软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2019,30(2):231−243 [doi: 10.13328/j.cnki.jos.005584]

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

用于验证多智能体系统的 APTL 模型检测器*

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](综合业务网理论及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: zhhduan@mail.xidian.edu.cn; 田聪, E-mail: ctian@mail.xidian.edu.cn

摘 要: 由于经典的线性时序逻辑表达能力有限,设计并开发了基于交替投影时序逻辑(alternating projection temporal logic,简称 APTL)的模型检测工具.根据王海洋等人提出的 APTL 符号模型检测方法,设计并实现了 APTL 模型检测器 MCMAS_APTL.该工具可用于多智能体系统(multi-agent system,简称 MAS)的性质验证.MCMAS_APTL 检查 MAS 是否满足具体性质的过程如下:首先,用解释系统编程语言(interpreted system programming language,简称 ISPL)描述要验证的系统 IS,用 APTL 公式 *P* 描述要验证的性质;然后,符号化表示系统 IS,并将非 *P* 转化为范式;最后,计算所有满足非 *P* 的路径的起始状态集合.如果得到的状态集合中包含系统的初始状态,则说明系统不满足公式 *P*;反之,则说明系统满足公式 *P*.详细阐述了实现 MCMAS_APTL 的过程,并且通过验证机器人足球赛的例子展示了 MCMAS_APTL 的性能.

关键词: 交替投影时序逻辑;多智能体系统;模型检测

中图法分类号: TP311

中文引用格式: 王海洋,段振华,田聪.用于验证多智能体系统的 APTL 模型检测器.软件学报,2019,30(2):231−243. http://www.jos.org.cn/1000-9825/5584.htm

英文引用格式: Wang HY, Duan ZH, Tian C. APTL model checker for verifying multi-agent systems. Ruan Jian Xue Bao/Journal of Software, 2019,30(2):231−243 (in Chinese). http://www.jos.org.cn/1000-9825/5584.htm

APTL Model Checker for Verifying Multi-agent Systems

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 Services Networks (Xidian University), Xi'an 710071, China)

Abstract: The model checking tool based on Alternating Projection Temporal Logic (APTL) is designed and implemented to improve the ability of expressing for linear temporal logic in this study. The supporting tool MCMAS_APTL is developed inspired by the method for symbolic model checking of APTL provided by Wang *et al*. The tool MCMAS_APTL could be used for verifying the properties of Multi-agent Systems (MASs) effectively. The detailed procedures of checking whether a MAS satisfies a property by MCMAS_APTL are given as follows. Firstly, describing the system IS with the Interpreted System Programming Language (ISPL) and specifying the property of the system by an APTL formula *P*. Then, symbolically representing the system IS and transforming the negation of *P* into normal form. Finally, calculating the set of states from which there is at least one existing path satisfying the negation of *P*. If the obtained state set contains an initial state, then the system does not satisfy the formula *P*; otherwise, the system satisfies the formula *P*. The details of implementing MCMAS_APTL are provided in this paper, and a robotic soccer game is presented to show how the model checker MCMAS_APTL works in practice.

Key words: alternating projection temporal logic; multi-agent system; model checking

* 基金项目: 国家自然科学基金(61732013, 61420106004)

Foundation item: National Natural Science Foundation of China (61732013, 61420106004)

收稿时间: 2017-11-17; 修改时间: 2018-01-14, 2018-03-24; 采用时间: 2018-04-10; jos 在线出版时间: 2018-04-27

CNKI 网络优先出版: 2018-04-27 14:58:37, http://kns.cnki.net/kcms/detail/11.2560.TP.20180427.1458.010.html

随着计算机软硬件系统的飞速发展,人工智能(artificial intelligence,简称 AI)[1]理论与技术日益成熟,并且在计算机领域内得到了前所未有的重视,其应用领域在逐渐扩大及深入.其中,智能机器人越来越多地出现在人们的日常生活中,例如体育产业中的足球机器人、日常生活中的家用伺服机器人以及工业自动焊接机器人等.所以,保障 AI 产品的安全性和可靠性,是如今人们面临的刻不容缓的问题.模型检测(model checking)[2]是一种简单明了并且实现了自动化的、用于验证软硬件系统是否满足人们期望性质的技术.它已经广泛应用到各类软硬件系统的性能检测中,如运输控制系统、电子商务、航空航天领域.近年来,模型检测方法也已经应用到 AI 产品的性能检测中.模型检测是一种基于逻辑的、应用广泛的、对有限状态系统进行自动化程序验证的技术,最早由著名学者 Clarke 与 Emerson 以及 Quielle 与 Sifakis 于 20 世纪 80 年代分别提出.在模型检测中,系统 S 编码为迁移系统 M_S,要验证的性质规约 P 用逻辑公式 ϕ_P 表示,验证系统 S 是否满足规约 P,转化为模型 M_S 是否满足逻辑公式 ϕ_P.其中,M_S 满足公式 ϕ_P 简写为 $M_S|=\phi_P$.

验证 AI 产品的性能需要将其形式化规约,而 MAS[2]是 AI 中应用广泛的分布式系统,它可以将大而复杂的系统描述成多个彼此可以互相协调通信的小而易管理的系统,从而完成各自的或者共同的目标.MAS 在表达实际系统时,通过各智能体之间的通信、合作、互解、协调、调度、管理及控制来表达系统的结构、功能及行为特性.采用多智能体系统解决实际应用问题.该系统具有很强的鲁棒性和可靠性,并具有较高的问题求解效率.MAS 已经广泛应用于各个领域,如智能机器人、交通控制、分布式预测、监控及诊断、分布式智能决策及虚拟现实.MAS 引起了众多技术领域的广泛关注,因此,设计、实现及验证该系统的工具显得尤为迫切.特别是当 MAS 应用于安全验证相关领域时,验证其是否满足设计要求就显得尤其重要了.针对多智能体系统的模型检测方法的研究,已有学者取得了突出成果[4,5].

本文根据文献[6]中提出的 APTL 模型检测方法实现了多智能体系统模型检测器 MCMAS_APTL,其用 APTL 公式表示 MAS 的性质用于检测 MAS 的性能.APTL 公式不仅可以描述经典时序逻辑 LTL 公式可以描述的性质,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的与合作和博弈相关的性质,因此,用 APTL 公式可方便描述多智能体系统的性质.

状态爆炸问题是模型检测过程面临的常见问题之一,即随着变量个数的增多,系统的状态空间呈指数级增长.在模型检测过程中,能够缓解空间爆炸问题的技术有二值判断图(BDD)[7]、抽象技术、限界模型检测、化简和假设-保证推理.MCMAS_APTL 是基于 APTL 的多智能体系统的符号化模型检测器.它利用 APTL 公式表达多智能体系统的性质,借助工具 MCMAS[8]中的符号化多智能体系统模块符号化表示要验证的系统,进而验证模型是否满足给定的 APTL 公式.该工具不仅可以验证多智能体系统,还可用于验证普通的反应系统.

MAS 的模型检测方法已引起众多学者的广泛关注并且进行了深入研究.文献[9]中提出了验证 MAS 性质的限界模型检测方法,通过扩展 CTL*得到包括认知模态的逻辑 CTL*K.CTL*K 可描述同步解释系统的性质,而且可描述时序和认知两个方面的性质.文献[10]提出了一个基于 AUML 状态机模型的 MAS 模型检测机制.文中利用了比较高效的模型检测工具 MCMAS.它不仅可用于检测 MAS 的认知性质,而且可以检测 MAS 的智能体的行为是否正确或智能体之间的合作关系,但该文献不能检测 MAS 中智能体之间的博弈性质.文献[11]提出了一种自动生成控制器的机制,当输入一种机器人模型和任务环境以及一种任务或者机器人的行为后,该控制器就能保证机器人完成任务.该文献中的方法是首先创造一种符合规约 LTL 公式的控制器,其中,任务规约用 LTL 公式描述,如机器人的搜索、救援、覆盖面以及障碍躲避等.文献[12]提出了一种在语法层对智能体策略类型进行刻画的系统模型.它允许不同智能体具备不同的策略类型,研究了基于新模型的 ATL 模型检测方法并且开发了检测工具.文献[13]研究了下推多智能体系统的模型检测方法,引入了下推认知博弈结构作为模型,涉及到的时序逻辑为 ATEL、ATEL*和 AEMC.本文中的 MCMAS_APTL 工具利用了比较高效的模型检测工具 MCMAS 中的符号化表示系统部分,性质规约用 APTL 公式描述,其中,APTL 公式简单易懂,表达能力强,APTL 公式可以方便地描述多智能体系统的性质,所以该工具可方便地用于多智能体系统的性质验证.

本文第 1 节简单介绍交替投影时序逻辑语法、语义及其他基本概念.第 2 节阐述多智能体系统模型检测器 MCMAS_APTL 的框架和实现过程.第 3 节给出机器人足球赛模型检测实例.第 4 节对本文进行总结.

1 基础概念

1.1 交替投影时序逻辑

语法. APTL 的语法定义如下:

$$P::=p|\neg P|P\vee Q|\bigcirc_{\langle\langle A\rangle\rangle}P|(P_1,\ldots,P_m)prj_{\langle\langle A\rangle\rangle}Q.$$

令$\mathcal{P}$为原子命题的可数集合,$\mathcal{A}$为代理的可数集合.其中,$p\in\mathcal{P}$,$A\subseteq\mathcal{A}$,$P_i(i=1,\ldots,P_m)$,P 和 Q 为 APTL 公式,$\bigcirc_{\langle\langle A\rangle\rangle}$(next)和 $prj_{\langle\langle A\rangle\rangle}$(projection)为基本的 APTL 时序操作符.

语义. 为了定义 APTL 公式的语义,首先需要介绍并发博弈结构(concurrent game structure,简称 CGS)[6,8]. CGS 是一个七元组 $C=\langle\mathcal{P},\mathcal{A},S,S_0,l,\Delta,\tau\rangle$,其中,$\mathcal{P}$是原子命题的有穷非空集合;$\mathcal{A}$是代理的有穷集合;$S$ 是状态的有穷非空集合;S_0 是初始状态的有穷非空集合;$l:S\to 2^{\mathcal{P}}$ 为标记函数,每个状态被原子命题集合的子集标记;$\Delta^a(s)$是代理 $a\in\mathcal{A}$在状态 s 的可以做出决策的非空集合;$\Delta^A(s)=\Delta^{a_1}(s)\times\ldots\times\Delta^{a_k}(s)$ 是代理集合 $A=\{a_1,\ldots,a_k\}\in 2^{\mathcal{A}}$ 在状态 s 的决策向量的非空集合;相应地,$\Delta^A(s)$ 简单表示为$\Delta(s)$,表示$\mathcal{A}$中的代理的决策集合;一个决策 d_a,表示代理 a 在决策 d 上的决策,d_A 表示代理集合 $A\subseteq\mathcal{A}$在决策 d 的决策.对于每一个状态 $s\in S$ 和决策 $d\in\Delta(s)$,$\tau(s,d)\in S$ 将状态 s 和代理集合$\mathcal{A}$的决策 d 映射到新的状态.f_A 为代理集合 A 在系统 C 中的一个策略,该策略使得系统生成多条以初始状态 s_0 为起始状态的路径.定义 $out(s,f_A)$为以状态 s 为起始状态策略 f_A 使得系统执行生成的所有的路径集合.详细讲解见文献[6].

根据 CGS 的定义,在$\mathcal{P}$上定义了一个状态 s,为$\mathcal{P}$到 $B=\{\text{true},\text{false}\}$的投影,即 $s:\mathcal{P}\to B$.在一个 CGS 中,以状态 s 为起始节点的一条路径$\lambda(s)$满足 APTL 公式 P,标记为$\lambda(s)|=P$.一个 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路径满足公式 P,标记为 $C|=P$.

关系定义如下:

- $\lambda(s)|=p$ 对于命题 $p\in\mathcal{P}$,当且仅当 $p\in l(s)$.
- $\lambda(s)|=\neg P$ 当且仅当$\lambda(s)|\neq P$.
- $\lambda(s)|=P\vee Q$ 当且仅当$\lambda(s)|=P$ 或$\lambda(s)|=Q$.
- $\lambda(s)|=\bigcirc_{\langle\langle A\rangle\rangle}P$ 当且仅当$|\lambda(s)|\geqslant 2$,并且 A 存在一个策略 f_A,使得$\lambda(s)\in out(s,f_A)$,并且$\lambda(s)[1,|\lambda|]|=P$.
- $\lambda(s)|=(P_1,\ldots,P_m)prj_{\langle\langle A\rangle\rangle}Q$ 当且仅当 A 存在一个策略 f_A,$\lambda(s)\in out(s,f_A)$,$0=r_0\leqslant r_1\leqslant\ldots\leqslant r_m\leqslant|\lambda(s)|$,使得$\lambda(s)[r_{i-1},r_i]|=P_i$,$0<i\leqslant m$ 并且$\lambda|=Q$,λ由以下两种情况得到:
  a) $r_m<|\lambda(s)|$,那么$\lambda=\lambda(s)\downarrow(r_0,\ldots,r_m)\cdot\lambda(s)[r_m+1,\ldots,|\lambda(s)|]$;
  b) $r_m=|\lambda(s)|$,那么$\lambda=\lambda(s)\downarrow(r_0,\ldots,r_m)$.

APTL 范式. 将 APTL 公式转化为范式(normal form)[6]形式,是 APTL 模型检测中不可或缺的环节.接下来简单介绍 APTL 公式的范式和完全范式(complete normal form)的定义.

定义 1(范式(normal form)). Q_P 为 APTL 公式 Q 中的原子命题集合,公式 Q 的范式定义为

$$Q\equiv Q_e\wedge\varepsilon\vee\vee_{i=0}^{n}(Q_{ci}\wedge Q_i),Q_i\equiv\wedge_{j=1}^{r}\bigcirc_{\langle\langle A_j\rangle\rangle}Q_{ij}.$$

Q_e 为状态公式,$Q_{ci}\equiv\wedge_{k=1}^{l}\dot{q}_k,q_k\in Q_p,\dot{q}_k$ 为 q_k 或$\neg q_k$,如果 $i\neq j$,则 $Q_{ci}\neq Q_{cj}$;Q_{ii} 为 APTL 公式.

APTL 公式的范式包含两部分:$Q_e\wedge\varepsilon$为终止部分,若只有 1 个状态 s_0 的路径满足公式 Q_e,则满足 $Q_e\wedge\varepsilon$;$Q_{ci}\wedge Q_i$ 为非终止部分,公式 $Q_{ci}\wedge Q_i$ 的模型为一个生成树,其中,该生成树的根节点满足 Q_{ci}.

定义 2(完全范式(complete normal form)). Q_P 为 APTL 公式 Q 中出现的原子命题组成的集合,Q 的完全范式定义为 $Q\equiv Q_e\wedge\varepsilon\vee\vee_{i=0}^{n}(m_i\wedge M_i)$, 其中, $M_i\equiv\wedge_{j=1}^{r}\bigcirc_{\langle\langle A_j\rangle\rangle}M_{ij}$. Q_e 为状态公式; $m_i\equiv\wedge_{k=1}^{l}\dot{q}_k,q_k\in Q_p,\dot{q}_k$ 为 q_k 或$\neg q_k$; $\vee_{i=0}^{n}m_i\equiv\text{true}$ 并且$\vee_{i\neq j}(m_i\wedge m_j)\equiv\text{false}$,$m_i$ 是原子命题的最小项,如果有 n 个原子命题,就有 2^n 个最小项 $m_0,m_1,\ldots,m_{2^n-1}$; M_{ij} 是原子命题的最大项,也有 2^n 个最大项.

文献[14]中的定理 4 已经证明,任意一个 APTL 公式都可以转化为范式.将公式 Q 转化为范式后,进一步可以转化为完全范式.如果 Q 的完全范式为 $Q\equiv Q_e\wedge\varepsilon\vee\vee_{i=0}^{n}(m_i\wedge M_i)$, 则$\neg Q$ 的范式为 $\neg Q\equiv\neg Q_e\wedge\varepsilon\vee\vee_{i=0}^{n}(m_i\wedge$

$\neg M_i$).

1.2 符号模型检测

模型检测是一种有效验证有限状态系统性质的技术.由于基于枚举的模型检测方法容易造成状态爆炸问题,所以在实际应用中会遇到很多问题.研究者们尝试解决该问题,提出了一些缓解模型检测过程中状态爆炸问题的有效方法,如抽象技术、限界模型检测以及基于 BDD 的符号模型检测.符号模型检测的核心思想是建立在将状态集合和迁移关系集合用布尔方程表示的基础上的,用布尔方程隐式地描述状态集合和迁移关系集合,比显式地利用枚举方式描述能够显著节约存储空间.符号化模型检测方法是以状态集合为操作对象,而不是单个状态.

符号化表示方法能够更加简洁地表示系统状态,本文利用二值判断图(binary decision diagrams,简称 BDD)表示状态集合.通过对 BDD 操作,容易实现求状态集合的前驱状态集合、后继状态集合以及状态集合的合并等.符号化模型检测算法与枚举模型检测算法的基本搜索过程相似,本质区别在于,一般的模型检测中以单个状态为操作对象;而符号模型检测中以状态集合为操作对象,且借助 BDD 等数据结构,已经有功能强大的操作工具包作为后续开发基础.

二值判断图(BDD). BDD 表示布尔函数的有根无环有向图,内部节点标记对应公式中的变量,叶子节点标记为 0 或 1.如图 1 所示为布尔函数 $f(a,b,c,d)=a\wedge b\vee c\wedge \overline{d}$ 的 BDD.

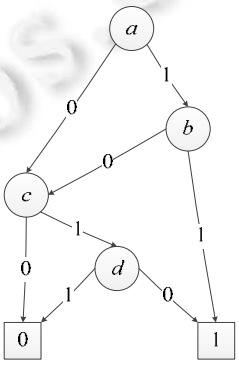

Fig.1 BDD of $f(a,b,c,d)=a\wedge b\vee c\wedge \overline{d}$

图 1 $f(a,b,c,d)=a\wedge b\vee c\wedge \overline{d}$ 的 BDD

一个布尔公式可由不同的 BDD 表示,为确保每个布尔公式对应唯一的 BDD,对 BDD 有两种限制:(1) 对布尔公式中的变量进行排序,如图 1 中布尔公式变量排序为 $a<b<c<d$;(2) BDD 中不存在冗余子树和多余的节点.因此,基于这两种限制,多余的叶子节点、同构子树以及重复的非终节点可被删除,得到最简 BDD,如图 1 所示即为该公式对应的最简排序 BDD.

1.3 MCMAS

MCMAS 是一种多智能体模型检测工具,MCMAS 的框架结构如图 2 所示.

MCMAS 具有成熟、高效的符号化技术,我们借助 MCMAS 中模型的符号化模块实现 APTL 模型检测工具 MCMAS_APTL.MCMAS 利用解释系统编程语言(interpreted system programming language,简称 ISPL)描述要验证的系统,支持的逻辑为 CTLKD-A^{DC}[8].将要验证的系统和一组公式输入 MCMAS,可计算出系统是否满足公式.如果公式不成立,则输出一条反例路径;否则,输出一条证据路径.

在 MCMAS 中要验证的程序形式化表示为解释系统(interpreted system)并用语言 ISPL 描述.ISPL 程序将一种多智能体系统描述为包含多个智能体和环境的系统,对于 ISPL 描述的智能体简单介绍如下.局部状态(local states)是智能体的私有内部状态,是由变量描述的,对于其他智能体是不可见的.智能体与其他智能体和环境的

互动是通过将局部状态转化为公共可视化的变量完成的.智能体采取的动作(action)由智能体的局部协议(local protocol)决定.局部状态随局部演变函数变化,该类函数根据当前的局部状态和所有智能体的全局动作(joint action)给出可能的下一个局部状态(next local state).ISPL 程序的结构是根据 IS 的语义给出的,并且可以广泛应用于描述多智能体系统.在程序中,环境用关键字 Environment 表示,Environment 中的一些局部变量对其他智能体可见.

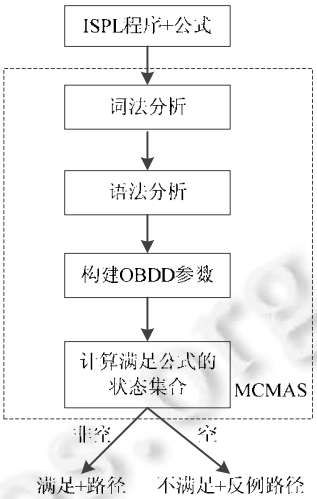

Fig.2 Architecture of model checking tool MCMAS

图 2 模型检测工具 MCMAS 的结构框架

2 原型工具

从底层开发实现模型检测工具耗时、耗力,不可控因素较多,本文以开源工具为基础实现我们提出的模型检测器.鉴于 MCMAS 工具所具有的特性和优点,最终选择以 MCMAS 为基础实现 APTL 符号模型检测算法.

2.1 工具框架

我们开发实现了用于验证多智能体系统的 APTL 符号模型检测器 MCMAS_APTL.该工具借助了 MCMAS 的符号化表示模型.MCMAS_APTL 的架构如图 3 所示.

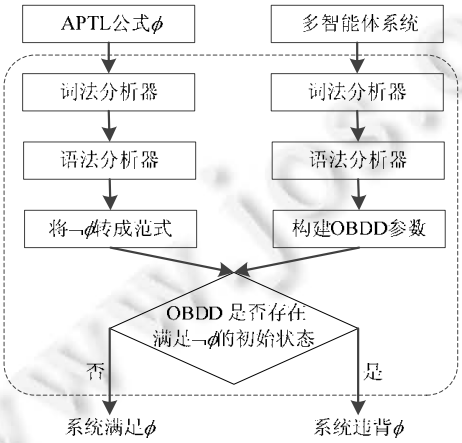

Fig.3 Architecture of model checking tool MCMAS_APTL

图 3 模型检测工具 MCMAS_APTL 的结构框架

MCMAS_APTL 采用 C++语言开发,由 3 个模块组成,分别为:用布尔函数符号化表示解释系统模块;将 APTL 公式转化为范式模块;检查解释系统是否满足 APTL 公式模块.第 1 个模块是将解释系统符号化表示,该模块是借助工具 MCMAS 中的对应的部分;第 2 个模块是将 APTL 公式转化为范式;第 3 个模块是通过计算满足公式的状态集合验证输入的解释系统是否满足给定的 APTL 公式.

2.2 模型检测方法

符号模型检测方法有效缓解状态爆炸问题的关键环节是用布尔方程符号化表示模型,进而用ROBDD高效地表示;后续整个检查过程操作都是在ROBDD上进行的.APTL 符号模型检测方法是将多智能体系统模型化为解释系统后进行检测的.解释系统是 Kripke 结构的一种扩展,与 CGS 相比更适合 MAS 的建模.在 CGS 中系统只有全局状态这一概念,IS 的全局状态是由内部所有智能体的局部状态组成.详细的比较见文献[6].

2.2.1 符号化表示解释系统

下面简单介绍符号化表示解释系统(interpreted system)的方法.给定一个解释系统 $IS=\langle(L_i,Act_i,P_i,t_i)_{i\in\Sigma},S_0,h\rangle$.

- 编码一个智能体 $i(i\in\mathcal{N})$的局部状态需要 $nv(i)=\lceil\log_2|L_i|\rceil$个布尔变量,编码系统的全局状态 g 为布尔向量 $\bar{v}=(v_1,...,v_N)$,其中, $N=\sum_i nv(i)$.
- 编码一个智能体的局部行为需要 $na(i)=\lceil\log_2|Act_i|\rceil$个布尔变量,编码系统的全局行为 a 为布尔向量 $\bar{w}=(w_1,...,w_M)$,其中, $M=\sum_i na(i)$.
- 一个智能体的协议可用局部状态和行为的布尔公式蕴含关系编码. $P(\bar{v},\bar{w})$ 是所有协议的布尔函数组合得到的整个系统协议布尔函数.
- 一个智能体的演变函数是由该智能体的局部变量和其他智能体行为以及环境的可视化变量组成的布尔函数表示, $t(\bar{v},\bar{w},\bar{v}')$ 为所有智能体演变函数的布尔函数组合得到的整个系统演变关系布尔函数.
- 系统的初始状态集合可用一个布尔函数 $S_{0\bar{v}}$ 表示.

解释系统 IS 的时序迁移关系可以用布尔方程 $R_t(g,g')$表示.该方程由所有代理的演变方程 t_i 得到,即

$$R_{t(\bar{v},\bar{v}')}=\vee_{\bar{w}\in Act}(t(\bar{v},\bar{w},\bar{v}')\wedge P(\bar{v},\bar{w})).$$

该公式描述全局状态间的布尔关系,应用于时序逻辑操作符的计算.可达全局状态集合 G 可以用一个布尔公式表示,并通过求解 $\tau(Q)=(S_{0\bar{v}}\vee\exists(\bar{v}')(R_{t(\bar{v}',\bar{v})}\wedge Q_{\bar{v}'}))$ 的不动点得到.其中,Q 为系统状态集合.τ的不动点可通过迭代计算$\tau(\varnothing)$得到[1].

2.2.2 APTL 符号模型检测器的实现

在APTL模型检测方法中,需要验证的系统描述为解释系统$IS=\langle(L_i,Act_i,P_i,t_i)_{i\in\Sigma},S_0,h\rangle$,要验证的性质用APTL公式表示.$IS,\lambda\models\phi$表示 IS 中的路径$\lambda$满足$\phi$,简写为$\lambda|=\phi$.$IS|=\phi$当且仅当以初始状态为起始状态的任意一条路径满足公式ϕ.

定义 3(满足性质的状态集合). 对于一个 APTL 公式ϕ和解释系统$IS=\langle(L_i,Act_i,P_i,t_i)_{i\in\Sigma},S_0,h\rangle$,其中一条执行路径$\lambda$是系统 IS 中的非空状态序列,$\lambda$为有穷或者无穷路径.集合 $Sat(\phi)\subseteq G$ 包含了所有的至少有 1 条以其为初始节点的路径满足公式ϕ的状态:

$$Sat(\phi)=\{g\in G|\exists\lambda\in Paths(G)|=\phi \text{ and } \lambda[0]=g\},$$

其中,G 为系统 IS 的全局状态集合,$Paths(G)$表示全局状态组成的所有的路径集合.

对于一个解释系统 $IS=\langle(L_i,Act_i,P_i,t_i)_{i\in\Sigma},S_0,h\rangle$,给定一个代理集合 $A\subseteq\Sigma$,全局状态集合 $G_1\subseteq G$ 和迁移关系 $t'\subseteq t$,函数 *PRE_img* 返回状态集合 G_1 中状态的前驱状态集合.Σ为系统的代理集合,G 为可达全局状态集合,t 为全局状态的演变函数.函数 *PRE_img* 定义如下:

$$PRE\_img(A,G_1,t')=\{g\in G_1|\exists g'\in G_1.t'(g,P_A)=g'\}.$$

解释系统 $IS=\langle(L_i,Act_i,P_i,t_i)_{i\in\Sigma},S_0,h\rangle$和 APTL 公式$\phi$中,$IS|=\phi$当且仅当 $Sat(\neg\phi)$和 S_0 的交集为空.依据以上的基础概念,我们实现了函数 *APTL_model_checking*(*bdd_parameters*para,char*ϕ)检查解释系统 $IS=\langle(L_i,Act_i,P_i,t_i)_{i\in\Sigma},S_0,h\rangle$是否满足 APTL 公式$\phi$,其中,*para* 为系统模型的符号化表示参数.检查函数首先调用函数 *cal_aptl_bdd*(*bdd_*

*parameters*para,char*φ*)计算 $Sat(\neg\phi)$的特征函数 $Sat_{\bar{v}}(\neg\phi)$,然后判断 $Sat_{\bar{v}}(\neg\phi)\cdot S_{0\bar{v}}$ 是否为 0.

- 若 $Sat_{\bar{v}}(\neg\phi)\cdot S_{0\bar{v}}=0$,则表明 $Sat(\neg\phi)$和 S_0 的交集为空,即不存在一条路径$\lambda\in Paths(g_0)(g_0\in S_0)$,使得$\lambda|=\neg\phi$.即 *IS* 中的所有执行路径满足公式$\phi$;
- 如果 $Sat_{\bar{v}}(\neg\phi)\cdot S_{0\bar{v}}\neq 0$,则说明在 *IS* 中至少存在 1 条以状态集合 $Sat(\neg\phi)\cap S_0$ 中的状态为起始状态的执行路径λ_{ce},使得$\lambda_{ce}|=\neg\phi$.

函数 *APTL_model_checking*(*bdd_parameters*para,char*φ*)的伪代码如下所示.

void *APTL_model_checking*(*bdd_parameters*para,char*φ*){

1. *BDD b*←*cal_aptl_bdd*(*para*,¬*φ*);
2. *BDD temp*←*b**(*(*para*→*in_st*));
3. if *temp*==0
   return *IS*|=*φ*;
4. else
   return *IS*|≠*φ*;

}

其中,函数 APTL_model_checking 中的第 1 行是计算满足公式¬*φ*的状态集合的 BDD,第 2 行是计算满足公式¬*φ*的状态集合与系统的初始状态集合的交集,*para*→*in_st* 为表示系统的初始状态集合的 BDD.

函数 *BDD cal_aptl_bdd*(*bdd_parameters*para, char*φ*)计算满足 APTL 公式*φ*的状态集合的 BDD,形参为系统模型的符号化表示参数和 APTL 公式*φ*,返回的是表示满足公式*φ*的状态集合的 BDD.其中,"+"和"·"分别代表逻辑"或"和"与", $Sat_{\bar{v}}(\phi)$ 为 $Sat(\phi)$的布尔方程.$NF(\phi)$是将公式*φ*转化为范式的过程, $PRE\_img(A,Sat_{\bar{v}}(\varphi),R_t)$ 是计算 $Sat_{\bar{v}}(\varphi)$ 的前驱状态函数.$FIXPOINT(c(Sat_{\bar{v}}(\phi_{ir_e}),R_t))$ 计算 $c(Sat_{\bar{v}}(\phi_{ir_e}),R_t)$ 的不动点.

BDD cal_aptl_bdd(*bdd_parameters*para,char*φ*){

1. *mark*[*φ*]=0, $Sat_{\bar{v}}(\phi)=0$;
2. $NF(\phi)=\vee_{i=1}^{n}\psi_i$;
3. *mark*[$\phi_{i'j}$]=0;
4. *vector*⟨*BDD**⟩**R*$_t$←(*para*→*vec_reachRT*);
5. for (int *i*=1; *i*≤*n*; *i*++){
6. if $\psi_i\equiv\phi_e\wedge\varepsilon$ then
   $Sat_{\bar{v}}(\psi_i)=Sat_{\bar{v}}(\phi_e)\cdot PRE(\varnothing,Sat_{\bar{v}}(\varepsilon),R_t)$;
7. else if $\psi_i\equiv\phi_i\wedge\wedge_{j=1}^{m}\bigcirc_{\langle\langle A_j\rangle\rangle}\phi_{ij}$,对于该公式中的公式$\phi_{ii}$,存在 1≤*k*≤*m*,*mark*[$\phi_{ik}$]==0,或者 1≤*r*≤*m*,*mark*[ϕ_{ir}]==1 then{
   对于所有的ϕ_{ik},令 *mark*[ϕ_{ik}]=1;
   $Sat_{\bar{v}}(\psi_i)=Sat_{\bar{v}}(\phi_i)\cdot\prod_k(PRE(A_{ik},cal\_aptl\_bdd(para,\phi_{ik}),R_t))$
   $\prod_r(PRE(A_{ir},FIXPOINT(c(Sat_{\bar{v}}(\phi_{ir_e}),R_t)),R_t))$;
   其中,$\phi_{ir}\equiv\phi_{ir_e}\wedge\wedge_{x=1}^{y}\bigcirc_{\langle\langle A_{ir_{ex}}\rangle\rangle}\phi_{ir_{ex}}$,$c(X,R_t)$计算 *X* 的值;
   }
   }
8. $Sat_{\bar{v}}(\phi)=Sat_{\bar{v}}(\psi_1)+Sat_{\bar{v}}(\psi_2)+...+Sat_{\bar{v}}(\psi_n)$;
9. return $Sat_{\bar{v}}(\phi)$;

}

函数 *BDD cal_aptl_bdd*(*bdd_parameters*para,char*φ*)中,第 2 行将公式*φ*转化为范式,其中,$\psi_i\equiv\phi_e\wedge\varepsilon$或者 $\psi_i\equiv\phi_{i'}\wedge\wedge_{j=1}^{m}\bigcirc_{\langle\langle A_{ij}\rangle\rangle}\phi_{i'j}$;第 4 行中,*para*→*vec_reachRT* 为系统模型的迁移关系.

函数 cal_aptl_bdd 首先将 APTL 公式ϕ转化为范式,后续分别处理各项ψ_i.

- 如果$\psi_i \equiv \phi_e \wedge \varepsilon$,根据范式的定义,$\phi_e$为状态公式,$Sat_{\bar{\nu}}(\psi_i)$表示满足$\phi_e$的状态集合,并且这些状态是有穷执行路径的最终状态.
- 如果$\psi_i \equiv \phi_i \wedge \wedge_{j=1}^{m} \bigcirc_{\langle\langle A_j \rangle\rangle} \phi_{ij}$,首先调用函数 cal_aptl_bdd 计算$Sat_{\bar{\nu}}(\phi_i)$.在计算过程中,如果生成了在$\phi$的转化中没有出现过的子公式$\phi_{ik}(1 \leqslant k \leqslant m)$,计算$PRE_img(A_{ik}, cal_aptl_bdd(\phi_{ik}, R_t), R_t)$;如果生成了在转化$\phi$的过程中出现过的子公式$\phi_{ir}(1 \leqslant r \leqslant m)$,说明公式对应的图中存在环,所以计算$Sat_{\bar{\nu}}(\phi_{ir})$变的很复杂.这个问题可以通过计算不动点得到$Sat_{\bar{\nu}}(\phi_{ir})$,然后计算前驱集合$PRE_img(A_{ir}, Sat_{\bar{\nu}}(\phi_{ir}), R_t)$.通过将上面求到的布尔方程求"与",得到$Sat_{\bar{\nu}}(\psi_i)$.

最后,布尔方程$Sat_{\bar{\nu}}(\phi)$可以将所有的$Sat_{\bar{\nu}}(\psi_i)$求"或"得到,即$Sat_{\bar{\nu}}(\phi) = Sat_{\bar{\nu}}(\psi_1) + Sat_{\bar{\nu}}(\psi_2) + ... + Sat_{\bar{\nu}}(\psi_n)$.

3 机器人足球赛模型检测实例

机器人足球赛[15]是多智能体系统的典型应用,其中涉及到了机器人之间的合作与竞争.在足球比赛过程中,同一个组的机器人合作,尽可能地将球踢入对方球门,而对方机器人会尽力阻止足球进入自己球队球门.这个过程要求机器人能够根据不同的情景选取不同的策略,其中的合作和博弈性质可以方便地用 APTL 公式描述.所以,本文通过机器人足球赛的简单示例展示该工具的工作效果.

3.1 机器人足球赛的策略模型

本节介绍机器人足球赛的策略模型.

3.1.1 足球场地模型

机器人足球赛场地为长方形,一般是长 9 000mm、宽 6 000mm.整个球场分为 3 个区域:前场、中场、后场,这 3 个区域是用直线分隔开的.足球场地模型如图 4 所示,场地被量化为长 30 个单位、宽 20 个单位,每个单位代表 300mm.(x,y)坐标表示球场中的位置,中央点的坐标为(15,10),中圈的半径为 3 个单位长度.球门宽度为 4 个单位长度,两个球门线的坐标分别为(0,8),(0,12)和(30,8),(30,12).禁区宽为 4 个单位长度,长为 8 个单位长度.

Fig.4 Soccer field model

图 4 足球场模型

3.1.2 策略模型

由于机器人足球赛的参数随着比赛的进行而变化,所以机器人足球赛的路径规划问题比较复杂.在比赛中,足球机器人不仅要考虑本队的策略和规划,而且要考虑对方球队机器人可能的策略.所以在一个特定的场景下,可能有多个不同的最优选择.机器人足球赛的规则和人类足球赛规则类似.一般地,一个机器人可以采取的行动如下.

- 开球(kick off):在比赛开始时,一个机器人将球踢出.
- 防护(go to defend):机器人去防护对方球员.

- 靠近球(go to the ball):机器人靠近球.
- 截球(intercept the ball):当对方球员持球时,试图截取球.
- 传球(pass the ball to its teammate):当持球的机器人被对方球员拦截时,该机器人将球传给本队队员.
- 射门(shoot the ball into the goal):机器人球员试图将球踢入对方球门.
- 带球前行(go forward with the ball):机器人持球向距离对方球门近的地方前行.

在比赛过程的任意时刻,每一个机器人都能获取球的位置、两个球门的位置、自身的位置和其他机器人的位置.根据不同的情形,团队中的球员担当不同的角色,其中包括前锋(striker)、中场(midfielder)和后卫(defender).守门员与其他球员不同.在整场比赛过程中,守门员的角色不变,其程序也比较简单,仅是一直在寻找、发现和观察球的情况.当球靠近球门时,守门员试图将球踢出以防对方球队进球,并且试图将球踢到距离本队球门较远的位置.对于一场机器人足球赛,持球的球队采取攻击策略(attack tactic),对方球队采取防卫策略(defensive tactic).下面详细介绍包含两个球队 *A* 和 *B* 的足球赛策略,每队各有 4 名球员.

防卫策略(defensive tactic).当足球在 *A* 队的中场或者后场时,*A* 队采取防卫策略.如果 *A* 队没有队员持球,那么 *A* 队中距离球最近的队员试图截球,该球员作为后卫.距离对方球门最近的队员跑去中场并作为前卫,当本队球员截取到球后等待着传球.另外一个球员作为中场队员,并且试图去拦截对方球员的传球.该场景如图 5(a)所示,红色队员属于 *A* 队,黄色队员属于 *B* 队.其中 GK 为守门员、S 为前锋、M 为中场、D 为后卫.如果 *A* 队队员持球,持球的机器人作为中场队员且必须将球传给前锋.剩余的一个队员作为后卫并且停留在罚球点(penalty spot).当前锋得到球后采取新的策略,该队员将会有新的角色或者行为.该情形如图 5(b)所示.

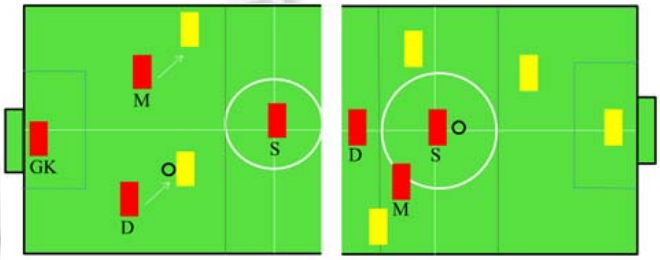

Fig.5 Role assignment in a defensive tactic

图 5 防卫策略中的角色分配

进攻策略(offensive tactic). 当球在前场时,*A* 队采取进攻策略.如果 *A* 队队员持球,持球队员作为前锋,距离本队球门最近的队员作为后卫且站在罚球点以防对方球员进球.另一个队员作为中场队员与前锋保持在一个水平线上并且在离对方球门近的地方,如果对方球员没有在球门和前锋之间拦截球,那么前锋将球踢向对方球门.如果对方球员试图拦截球,那么前锋将球传给中场队员,并选择新的策略.以上两种情形如图 6 所示.如果 *A* 队没有队员持球,*A* 队中的前锋试图截球.后卫必须在后场以防对方球员进球.中场球员阻挡对方队员靠近自己区域以防对方队员之间传球.

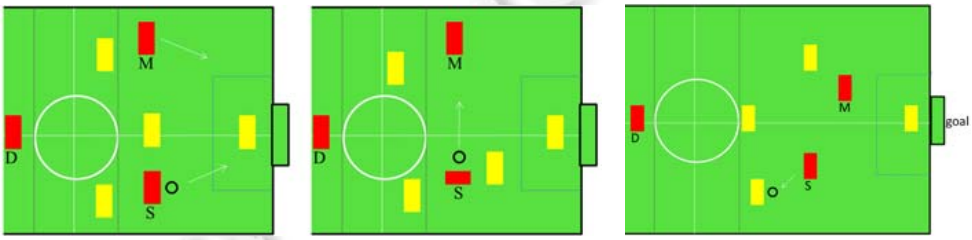

Fig.6 Role assignment in a offensive tactic

图 6 进攻策略中的角色分配

3.2 实验展示

首先,我们根据上述介绍的足球赛策略用 ISPL 描述机器人足球赛过程;然后,用工具 MCMAS_APTL 展示比赛过程并验证是否两个球队都有可能进球.球队分为红、黄两个队,每队有 3 名队员.足球场地模型如图 4 所示,守门员的活动区域为罚球区.

守门员可以采取的动作包括 act_none、run、intercept 和 kick the ball.显然,act_none 表示守门员不行动;run 表示守门员前进;intercept 表示守门员拦截对方球员进球;kick the ball 表示守门员踢球.前锋和中场可以采取的动作包括 act_none、kick off、kick the ball、run、intercept、shoot、pass the ball、take a pass 和 dribbling.kick off 表示球员在比赛开始时开球;pass the ball 表示球员将球传递给本队球员;shoot 表示球员试图将球踢入对方球门;take a pass 表示球员得到本队球员传的球;dribbling 表示球员持球前行.

足球赛的模型描述为一个解释系统,红队包含 3 个代理(agent):*r_gk*、*r*_1 和 *r*_2,其中,*r_gk* 为守门员;*r*_1 和 *r*_2 根据实际情况有 3 个角色——前锋(striker)、中场(midfielder)和后卫(defender).黄队也有 3 个代理:*y_gk*,*y*_1 和 *y*_2,角色分配与红队类似.代理 environment 包含可视变量用于描述比赛过程中的环境.

以 *r*_1 为例介绍机器人的部分行为,kick off 表示代理 *r*_1 在比赛开始时开球,代码如下所示.

```
Agent r_1
  Vars:
    state:{s_none,kick_off,kick_the_ball,run,intercept,shoot,pass_the_ball,dribbling}
  end Vars
  Actions={act_none,kick off,kick the ball,run,intercept,shoot,pass the ball,dribbling}
  Protocol:
  ...
  state=kick_off:{kick off};
  ...
  end Protocol
  Evolution:
  ...
  state=kick_off if Environment.state=r_kickoff and Environment.ballx=15 and Environment.bally=10
    and (Environment.r_2x<12 or Environment.r_2x>18) and (Environment.r_2y>13
    or Environment.r_2y<7) and (Environment.y_1x<12 or Environment.y_1x>18)
    and (Environment.y_1y>13 or Environment.y_1y<7) and (Environment.y_2x<12
    or Environment.y_2x>18) and (Environment.y_2y>13 or Environment.y_2y<7);
  ...
  end Evolution
end Agent
```

其中,(*Environment.ballx*,*Environment.bally*)表示球的位置坐标,(*Environment.red_gkx*,*Environment.red_gky*)表示代理 *red_gk* 的位置坐标,其他类似.*Vars* 中的 *state* 为代理 *r*_1 的状态变量,括号内为 *r*_1 的所有可能的状态取值.*Actions* 为 *r*_1 的所有的行为动作.*Protocol* 为 *r*_1 的协议,如,*state=kick_off*:{*kick off*}表示当 *r*_1 的状态为 *kick_off* 时,其采取的行动为 *kick off*.*Evolution* 是代理 *r*_1 的演变函数,例如代码中 *r*_1 的演变函数,当 if 后面的公式的值为真时,代理 *r*_1 的状态为 *kick_off*;如果代理在某一状态时同时有多个 if 后面的公式的值为真,则代理的状态是在几个可能的状态中随机选取的.

行为 *pass the ball* 表示代理 *r*_1 传球给队友 *r*_2.如果 *r*_1 持球并且无法将球踢入对方球门,那么 *r*_1 采取 *pass the ball* 行动,同时,对方球队队员会试图截球,*pass the ball* 的代码如下所示.

```
Agent r_1
```

...
Evolution:
...
state=pass_the_ball
if *Environment.state=r*1*_ball* and *Environment.r_*2*x*>*Environment.r_*1*x*
and *Environment.r_*2*x*–*Environment.r_*1*x*≤5...
and *Environment.y_*2*y*<*Environment.r_*2*y*)) or (*Environment.r_*1*y*>*Environment.r_*2*y*
and *Environment.r_*1*y*–*Environment.r_*2*y*≤5...
and *Environment.y_*2*y*<*Environment.r_*1*y*)) or (*Environment.r_*1*y*=*Environment.r_*2*y*
...));
...
end *Evolution*
end *Agent*

shoot 表示代理 *r_*1 将球踢入对方球门,ISPL 代码如下所示:

*Agent r_*1
...
Evolution:
...
state=shoot if *Environment.state=r*1*_ball* and ... or (*Environment.y_*1*y*<*Environment.r_*1*y*...
or (*Environment.y_*2*y*<*Environment.r_*1*y* and *Environment.r_*1*y*–*Environment.y_*2*y*≤3)));
state=shoot if *Environment.state=r*1*_ball* and *Environment.r_*1*x*≥20;
...
end *Evolution*
end *Agent*

当 *y_*1 或者 *y_*2 持球并且条件对 *r_*1 有利,*r_*1 可能采取 *intercept* 行为,该情况下 ISPL 代码如下所示:

*Agent r_*1
...
Evolution:
...
state=intercept if *Environment.state=y*1*_ball* and ((*Environment.y_*1*x*–*Environment.r_*1*x*)
*(*Environment.y_*1*x*–*Environment.r_*1*x*)+(*Environment.y_*1*y*–*Environment.r_*1*y*)
*(*Environment.y_*1*y*–*Environment.r_*1*y*))≤((*Environment.y_*1*x*–*Environment.r_*2*x*)
*(*Environment.y_*1*x*–*Environment.r_*2*x*)+(*Environment.y_*1*y*–*Environment.r_*2*y*)
*(*Environment.y_*1*y*–*Environment.r_*2*y*));
state=intercept if *Environment.state=y*2*_ball* and ...)≤((*Environment.y_*2*x*–*Environment.r_*2*x*)...);
...
end *Evolution*
end *Agent*

对于该系统,原子命题集合 *AP*={*redscore*,*yellowscore*},*redscore* 表示红队进球得分,*yellowscore* 表示黄队进球得分;代理分为两个组 g_1={*r_gk*,*r_*1,*r_*2}和 g_2={*y_gk*,*y_*1,*y_*2}.为了得到黄队进球得分的路径,我们给出 APTL 公式 $\neg\Diamond_{\langle\langle g_2\rangle\rangle}$*yellowscore* ,如果黄队能够进球,会输出一条路径满足 $\Diamond_{\langle\langle g_2\rangle\rangle}$*yellowscore* .其中,公式 $\Diamond_{\langle\langle g_2\rangle\rangle}$*yellowscore* 的语义为代理集合 g_2 存在策略使得执行路径上存在状态满足原子命题公式 *yellowscore*.如图 7 所示为一条满

足公式 $\Diamond_{\langle\langle g_2\rangle\rangle}$*yellowscore* 的路径,其中,足球从中央点(15,10)经由曲线到达点(0,11),球到点(0,11)表明黄队已进球,该曲线为比赛过程中足球的运动轨迹.

另外,本文实现了一场每队有两名队员的机器人足球赛,球队的策略与第 3.1 节介绍的策略类似.原子命题集合为 *AP*={*redscore*,*yellowscore*},*redscore* 表示红队得分进球,*yellowscore* 表示黄队得分进球.系统包含两支球队分别为 g_1={*red_gk*,*red_f*},g_2={*yellow_gk*,*yellow_f*}.

在球赛开始时,代理 *red_f* 开球.如果想要得到红队进球得分的一条路径,给出 APTL 公式 $\neg\Diamond_{\langle\langle g_1\rangle\rangle}$*redscore*, 当红队进球得分时,MCMAS_APTL 输出一条路径满足公式 $\Diamond_{\langle\langle g_1\rangle\rangle}$*redscore* ,如图 8 所示.

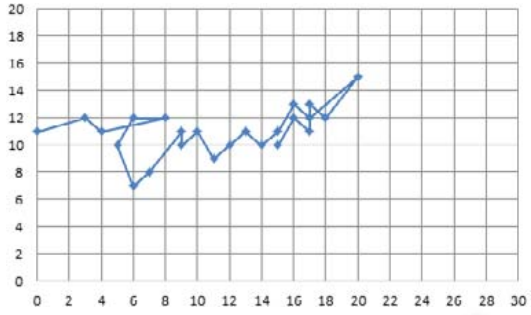

Fig.7 A path satisfies $\Diamond_{\langle\langle g_2\rangle\rangle}$*yellowscore*

图 7 满足 $\Diamond_{\langle\langle g_2\rangle\rangle}$*yellowscore* 的路径

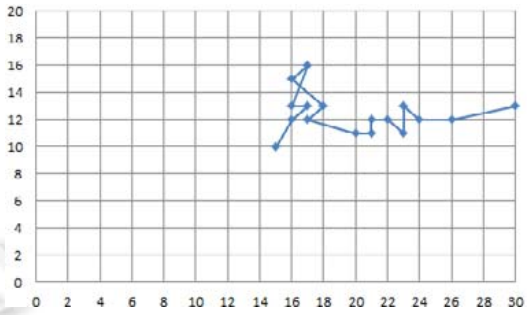

Fig.8 A path satisfies $\Diamond_{\langle\langle g_1\rangle\rangle}$*redscore*

图 8 满足 $\Diamond_{\langle\langle g_1\rangle\rangle}$*redscore* 的路径

另外,本文实现了每个球队包含两个队员的实验,其中,模拟过程执行时间、可达状态数以及所用的 BDD 存储空间如表 1 中 2×2 这一行所示.与每个球队有两个球员的比赛相比,每个球队有 3 个球员的比赛所用时间明显增多,状态空间爆发式增长.这两种实验都是在 2.93GHZ Intel Core i7,8GB RAM 上完成的.实验结果表明,工具 MCMAS_APTL 具有一定的实用性,但是要高效地验证复杂的多智能体系统,还需提高工具 MCMAS_APTL 的性能.

Table 1 Performance of MCMAS_APTL for robotic soccer games

表 1 机器人足球赛在 MCMAS_APTL 上的执行

	时间(s)	状态数	BDD memory
2×2	38.839	204 880	25 720 812
3×3	199.835	1.99141e+09	134 562 604

4 结 论

本文根据 APTL 符号模型检测算法实现了模型检测器 MCMAS_APTL,将 APTL 模型检测算法实现并且融合到 MCMAS 中,实现了一种基于 APTL 的多智能体系统模型检测器.在实现 MCMAS_APTL 时,借助了 MCMAS 的符号化系统模块,该部分可高效地符号化表示要验证的系统,从而提高了模型检测的效率.在工具 MCMAS_APTL 中实现了利用 APTL 公式验证多智能体系统的时序性质.在未来的工作中,将会研究系统的认知性质并且实现对多智能体系统认知性质的检测.

References:

[1] 蒋新松.人工智能及智能控制系统概述.自动化学报,1981,7(2):148−156.

[2] Visser W, Havelund K, Brat G, Park SJ, Lerda F. Model checking programs. Automated Software Engineering, 2003,10(2): 203−232.

[3] Halpern JY. Reasoning about knowledge: A survey. In: Handbook of Logic in Artificial Intelligence & Logic Programming. 1995. 1−34.

[4] Chen TL, Song F, Wu ZL. Verifying pushdown multi-agent systems against strategy logics. In: Proc. of the 25th Int'l Joint Conf. on Artifficial Intelligence (IJCAI 2016). New York, 2016.

[5] Chen TL, Song F, Wu ZL. Global model checking on pushdown multi-agent systems. In: Proc. of the 30th AAAI Conf. on Artifficial Intelligence (AAAI 2016). Arizona, 2016.

[6] Wang HY, Duan ZH, Tian C. Symbolic model checking for alternating projection temporal logic. In: Proc. of the COCOA. 2015. 481−495.

[7] Bryant RE. Graph-based algorithms for Boolean function manipulation. IEEE Trans. on Computers, 1986,35(8):677−691.

[8] Lomuscio A, Qu HY, Raimondi F. MCMAS: An open-source model checker for the verification of multi-agent systems. Int'l Journal on Software Tools for Technology Transfer, 2017,19(1):9−30.

[9] Luo XY, Su KL, Sattar A, Reynolds M. Verification of multi-agent systems via bounded model checking. In: Proc. of the Australian Conf. on Artificial Intelligence. 2006. 69−78.

[10] Zhang DP, Ji X, Wang XS. An AUML state machine based method for multi-agent systems model checking. In: Proc. of the Intelligent Information Processing. 2014. 106−112.

[11] Kress-Gazit H, Fainekos GE, Pappas GJ. Temporal-logic-based reactive mission and motion planning. IEEE Trans. on Robotics, 2009,25(6):1370−1381.

[12] Zhang YD, Song F. Model-checking for heterogeneous multi-agent systems. Ruan Jian Xue Bao/Journal of Software, 2018,29(6): 1582−1594 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/29/5462.htm [doi: 10.13328/j.cnki.jos.005462]

[13] Chen TL, Song F, Wu ZL. Model checking pushdown epistemic game structures. In: Proc. of the 19th Int'l Conf. on Formal Engineering Methods (ICFEM 2017). Xi'an, 2017. 36−53.

[14] Tian C, Duan ZH. Alternating interval based temporal logics. In: Proc. of the ICFEM. 2010. 694−709.

[15] Guarnizo JG, Mellado M, Low CY, Aziz N. Strategy model for multi-robot coordination in robotic soccer. Applied Mechanics and Materials, 2013,393(6):592−597.

附中文参考文献:

[12] 张亚迪,宋富.异构多智能体系统模型检查.软件学报,2018,29(6):1582−1594. http://www.jos.org.cn/1000-9825/29/5462.htm [doi: 10.13328/j.cnki.jos.005462]

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

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

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