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,$Sat_{\bar{v}}(\psi_i)$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$Sat_{\bar{v}}(\phi_i)$.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$Sat_{\bar{v}}(\phi_{ir})$变的很复杂.这个问题可以通过计算不动点得到$Sat_{\bar{v}}(\phi_{ir})$,然后计算前驱集合$PRE_img(A_{ir}, Sat_{\bar{v}}(\phi_{ir}), R_t)$.通过将上面求到的布尔方程求"与",得到$Sat_{\bar{v}}(\psi_i)$.

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

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.

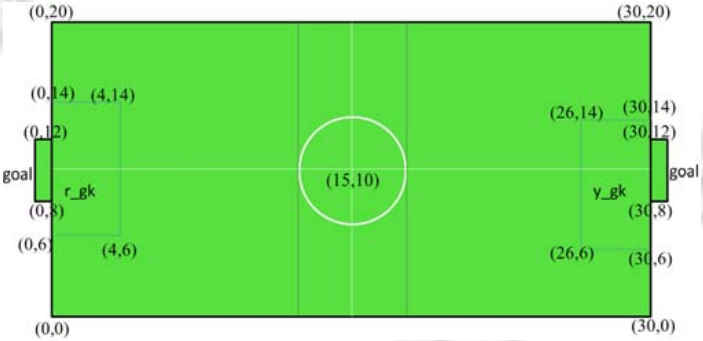

Fig.4 Soccer field model

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.

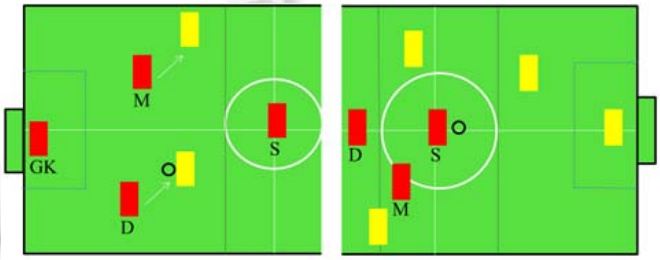

Fig.5 Role assignment in a defensive tactic

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.

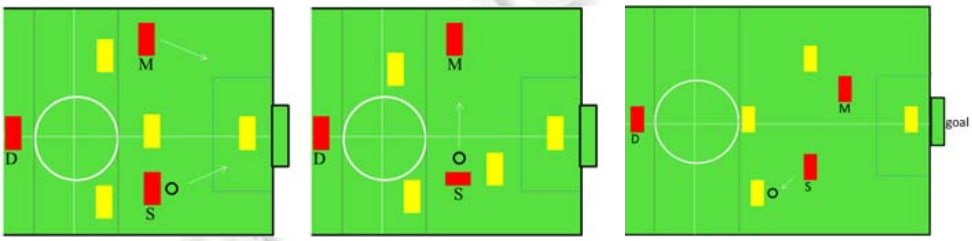

Fig.6 Role assignment in a offensive tactic

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.

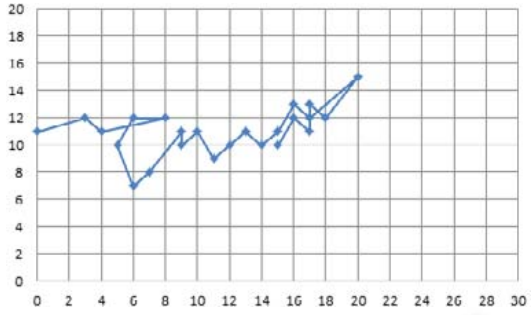

Fig.7 A path satisfies $\Diamond_{\langle\langle g_2\rangle\rangle}$*yellowscore*

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

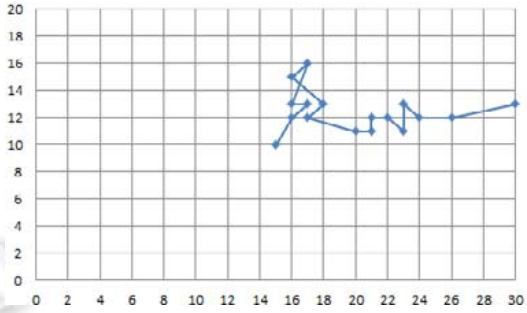

Fig.8 A path satisfies $\Diamond_{\langle\langle g_1\rangle\rangle}$*redscore*

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.