game related properties of open systems and multi-agent systems. To verify whether a system satisfies an APTL formula, the satisfiability of the APTL formula should be checked. In this paper, based on the method for checking satisfiability of an arbitrary APTL formula provided, a supporting tool APTL2BCG has been developed. The details of implementation are given as follows. Firstly, LNFG of P is constructed according to the normal form of the formula P. Then, the LNFG is transformed to a generalized alternating Büchi automaton over concurrent game structure (GBCG). Finally, the alternating Büchi automaton is developed over concurrent game structure (BCG) from the obtained GBCG, and the BCG is simplified for checking the satisfiability of the formula P.
Key words: alternating projection temporal logic; normal form; LNFG (labeled normal form graph); BCG (alternating Büchi automaton over concurrent game structure); satisfiability

* 基金项目: 国家自然科学基金(61732013, 61420106004)
Foundation item: National Natural Science Foundation of China (61732013, 61420106004)
本文由形式化方法的理论基础专题特约编辑傅育熙教授、李国强副教授、田聪教授推荐.
收稿时间: 2017-06-26; 修改时间: 2017-09-01; 采用时间: 2017-11-06; jos 在线出版时间: 2017-12-28
CNKI 网络优先出版: 2017-12-29 13:19:05, http://kns.cnki.net/kcms/detail/11.2560.TP.20171229.1318.001.html

基于区间的交替时序逻辑在文献[1]中提出,其中包括交替区间时序逻辑(alternating interval temporal logic,简称 AITL)和交替投影时序逻辑(alternating projection temporal logic,简称 APTL)[2,3].APTL 是由命题投影时序逻辑(propositional projection temporal logic,简称 PPTL)[4]扩展得到的,并且是在 AITL 中引入投影操作符 *prj* 得到的,因此,APTL 也是 AITL 的扩展逻辑.时序逻辑是形式化验证的关键元素之一,Pnueli 于 1977 年提出了线性时序逻辑(linear temporal logic,简称 LTL)[5]用于规约系统的线性时序性质.LTL 公式描述的是系统的任意一次运行中的状态以及它们之间的关系.

Clarke 和 Emerson 于 1980 年提出了分支时序逻辑,即计算树逻辑(computation tree logic,简称 CTL)[5],CTL 公式不仅可描述系统中状态的先后关系,而且可描述系统存在一条路径和任意一条路径满足的性质.经典逻辑 LTL,CTL 存在局限性,如无法描述某些循环和并发的性质.区间时序逻辑(interval temporal logic,简称 ITL)[4]和投影时序逻辑(projection temporal logic,简称 PTL)[2,6]可方便地描述系统的顺序、循环性质,另外,PTL 公式还可方便地描述系统的并发性质.为了描述开放系统和多智能体系统中合作及博弈相关的性质,文献[1]中提出了 APTL.APTL 公式不仅可以描述经典时序逻辑 LTL 可以描述的性质,而且可以描述与区间相关的顺序和循环性质以及开放系统和多智能体系统中与博弈相关的性质.多智能体系统是现阶段各领域的研究热点,并得到了广泛应用.比如人工智能、竞拍.验证多智能体是否满足人们所期望的性质,是人工智能中面临的关键问题之一,APTL 可用于描述多智能体系统的各种性质.

APTL 中包括 chop 和 projection 时序操作符,使得 APTL 公式可方便地描述系统的顺序和并发性质.例如,智能体集合 *A* 使得 *p* 在第 10 和第 16 个状态之间成立,可用 APTL 公式表示为

$$len(10);_{\ll\varnothing\gg} len(6)\wedge\Diamond_{\ll A\gg} p;_{\ll\varnothing\gg} \text{true}.$$

对于 APTL 公式 $(P_1,\ldots,P_m)\,prj_{\ll A\gg}Q$,公式 $P_1;_{\ll A\gg}\ldots;_{\ll A\gg}P_m$ 和 Q 是相对独立的,每一个公式对应一个区间模型;公式 Q 与公式 $P_1;_{\ll A\gg}\ldots;_{\ll A\gg}P_m$ 的模型是并行的,他们只是在特定的汇合点上通信,而且两个公式可能在不同的时间点结束.

判断公式的可满足性是程序验证过程中不可或缺的一个环节,根据文献[1]中给出的 APTL 公式的可满足性检查方法,开发实现了检查工具 APTL2BCG.具体细节如下:首先,将给定的 APTL 公式转化为范式,构造标记范式图(labeled normal form graph,简称 LNFG);然后,将 LNFG 转化为对应的 GBCG,并将 GBCG 转化为 BCG;最后,根据本文给出的 BCG 的化简方法将 BCG 化为最简形式,检查 BCG 是否为空,得出公式是否可满足.

本文第 1 节介绍 APTL 的语法语义.第 2 节详细描述工具 APTL2BCG 的框架和实现过程.第 3 节展示工具效果.第 4 节总结全文并展望将来的工作.

1 基础概念

1.1 APTL语法语义

语法. 令$\mathcal{P}$为原子命题的可数集合,$\mathcal{A}$为代理的可数集合.APTL 的语法定义如下:

$$P ::= p\mid\neg P\mid P\vee Q\mid\bigcirc_{\ll A\gg}P\mid(P_1,\ldots,P_m)\,prj_{\ll A\gg}Q,$$

其中,$p\in\mathcal{P}$,$A\subseteq\mathcal{A}$,$P_i(i=1,\ldots,m)$,P 及 Q 为 APTL 公式,$\bigcirc_{\ll A\gg}$(next)及 $prj_{\ll A\gg}$(projection)是基本的时序操作符.

语义. APTL 公式的语义是在并发博弈结构(concurrent game structure,简称 CGS)[1,7]的基础上定义的,CGS 是七元组 $C=\langle\mathcal{P},\mathcal{A},S,S_0,l,\Delta,\tau\rangle$,详细的解释见文献[1].

根据 CGS 的定义,在$\mathcal{P}$上定义了一个状态 s,是$\mathcal{P}$到 $B=\{\text{true},\text{false}\}$的投影,$s:\mathcal{P}\to B$.在 CGS 中,以状态 s 为起始节点的一条执行路径$\lambda(s)$满足 APTL 公式 P,标记为$\lambda(s)\vDash P$.CGS C 满足 APTL 公式 P 当且仅当所有以 CGS 的初始节点为起始节点的执行序列满足公式 P,标记为 $C\vDash P$.

$\vDash$关系定义如下:

- $\lambda(s)\vDash p$ 当且仅当命题 $p\in\mathcal{P}$有 $p\in l(s)$;
- $\lambda(s)\vDash\neg P$ 当且仅当$\lambda(s)\nvDash P$;

- $\lambda(s)\vDash P\vee Q$ 当且仅当$\lambda(s)\vDash P$ 或$\lambda(s)\vDash Q$;
- $\lambda(s)\vDash\bigcirc_{\ll A\gg}P$ 当且仅当$|\lambda(s)|\geq 2$,并且 A 存在一个策略 f_A,使得$\lambda(s)\in out(s,f_A)$并且$\lambda(s)[1,|\lambda|]\vDash P$;
- $\lambda(s)\vDash(P_1,\dots,P_m)\,prj_{\ll A\gg}Q$ 当且仅当 A 存在一个策略 $f_A,\lambda(s)\in out(s,f_A),0=r_0\leq r_1\leq\dots\leq r_m\leq|\lambda(s)|$,使得 $\lambda(s)[r_{i-1},r_i]\vDash P_i,0<i\leq m$ 并且$\lambda\vDash Q.\lambda$有以下两种情况得到.
  a) $r_m<|\lambda(s)|$,那么$\lambda=\lambda(s)\downarrow(r_0,\dots,r_m)\cdot\lambda(s)[r_m+1,\dots,|\lambda(s)|]$;
  b) $r_m=|\lambda(s)|$,那么$\lambda=\lambda(s)\downarrow(r_0,\dots,r_m)$.

1.2 交替Büchi自动机

基于并发博弈结构的自动机在文献[1,8]中已经介绍,本节简单介绍了交替 Büchi 自动机[9]的定义.在计算中,传统的自动机只包含存在性迁移关系,交替自动机[10]包含了存在性迁移关系和全局性迁移关系.

定义(交替 Büchi 自动机). 交替 Büchi 自动机(alternating Büchi automaton)是一个五元组 $A=(\mathcal{P},Q,q_0,\delta,F)$,其中,$\mathcal{P}$为有穷字母表,$Q$ 为有穷状态集合,q_0 为初始状态,$F\in 2^Q$ 为接收状态集合,$\delta:Q\times\mathcal{P}\to\mathbb{B}^+(Q)$为迁移函数.由于交替自动机中有全局性迁移关系,所以交替Büchi自动机读入一个字生成无穷树而不是无穷序列.如图1所示为一个交替自动机$(\{a,b\},\{s,p,q\},s,\delta,\{p\})$,其中,从状态 s 出发的两个标记有 b 的迁移为全局性迁移关系,即:图中加粗的两条边为全局性迁移,其他都是选择性迁移关系.给定一个字 *abbab*…,自动机读入该字的一个生成树如图 2 所示.

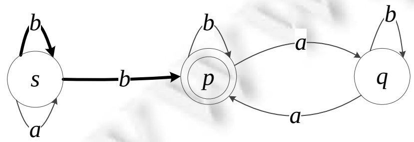


Fig.1 An alternating automaton

图 1 交替自动机

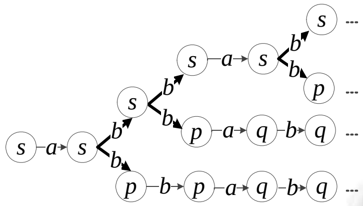


Fig.2 A spanning tree of the automaton in Fig.1 on the word *abbab*…

图 2 图 1 中的自动机读入字 *abbab*…的一个生成树

如果生成树中的每一条分支都满足 Büchi 条件,那么这个生成树是被接收的.一个字是被交替 Büchi 自动机接收的当且仅当存在一个可接收的生成树.

2 原型工具

2.1 工具框架

我们开发了检查 APTL 公式的可满足性的原型工具 APTL2BCG,该工具是基于 Windows 平台采用 C++语言开发的,工具架构如图 3 所示.

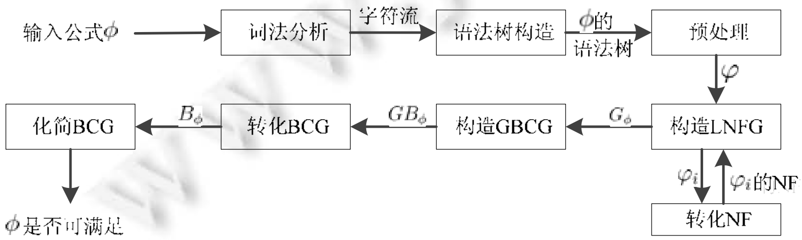


Fig.3 Architecture of APTL2BCG

图 3 APTL2BCG 的架构

APTL2BCG 主要包括以下几个模块,分别是词法分析与语法树构造模块、预处理模块、转化范式模块、构造 LNFG[1]模块、转化 GBCG[1]模块、将 GBCG 转化为 BCG[1]模块、BCG 化简模块,根据得到的 BCG 是否

为空检查公式的可满足性.该工具以 APTL 公式作为输入,格式为文本文件.在该文件中,公式的各操作符的表示方法如下:!表示¬,|和&分别表示∨和∧,()表示○,<>表示◊,[]表示□,<<>>表示≪≫.例如,公式 $\bigcirc_{\ll A\gg}P\wedge\Diamond_{\ll B\gg}\neg Q$ 表示为()<<*A*>>*P*&<><<*B*>>!*Q*.

检查 APTL 公式ϕ的可满足性的具体过程是:

- 首先,对输入的公式ϕ进行词法分析,词法无误则构造语法树,对公式进行预处理;
- 然后,构造公式的 LNFG,构造过程需要不断的调用范式转化模块将产生的新节点转化为范式,并据此生成新的节点和边,直到所有的节点表示的子公式均已被处理;
- 接着,将 LNFG 转化为 GBCG;
- 最后,将 GBCG 转化为 BCG 并化简,接着检查最简 BCG 是否为空:如果 BCG 为空,则公式ϕ不可满足;反之亦然.

在检查公式的可满足性时,将公式以文本形式输入,在检查过程中输出描述 LNFG,GBCG 和 BCG 的文本文档,文本格式是工具 Graphviz 的输入文本的格式,调用 Graphviz 工具可将 LNFG,GBCG 和 BCG 的图输出,最后输出公式的可满足性.

2.2 语法树构造

在构造 BCG 的过程中,针对节点和边的操作都是在语法树的概念上进行的,CTreeNode(NODETYPE ptype, int typeofAgentset,string agentstr,string pstr,int istr,CTreeNode *s_1,CTreeNode *s_2,CTreeNode *s_3)表示树节点.其中,ptype 为节点类型,如节点类型为 CHOP,则以该节点为根的子树表示的是 *chop* 类型的公式;节点类型为 PROJECTION,则以该节点为根的子树表示的是 *prj* 类型的公式,等等.typeofAgentset 表示代理集合的类型,其中:1 代表≪≫,≪*A*≫表示无论其他代理采取什么策略,集合 *A* 中的代理总能使得公式成立;2 代表[[]],[[*A*]]表示集合 *A* 中的代理无论采取什么策略都不能使公式不成立.agentstr 表示代理的字符串,pstr 是该节点的字符串,istr 为该节点的整数值,s_1,s_2,s_3 分别为该节点的 3 个孩子节点.没有孩子节点的节点表示为 CTreeNode(NODETYPE ptype,int typeofAgentset,string agentstr,string pstr,int istr),如果树节点有一个孩子节点那么 s_2 和 s_3 为空,类似的有两个孩子节点的树节点的 s_3 为空.部分 APTL 公式的语法树结构如图 4 所示.

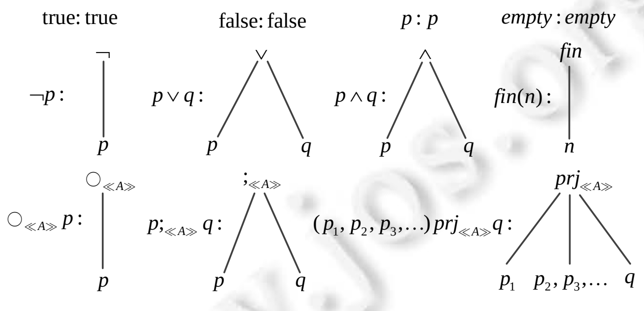


Fig.4 Syntax trees of APTL formulas

图 4 APTL 公式的语法树

由图 4 可以看出,语法树的根节点能够表示相应的公式类型.true,false,*empty*,*skip* 以及原子命题的语法树均不含孩子节点.公式$\neg p$,$fin(n)$,$len(n)$的语法树均含有一个孩子节点,分别为 p,n,n.在公式 $p\vee q$(或 $p\wedge q$)的语法树中,左右孩子节点分别对应公式的左右析取项(或合取项). $\bigcirc_{\ll A\gg}p$,$\Diamond_{\ll A\gg}p$ 和 $\Box_{\ll A\gg}p$ 的语法树的根节点均表示节点的类型和代理集合,而且均只有一个孩子节点 p.公式 $p;_{\ll A\gg}q$ 的语法树的左右孩子分别为 p 和 q.公式$(p_1,p_2,p_3,\ldots)\ prj_{\ll A\gg}q$ 的语法树有 3 个孩子节点,分别对应 $p_1,p_2,p_3,\ldots$和 q.

2.3 公式的预处理

由于输入的 APTL 公式本身可能存在冗余的情况,因此要先经过预处理过程对输入的合法的 APTL 公式进

行等价化简.预处理函数的伪代码如函数 *PRE*(*CtreeNode *ptree*)所示,对公式的预处理过程是在语法树上进行的.其中,输入的是待处理公式的语法树,输出化简后的公式的语法树.

2.4 BCG构造

任意一个合法的 APTL 公式都可以构造相应的 BCG.首先,根据文献[1]中的算法,实现了将 APTL 公式转化为 LNFG 的过程,并将 LNFG 转化为相应的 GBCG;然后,将 GBCG 转化为对应的 BCG.

```
void PRE(CtreeNode *ptree){
if (ptree==NULL) return;
switch(ptree){
case P→Q
    ptree←¬P∨Q;
case P↔Q
    ptree←¬P∧¬Q∨P∧Q;
case ¬¬P
    ptree←P;
case skip
    ptree ←○≪∅≫ε ;
case ¬ε
    ptree ←○≪∅≫ true ;
case len(n)
    ptree ←○^n≪∅≫ ε ;
case ◊≪A≫P
    ptree ← true;≪A≫ P ;
case P∧Q
  if P==true
    ptree←Q;
  …
  if  P == ε ∧ Q ==○≪A≫Q′
    ptree←false;
case P∨Q
  if P==Q
    ptree←P;
  if P==tru∨Q==true
    ptree←true;
  …
  if Q==false
    ptree←P;
  …
  }
}
```

构造 LNFG 模块通过循环调用函数 NF 来构造公式的 LNFG.函数 *NF*(*CtreeNode *ptree*)将 APTL 公式转化为范式,其中输入的是 APTL 公式的语法树,得到 APTL 公式的范式的语法树.构造 APTL 公式的范式图的函数为 *LNFG*(*CtreeNode* ptree*,*CGraph* pgraph*),其中,*ptree* 为公式的范式的语法树,*pgraph* 为生成的 LNFG.

给定一个 LNFG $G=(Q,q_0,\mathcal{A},\delta,\varepsilon,\mathbb{L}=\{\mathbb{L}_1,\ldots,\mathbb{L}_m\})$存在一个 GBCG $GB=(Q',q_0',\mathcal{A}',\delta',F=\{F_1,\ldots,F_n\})$与之相对应.其中,$Q'=Q,q_0'=q_0,\mathcal{A}'=\mathcal{A},\delta'=\delta\cup\delta(d,\varepsilon)=(\ll\varnothing\gg,d),F=\{\mathbb{L}_1^{\sim},\ldots,\mathbb{L}_m^{\sim}\}$.

任意一个 GBCG $GB=(\mathcal{P},\mathcal{A},Q,q_0,\delta,F=\{F_1,\ldots,F_n\})$都可以转化为一个 BCG $B=(\mathcal{P},\mathcal{A},Q',q_0',\delta',F')$,根据文献[11]中的方法,实现了将 GBCG 转化为 BCG 的过程.如果 GB 的接收集合 F 中只有一个接收状态子集,也就是说,$F=\{F_1\}$,则无需转换,该 GBCG 即为 BCG,那么对应的 BCG 的接收集合为 $F'=F_1$.如果 $n\geqslant 2$,将 GB 的状态复制 n 遍,第 i 个 GB 中的非接收状态与第 i 个 GB 中的相应状态相连接,第 i 个 GB 的 F_i 的状态与第 i+1 个 GB 中相应的状态相连接.得到的 B 的接收集合 F 中的状态是第 1 个 GB 中的接收集合的 F_1 中的状态.B 的接收条件是 F 中的一个状态无穷次的出现,根据该 BCG B 的构造过程,能推出所有的接收状态集 F_i 都能够被无穷次地访问到.

将一个 GBCG $GB=(\mathcal{P},\mathcal{A},Q,q_0,\delta,F=\{F_1,\ldots,F_n\})$转化为 BCG $B=(\mathcal{P},\mathcal{A},Q',q_0',\delta',F')$的伪代码如下所示.

```
void BCG(CGraph *GBCG,CGraph *BCG){
Q′←Q×{1,…,n};
q′0 ←< q0,1>;
F′←{<qF,1>|qF∈F1};
if q∉Fi
   δ′(<q,i>,p,A′)←{<q′,i>|q′∈δ(q,p,A′)};
else if q∈Fi
   δ′(<q,i>,p,A′)←{<q′,i+1>|q′∈δ(q,p,A′)};
}
```

例子:通过构造公式 $p;_{\ll a\gg}\Box_{\ll a\gg}q;_{\ll b\gg}\Box_{\ll b\gg}r$ 的 BCG 过程,讲解给定一个 APTL 公式构造 BCG 的具体细节.

以下是构造公式 $p;_{\ll a\gg}\Box_{\ll a\gg}q;_{\ll b\gg}\Box_{\ll b\gg}r$ 的 LNFG 的过程.

首先,我们将公式写为 $p\wedge fin(l_1);_{\ll a\gg}\Box_{\ll a\gg}q;_{\ll b\gg}\Box_{\ll b\gg}r$ 并转化为范式;

$$p\wedge fin(l_1);_{\ll a\gg}\Box_{\ll a\gg}q;_{\ll b\gg}\Box_{\ll b\gg}r\equiv p\wedge q\wedge r\wedge l_1\wedge\varepsilon\vee p\wedge q\wedge r\wedge l_1\wedge\bigcirc_{\ll b\gg}\Box_{\ll b\gg}r\vee p\wedge l_1\wedge q\wedge \bigcirc_{\ll a\gg}(\Box_{\ll a\gg}q;_{\ll b\gg}\Box_{\ll b\gg}r)\vee p\wedge\bigcirc_{\ll\varnothing\gg}(fin(l_1);_{\ll a\gg}\Box_{\ll a\gg}q;_{\ll b\gg}\Box_{\ll b\gg}r);$$

接着,将新生成的 APTL 公式 $\Box_{\ll b\gg}r,\Box_{\ll a\gg}q;_{\ll b\gg}\Box_{\ll b\gg}r$ 和 $fin(l_1);_{\ll a\gg}\Box_{\ll a\gg}q;_{\ll b\gg}\Box_{\ll b\gg}r$ 转化为范式;

$$\Box_{\ll b\gg}r\equiv r\wedge\varepsilon\vee r\wedge\bigcirc_{\ll b\gg}\Box_{\ll b\gg}r$$

$$\Box_{\ll a\gg}q\wedge fin(l_2);_{\ll b\gg}\Box_{\ll b\gg}r\equiv q\wedge r\wedge l_2\wedge\varepsilon\vee q\wedge r\wedge l_2\wedge\bigcirc_{\ll b\gg}\Box_{\ll b\gg}r\vee q\wedge \bigcirc_{\ll a\gg}(\Box_{\ll a\gg}q;_{\ll b\gg}\Box_{\ll b\gg}r)\wedge q\wedge\bigcirc_{\ll\varnothing\gg}(fin(l_2);_{\ll b\gg}\Box_{\ll b\gg}r)$$

$$fin(l_1);_{\ll a\gg}\Box_{\ll a\gg}q;_{\ll b\gg}\Box_{\ll b\gg}r\equiv q\wedge r\wedge l_1\wedge\varepsilon\vee q\wedge r\wedge l_1\wedge\bigcirc_{\ll b\gg}\Box_{\ll b\gg}r\vee q\wedge l_1\wedge \bigcirc_{\ll a\gg}(\Box_{\ll a\gg}q;_{\ll b\gg}\Box_{\ll b\gg}r)\vee\bigcirc_{\ll\varnothing\gg}(fin(l_1);_{\ll a\gg}\Box_{\ll a\gg}q;_{\ll b\gg}\Box_{\ll b\gg}r);$$

然后,将生成的新的 APTL 公式 $fin(l_2);_{\ll b\gg}\Box_{\ll b\gg}r$ 转化为范式;

$$fin(l_2);_{\ll b\gg}\Box_{\ll b\gg}r\equiv l_2\wedge r\wedge\varepsilon\vee l_2\wedge r\wedge\bigcirc_{\ll b\gg}\Box_{\ll b\gg}r\vee\bigcirc_{\ll\varnothing\gg}(fin(l_2);_{\ll b\gg}\Box_{\ll b\gg}r);$$

最后,根据公式的 NF 构造 LNFG.

公式 $p;_{\ll a\gg}\Box_{\ll a\gg}q;_{\ll b\gg}\Box_{\ll a\gg}r$ 的 LNFG 如图 5 所示.

节点 1 表示公式 $p\wedge fin(l_1);_{\ll a\gg}\Box_{\ll a\gg}q;_{\ll b\gg}\Box_{\ll a\gg}r$;

节点 2 表示 $\Box_{\ll b\gg}r$;

节点 3 表示 $\Box_{\ll a\gg}q\wedge fin(l_2);_{\ll b\gg}\Box_{\ll b\gg}r$;

节点 4 表示 $fin(l_1);_{\ll a\gg}\Box_{\ll a\gg}q;_{\ll b\gg}\Box_{\ll b\gg}r$;

节点 5 表示 $fin(l_2);_{\ll b\gg}\Box_{\ll b\gg}r$;

节点 6 表示 *empty*.

将图 5 中的 LNFG 转化为 GBCG,如图 6 所示.其中,节点 1 为初始节点,节点 1、节点 4 标记有 l_1,节点 3、节点 5 标记有 l_2,接收集合为 $F=\{F_1=\{2,3,5,6\},F_2=\{1,2,4,6\}\}$.

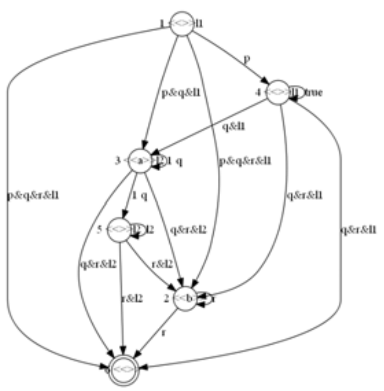


Fig.5 LNFG of formula $p;_{\ll a\gg}\Box_{\ll a\gg}q;_{\ll b\gg}\Box_{\ll b\gg}r$

图 5 公式 $p;_{\ll a\gg}\Box_{\ll a\gg}q;_{\ll b\gg}\Box_{\ll b\gg}r$ 的 LNFG

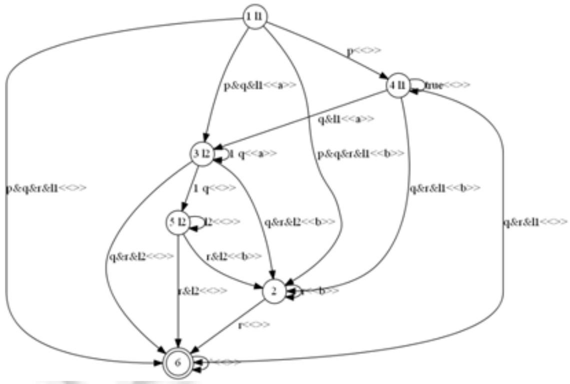


Fig.6 GBCG of formula $p;_{\ll a\gg}\Box_{\ll a\gg}q;_{\ll b\gg}\Box_{\ll b\gg}r$

图 6 公式 $p;_{\ll a\gg}\Box_{\ll a\gg}q;_{\ll b\gg}\Box_{\ll b\gg}r$ 的 GBCG

将图 6 的 GBCG 转化为 BCG 如图 7 所示,其中,状态(1,1)为初始状态,接收集合为 $F'=\{(2,1),(3,1),(5,1),(6,1)\}$.

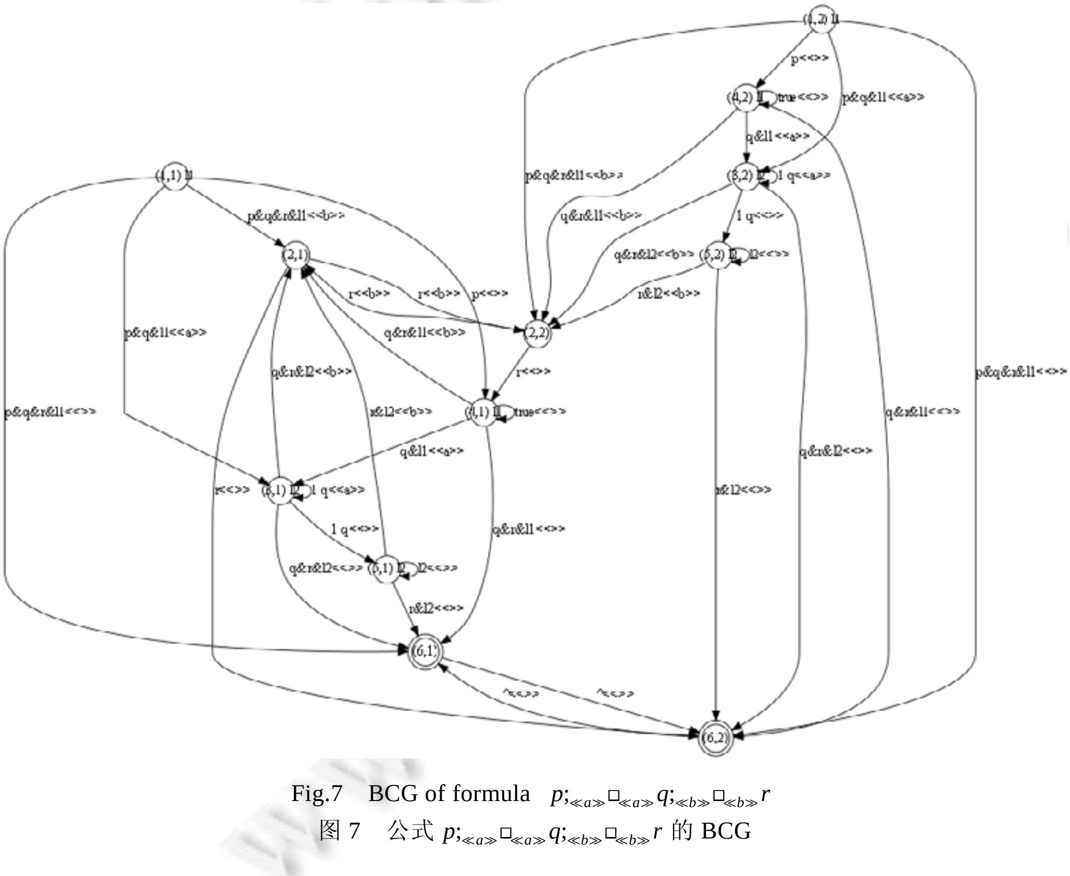


Fig.7 BCG of formula $p;_{\ll a\gg}\Box_{\ll a\gg}q;_{\ll b\gg}\Box_{\ll b\gg}r$

图 7 公式 $p;_{\ll a\gg}\Box_{\ll a\gg}q;_{\ll b\gg}\Box_{\ll b\gg}r$ 的 BCG

2.5 BCG的化简及检查

为了检查 APTL 公式的可满足性,必须将得到的 BCG 进行化简,因此,我们给出了化简 BCG 的算法 *SIMPLIFY*.对于一个 BCG,可能存在没有后继节点的节点或者存在没有出边的非接收状态节点,这些多余的节点和边应该被删除.由于交替自动机中存在一些全局的迁移关系,也就是说存在从同一个节点出发的多条边应同时满足相应的性质,所以做完以上删除操作后,可能会产生非初始节点入度为 0 的节点,这些节点也应该被删

除,具体过程如下.

SIMPLIFY(B).

输入:APTL 公式φ的 BCG $B=(\mathcal{P},\mathcal{A},Q,q_0,\delta,F)$;

输出:*SIMPLIFY*(B)计算出一个φ的最简 BCG $B'=(\mathcal{P}',\mathcal{A}',Q',q_0',\delta',F')$.

$B'=B$;

while $\exists q\in Q'$且节点 q 没有出边或者 q 为非接收节点且没有出边从 q 节点到其他节点

do ($Q'=Q'\backslash q$;

$\delta'=\delta'\backslash\bigcup_i(q_i,p_i,A_i,q)$; /* $\bigcup_i(q_i,p_i,A_i,q)$ 表示节点 q_i 到节点 q 的连边集合*/

for each i

$\delta'=\delta'\backslash\bigcup_j(q_i,p_i,A_j,r_j)$;

/* $\bigcup_j(q_i,p_i,A_j,r_j)$ 表示边的集合,在 APTL 公式的范式中,存在 $p_i\wedge\bigcirc_{\ll A_i\gg}R_{q_i}\wedge\bigcirc_{\ll A_j\gg}R_{r_j}$,其中,节点 q_i 和 r_j 分别由公式 R_{q_i} 和 R_{r_j} 标记*/

if $q\in F'$ **then**

$F'=F'\backslash q$;

)

end while

while $\exists q\in Q'$并且 q 不是初始节点,入度为 0

do ($Q'=Q'\backslash q$;

$\delta'=\delta'\backslash\bigcup_j\bigcup_i(q,p_i,A_i,r_j)$; /* $\bigcup_j\bigcup_i(q,p_i,A_i,r_j)$ 为连接节点 q 的边*/

)

end while

return B';

将图 7 中的 BCG 化简得到最简 BCG,如图 8 所示,其中接收集合为 F={(2,1),(3,1),(5,1),(6,1)}.

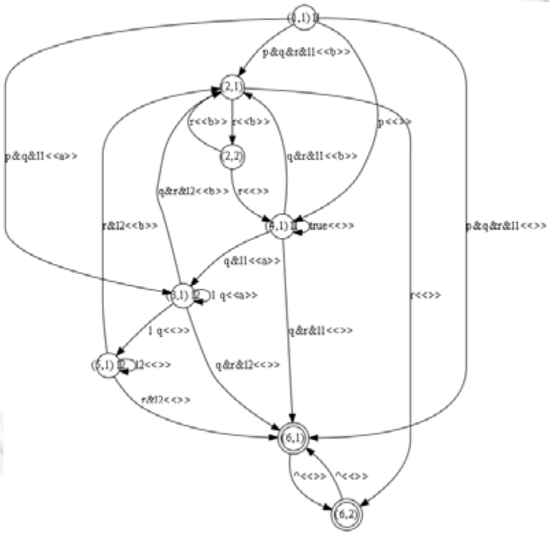


Fig.8 BCG of formula $p;_{\ll a\gg}\square_{\ll a\gg}q;_{\ll b\gg}\square_{\ll b\gg}r$

图 8 公式 $p;_{\ll a\gg}\square_{\ll a\gg}q;_{\ll b\gg}\square_{\ll b\gg}r$ 的 BCG

图 8 中的最简 BCG 不为空,所以公式 $p;_{\ll a\gg}\square_{\ll a\gg}q;_{\ll b\gg}\square_{\ll b\gg}r$ 是可满足的.

通过检查化简后的 BCG 是否为空,来判断对应的 APTL 公式的可满足性.检查任意一个 APTL 公式的可满足性过程如算法 *CHECK* 所示,其中,函数 *LNFG* 构造 APTL 公式 *P* 的 LNFG,函数 *TR* 将公式的 LNFG 转化为 GBCG,函数 *BCG* 将公式的 GBCG 转化为 BCG,函数 *SIMPLIFY* 对生成的 BCG 进行化简.

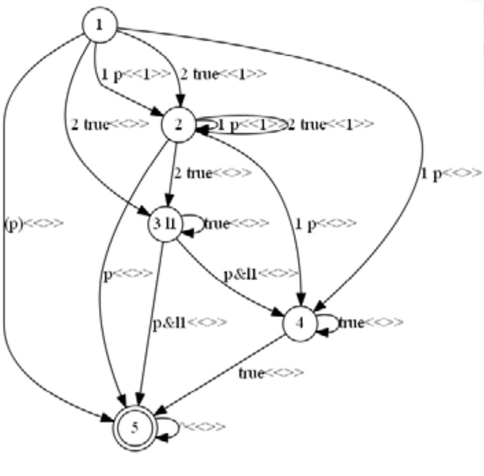
CHECK(*P*).
输入:APTL 公式 *P*;
输出:公式 *P* 是否可满足.
G=*LNFG*(*P*); /*构造公式 *P* 的 LNFG *G**/
GB=*TR*(*G*); /*将 LNFG *G* 转化为 GBCG *GB* */
BCG(*GB*,*B*); /*将 GBCG *GB* 转化为 BCG *B**/
G′=*SIMPLIFY*(*B*);
if *G*′为空
  return 不可满足;
else return 可满足;

3 工具展示

3.1 实例展示

本节通过检查哲学家就餐问题需要满足性质的可满足性,展示工具 APTL2BCG 的工作效果.哲学家围坐在一张圆桌旁,每人面前各摆放一碗面条,每两位哲学家之间只摆放一支筷子,哲学家的行为是反复地吃饭和思考.当哲学家饥饿时,他必须能够同时拿起左、右两支筷子才能吃到面条,餐后放下两支筷子继续考虑.*i* 是哲学家的编号(比如有 5 位哲学家,那么 1≤*i*≤5),每个哲学家存在以下 5 种状态:哲学家 *i* 思考表示为 *i_Thk*,哲学家 *i* 处于饥饿状态表示为 *i_Hug*,哲学家 *i* 拿筷子表示为 *i_Tch*,哲学家 *i* 吃饭表示为 *i_Eat*,哲学家 *i* 放下筷子表示为 *i_Pch*.哲学家的行为构成循环,依次是思考、感觉到饥饿后准备拿筷子、当两边的筷子都闲置的时候拿起筷子、接着吃饭、吃完后放下筷子,继续思考,依次循环下去.

哲学家 *i* 总能吃到面条,不会出现一直饥饿的情况用 APTL 公式表示为 $\Box_{\ll i\gg}\Diamond_{\ll i\gg} i\_Eat$.以哲学家 1 为例,用工具 APTL2BCG 检查公式 $\Box_{\ll 1\gg}\Diamond_{\ll 1\gg} 1\_Eat$ 的可满足性,为了使输出的 BCG 简单明了,令 *p*=*i_Eat*,将公式 []<<1>><><<1>>*p* 输入,得出的最简 BCG 如图 9 所示,BCG 不为空,表示公式 $\Box_{\ll 1\gg}\Diamond_{\ll 1\gg} 1\_Eat$ 是可满足的.



$1: \Box_{\ll 1\gg}\Diamond_{\ll 1\gg} p$
$2: \Box_{\ll 1\gg}(\mathrm{true};_{\ll 1\gg} p)$
$3: \mathrm{true};_{\ll 1\gg} p$
$4: \mathrm{true}$
$5: empty$
$F = \{1,2,4,5\}$

Fig.9 BCG of formula $\Box_{\ll 1\gg}\Diamond_{\ll 1\gg} p$
图 9 公式 $\Box_{\ll 1\gg}\Diamond_{\ll 1\gg} p$ 的 BCG

哲学家在吃面条后需要放下筷子接着再思考,以免相邻的哲学家拿不到筷子.以哲学家 1 为例,哲学家 1 吃完面条就放下筷子用 APTL 公式表示为: $\square_{\ll\emptyset\gg}(fin(1\_Eat);_{\ll 1\gg}\bigcirc_{\ll 1\gg}1\_Pch)$.令 p 代表 1_*Eat*,q 代表 1_*Pch*,检查公式 $\square_{\ll\emptyset\gg}(fin(p);_{\ll 1\gg}\bigcirc_{\ll 1\gg}q)$ 的可满足性.将公式[]<<>>(*fin*(*p*);<<1>>()<<1>>*q*)输入到工具 APTL2BCG 中,得到如图 10 所示的 BCG,表明公式 $\square_{\ll\emptyset\gg}(fin(1\_Eat);_{\ll 1\gg}\bigcirc_{\ll 1\gg}1\_Pch)$ 是可满足的.

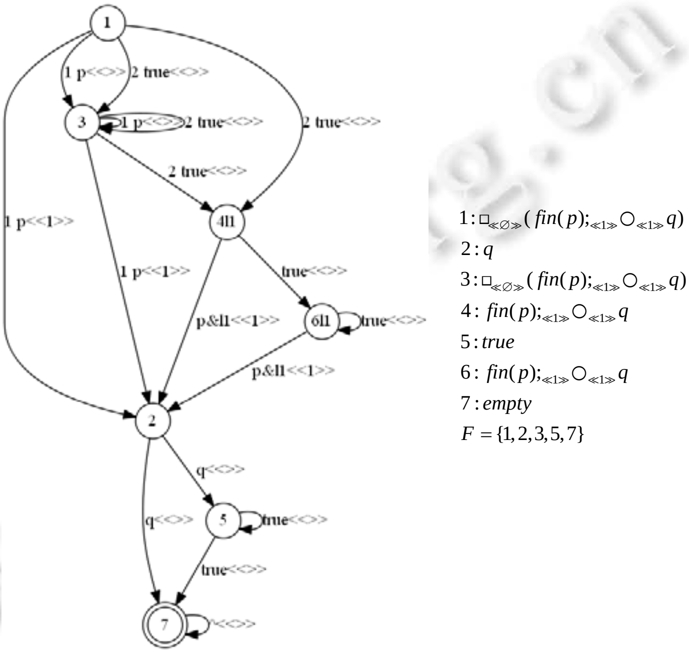


Fig.10 BCG of formula $\square_{\ll\emptyset\gg}(fin(p);_{\ll 1\gg}\bigcirc_{\ll 1\gg}q)$

图 10 公式 $\square_{\ll\emptyset\gg}(fin(p);_{\ll 1\gg}\bigcirc_{\ll 1\gg}q)$ 的 BCG

3.2 实验与分析

本节通过检查 3 类 APTL 公式的可满足性,展示工具 APTL2BCG 的实用性和工作效率.对以下 3 类公式进行可满足性检查:

$$len(n)\wedge\Diamond_{\ll A\gg}p \tag{1}$$

$$\Diamond_{\ll A1\gg}P_1;_{\ll A\gg}\Diamond_{\ll A2\gg}P_2;_{\ll A\gg}\dots;_{\ll A\gg}\Diamond_{\ll An\gg}P_n \tag{2}$$

$$(P_1,\dots,P_n)\,prj_{\ll A\gg}Q \tag{3}$$

我们用工具 APTL2BCG 检查以上 3 类公式的可满足性,其中,

- 路径λ满足公式(1)当且仅当智能体集合 A 的策略使得$\lambda[i]\vDash p(0\leqslant i\leqslant n)$成立;
- 路径λ满足公式(2)当且仅当智能体 A 的策略使得公式 $\Diamond_{\ll A1\gg}P_1,\Diamond_{\ll A2\gg}P_2,\dots,\Diamond_{\ll An\gg}P_n$ 在路径上顺序成立,其中, $\Diamond_{\ll A1\gg}P_1$ 表示智能体 A_1 的策略使得 P_1 在区间内可满足;
- 路径λ满足公式(3)当且仅当智能体集合 A 的策略使得性质 $P_1,P_2,\dots,P_n$ 依次在路径上成立,并且使得性质 Q 成立,公式 $P_1;_{\ll A\gg}P_2;_{\ll A\gg}\dots;_{\ll A\gg}P_n$ 与公式 Q 是相对独立的,他们的模型是并行的只是在特定的汇合点通信.

实验环境为 2.93GHz,Intel(R) Core(TM) i7 CPU 870,8G 内存 PC.实验结果见表 1.在表 1 中,

- 第 1 列表示公式中 n 的大小;
- 节点数和边数所在列分别表示生成的相应公式的 BCG 的节点数和边数,单位为个;
- 内存所在列表示程序运行时占用的内存,单位为 kb;

- 时间所在列表示判断公式的可满足性所需时间,单位为 s.

Table 1 Experimental results

表 1 实验结果

n	F.(1)				F.(2)				F.(3)			
	节点数	边数	内存	时间	节点数	边数	内存	时间	节点数	边数	内存	时间
0	2	2	1 792	0.04	–	–	–	–	4	5	1 824	0.09
1	4	4	1 824	0.09	6	9	1 828	0.1	4	5	1 832	0.11
2	6	7	1 844	0.23	10	20	1 988	0.5	8	13	1 920	0.39
3	8	10	1 872	0.34	14	35	2 408	1.26	13	27	2 208	1.33
4	10	13	1 920	0.50	18	54	3 328	3.141	18	45	2 848	2.92
5	12	16	1 988	0.69	22	77	5 028	5.92	23	67	3 976	5.722
6	14	19	2 044	0.87	26	104	7 704	11.811	29	93	5 808	11.378
7	16	22	2 136	1.22	30	135	12 136	28.657	34	123	8 672	15.009
8	18	25	2 256	1.54	34	170	18 292	110.32	38	157	12 720	35.333
9	20	28	2 416	2.02	38	209	27 152	689.233	43	195	18 504	169.63
10	22	31	2 584	2.415	42	252	39 340	5 055.91	48	237	26 512	1 054.2

表 1 的实验结果显示:n 的取值在一定范围时,此 3 类公式随着 n 的增大,生成的 BCG 的节点数、边数和所用内存空间都是线性增长.对于公式(2)和公式(3),随着公式中 n 的增大增长迅速.

但是实验结果看,工具 APTL2BCG 的实际应用效果比较好,有较好的实用性.

4 结 论

APTL 作为一种语法简洁、表达能力强的逻辑,其可用于规约多智能体系统的性质.APTL 公式的可满足性检查是系统验证的前提,本文开发了 APTL 公式的可满足性检查工具 APTL2BCG.检查过程是:首先,利用 APTL 公式的范式、LNFG 以及 GBCG 构造公式的 BCG;然后,化简 BCG 并检查其是否为空,得出公式是否可满足.

此外,本文通过实验展示了 APTL2BCG 在实践中的可用性和实用性.未来将进一步优化检查工具 APTL2BCG,并在此基础上开发基于 BCG 的 APTL 模型检测工具.

References:

[1] Tian C, Duan ZH. Alternating interval based temporal logics. In: Proc. of the ICFEM. 2010. 694–709. [doi: 10.1007/978-3-642-16901-4_45]

[2] Duan ZH. Temporal Logic and Temporal Logic Programming. Beijing: Science Press, 2005.

[3] Wang HY, Duan ZH, Tian C. Symbolic model checking for alternating projection temporal logic. In: Proc. of the COCOA. 2015. 481–495. [doi: 10.1007/978-3-319-26626-8_35]

[4] 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]

[5] Baier C, Katoen JP. Principles of Model Checking. Cambridge: MIT Press, 2008.

[6] Duan ZH, Tian C. A practical decision procedure for propositional projection temporal logic with infinite models. Theoretical Computer Science, 2014,554:169–190. [doi: 10.1016/j.tcs.2014.02.011]

[7] Alur R, Henzinger TA, Kupferman O. Alternating-Time temporal logic. JACM, 2002,49:672–713. [doi: 10.1145/585265.585270]

[8] Schewe S, Finkbeiner B. Satisfiability and finite model property for the alternating-time Mu-calculus. In: Proc. of the CSL. 2006. 591–605. [doi: 10.1007/11874683_39]

[9] http://www.cmi.ac.in/~kumar/words/lecture06.pdf

[10] Manna Z, Sipma H. Alternating the Temporal Picture for Safety. In: Proc. of the ICALP. 2000. 429–450. [doi: 10.1007/3-540-45022-X_37]

[11] Gastin P, Oddoux D. Fast LTL to Buchi Automata Translation. In: Proc. of the CAV. 2001. 53–65. [doi: 10.1007/3-540-44585-4_6]

王海洋(1989－),女,山东聊城人,博士,CCF学生会员,主要研究领域为时序逻辑,模型检测.

段振华(1948－),男,博士,教授,博士生导师,主要研究领域为网络计算,高可信软件理论和技术.

田聪(1981－),女,博士,教授,博士生导师,主要研究领域为形式化方法,时序逻辑,模型检测.

www.jos.org.cn