Multi-Robot Coordination using Setplays in the Simulation League

Luís Mota
Laboratório de Inteligência Artificial e Ciência de Computadores (LIACC), and Instituto Universitário de Lisboa (ISCTE - IUL), Lisboa, Portugal
luis.mota@iscte.pt

Luís Paulo Reis
Laboratório de Inteligência Artificial e Ciência de Computadores (LIACC), Faculdade de Engenharia da Universidade do Porto (FEUP), Porto, Portugal
lpreis@fe.up.pt

Nuno Lau
Instituto de Engenharia Electrónica e Telemática de Aveiro (IEETA) Universidade de Aveiro, Portugal
nuno.lau@ua.pt

***Abstract*— Strategic planning and multi-agent coordination are major research topics in the domain of RoboCup. Innovations in these areas are, however, often developed and applied to only a single RoboCup league and/or one domain, without proper generalization. Moreover, the more technical leagues, like Middle-size and Humanoid, tend to focus development on low-level skills, that often suffice to gain a competitive edge over other teams. In these leagues, the development of high-level cooperation is secondary.**

Although the importance of the concept of Setplay, to structure a robotic soccer team behaviour, has been acknowledged by many researchers, no general framework for the development and execution of generic Setplays has been introduced in the context of RoboCup. This paper presents such a framework for high-level setplay definition and execution, applicable to any RoboCup cooperative league and similar domains. The framework is based on a flexible, standard and league-independent language, which defines setplays that are interpreted and executed at run-time, using inter-robot communication.

An initial major step in the development of the Setplay framework is its usage and testing in the scope of the FCPortugal team, which participates in the RoboCup 2D-simulation and 3D-simulation leagues, where it won several titles in both leagues. After this successful implementation, this framework will be used in the mid-size league.

I. INTRODUCTION

RoboCup[1] [1] is an international initiative to promote Artificial Intelligence, robotics, and related fields. It fosters research by providing a standard problem where a wide range of technologies can be integrated and examined. RoboCup uses the soccer game as a central topic of research, aiming at innovations to be applied for socially significant problems and industries. Research topics include design principles of autonomous agents, strategy acquisition, real-time reasoning, robotics, and multi-agent collaboration, which this paper aims at contributing to.

Robotic Soccer needs, as the research in the domain develops, coordination at team scope, which involves planning at many levels. This paper deals with representing and executing high level, flexible plans for robots playing in different RoboCup leagues. A framework for representing, executing and evaluating such plans is presented, relying on a high-level setplay definition language and inter-robot communication.

Setplays are commonly used in many team sports such as soccer, rugby, handball, basketball and baseball. There are surely several important differences between robot soccer and human sports, but setplays are nonetheless a useful tool for high-level coordination and cooperation.

A. Motivation and Requirements

Setplays are used in a high number of situations in human sports. One of the situation where they are typically used are set-pieces, i.e., situations where the ball is put back to play after an interruption, like kick-offs or corner kicks. In these situations the players can position themselves and perform quick plans that can bring them a competitive edge.

A particular example is the corner kick, where some players position themselves in specific places, rapidly pass the ball sequentially between themselves until it reaches a well-positioned player, who will shoot at goal.

Since the tactics and skills of robots are continuously improving, and opponents try to react to new developments and counter them, it would be very useful for every team if there would be a way of freely defining setplays, through configuration files or even a generic setplay graphical editor, like the one already developed for setplay and formations definition in the FC Portugal team [2].

This kind of collective play should be described and shared in a standard, league-independent and flexible way, which would be interpreted and executed at run-time. The first benefit is the possibility of writing arbitrary Setplays, which can dynamically be used during the game, opening horizons to new plays which could, for instance, differ from game to game, to better deal with each opponents' characteristics. In this sense, the Setplays can also be used in different leagues. Furthermore, since any player can have access to the definition of Setplays and interpret their content, Setplays can also be a means for the creation of mixed teams, where heterogeneous robots would play together: when the Setplays are being executed, players simply have to follow the steps in the Setplay in order to cooperate.

To fulfil these requirements, one needs a standard language, where Setplays can be defined and interpreted by

[1] http://www.robocup.org

any player in any league. The basic concepts of soccer (moves, conditions, actions, skills) need a clear and concrete definition. Also, the transitions between intermediary steps have to be expressed, as well as termination conditions. Such a language is thus the scientific subject being presented in this article.

Further, a framework has been developed in C++ to ease the implementation of Setplays in any team: this framework already provides different features: a parser for setplay definition files and an engine to run the Setplays. As such, in order to implement Setplays in a new team, only two tasks have to be carried out: implement the testing of soccer conditions and the execution of actions in this domain. Details on these tasks will be given in section III-B.

B. Article Outline

A brief State of the Art in cooperative team-play in the RoboCup domain, as well as related work, are presented in section II. The framework that models this language is presented in section III, including its distribution as a C++ library. The implementation of the framework in the 2D simulation league [3] was a major step in the testing of this framework, and is described in section IV. This implementation was done on top of the code of the FCPortugal team[2]. Finally, conclusions are drawn, and future lines of research are presented in section V.

II. State of the Art

A. Coordination through Plays

The concept of *Setplay* is present in a teamwork and communication strategy for the 2D simulation league, presented by [4]. These *Setplays*, however, lack some of the most relevant features now presented. Namely, they are meant to be used only in very specific situations, like corner kicks and throw-ins, which are decided by the referee, and are unique for each of these situations. Thus, the question of Setplay activation and choice is not considered. Further, there is no mention to Parameters, though Player Roles are proposed. Most important, a *Setplay* is limited to a sequence of Steps, without alternatives, which excludes the need of choice announcing, and therefore the use of communication with this purpose.

The RFC Stuttgart/CoPS team uses Special Interaction Nets [5], a simplified version of Interaction Nets adapted to cooperation in multi-agent environments. These diagrams include states, representing actions, transitions, which model conditions, eventually global, and sub-nets, with the former components. Certain conditions can model time-dependent issues, and can be used to synchronise multi-agent behaviour. Messages can also be used to synchronise multiple networks. The model does not present a standard set of concepts, which does not enforce generalisation and may lead to the developing of very specific cooperative strategies.

XABSL is a language to describe behaviors for autonomous agents based on hierarchical finite state machines, and has been used by different teams in RoboCup, namely the German Team [6]. Recent developments [7] have allowed the use of the language to develop cooperative multi agent behaviour, through synchronisation elements, that allow the specification of minimum or maximum number of robots is a given state.

[2]http://www.ieeta.pt/robocup

III. Setplay Framework

The Setplay framework was designed with the goal of being general, flexible, parameterizable and applicable to any robotic soccer league. Its general structure is shown schematically in fig. 1.

At the top level, a *Setplay* is identified by a name, and has *parameters*, which can be simple data types like integers and decimals, or more sophisticated concepts as *points* and *regions*. *Setplays* also have *Player References*, which identify players taking part in the Setplay. The *Player References* can point to specific players, or be *Player Roles*, i.e., abstract representations of a particular role in the *Setplay*, identified by a name (e.g., attacker, supporter). *Parameters* and *Player Roles* will be instantiated at run-time, allowing a flexible use of the *Setplay*.

Steps are the main building block of a *Setplay*, which contains an arbitrary number of *Steps*, gathered in a list. A *Step* can be seen as a state in the execution of a *Setplay*. By convention, the first *Step* in a *Setplay* is always labelled with 0 as its *id*. The players participating in a *Setplay* will follow some, or all, of these *Steps* in order to accomplish the successful execution of the *Setplay*.

A *Step* has an *id*. To control the *Step*'s execution, *Wait time* is the amount of time the player should wait, after entering the *Step*, before starting the transition to another *Step*, or simply finishing the *Setplay*, and *abort time* is the threshold after which the players will abandon the *Setplay*, if it was not possible to progress from this *Step* to another. A *Step* also has a *Condition*, which must be satisfied before entering the *Step*. A list of *Player References*, in this scope called *participants*, identifies the players taking part in the *Step*.

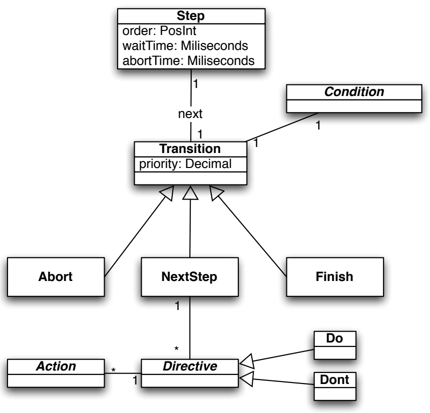


Fig. 2. Transitions between Steps.

There are several possible ways out of a *Step*, which are defined as *Transitions*, see fig. 2. All *Transitions* can have a

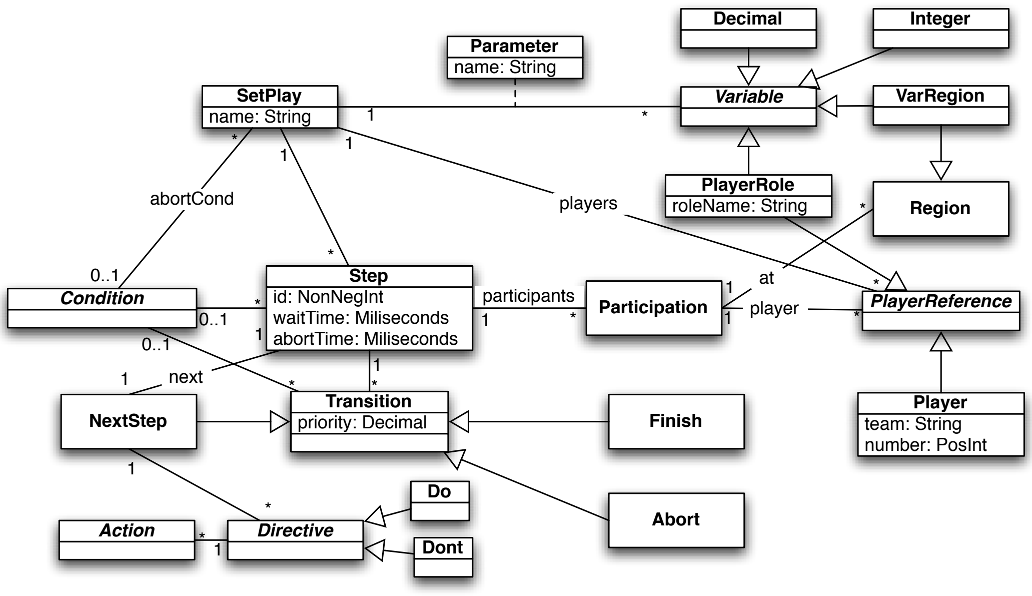


Fig. 1. Setplay definition

Condition, which must be satisfied for the *Transition* to be followed. An *Abort Transition* represents a situation where the *Setplay* must be abandoned, either because it is no longer judged useful, or it is thought that it will not reach its goal. The *Finish Transition* represents that the *Setplay* has reached its intended goal and should stop at this point. The main *Transition*, that is used to link between the different *Steps* is defined as *NextStep*. It includes the id of the next *Step* to be reached, and contains a list of *Directives* that will be applied in order to accomplish the *Transition*.

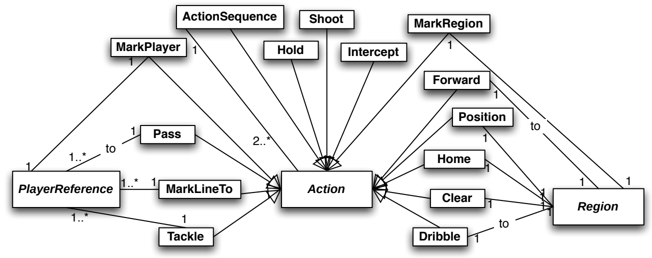


Fig. 3. Action definition

Directives include *Actions* and can be of two kinds: *Do* and *Don't*, meaning respectively that the contained *Actions* should, or should not, be executed. In this context, *Actions*, depicted in fig. 3, are high-level concepts that represent skills and moves, both simple and complex, that can be executed by a player. Examples of such *Actions* are passing the ball to a player or region, shooting at goal, intercepting the ball, or dribbling. In this *Setplay* framework, the *Action* concepts were inspired by the ones defined by Clang[8], the coaching language used in the simulation league.

The concept of *Condition*, already mentioned while introducing *Steps*, plays also a central role. Such *Conditions* have a wide field of application, and deal with the whole domain of robotic soccer. Similarly to the *Actions*, the majority of the *Conditions* in this framework were inspired in Clang. Examples of such *Conditions*, partially depicted in fig. 1, are players and ball positions, ball ownership and play mode. In this case, however, several new *Conditions* had to be introduced, in order to model complementary situations. Particularly, some *Conditions* refer to the possibility of accomplishing passes and shots, i.e., modelling the success of passes to players and regions, and shots at goal. One should pay special attention to this kind of *Conditions*: they are not based on a verifiable state-of-the world, but instead are an estimation of a success rate. This could be considered as intrinsically different from *Conditions* like player position, which are tangible and verifiable. Even these *Conditions* are, in the scope of robotic soccer, also somehow an estimation: the players do not know the real state-of-the-world, they simply have their own view, built from own observation and information shared by other team-mates. Therefore, for the sake of simplicity and expressiveness, all these concepts are indistinguishable considered *Conditions*.

Regions are another concept in the core of the definition of *Setplay*, and are depicted in the diagram in fig. 4. Once again, these concepts originate from Clang, including spatial entities like points, *Triangles*, *Arcs* and *Rectangles*. Similarly, the concept of *Dynamic Point*, referring to the location of a player or of the ball, is also introduced. Named regions are introduced to model intuitive locations like 'our mid-field' or 'their penalty box', as defined in [9].

A. Inter-robot communication

A relevant issue in the usage of the framework is how to achieve coordination between the robots when executing a *Setplay*. Naturally, a complex *Setplay* must follow several

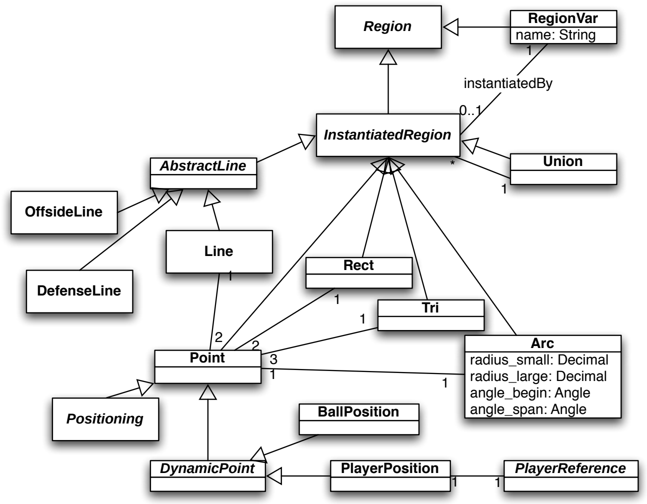


Fig. 4. Region definition

steps, and all participating players must be tightly synchronized in order to achieve fruitful cooperation. The first step towards this objective was to define a communication and synchronization policy, which should be as straightforward as possible, and can be seen in fig. 5.

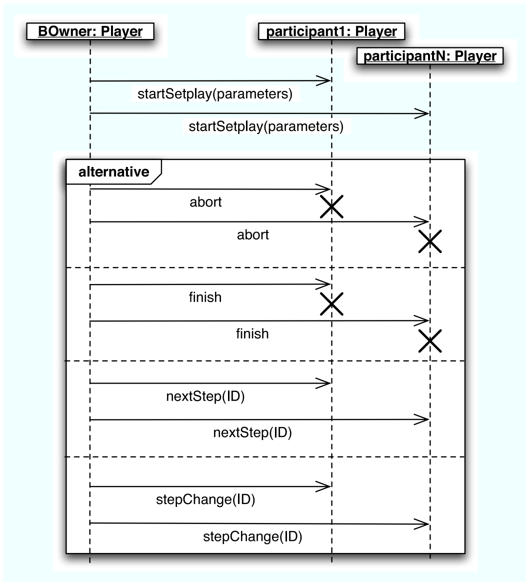


Fig. 5. Setplay interaction scheme

Each step will be led by the so-called lead player, who will normally either the player with ball possession, or a special agent (e.g. the *coach*) since it is the one which has to take the most important decisions, while manipulating the ball. This player, is naturally not fixed throughout the *Setplay*, and will change from step to step, while monitoring the execution of the *Setplay*, instructing the other players on *Setplay* begin, step entry and transition choice. The entry into a new step, which is decided by the lead player in charge of the previous step, normally implies the change of the lead player. The implementation of this communication policy is described in more detail in section IV.

B. Implementation as a C++ Library

The Setplay framework was, from the beginning, intended to be applicable to different teams and leagues: it should be possible to mix players with different originating teams in one single team, while executing Setplays, and, further, the framework should be applied in different leagues, as described in the present article.

Since the initial implementation and testing were conducted on top of the FCPortugal team, implemented using C++, it was decided to develop a C++ library, with two goals: ease the implementation in these teams, or any other implemented in C++, and maintain common framework that would be applied to any team.

This implementation intended to provide as much tools as possible, and therefore the following two features were developed:

- **Setplay definition parser** Since setplays can freely be defined using the model described in the last section, it was obviously necessary to develop a parser to load files and translate them to C++ objects. This parser was developed using the Spirit library included in the latest Boost [3] distributions.
- **Setplay execution engine** A Setplay execution is trivially determined by its definition. Therefore, the framework can provide an execution engine to be immediately used, out-of-the-box. This way, a team using the framework does not have to worry about the execution of a Setplay: it has to define the domain specific actions and conditions (see next section), and to launch the Setplay.

C. Framework usage

With the provided library and tools, each team wishing to use the framework only has to accomplish four tasks:

- **Setplay definition** Each team will use a different set of Setplays, according to their strategy and skills. These Setplays can be defined directly in the Setplay definition language, as described above, or use the graphical editor[2].
- **Implement Conditions** The conditions in the Setplay framework are league specific, and must be adapted to the teams' State-of-the-World implementation. Each Condition class has an abstract method to evaluate it that must be implemented.
- **Implement Actions** The actions in the Setplay framework are also league and team specific, and must be translated to actual actions or skills. Each Action class has an abstract method to execute it that must be implemented.
- **Deal with communication** The Setplay framework needs messages being exchanged between the players, in order to synchronize the execution of the Setplay. The framework makes this task quite simple: at each

[3]http://www.boost.org/

moment it is possible, through the invocation of methods, to know if there is a message to be sent and to access its content in plain text. There is also a method to interpret a received message. Thus, to deal with the communication issues, it suffices to check regularly if there is a message to be sent, sending it when appropriate, and, at the same time, report each received message to the framework for proper interpretation.

To actually use the Setplays, the team has to start the execution of the Setplay by instantiating its parameters, and regularly (i.e.: in every execution cycle) update the Setplay status through an update method, which must suply ball and players positions.

An initial prototype [10] of the implementation of the Setplay framework was applied to the Simulation 3D league, namely to the FCPortugal3D team[11], which won RoboCup 2006 in this league. This prototype was implemented on top of the 2006 simulator version, where the players were modeled as spheres. In the following year, the players were changed to Humanoids, with very complex dynamics, very slow movement and unsure skills. In such an environment, there is no use for high-level coordination: there are presently only three players on each team and development focus on low-level skills. The implementation on this league was thus abandoned and was not subject of real game testing.

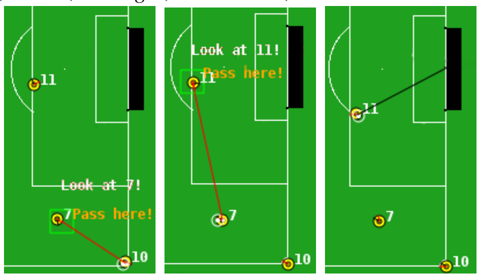
IV. Usage in the Simulation 2D League

As a primary test-bed for the *Setplays*, the code of the FCPortugal[12], which participates in RoboCup since 2000, was used. This code already had the main building blocks for the implementation of *Setplays*: a mature state-of-the-world, which considers both own observations and information shared by other players, and which includes prediction of actions' and interactions' effects; and a set of actions and skills that allows the easy mapping of actions as defined in the *Setplay* framework to concrete executions in the 2D simulator.

This implementation was achieved after following several steps. First, the Setplay usage scenario was chosen: in the current level of play, it was considered interesting to use Setplays in situations like corners, kick-ins and corners, since these situations are clearly announced by the server, and thus all players can prepare to participate in the Setplay. Secondly, *Conditions* and *Actions* as defined in the framework were implemented based on the existing code dealing with world-state, skills and action. Finally, the message exchange needed for the execution of setplays was implemented, as discussed in section IV-A.

A. Inter-robot communication

The major challenge in this implementation was how to deal with the limited communication means allowed by the server. To cope with the limited, single-channel communication, the lead player (i.e, the player with ball possession) will be the only player allowed to send messages. The content of communication must be as concise as possible, in order to follow the 2D simulator's limitations (messages under 10 bytes in length, only one message per team and cycle) and to leave enough place for the sharing of world-state information, necessary for the maintenance of a satisfactory and up-to-date world model by all players. In order to comply with these limitations, it was chosen to only use Setplays without arguments, since these would use much space in the startup messages (as explained in section III-A). The Setplay startup message does therefore only have the participating player numbers as arguments.



(a) (b) (c)

Fig. 6. Corner setplay execution steps.

B. Example Setplay

In this section, a simple example is presented, to illustrate the actual definition of *Setplays*, how the players in the 2D simulation league deal with it, and how inter-robot communication is deployed. In this case, the execution of the Setplay is shown without opponents, for clarity's sake.

A situation where a Setplay can be properly used is the corner-kick: it is an offensive situation close to the opponent goal and holes in the defense can be exploited. To keep this example clear, a simple situation, with only three participants and no opponents, will be described. The Setplay initiator triggers execution, after choosing the participating players from their distance to the positions in the setplay, by sending a instantiation message- In this case, the lead player is nr. 10 (taking role *cornerP*), and the two other participants nrs. 7 and 11 (roles *receiver* and *shooter*), thus the message sent is as follows:

```
S0 10 7 11
```

In step 0 of the Setplay, the participating players reach their positions, after which the *cornerP* tries to reach step 1, passing the ball to the *receiver*, as depicted in fig. 6(a). Upon gaining possession of the ball, *receiver* starts being the new lead player and therefore sends a message to the other players, informing them that the Setplay is currently in step 1, and that the *receiver* will try to reach step 2, as follows:

```
1 2
```

When the *receiver* verifies that it can make a pass to the *shooter*, it will do so, as depicted in fig. 6(b). In this figure, the *receiver* is looking at the *shooter* in order to accomplish a good pass, as it was the case in the precedent image.

Finally, as soon as it considers that shooting at goal is possible, the *shooter* executes the shot (see fig. 6(c)) and moves to step 3, which simply finishes the Setplay. At this

```
(setplay :name simpleCorner
 :players (list (playerRole :roleName cornerP)
     (playerRole :roleName receiver)
     (playerRole :roleName shooter))
 :steps
   (seq (step :id 0 :waitTime 15 :abortTime 70
       :participants (list (at cornerP (pt :x 52 :y 34))
           (at receiver (pt :x 40 :y 25))
           (at shooter (pt :x 36 :y 2)))
       :condition (playm ck_our) :leadPlayer cornerP
       :transitions (list
           (nextStep :id 1
             :condition (canPassPl :from cornerP
                                   :to receiver)
             :directives (list
               (do :players cornerP :actions
                      (bto :players receiver))
               (do :players receiver :actions
                      (intercept))))))
   (step :id 1 :waitTime 5 :abortTime 70
     :participants (list (at cornerP (pt :x 52 :y 34))
       (at receiver (pt :x 40 :y 25))
       (at shooter (pt :x 36 :y 2)))
     :condition (and (bowner :players receiver)
          (playm play_on))
     :leadPlayer receiver
     :transitions (list
       (nextStep :id 2
         :condition (canPassPl :from receiver :to shooter)
         :directives (list
           (do :players receiver
               :actions (bto :players shooter))
           (do :players shooter:actions (intercept))))))
   (step :id 2 :abortTime 70
     :participants (list (at cornerP (pt :x 52 :y 34))
        (at receiver (pt :x 40 :y 25))
        (at shooter (pt :x 36 :y 2)))
     :condition (and (bowner :players shooter)
                     (playm play_on))
     :leadPlayer shooter
     :transitions (list
        (finish :condition (canShoot :players shooter)
          :directives (do :players shooter
                          :actions (shoot)))))))
```

Fig. 7. Corner Setplay definition

moment, the *shooter* will send a message stating that it reached step 3, and that there is no further step in the Setplay:

```
3 -1
```

The described Setplay was defined in a configuration file, read upon player startup, as seen in fig. 7.

V. Future Work and Conclusions

The Setplays framework has shown to be flexible, since it allows the expression of very different plans, from a very simple kick-in to complex corners and ball exchanges in square in the simulation league. This flexibility also entails that the framework, and its underlying Setplay definition language, is abstract enough to deal with the whole of the robotic soccer domain.

In order to show the framework's generality, it will be applied to the middle-size RoboCup league, without any kind of change in its core. This is clearly a positive step towards a completely general setplay framework applicable to any RoboCup league.

Since the framework is presented as a stand-alone library, its usage is also quite simple: a new team wishing to use it only needs to define the domain specific concepts (actions and conditions on the State-of-the-World), and deal with Setplay selection and player and parameter choice. From this point on, it suffices to update the ball and players positions regularly to have Setplays executed.

Machine learning techniques will be applied on the selection of Setplays, considering the opponent and the game state. As a first approach, Case-based reasoning will be investigated with this goal in mind. On the long term, it also should be investigated how to extract new Setplays from real-game situations: in fact, if some team-play is successful, it could be possible to analyse logs, or images, and extract a Setplay definition for future usage.

VI. Acknowledgments

This work was partially supported by the Portuguese National Foundation for Science and Technology FCT-PTDC/EIA/70695/2006 Project - "ACORD: Adaptive Coordination of Robot Teams".

References

[1] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, E. Osawa, and H. Matsubara, "Robocup: A challenge problem for AI," *AI Magazine*, vol. 18(1), pp. 73–85, 1997.

[2] R. Lopes, "Coordination methodologies applied to robocup: a graphical definition of setplays," Master's thesis, Faculdade de Engenharia da Universidade do Porto, 2009.

[3] I. Noda, H. Matsubara, K. Hiraki, and I. Frank, "Soccer server: A tool for research on multiagent systems," *Applied Artificial Intelligence*, vol. 12, no. 2–3, pp. 233–250, 1998.

[4] P. Stone and M. Veloso, "Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork," *Artificial Intelligence*, vol. 110, no. 2, pp. 241–273, 1999.

[5] O. Zweigle, R. Lafrenz, T. Buchheim, U.-P. Käppeler, H. Rajaie, F. Schreiber, and P. Levi, "Cooperative agent behavior based on special interaction nets," in *Intelligent Autonomous Systems 9: IAS-9*, 2006.

[6] T. Röfer, J. Brose, E. Carls, J. Carstens, D. Goehring, M. Juengel, T. Laue, T. Oberlies, S. Oesau, M. Risler, M. Spranger, C. Werner, and J. Zimmer, "Germanteam 2006 the german national robocup team," in *Robocup 2006 Symposium*. Deutsches Forschungszentrum fuer Kuenstliche Intelligenz, Universitaet Darmstadt, Universitaet Bremen and Humboldt-Universitaet zu Berlin, 2006.

[7] M. Risler and O. von Stryk, "Formal behavior specification of multi-robot systems using hierarchical state machines in XABSL," in *AAMAS08-Workshop on Formal Models and Methods for Multi-Robot Systems*, Estoril, Portugal, 2008.

[8] M. Chen, E. Foroughi, F. Heintz, S. Kapetanakis, K. Kostiadis, J. Kummeneje, I. Noda, O. Obst, P. Riley, T. Steffens, Y. Wang, and X. Yin, *Users manual: RoboCup soccer server manual for soccer server version 7.07 and later*, 2003. [Online]. Available: http://sourceforge.net/projects/sserver/

[9] L. P. Reis and N. Lau, "Coach unilang - a standard language for coaching a (robo) soccer team," in *RoboCup-2001: Robot Soccer World Cup V*, ser. LNAI. Springer Verlag, 2002, vol. 2377, pp. 183–192.

[10] L. Mota and L. P. Reis, "Setplays: Achieving coordination by the appropriate use of arbitrary pre-defined flexible plans and inter-robot communication," in *First International Conference on Robot Communication and Coordination (ROBOCOMM 2007)*, ser. ACM International Conference Proceeding Series, A. F. T. Winfield and J. Redi, Eds., vol. 318. IEEE, 2007.

[11] N. Lau and L. P. Reis, "Coordination methodologies developed for FC Portugal 3D 2006 team," in *10th Robocup 2006 Symposium CD*, Bremen, Germany, 2006.

[12] L. P. Reis and N. Lau, "FC Portugal team description: Robocup 2000 simulation league champion," in *RoboCup-2000: Robot Soccer World Cup IV*, ser. LNAI, P. Stone, T. Balch, and G. Kraetzschmar, Eds., vol. 2019. Springer, 2000, pp. 29–40.