Setplays: Achieving Coordination by the appropriate Use of arbitrary Pre-defined Flexible Plans and Inter-Robot Communication

Luís Mota[*†], Luís Paulo Reis[†‡]
[*]Instituto Superior de Ciências do Trabalho e da Empresa (ISCTE), Lisboa, Portugal
Email: luis.mota@iscte.pt
[†]Laboratório de Inteligência Artificial e Ciência de Computadores (LIACC)
[‡]Faculdade de Engenharia da Universidade do Porto (FEUP), Porto, Portugal Email: lpreis@fe.up.pt

Abstract—**Multi-agent coordination and strategic planning are two of the major research topics in the context of RoboCup. However, innovations in these areas are often developed and applied to only one domain and a single RoboCup league, without proper generalization. Also, although the importance of the concept of Setplay, to structure the team's behaviour, has been recognized by many researchers, no general framework for the development and execution of generic Setplays has been presented 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 in a standard, league-independent and flexible language that defines setplays which may be interpreted and executed at run-time through the use of inter-robot communication. The implementation of this framework in the 3D simulation league is also described with concrete examples of Setplay definition, management and execution. The first results achieved show the usefulness of this approach and motivate us to use it as the main coordination of all our teams participating in the simulation, small-size, middle-size and legged leagues of RoboCup.**

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.

Robot Soccer needs, as the research in the domain develops, co-ordination 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 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 can nonetheless be a useful tool for high-level co-ordination and cooperation.

One additional motivation for high level planning has recently been raised in the Middle size league: the Technical Comittee has decided that some of the teams will have to join efforts to pairwise create common teams, in order to decrease the total number of participating teams. This means that the challenge of building multi-partner teams has to be dealt with in the near feature, in order to allow these teams to enter the competition. One possible way to integrate heterogeneous players would be to present them a list of Setplays, understandable to all, and simply tell them when and how each of these setplays should be executed. A full scenario of such a mixed team integration has already been presented in [2].

II. Motivation and Requirements

The 5dpo[2] mid-size team has its' collective behaviour organised around so-called *Setplays* [3]. Such *Setplays* are dynamically used in particular situations during the game. An example is used for the execution of free-kicks near the opposite goal: three players are involved. The *Initiator* is responsible to start the *Setplay* by passing the ball to the *Kicker*, which is located half a meter to the right of the ball. A third player, the *Supporter*, will be located behind the ball, in order to deal with lost or stray balls. The *Setplay* starts when the *Initiator* passes the ball to the *Kicker*, which tries to shoot at goal. This *Setplay* was fine-tuned and became a very successful collective move.

The concept of Setplay, as described, is used to structure the team's behaviour. But what if this kind of collective move could be described and shared in a standard, league-independent and flexible way, which would be interpreted and executed at run-time? The first benefit would be the possibility of writing arbitrary Setplays, which would dynamically be used during the game, opening horizons to new plays which could, for instance, differ from game to game, to better deal

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

[2]URI: http://paginas.fe.up.pt/∼robosoc/

ROBOCOMM 2007, 14th – 16th Oct 2007, Athens, Greece.
Copyright © 2011 – 2012 ICST
ISBN 978-963-9799-08-0
DOI 10.4108/ICST.ROBOCOMM2007.2209

with each opponents' characteristics. In that case, the Setplays could also be used in different leagues. Furthermore, since any player could have access to the definition of Setplays and interpret their content, Setplays could 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 co-operate.

To fulfil these requirements, one would need a standard language, where Setplays could be defined and interpreted by any player in any league. The basic concepts of soccer (moves, conditions, actions, skills) would 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 paper.

The framework that models this language is presented in section III. In section IV, synchronisation and communication issues are discussed, and a interaction policy is proposed.The implementation of this framework in the 3D simulation server [4] is described in section V. A concrete example of a Setplay and its' execution is the subject of section VI. Related work is presented and discussed in section VII. Finally, conclusions are drawn, and future lines of research are presented in section VIII.

III. The Setplay Framework

The Setplay framework was designed with the goal of being general, flexible, parameterizeable and applicable to any robotic soccer league. Its' general structure is shown schematically in Figure 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 blocks 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*, which is a non-negative integer. In order to control the *Step*'s execution, the concepts of *wait time* and *abort time* are introduced. *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*. The *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 one. A *Step* also has a *Condition*, which must be

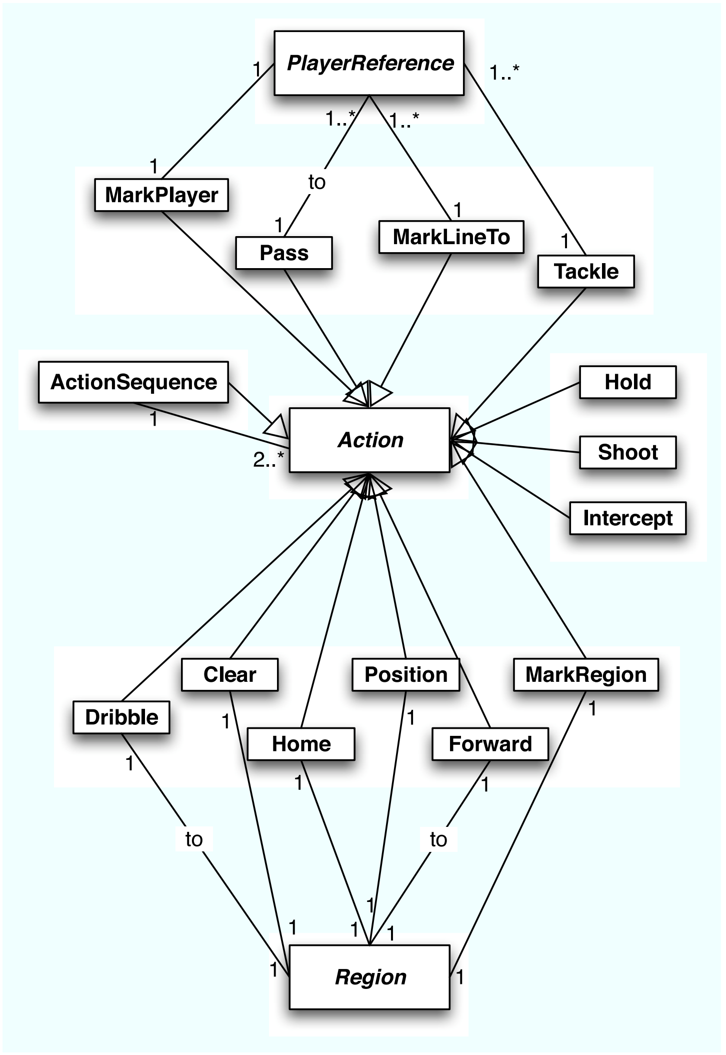

Fig. 2. Action definition

satisfied before entering the *Step*. A list of *Player References*, in this scope called *participants*, identifies the players taking part in the *Step*.

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

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 Figure 2, are high-level concepts that represent skills and

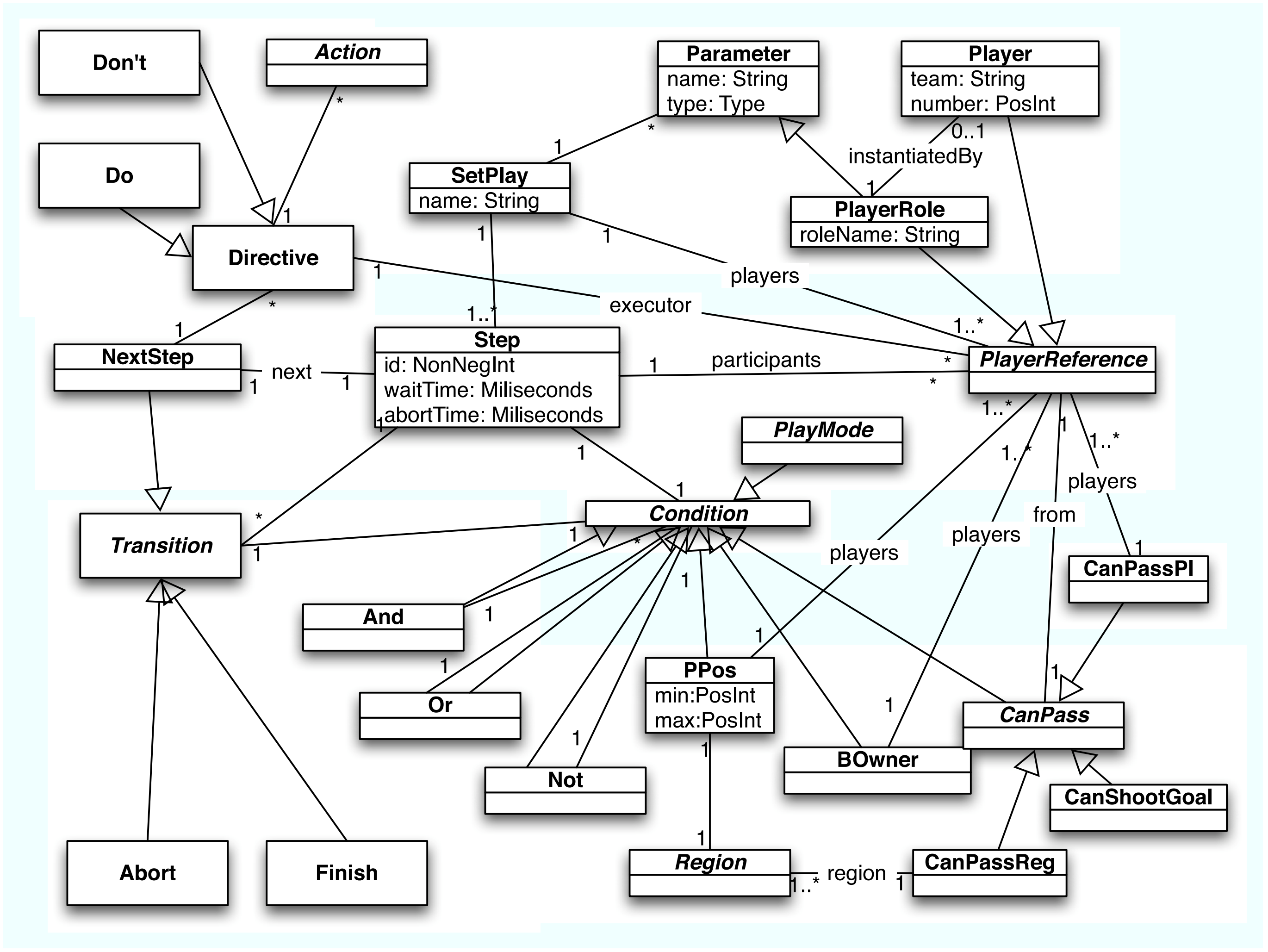

Fig. 1. Setplay definition

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 [5], the coaching language used in the simulation league. There is, however, one added *Action* which is absent from this language: the concept of *Action Sequence*, where several actions are to be executed following a particular order.

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 Figure 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 indistinguishably considered *Conditions*.

Regions are another concept in the core of the definition of *Setplay*, and are depicted in the diagram in Figure 3. 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 [6].

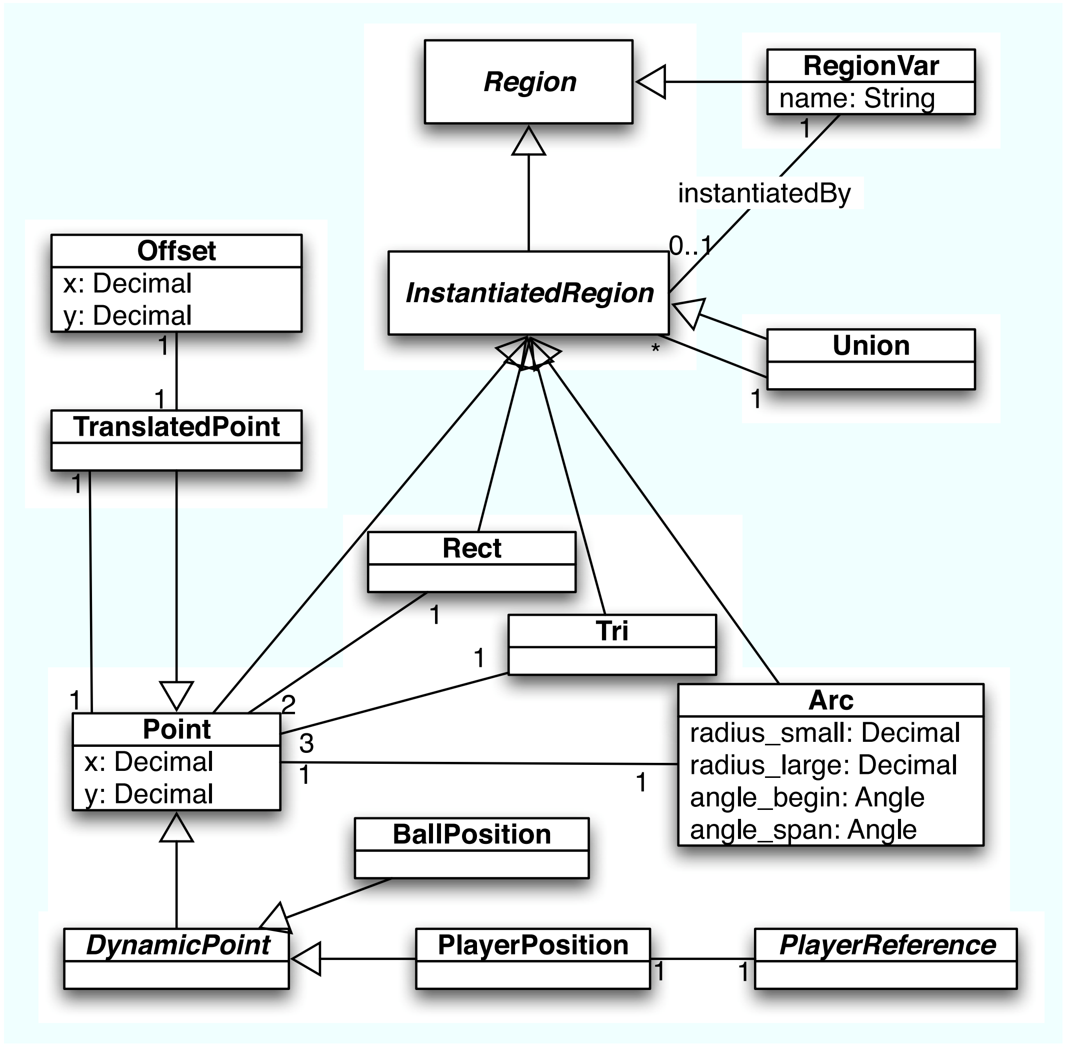

Fig. 3. Region definition

IV. Inter-robot communication

The major issue in this framework was how to achieve co-ordination between the robots when executing a *Setplay*. Naturally, a complex *Setplay* must follow several steps, and all participating players must be tightly synchronised in order to achieve fruitful co-operation. The first step towards this objective was to define a communication and synchronisation policy, which should be as straightforward as possible, and can be seen in Figure 4.

Each step will be led by the so-called lead player, who has ball possession, 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 V.

V. Implementation in the 3D simulation league

As a primary testbed for the *Setplays*, the code of the 2006 champion in the 3D simulation league, FC Portugal [7], 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 3D simulator.

This implementation was achieved after following several steps. First, a parser for *Setplay* definitions was implemented, in order to allow the players to read any *Setplay* defined in the standard language. Secondly, *Conditions* and *Actions* were implemented based on the existing code dealing with world-state, skills and action. Finally, the dynamics of the *Setplays*' steps and transitions were implemented.

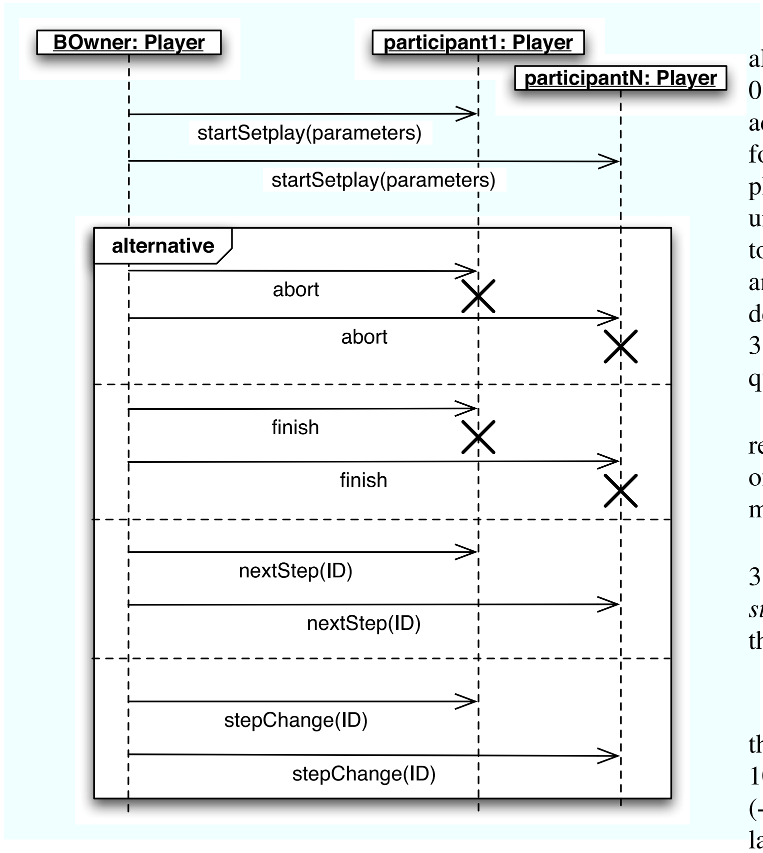

Fig. 4. Setplay interaction scheme

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 3D simulator's limitations 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. Examples and details of the transmitted information are presented in the next section.

VI. Example Setplay

In this section, a simple example is presented, to illustrate the actual definition of *Setplays*, how the players in the 3D simulation league deal with it, and how inter-robot communication is deployed. To emphasise the execution details, this *Setplay* is not an actual game situation, but instead takes place in a scenario without opponents. Thus, the players, identified by roles *P1*, *P2* and *P3*, are laid out in the vertices of a triangle, defined by the point parameters *P1Pos*, *P2Pos* and *P3Pos*. To introduce some tolerance to the players positions, another parameter, *PosTolerance* is defined, and will be used to define circles around the players' intended positions. The execution of the *Setplay* consists of the players simply passing the ball among them.

The formal definition of this *Setplay*, which is known by all players, consists of 4 different steps, as follows: in step 0, the players simply move to their assigned positions, and, additionally, *P1*, which has ball possession, is also responsible for carrying the ball to its own position. In steps 1, 2, and 3, the players P1, P2 and P3 have, respectively and in turn, the ball under possession, being responsible for choosing the transition to be followed, i.e., passing the ball to one of the other players, and communicating this decision to the other participants. The definition of this *Setplay* is included in figure 5. Steps 2 and 3 were omitted due to space limitations and because they are quite similar to step 1.

One application of *Triang* was conducted, its' results recorded, and will be used to illustrate the general execution of Setplays. The terms in Figure 4 will be used to identify the messages sent.

The *Setplay* was was started by initiative of player number 3. After deciding to start, it broadcasts a message of type *startSetplay* with the data necessary for the instantiation of the *Setplay*, as follows:

```
Triang -10.0 -17.7 -17.7 2 3 2 4
```

This message identifies the *Setplay* by its' name. Next, the three *Point* parameters are included: *P1Pos* will be the *Point* (-10,0), and *P2Pos* and *P3Pos* will, respectively, have the values (-17,7) and (-17,-7). *PosTolerance* will have 2 as its' value. The last three arguments indentify the participants in the *Setplay*: players 3, 2 and 4. After instantiating the *Setplay*, all players follow the only *Directive* in *Step* 0 and move to their assigned positions, as seen in Figure 6(a). The lead player will also move the ball to his position. When the players have reached their positions, the lead player, which has ball possession and is identified by its' first position in the *Participants* list, decides to make the *Transition* to *Step* 1. At this moment, it will issue a *stepChange* message, identifying the new *Step*:

```
1
```

Next, the lead player, which remains *P1* in *Step* 1, chooses one of the available *Transitions*, in this particular texecution to *Step* 2, and broadcasts a new *nextStep* message, which, with the purpose of overcoming communication failures, also includes the present *Step* number in the beginning. The character 'A' is used as a separator between the two *Step* ids:

```
1A2
```

Player *P1* then proceeds to executing the transition, by forwarding the ball to *P2Pos* (see Figure 6(b)), and then moves back to its' position. After finishing the transition, P1 again informs the other players, through a new *stepChange* message, that the Step has changed to Step 2. Upon reception of this last message, *P2* takes the role of lead player and will therefore be responsible for the choice of *Transitions* and their communication to other players. Screenshots of subsequent *Transitions* between *Steps* 2 and 3, 3 and 2, 2 and 1 and, finally, 1 and 3 can be seen in Figure 6(b-f). As it can be observed in these images, the *Setplay* is smoothly executed from step to step. In this particular *Setplay* no end transition is included, so it will eventually be executed forever.

```
(setplay :name Triang
   :parameters
      (list (parameter :name P1Pos :type point)
         (parameter :name P2Pos :type point)
         (parameter :name P3Pos :type point)
         (parameter :name PosTolerance :type decimal))
   :players (list (playerRole :roleName P1)
         (playerRole :roleName P2) (playerRole :roleName P3))
   :steps (list (step :id 0 :waitTime 0 :abortTime 30
         :participants (list P1 P2 P3)
            :condition (and (playm play_on) (bowner :players P1))
            :transitions (list (nextStep :id 1
                     :directives (list
                        (do :executors P1 (seq (fwd :region P1Pos) (pos :region P1Pos)))
                        (do :executors P2 (pos :region P2Pos))
                        (do :executors P3 (pos :region P3Pos))))))
         (step :id 1 :waitTime 20 :abortTime 30
            :participants (list P1 P2 P3)
            :condition (and (playm play_on)
                  (ppos :players P1
                   :region (arc :center P1Pos :radius_large PosTolerance))
                  (ppos :players P2
                   :region (arc :center P2Pos :radius_large PosTolerance))
                  (ppos :players P3
                   :region (arc :center P3Pos :radius_large PosTolerance))
                  (bowner :players P1))
            :transitions (list (nextStep :id 2
                     :condition (canPassReg :from P1 :region P2Pos)
                     :directives (list
                        (do :executors P1 (seq (fwd :region P2Pos) (pos :region P1Pos)))
                        (do :executors P2 (pos :region P2Pos))
                        (do :executors P3 (pos :region P3Pos))))
                  (nextStep :id 3
                     :condition (canPassReg :from P1 :region P3Pos)
                     :directives (list
                        (do :executors P1 (seq (fwd :region P3Pos) (pos :region P1Pos)))
                        (do :executors P2 (pos :region P2Pos))
                        (do :executors P3 (pos :region P3Pos))))))
         (step :id 2 (...))
         (step :id 3 (...))))
```

Fig. 5. Triangle setplay definition

VII. Related Work

A strategy for role assignment in the four-legged league was introduced by [8]. This strategy implies the communication of the currently chosen *Play*, which provides a set of *Roles* to be assigned to all the available players in the team. The strategy assures co-ordination by the existence of a leader that selects the best momentary *Play* and instructs the other robots on what *Roles* to take. Each *Role* fully determines the player's behaviour. The strategy does not, however, define a concept of *Setplay* with intermediary states, and *Plays* do not have *Parameters*.

The concept of *Setplay* is present in a teamwork and communication strategy for the 2D simulation league, presented by [9]. 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.

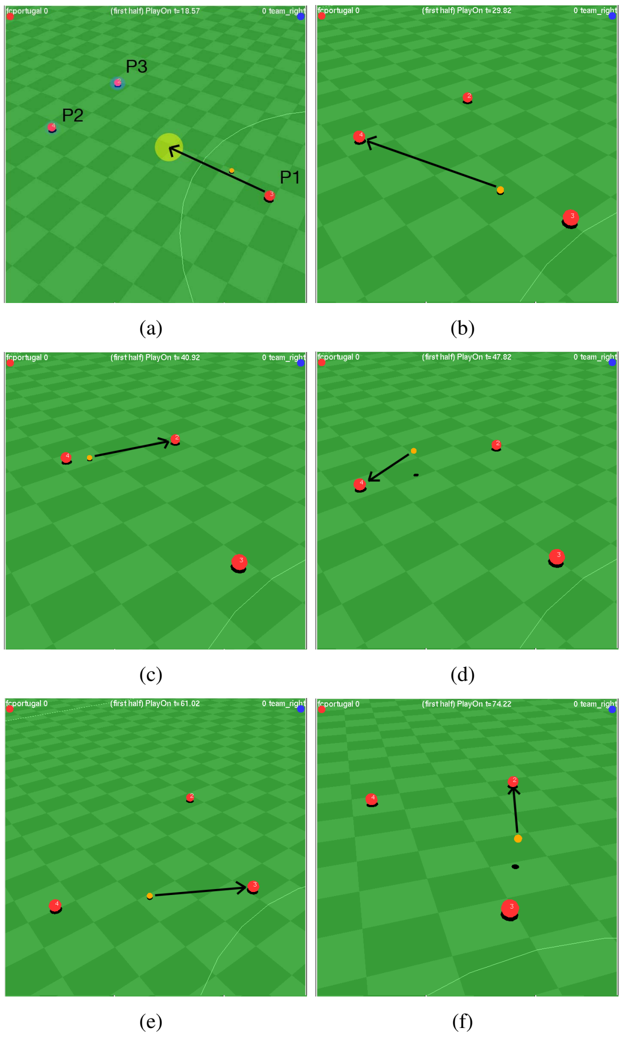

Fig. 6. Setplay example

VIII. Conclusions and Future Development

Since the presented framework is, to the best of our knowledge, an innovative contribution to the domain of co-operation in robotic soccer, and agent systems in general, it opens a wide range of new research opportunities. A league-independent language for arbitrary setplay definition is presented, which can be used by any team (including mixed teams), relying on communication to manage setplay execution and player synchronisation.

The first task will be to fully integrate the *Setplays* in the top-level behaviour of the FCPortugal 2D and 3D simulation teams. This integration will need a strategy for *Setplay* activation and instantiation, which will initially be made using Case-based Reasoning. It seems clear that this is a good scenario for research on this kind of reasoning. Since there is also a strong connection to the 5dpo mid-size team, *Setplays* will also be implemented in this league and used to enhance the team's current use of *Setplays*.

At present stage of development, all *Transitions* are considered equal and chosen randomly. This can be changed through the consideration of weights, that will label a *Transition* as more or less desirable. Such weights will surely be useful, since there are situations where one way is preferable to another.

Further, since the players knowledge of the world is imperfect and liable to uncertainties, *Conditions* could be considered fuzzy and take benefit of the usage of thresholds of confidence. This is surely suited to the robotic soccer domain, and will be a subject of research in the near future.

Acknowledgment

The authors would like to thank Nuno Lau, also from FCPortugal, for valuable inputs on the *Setplay* definition language.

This work was partially supported by a grant from Fundação para a Ciência e a Tecnologia from the portuguese government.

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] L. Mota, L. P. Reis, and H.-D. Burkhard, “Communication challenges raised by open co-operative teams in RoboCup,” in *Encontro Científico do Festival Nacional de Robótica*, 2006.

[3] A. S. Conceição, A. P. Moreira, L. P. Reis, and P. J. Costa, “Architecture of cooperation for multi-robot systems,” in *First IFAC Workshop on Multivehicle Systems (MVS'06)*, 2006.

[4] M. Kögler and O. Obst, “Simulation league: The next generation,” in *RoboCup 2003: Robot Soccer World Cup VII*, ser. Lecture Notes in Artificial Intelligence, D. Polani, A. Bonarini, B. Browning, and K. Yoshida, Eds. Berlin, Heidelberg, New York: Springer, 2004, vol. 3020, pp. 458–469.

[5] 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/

[6] 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. Lecture Notes in Artificial Intelligence. Springer Verlag, 2002, vol. 2377, pp. 183–192.

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

[8] C. McMillen and M. Veloso, “Distributed, play-based role assignment for robot teams in dynamic environments,” in *8th International Symposium on Distributed Autonomous Robotic Systems (DARS 2006)*, 2006.

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