# An Experiment of Verification of Multi-agent Robotic Soccer Plans with Model Checking

Rui C. Botelho A. S.[1], Aline M. S. Andrade[2], Augusto Loureiro da Costa[3], Frederico J. R. Barboza[2]
[1]Post-graduation Program on Mechatronics – Master, [2]Distributed Systems Laboratory, [3]Robotics Laboratory
Federal University of Bahia, UFBA
Salvador, Brazil
{ruicbs,fred.barboza}@gmail.com,{aline,augusto.loureiro}@ufba.ba

***Abstract*—In this paper we present an experiment of model checking which consists of the verification of plans of a multi-agent system for simulated robot soccer. This system is of considerable complexity because it is concurrent, nondeterministic and with partial vision of the environment. Some solutions adopted relative to modeling and process of verification to circumvent state space explosion are reported.**

***Keywords—Model Checking; Planning; Multi-agent Systems, Autonomous Agents***

## I. INTRODUCTION

In recent decades, research involving the development of Autonomous Agents - AAs and Multi-agent Systems - MAS has increased in academia and industry. This is due to the use of these systems, whether real or virtual entities, in various kinds of applications: autonomous robots, multi-robot systems, unmanned vehicles, control systems, flight plans of aircraft combat systems, air traffic control, simulation, matches of physical or simulated robots in software, softbots, among others.

The use of MAS and AAs are motivated by their inherent characteristics such as autonomy, collaboration, proactivity among others. The main activities of these entities are related to the achievement of objectives that are performed based on planning. It is necessary to ensure that AAs individually or MAS collectively have correct plans to guarantee that they do not behave unintendedly, and have desirable outcomes. Formal methods can be used in this context, particularly model checking, which has been used in several works published in this area [1, 2, 3, 4, 5, 6, 7].

In this paper we present the results of the verification of plans of robot soccer, the Mecateam [8], which uses the simulated environment Robocup [9]. Our interest in this team is mainly due to the fact that it is a multi-agent system with a multilayer architecture, which imposes a parallelism in the internal plans of the agents increasing the state space of the problem.

In robot soccer the environment is non-deterministic and players only have a partial vision of it. These features combined with the three-layer architecture of Mecateam add complexity to its planning, requiring care in both modeling and verification of the plans. With this in mind some solutions were carried out: abstractions were done to overcome the state space explosion; a decomposition of the plans based on the multilayer organization of the agents was considered; the process of verification was implemented in an incremental way considering an evolution of the models of the plans from the individual plans of the agents to their collective ones. These solutions together allowed the verification of a significant part of the state space of the problem.

For our experiment the UPPAAL model checker [10] was used. UPPAAL is a tool set applied in modeling and verification of systems which uses timed automata formalism to model the system and a subset of TCTL (Timed Computation Tree Logic)[10] for the specification of the system properties.

The contributions of this paper include solutions adopted in modeling and verification of a MAS complex real application with model checking which may be useful in similar multi-agent systems.

This paper is organized as follows: in the following section related works are presented; section 3 presents an introduction of the robot soccer team; section 4 presents the specification of the plans as automata; section 5 presents the plan model checking; and finally, in section 6, some conclusions are presented.

## II. RELATED WORKS

In [1,2] the validation of plans for a multi-agent simulation environment for tactical fighter aircraft is considered. This is highly dynamic, non-deterministic and has partial vision. The multi-agent plans are modeled as a network of hybrid automata [3] and the agents have more reactive than cognitive behavior.

In [4,5] the plans verification of a controller and scheduler system that composes the remote control of the robot Deep Space 1 used in U.S. space agency missions (NASA - National Aeronautics Space Administration)was carried out using UPPAAL. This work deals with the reactive behavior of the individual agents without considering the interaction with other agents.

In [6] the potential use of hybrid automata using the HYTEC tool to model and verify plans of autonomous agents

is demonstrated considering an unmanned aerial reconnaissance case study. In this case the plans are almost always deterministic, the environment is non-dynamic and the agents do not interact with each other.

The work presented in [7] uses model checking for verification and simulation of soccer teams of robots at runtime. It considers a specific platform, the Ericson Company platform and the ERLANG specification language and McERLANG verifier. The verification of the agent's code is considered as a whole, which comprises planning activities, interface code, actions done in the environment, etc. which make it difficult to distinguish planning errors from other errors. In this work the team is modeled as a single component which prevents the analysis of agents in isolation. Finally, the verification is performed at runtime using the Soccerserver simulation environment to simulate the match.

The works presented above focus on different features of MAS, encompassing their several planning characteristics. However, none of them consider as many aspects together as our work does, namely: non-determinism, communication among agents, partial vision, structured planning in three-ties, verification of individual agents and collectively. These aspects together increase the complexity of the problem and require creative solutions to abstract the model and a systematization of the verification process in order to model and check the plans.

## III. Robotic Soccer

Robot soccer is used as a platform for the study of a wide variety of problems inherent in AAs, robotics and cooperative MAS. Proposed by Robocup Federation, the robot soccer Robocup has emerged as a laboratory for the study of Distributed Artificial Intelligence (DSI) and its ramifications [9].

In a robot soccer match the robots must be able to: recognize their position and all references of location in the soccer field, represent the environment of the game, set objectives, plan and execute actions to achieve their goals. To deal with and provide consistent and intelligent behavior, agents must combine a sequence of actions associated with a primary set of concurrent actions, first individually and then collectively, according to a planning that meets individual and group objectives for each state of the environment.

### A. The robot team Mecateam

The Mecateam robot soccer team was designed according the Concurrent Autonomous Agent Architecture [11] which comprises a cognitive, an instinctive and a reactive layer as presented in Fig. 1. This team has been participating in major competitions and has emerged as one of leading teams in simulated robot soccer in Brazil. The cognitive layer is responsible for the planning of the team, the reactive layer perceives and acts on the environment and the instinctive layer intermediates the communication between the cognitive and the reactive layer. The plans of the team are covered by cognitive and instinctive rules. The cognitive rules are responsible for defining the objectives of the agents and between each current objective and every future goals of the team, the cognitive rules promote a change in the state of the agent according to information about the environment from the instinctive rules. The instinctive rules receive this information from the reactive layer and determine the behaviors to be carried out by the reactive layer in the environment.

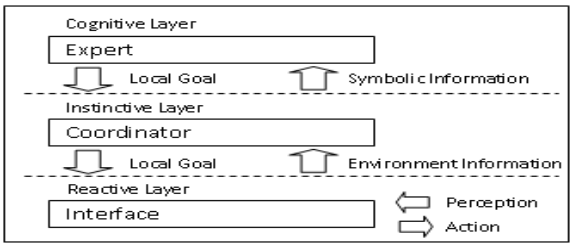


Fig. 1. Representation of the architecture of [10] (Adapted).

The individual plans of each player are composed of their cognitive and instinctive rules and their interactions. The performance of each player depends on its plans, changes in the environment and its internal state. In turn, the plans of the team are composed of the union of all individual plans and represent the possibilities of collective action in response to changes that occur in the environment.

The reactive layer has no decision-making behavior and for this reason it was not considered in the scope of this work.

### B. Cognitive rules

The cognitive plan has some similar rules for the groups of agents (players): defenders (players 2, 3, 4 and 5) and midfielders and forwards (players 6, 7, 8, 9, 10 and 11). Table I presents two examples of cognitive rules by groups of players. In the first rule (rule 0) the *local_goal current* of the player is *none*. This corresponds to the beginning of the game, the goalkeeper and the defenders go to advance which means taking offensive actions in the game. The assertion *local_goal status active* activates the current local goal, now *advance*. The second rule (rule 3) changes the current local goal *side-attack* to the current local goal *mark* if the local goal *side_attack*fails. In this case the state *mark* is activated (*local_goal status active*).

TABLE I. Examples of cognitive rules

| Goalkeeper and Defenders | Midfields and Forwards |
|---|---|
| (rule_0_start<br>(if<br>(logic(local_goal current none)) )<br>(then<br>(logic(local_goal current advance))<br>(logic(local_goal status active)) )) | (rule_3_side_attack<br>(if<br>(logic(local_goal current side_attack))<br>(logic(local_goal status fail )))<br>(then<br>(logic(local_goal currentmark))<br>(logic(local_goal status active )) )) |

In general the variable *local_goal current* has a set of possible values: none, mark, advance, side_attack and ending; and the variable *local_goal status* can assume the values: active, achieve and fail. Combinations among these values determine the individual player's goals. The set of particular values of *local_goal current* of a player depends on its role in the team, unlike the *local_goal status* of each player which has

the same set of values for all players. Table II shows the values of these variables per group of players.

TABLE II. SETS OF LOCAL GOAL CURRENT / STATUS BY GROUP OF PLAYERS.

| | | Players | | |
|---|---|---|---|---|
| **Variables** | | ***1*** | ***2,3,4,5*** | ***6,7,8,9,10,11*** |
| ***local_goal*** | ***current*** | mark, side_attack, none | | |
| | | advance | | |
| | | ending | | |
| | ***status*** | fail, active, achieved | | |

### C. Instinctive rules

The instinctive level of each agent contains a set of instinctive rules and a subset of it is presented in Fig. 2. These rules transmit the vision of the environment, from the reactive level to the cognitive level, and determine the behaviors of the reactive level on the environment. Between each current and each future goal of the team, the cognitive level should promote a change in the current state of the agent. This change depends on the results of the execution of a set of instinctive rules. Therefore, the instinctive rules intermediate the communications between the reactive and the cognitive level. The reactive level is responsible for: perceiving the situation of the environment, acting on the environment, activating reactive actions such as *intercept ball*, *drive_ball_forward*, *hold_ball*, etc.

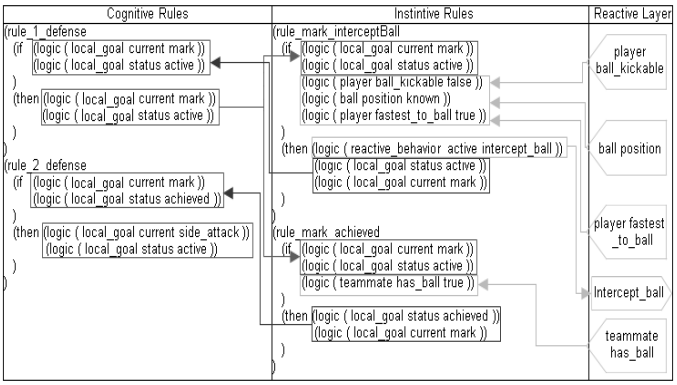


Fig. 2. Example of interactions between cognitive, instinctive and reactive rules.

## IV. SPECIFICATION OF THE PLANS AS AUTOMATA

In this section we present a brief description of timed automata of UPPALL model checker used to model the rules and some examples of models.

### A. UPPAAL automata

An automaton in UPPAAL is represented as a graph with a finite set of locations and transitions, represented as nodes and edges respectively. The initial location is represented by two concentric circles. Locations labeled "U" have priority over other locations. In the automaton of Fig. 6, the urgent location *SendingEnvironmentInfo* has priority over any other location because the environment state must be updated before any other action of the game is executed.

A transition can be controlled by guards and channels. The guards are logical expressions that determine the conditions for a transition to be triggered. For example the guard at the transition between the locations *None* and *Advance* (Fig. 3) determines that this transition will only be executed if the *LocalGoalCurrent == 4*. The channels synchronize the actions of two or more automata and can also be declared as urgent or broadcast to give priority to the corresponding transition and enable the communication with many automata at the same time, respectively.

### B. Models of the rules

The left and right sides of the rules specify pre and post conditions which are modeled by the guards and changing in the values of variables in the automaton. Part of the three groups of players presented in Table II has the same set of rules and they are modeled as shown in Fig. 3 and Fig. 4.

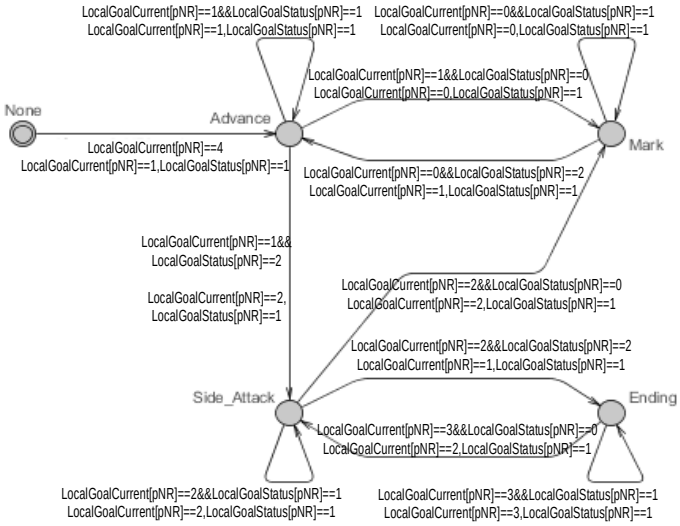


Fig. 3. Automaton of cognitive rules of the goalkeeper.

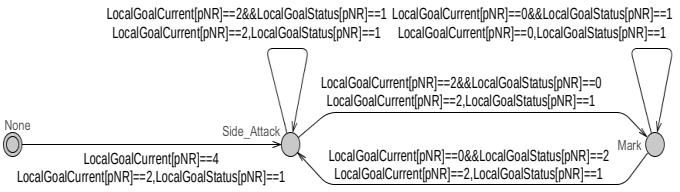


Fig. 4. Automaton of cognitive rules of the midfield and forward players.

The locations of the automata represent the current local goals of the players. InFig. 3, the locations corresponding to the possible goals of the cognitive rules of thegoalkeeper can be identified (see Table II): *None*, *Advance*, *Mark*, *Ending*, *Side_Attack*.In Fig. 4 the locations *None*, *Side_Attack* and *Mark* correspond to local current goalsof midfields and forwards players. For control purposes, the variables*LocalGoalCurrent* and *LocalGoalStauts* are declared in the automata. They correspondto the variables *local_goal current* and *local_goal status* of the rules, respectively.*LocalGoalCurrent* can assume the values: 0 to *mark*, 1 to *advance*, 2 to *side_attack*, 3 to*ending* and 4 to *none*.

*LocalGoalStatus* can assume the values: 0 to *fail*, 1 to *active* and2 to *achieved*. The combined values of both variables drive the change in location ofthe automaton. For example, *rule 2_defense* of Fig. 2 has state *mark* and status*achieved*. In the automaton of Fig. 4 that models this rule the corresponding location*Mark* has the guards *LocalGoalCurrent == 0* and *LocalGoalStatus == 2* thatcorrespond to state *Mark* and status *Achieved* respectively. So, a transition to thelocation *Side_Attack* occurs and the values of these variables are updated to*LocalGoalCurrent = 2* and *LocalGoalStatus == 1*.

Each instinctive rule was created as an elementary component to allow the removal or addition of rules for checking a particular player. Fig. 5 gives an idea of how the instinctive rules were modeled in automata.

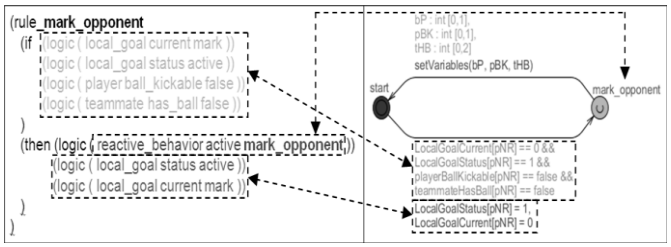


Fig. 5. Example of an instinctive rule and its respective automaton.

The automata presented up to here were used to verify individual players, without considering their interactions with the environment. For the verification of the entire team the model was enhanced with the communication channels *Startgame*, *ReStartGame* and *ActionInfo* between the environment and the players. Fig. 6a and Fig. 6b show examples of cognitive and instinctive rules resulting from the inclusion of these communications channels, respectively. A global boolean variable *EnvironmentInfo* was declared to control if the environment has processed the information resulting from the reactive actions in the previous round of the match and if the information about each player is available.

The channels *Startgame* and *ReStartGame* are declared broadcast to establish communication of the environment with all the players at the same time and they are declared urgent because the initial configuration of the game should be restored with priority over other actions. The channel *ActionInfo* represents the communication of each individual player with the environment and it is declared urgent because the environment should be restored to represent the actual situation of the game before any other action is executed.Fig. 7 shows the automaton of the environment. From its initial location a message (*Start Game!*) is sent to all players to inform that the environment is ready and that the players can start the game. In the location *WaitingForActions* the environment waits for all players to inform (through channel *ActionInfo*) the rules that were selected relative to a specific scenario of the match. After receiving all messages the transition to the location *SendingEnvironmentInfo* occurs and the function *setOpponentInformations()* is executed to update the current situation of the play considering the informations of the players and the actions of the adversary team which is simulated by the function *setOpponentInformations (oPHB,pTOA,pSOA)*. In case a goal is not done (*goalScored == false*) a new round is initiated by the function *enableNextRound()*, which is responsible to set the new state of the environment in order to the match continue. In case a goal is done (*goalScored == true*) the play is restarted. The function *setInitReinitValues()* carries out the configuration or reconfiguration of the environment in the beginning of the match or when a goal is done respectively. When a goal is done all players are informed by *ReStartGame!*channel.

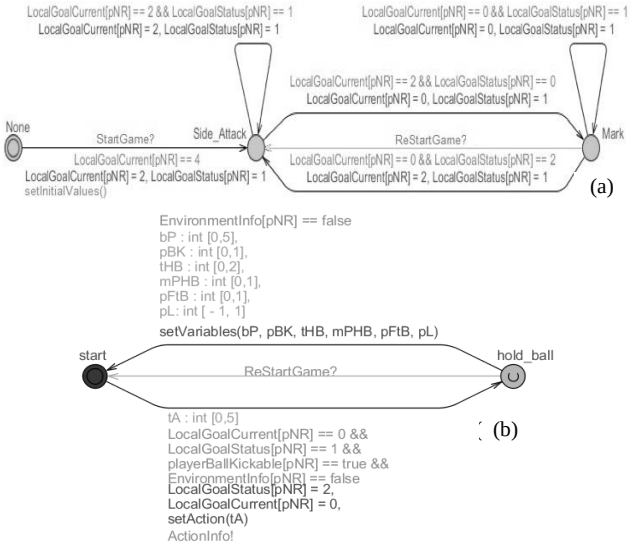


Fig. 6. Model of rules with communication channels: (a) cognitive rules of midfield and forward players, (b) an instinctive rule.

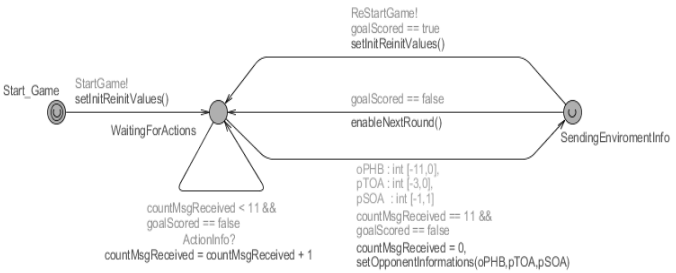


Fig. 7. Automaton of the environment.

The partial vision and the non-determinism of the environment were represented through a range of values previously defined by select label variable[1] of UPPAAL. For example, in Fig. 6 the select label variable (*oPHB: int[-11,0]*) receives a value between -11 and 0 each time a transition occurs from the location *WaitingForActions* to *SendingEnvironmentInfo* to indicate who player is with the ball when it is under control of the opponent team: {-1} – the ball is with the goalkeeper, {-2 to -5} – the ball is with the defensive players and {-5 to -11} – the ball is with the midfield or forward players. If the selected value is 0 the ball is in the field but it is not controlled by anyone.

## V. MODEL CHECKING OF THE PLANS

Model checking consists of the specification of a finite model (an automaton or a variation of this type of representation) of a system and in the verification of desired

[1]Select label variables will take a non-deterministic value in the range of their respective types [9].

properties of the system through an exhaustive scan of the model state space, which is done automatically [11].

UPPAAL is a model checker designed for the verification of real-time systems. The UPPAAL tool set comprises a modeling environment, a simulator that is used to interactively monitor the execution of the model and a property checker that uses a query language which is a subset of the TCTL. The models in UPPAAL are designed as a network of timed automata as shown in the previous section of this paper [11]. The problem of state explosion is an intrinsic characteristic of model checking and solutions should be adopted to overcome this problem, such as abstractions in the models, compositional model checking [11], and other solutions.

### A. Abstractions

To carry out model checking we did some abstractions in order to simply the model and reduce the state space: the soccer field was represented with one dimension, some match situations and time constraints were disregarded and the actions of the adversary team were simulated by a function in UPPAAL. In addition, we used an incremental verification process, presented in next section, which enabled the verification of individual agents initially, followed by the verification of groups of agents until the whole team in interaction with the environment was considered. Moreover, the multilayer architecture of Mecateam led to a modularization of the specification and verification considering the separation of cognitive and instinctive rules.

The representation of the field in one dimension (soccer simulation works with two dimensions) considered six regions, corresponding to the classic regions of a soccer field, which are: area, intermediate and midfield for both teams. This representation proved sufficient to explore relevant situations of the game such as the movements of the players and the partial view of the environment. It also allowed a significant simplification of the state space of the model and eliminated the problem of having to deal with calculations of movement and trajectory in the 2D environment, which should be resolved by a low level layer (reactive layer).

The following match situations were not represented because they are controlled by Soccerserver and are not part of the plans: corners, off sides, penalties, among others. In relation to time constraints, the cognitive and instinctive rules do not use this type of restriction. Although the Robocup environment considers a deadline for receiving agent messages, and our model has been devised to allow the inclusion of time constraints, we did not consider them in this version of the model. We emphasize that this did not cause any loss, considering the purpose of the cognitive and instinctive plan verification. The function that represents the adversary team simulates its behavior by choosing an offensive or defensive action, depending on the control of the ball is with the adversary or with it respectively. Random values are set through select label variables of UPPAAL to define which player will play, which actions will be executed and if the selected actions to be executed can succeed or not. So it was not necessary to create automata to represent the adversary team which has simplified the modeling.

We used pseudorandom values and data structures to confront the real situation of the match and the vision of each player with respect to it. This enabled a simple representation of the environment's uncertainty (caused by non-determinism) and the representation of the partial vision of the agents. An example is the position of the ball which is only partial determined by the agents and depends on their distances from it. The vision of each agent of the ball position is defined by the integer variable *bP: int[0,5]*, as seen in Fig. 7. This variable selects one value between 0 and 5 that corresponds to the soccer field areas, according to the position of the player in the field.

### B. Verification of the plans

Due to the complexity of the system we defined a modularization of the verification process at several stages from simple plans to collective plans. This facilitated the analysis of model checking results and allowed the detection of errors in the subspaces of the whole model.

The process of verification of the cognitive and instinctive plans was done in five stages: 1. Verification of common properties related to common rules of all players or group of players (defenders or midfield players or offensive players). 2. Verification of the individual plans which consist of the particular rules of each agent in the team. 3. Modeling and verification of the environment. 4. Verification of each agent in communication with the environment. 5. Verification of the entire team in communication with the environment.

For the verification of the individual plans of each player we modularized the process, similarly as in composition model checking technique assume-guarantee [11]: first we assumed that the environment, the reactive layer and the instinctive layer were correct and applied model checking to the cognitive rules, and then we verified the instinctive rules assuming now that the cognitive rules were correct. This facilitated the analysis of the model checking and identification of errors.

In Table III are presented the properties used to verify the plans: safety (first and second line) and reachability properties (third line). These rules are specified as processes in UPPAAL which are instantiated according to the type of the rules as specified by the BNFs in the table. The safety properties are used to verify if the automata are free of deadlock (first line) or if the automata are consistent with the rules (second line), i.e., if the models had been constructed correctly. The reachability properties are used to verify if any automata location, corresponding to each cognitive or instinctive rule, is achieved from the initial location to determine if these rules are in fact used by the agents meaning that the reactive behaviors are achievable and the player will drive them.

### A. Some results of model checking

As a result of the verification of the individual agents the occurrence of deadlock was observed in all players due to the bad construction of a rule (*rule_mark_hold_ball*) which led all plans to a cognitive state of the system in which there were no transitions possible. In addition, some locations were identified as not reachable in the instinctive rules, as for

example the state *advance* in the instinctive rule *Advanced_Pass_Ball_Forward* due to an error in this rule which was reflected in the model. After we had corrected the identified errors, the results of the model checking of the entire team showed that the model had preserved the reachability properties already verified before.

TABLE III. PROPERTIES.

| Description | Formulae |
|---|---|
| 1. Non-occurrence of deadlocks | A□ not deadlock |
| 2. In all paths never occurs that the value of local goal current of the player is different from X and the location of the cognitive automaton is equivalent to X. | A□ not (LocalGoalCurrent[<pNR>] != X and P<pNR>_C.<CognitiveLocation>)<br>Ex: A□ not (LocalGoalCurrent[1] != 0 and P1_C.Mark) |
| 3. Exist a path for any location from the initial state. | E◊ P<pNR>_<Rule><br>Ex: E◊ P1_C.Side_Attack |
| Where the symbols declared in < > are defined as:<br><pNR> ::= 1 \| 2 \| 3 \| 4 \| 5 \| 6 \| 7 \| 8 \| 9 \| 10 \| 11<br><Rule> ::= C.<CognitiveLocation> \| <InstinctiveRule><br><CognitiveLocation> ::= Mark \| Advance \| Side_Attack \| Ending \| None<br><InstinctiveRule> ::= <TypeOfCognitiveRuleAssociated> <AcronismAndInstinctiveRule><br><TypeOfCognitiveRuleAssociated> ::= A \| M \| SA \| E<br><AcronismAndInstinctiveRule> should be replaced by the name of an stinctive rule (which trigger a corresponding reactive action) such as: SB.seach_ball , HB.hold_ball , etc. | |

Table IV shows some data about the verification. The specification consisted of 47 rules (10 cognitive and 37 instinctive). A total of 161 automata were instantiated: 11 automata from cognitive rules (one for each player), 147 automata from instinctive rules and one automaton of the environment. A total of 204 properties were checked. In the verification of each individual player all properties were checked but in relation to the complete team sate explosion occurred in the verification of 63 properties. Despite this state explosion situation, 141 properties (69.11%) were verified: 114 (80.85%) were satisfied and 27 (19.15%) were not satisfied. The maximum verification time of a property was 27.83 hours and the minimum was 0.01 seconds, approximated. It is worth mentioning that all the individual rules were completed checked but when the rules were processed collectively only the cognitive rules were completely checked and approximately 50% of the instinctive rules were not checked because of state explosion.

TABLE IV. TOTAL OF VERIFICATION.

| Rules | Automata | Properties | | | |
|---|---|---|---|---|---|
| | | Quantity Data | | Time Data | |
| | | Total 204 | | Minimum | Maximum |
| 47<br>10 cog.<br>37 inst. | 161<br>11 cog.<br>149 inst.<br>1 env. | Verified 141 | Satisfied 114 | ~0.01s | 8,5931s<br>(~27.83h) |
| | | | Not satisfied 27 | | |
| | | Not verified 63 | - | - | - |

## VI. CONCLUSIONS

The verification of AAs and MASs plans is not straightforward, it is normally a laborious and complex activity.

This paper presented results of applying model checking on the verification of the agent plans of a robot soccer application with a three-tier architecture, where the environment is nondeterministic, dynamic and has partial vision. To minimize the state space explosion, some solutions were applied relative to abstractions of the model and for the process of verification. A modularization of the process of verification was carried out considering an evolution of the model from individual players to the whole team in order to identify errors as soon as possible, which otherwise could be difficult to interpret due to the size of the complete model.

The solutions proposed can be adapted for other similar applications and as a future work we intend to devise an interactive tool to support the development and verification of multi-agent systems with similar characteristics of this one.

## REFERENCES


[1] A. El Fallah Seghrouchni et al, "Modelling, Control and Validation of Multi-Agent Plans in Dynamic Context", AAMAS, vol. 1, pp.44-51, Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 1 (AAMAS'04), 2004.

[2] F. Marc, "Planification Multi-Agent sous Contraintes dans un Contexte Dynamique: Application Aux Simulations Aériennes", Thèse (Doctorat en Informatique), École Doctorale d'Informatique, Télécommunication et Électronique de Paris, Université Pierre et Marie Curie, 2005.

[3] T. A. Hezinger, "The Theory of Hybrid Automata", 28p, Electronics Research Laboratory, College of Engineering, University of California: Berkeley, 1996.

[4] L. Khatib et al, "Verification of plan models using UPPAAL", 1st International Workshop on Formal Approaches to Agent-Based Systems, Maryland, 2000.

[5] L. Khatib et al, "Mapping temporal planning constraints into timed automata", in: Proceeding of 8th International Syposium on Temporal Representation and Reasoning, 249p, IEEE Computer Society : Cividale Del Friuli, 2001.

[6] G. S. Costa, "Utilização da Verificação de Modelos para o Planejamento de Missões de Veículos Aéreos não-Tripulados", Dissertação (Mestrado em Engenharia Elétrica) - IME, Rio de Janeiro, 2008.

[7] C. B. Earle et al, "Verifying Robocup Teams", in: Proceeding of MoChArt 2008, 5th International Workshop on Model Checking and Artificial Intelligence, 189p, Patras, 2008.

[8] O. Santana Jr, C.F.G. Chavez, A. Loureiro da Costa, "MecaTeam Framework: An Infrastructure for the Development of Soccer Agents for Simulated Robots", IEEE Latin American Robotic Symposium, LARS, p 137-142, ISBN: 978-1-4244-3379-7, 2008.

[9] H. Kitano, "Robocup: The robot world cup initiative", in Proc. of The First International Conference on Autonomous Agent (Agents-97)) Marina del Ray, The ACM Press, 1997.

[10] G. Behrmann et al, "A Tutorial on Uppaal 4.0", Department of Computer Science, Aalborg University, 2006.

[11] A. Loureiro da Costa, G. Bittencourt, "From a concurrent architecture to a concurrent autonomous agents architecture", in: Third International Workshop in RoboCup, Springer Lecture Notes in Artificial Inteligence LNAI, 1856pp, pp.85-90, 1999.

[12] E. M. Clarke, O. Grumberg and D. A. Peled, "Model Checking". The MIT Press, 1999.