Mechatronics xxx (2010) xxx–xxx

Contents lists available at ScienceDirect

Mechatronics

journal homepage: www.elsevier.com/locate/mechatronics

Multi-robot coordination using Setplays in the middle-size and simulation leagues

Luís Mota [a,b,*], Luís Paulo Reis [b,c], Nuno Lau [d,e]

[a] Instituto Universitário de Lisboa (ISCTE-IUL), Lisboa, Portugal
[b] Laboratório de Inteligência Artificial e Ciência de Computadores (LIACC) da Universidade do Porto, Porto, Portugal
[c] Departamento de Engenharia Informática (DEI/FEUP), Faculdade de Engenharia da Universidade do Porto, Porto, Portugal
[d] Instituto de Engenharia Electrónica e Telemática de Aveiro (IEETA), Portugal
[e] Departamento de Electrónica e Telecomunicações da (DETUA), Universidade de Aveiro, Portugal

ARTICLE INFO

Article history:
Available online xxxx

Keywords:
Multi-agent cooperation
Setplay
Middle-size league

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 was 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 both in the 2D and 3D leagues. This framework was also recently implemented in the middle-size team CAMBADA. This team has, in the recent past and with previous versions of the control software, ranked first and third in RoboCup's 2008 and 2009 editions. The implementation is described with concrete examples of Setplay definition and execution, which shows the usefulness of this approach and motivate its use as a major coordination tool for teams participating in the simulation, small-size, middle-size, standard platform and humanoid leagues of RoboCup.

1. Introduction

RoboCup[1] [4] 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, since they allow the definition of how different players interact in key situations. In the sense employed in this article, a Setplay is a freely-definable, flexible and multi-step plan, which allows alternative execution paths, involving a variable number of robots.

1.1. Motivation and requirements

The CAMBADA team[2] has, in recent years, obtained very good results in the RoboCup competitions. The final ranks in the editions

* Corresponding author at: Instituto Universitário de Lisboa (ISCTE-IUL), Lisboa, Portugal.
E-mail address: luis.mota@iscte.pt (L. Mota).

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

[2] http://www.ieeta.pt/atri/cambada/

0957-4158/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.mechatronics.2010.05.005

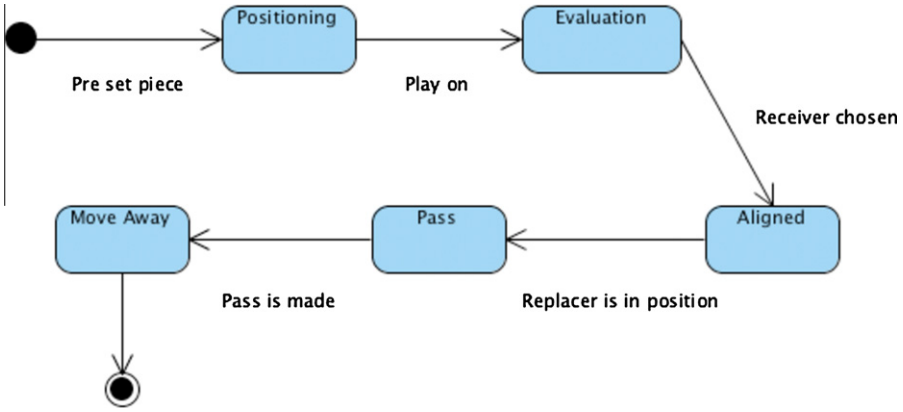

Fig. 1. Replacer's state machine, from [23].

since 2007 have been 5th, 1st and 3rd. These achievements are a result of a continuous effort to enhance the team performance. Namely, the team performance was highly improved when playing set pieces, i.e., throw-ins, free-kicks, goal-kicks and other similar situations decided by the referee. In recent years, rule changes were made in order to encourage team-play [14]. With the goal of avoiding that a single player dribbles around the field before eventually shooting at goal, set-pieces must include a pass, between the two players that first touch the ball, and these must have at least one meters between them.

Set pieces [7] are a very important part of CAMBADA game. For example, in the 2008 Robocup final, four of the seven goals scored resulted directly from a set piece. In a typical set piece scenario, there are three robots directly involved, while the rest are in strategic positions. A typical set piece, as it has been used in CAMBADA in recent years will be now described, to better motivate the subsequent development of the Setplay framework.

Set pieces are based in two roles: *Replacer*, the robot that makes the pass, and *Receiver*, the robot that will receive the ball and then go on playing, shooting at the goal. In each set piece there will be one *Replacer* and two *Receivers*.

The *Replacer* role (see the state machine in Fig. 1) is the one responsible for giving the first touch to the ball in every set piece. This role is the one in charge of beginning the set piece. It will position itself close to the ball (state "*Positioning*"), decide which *Receiver* to pass to (state "*Evaluation*"), align with the chosen *Receiver* (state "*Aligned*") and, after passing (state "*Pass*"), it will move away.

The role *Receiver* (see state machine in Fig. 2) is responsible for the positioning of two robots in the set pieces. One of those robots will receive a pass from the *Replacer*, as described before, after correctly positioning itself (state "*Positioning*"). It will then receive the ball (state "*Receive*") and subsequently kick the it into the opponent goal (state "*Kick*").

In this previous approach, the strategy for set pieces can be parameterized in a configuration file. In that file there is information about the positioning of the *Receivers* for every set piece and to whom the *Replacer* should pass first. The configuration file is written in XML and contains many informations, such as field dimensions and other parameters [3]. However, the configuration of set-pieces is limited: the general execution is hard-coded, and only the positioning of the participating roles can be changed. In the team that played in RoboCup 2009 there was no way of defining a new Setplay, like, e.g., the *Replacer* passing to a *ReceiverA*, which would in turn pass to a *ReceiverB*, which would eventually shoot at goal.

Since the skills of robots are continuously improving, and opponents try to react to new developments and counter them, it would be very useful for the CAMBADA team if there would be a new way of freely defining arbitrary Setplays, through configuration files or

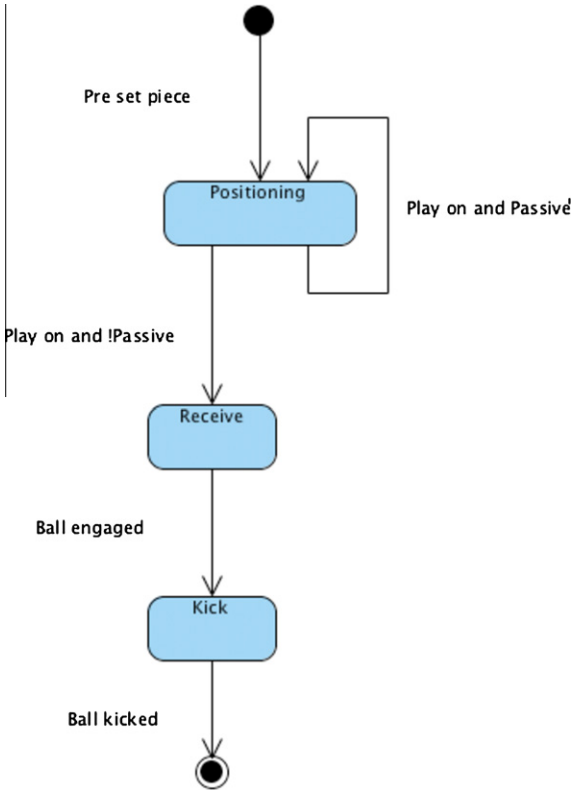

Fig. 2. Receiver's state machine, from [23].

even a generic Setplay graphical editor like the one already developed for Setplay and formations definition in the FC Portugal team [11].

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

Please cite this article in press as: Mota L et al. Multi-robot coordination using Setplays in the middle-size and simulation leagues. Mechatronics (2010), doi:10.1016/j.mechatronics.2010.05.005

To fulfil these requirements, one needs a new standard language, where Setplays can be defined and interpreted by 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 the remainder of this article.

Further, a new 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 3.2.

1.2. 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 2. The framework that models this language is presented in Section 3, including its distribution as a C++ library. The implementation of a full-blown prototype of usage of the framework in the 2D simulation server [15] was a major step in the testing of this framework, and is described in Section 4. This implementation was done on top of the code of the FCPortugal team. [3] The main focus of this article, the framework implementation in the middle-size team CAMBADA is presented in Section 5. Finally, conclusions are drawn, and future lines of research are presented in Section 6.

2. State of the art

2.1. Coordination in the middle-size league

The CAMBADA coordination [6] is based in Situation Based Strategic Positioning (SBSP) [18,16] strategies used in RoboCup 2D simulation league, adapted to the MSL specifications. For role assignment a dynamic algorithm that adapts the formation to a possible varying number of active robots, is used, which will assign each role/robot to the strategic positionings according to priorities and number of active robots [7]. CAMBADA has in recent years also been using simple Setplays in key situations like corners, kick-offs and free-kicks. Such Setplays can have their parameters configured, but cannot be freely defined: their execution structure is hard-coded.

TechUnited [1] team has recently switched from static to dynamic assignment of roles, which is done centrally by a dedicated module, based on world-state, namely player and ball positions.

The Brainstormers Tribots [5] have their roles, and the team formation, decided centrally by a dedicated, high-level module. The closest robot to the ball acts as a master and decides which play to use. Such plays are managed through communication and managed through a master player, but it is unclear how they are defined [10].

RFC Stuttgart, formerly CoPS [26], use dynamic role assignment, with sub-roles, e.g., role Defender and sub-role Left or Right. The role allocation is done locally by every robot, based on the shared world model, that integrates information from all robots. This being the case, inconsistencies are minimised, since all robots decide based on the same information. Role allocation will be potentially wrong only when the shared world model is inconsistent.

Coordination in this league is thus mainly structured around positional and role-allocation techniques. Setplay based cooperation is beginning to be used in specific situations, by teams CAMBADA and Brainstormers Tribots, but only using limited and rigid techniques.

2.2. Coordination through positioning

A method to achieve coordination based on repulsions and attractions called *Strategic Positioning by Attraction and Repulsion* (SPAR) was introduced by [24]. When an agent is positioning itself using SPAR, the agent maximizes the distance to other players and minimizes the distance to ball and to goal. This is achieved evaluating several forces: repulsion from opponents and team-mates, attraction to active team mate, ball and opponent goal. It also uses other constraints that have influence in agent's positioning: stay in an area near home position, stay within the field boundaries, avoid being at an offside position and stay in a position where is possible to receive a pass.

Later the *Situation Based Strategic Positioning* (SBSP) was introduced by [18,9]. If an agent is not involved, and will not be soon, in an active situation it will try to occupy its strategic position relative to the actual situation of the game, which can dynamically change when the situation changes. The dynamic changes of one player will influence the decision of others, which will avoid that several players take the same strategic position. Through the analysis of the tactic, formation, self positioning in the formation and player type, a player is able to define its base strategic positioning. This position is then adjusted accordingly to ball position and velocity and situation (i.e. attack, defence, etc.). The player type defines the player's strategic characteristics like ball attraction, admissible regions in the field, specific positional characteristics for some regions in the field, tendency to stay behind the ball, alignment in the offside line, and attraction by specific points in the field in some situations. Using a strategic positioning like SBSP the players will be more well distributed over the field than using a active one like SPAR, this is the reason why other teams adopted SBSP as the standard positioning method.

2.3. Coordination through plays

In this section, a Play is considered either a role allocation algorithm, or a small plan, involving all or part of the players in a team, which defines their positioning and/or choice of actions. A Role is a particular participation of a robot in a Play, which may include Parametres, that are variables that influence the Play's definition and execution.

The concept of *Setplay* is present in a teamwork and communication strategy for the 2D simulation league, presented by [25]. 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.

A strategy for role assignment in the now defunct four-legged league was introduced by [12]. 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 coordination 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*.

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

Please cite this article in press as: Mota L et al. Multi-robot coordination using Setplays in the middle-size and simulation leagues. Mechatronics (2010), doi:10.1016/j.mechatronics.2010.05.005

Also in the context of the four-legged league, [22] introduced a Case-based-reasoning inspired system, relying on the concept of *Play*. *Plays* are a more limited concept than Setplays, since they merely aim at distributing roles among all the teams players. It is therefore more of a coordination methodology than cooperation with actual plans, as is the case with Setplays.

The RFC Stuttgart/CoPS team uses Special Interaction Nets [27], 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 all 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.

The Extensible Agent Behaviour Specification Language (XABSL) is a language to describe behaviours for autonomous agents based on hierarchical finite state machines, and has been used by different teams in RoboCup, namely the German Team [21]. Recent developments [20] 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. This kind of cooperation does, though, not easily allow the incorporation of communication [19]. Also, the programming of cooperative behaviour requires the definition of *options*, through a dedicated language.

The described approaches that are more than role allocation algorithms are limited in some aspects: either they do not define a standard vocabulary, they do not allow real-time definition of parameters, they can be started only in particular situations, they do not allow alternatives in the definition of Setplays or they do not allow the quick prototyping of new Setplays through easily editable files. There is, thus, place for large improvements.

3. Setplay framework

As stated in Section 1, a Setplay is a freely-definable, flexible and multi-step plan, which allows alternative execution paths, involving a variable number of robots. 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. 3.

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*. An *abortCond* exists to define conditions that, if satisfied, will make the *Setplay* at any moment in its' execution.

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*, 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 satisfied before entering the *Step*. A list of *Participations*, in this scope called *participants*, identify the players taking part in the *Step*, optionally also defining their positioning.

There are several possible ways out of a *Step*, which are defined as *Transitions*, see Fig. 4. 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*

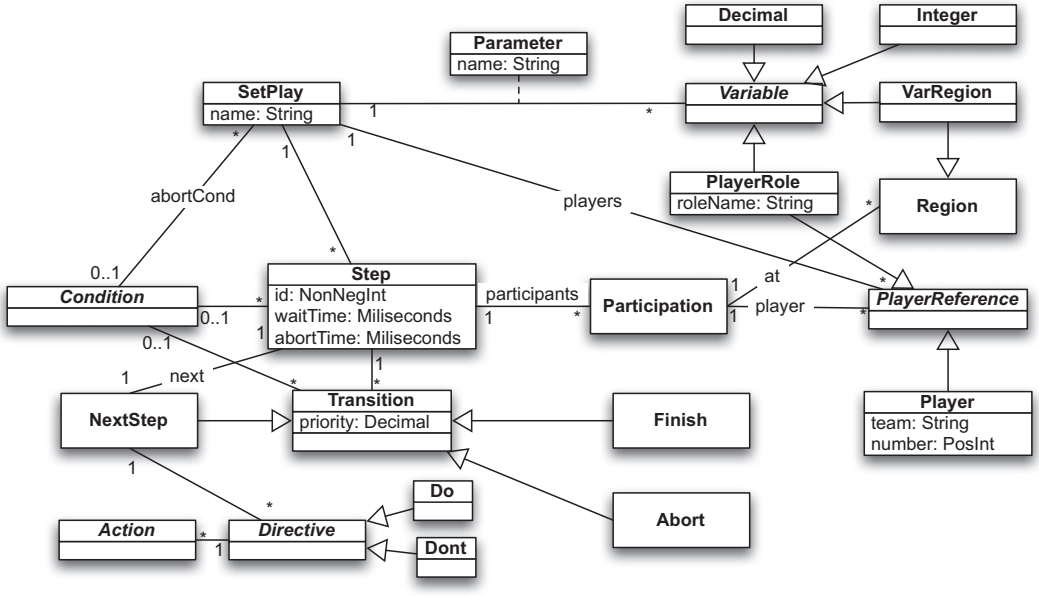

Fig. 3. Setplay definition.

Please cite this article in press as: Mota L et al. Multi-robot coordination using Setplays in the middle-size and simulation leagues. Mechatronics (2010), doi:10.1016/j.mechatronics.2010.05.005

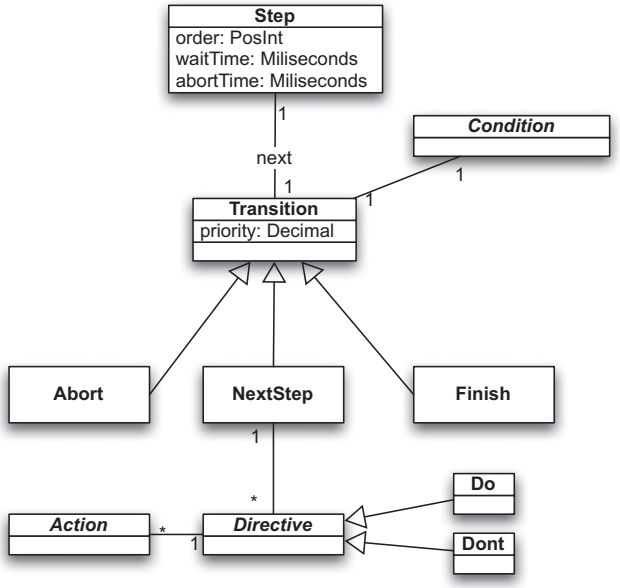

Fig. 4. Transitions between Steps.

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 connect players to *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. 5, are abstract, 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 [2], 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.

Conditions model high-level characteristics of the State of the World, specifically modelling the domain of robotic soccer. Examples of such *Conditions*, depicted in Fig. 6, are players and ball positions, ball ownership and play-mode. Similarly to the *Actions*, the majority of the *Conditions* in this framework were inspired in Clang. 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. 7. 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 [17].

3.1. 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 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 straight-forward as possible, and can be seen in Fig. 8.

Each step will be led by the so-called lead player, who will normally be either the player with ball possession (see Section 4.1), since it is the one which has to take the most important decisions, while manipulating the ball, or a special agent (e.g. the *coach*, see Section 5.1). This player is determined through the Setplay definition, and it must not, naturally, be fixed throughout the *Setplay*, which means it can change from step to step, while monitoring the execution of the *Setplay*.

On *Setplay* start, the lead player will instruct the other players on *Setplay* begin, communicating the *Setplay* number, the participating players and the other (optional) parameters (message *startSetplay* in Fig. 8). Along *Setplay* execution, the verification of step entry and the choice of a transition will be announced through *stepChange* and *nextStep* messages, as seen in Fig. 8. The entry into a new step, which is decided by the lead player in charge of the previous step, can imply the change of the lead player. Other possible messages are related to *Setplay* end: the lead player may ver-

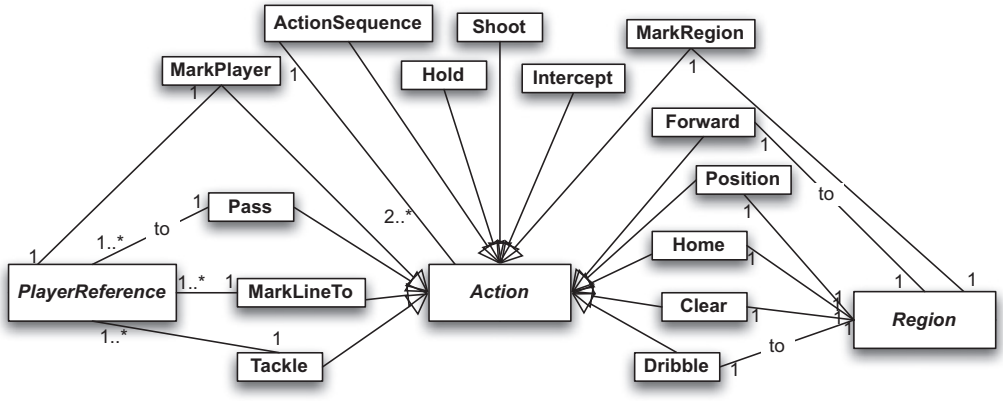

Fig. 5. Action definition.

Please cite this article in press as: Mota L et al. Multi-robot coordination using Setplays in the middle-size and simulation leagues. Mechatronics (2010), doi:10.1016/j.mechatronics.2010.05.005

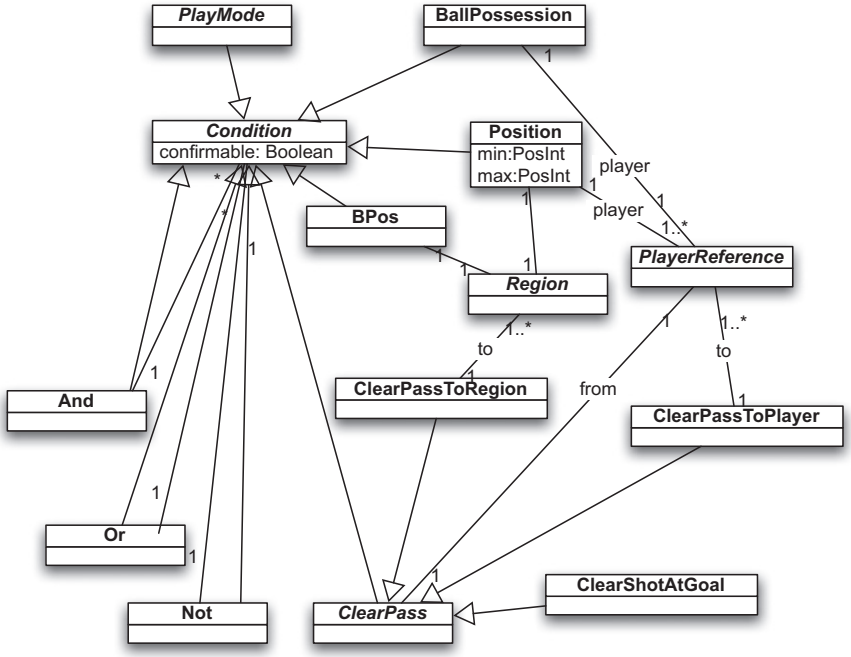

Fig. 6. Condition definition.

ify that a *Setplay* is finished, when it chooses a *Finish Transition*, or that it must be aborted, when a *Abort Transition* is followed or the general *Abort Condition* is satisfied. Such situations will be announced through the corresponding messages.

In momentary situations, the failure of the lead player, or communication problems, would impair the management of Setplay execution. Precisely to avoid these situations, Setplays have abort timeouts that will stop the execution in case of lead player's inaction.

The implementation of this communication policy is described in more detail through an example in Sections 4.2 and 5.2.

3.2. 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,

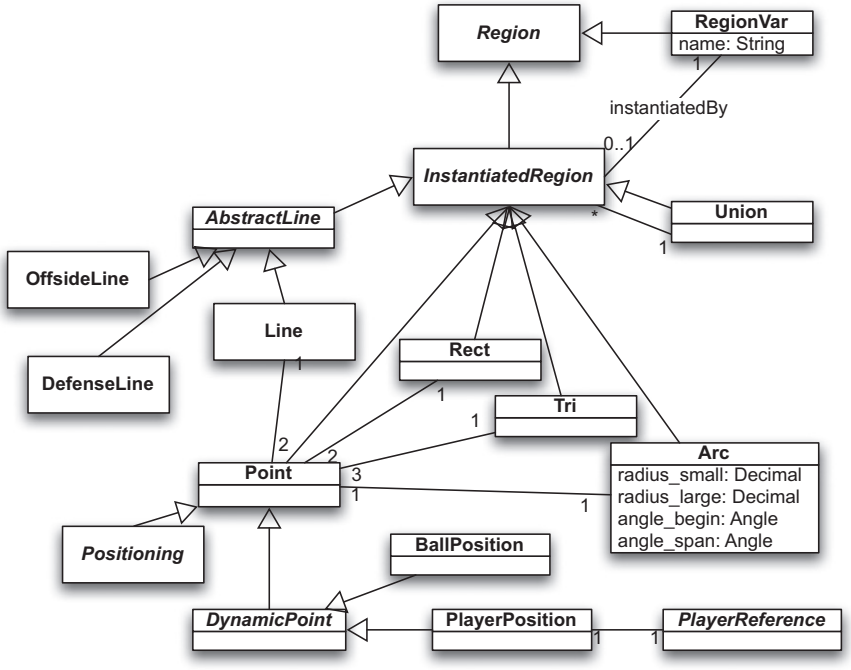

Fig. 7. Region definition.

Please cite this article in press as: Mota L et al. Multi-robot coordination using Setplays in the middle-size and simulation leagues. Mechatronics (2010), doi:10.1016/j.mechatronics.2010.05.005

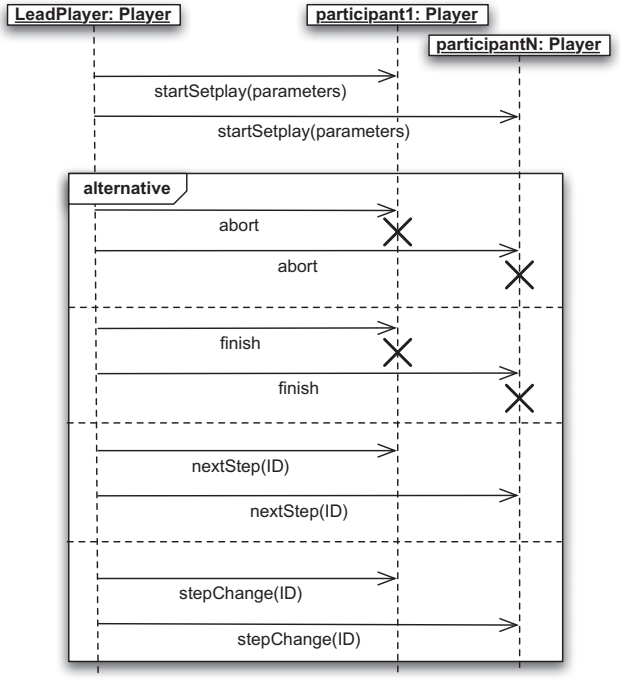

Fig. 8. Setplay interaction scheme.

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 and CAMBADA teams, both 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[4] 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.

3.3. 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 [11].
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 moment it is possible, through the invocation of methods, to know if the Framework requires a message to be sent and, in this case, to access its content in plain text. In such situations, the content of the message should be transmitted to other players through the communication channels available. There is also a method to, upon reception, interpret a 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 the parameters included in the Setplay definition, and regularly (i.e., in every execution cycle) update the Setplay status through an update method, which must supply ball and players positions, and check for message received or waiting for emission.

An initial prototype [13] of the implementation of the Setplay framework was applied to the simulation 3D league, namely to the FCPortugal3D team [8], which won RoboCup 2006 in this league. This prototype was implemented on top of the 2006 simulator version, where the players were modelled 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.

4. Usage in the simulation 2D league

As a primary test-bed for the *Setplays*, the code of the FCPortugal [16], 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 free-kicks, 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 4.1.

4.1. 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

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

Please cite this article in press as: Mota L et al. Multi-robot coordination using Setplays in the middle-size and simulation leagues. Mechatronics (2010), doi:10.1016/j.mechatronics.2010.05.005

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 3.1). The Setplay startup message does therefore only have the participating player numbers as arguments.

4.2. 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.

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. 9a. 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. 9b. 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. 9c) and moves to step 3, which simply finishes the Setplay. At this 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. 10.

5. Usage in the middle-size league

After successfully having implemented the framework's usage in the 2D simulation league, it was decided to use it in a real-world environment, the middle-size league. This league's development in recent years has seen the leading teams' effort to enhance the high-level performance, through team-level coordination and cooperation schemes, as seen in Section 2. Namely, as it has been described in Section 1.1, team CAMBADA has developed configurable team--play for set pieces. It was thus considered timely to take this kind of team-play one level higher: fully integrate the Setplay framework.

Development was, in a first moment, conducted in a simulated environment: CAMBADA has developed a full-blown simulator to ease the development process and avoid the constant use of the robots. The simulator fully emulates the robots' low-level functions (perceptors and actuators), and hence the high-level software can be directly applied to the robots, with only minor inconsistencies. Development in the simulated environment was quite straight-forward, and, after implementing the abstract *Conditions* and *Actions*, the robots were quickly executing Setplays. Special attention had to be given to passes between players: the *canPassPl Condition* had to assure that both players involved were well aligned, or else the pass would frequently fail, since the ball catching capabilities of the robots are very limited.

Setplay instantiation and startup were taken care of by the team *coach*, which has access to the team's shared State-of-the-World (see Section 5.1 further down for details). This intangible agent, that does not have to deal with low-level activities, has plenty computing power to manage the Setplay execution.

The integration of the framework in the CAMBADA team has proven to be more challenging than in the 2D simulation team, since the low-level perception and action mechanisms are far more complex, and had to be more finely tuned. The implementation, however, did not find any major problems nor did it demand changes in the framework.

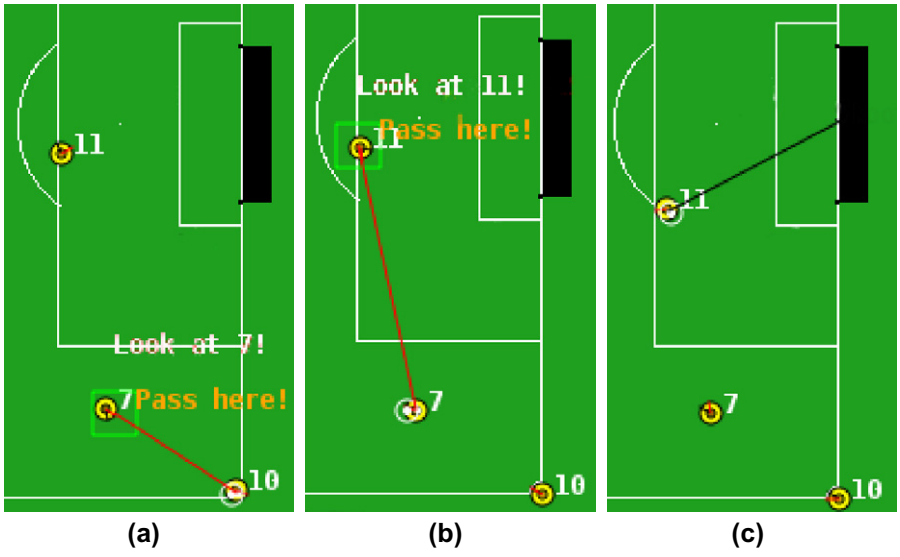

Fig. 9. Corner Setplay execution steps.

Please cite this article in press as: Mota L et al. Multi-robot coordination using Setplays in the middle-size and simulation leagues. Mechatronics (2010), doi:10.1016/j.mechatronics.2010.05.005

```
(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. 10. Corner Setplay definition.

5.1. Inter-robot communication

Communication in the middle-size league is not subject to strict limitations: although the robots are fully autonomous, they can freely communicate through a wireless network using standard protocol 802.11. In championship settings, it is common that the network has high traffic and is consequently over-loaded, but this situation in normally under control. In order to avoid message loss or synchronization issues, messages are repeatedly written a configurable number of times, which will ensure eventual reception of all messages.

The CAMBADA team uses a black-board approach with a shared State-of-the-World. This architecture, called Real Time Database (RTDB) [6] has been used both for posting perception information and coordination flags in previous tackles at team-level coordination. The RTDB was therefore used for the posting of the Setplay messages by the *coach*, since these can be read by all the team robots.

5.2. Example Setplay

In order to illustrate Setplay execution, a very simple interaction will be used as example, in this case in play-on mode, when the ball is in possession of our team, in a specific region (intersection of *mid_left* and *their_back*). There are two robots involved: the *striker* will turn and then pass the ball to the *shooter*, which, in turn, will position itself in a central point and, upon reception of the pass, will shoot at goal. Naturally, there can be more complex Setplays, with more parameters or extra receivers: such complexities are avoided in this example for clarity's sake. This Setplay's definition can be seen in Fig. 11.

The execution of this Setplay will be illustrated through images, displayed in Fig. 12, of an actual execution in CAMBADA's simulator, which uses the same code that runs on the actual robots.

The Setplay is initiated by the *coach* upon verification that the conditions for entry in step 0 are satisfied: play-mode is play-on, ball is in the desired region and one player has ball possession. The participating players are chosen according to their distance from the Setplay positions. At this moment (see diagram on Fig. 12a), the *coach* posts the following message on the RTDB, stating that the participant players have jersey numbers 6 (representing the coach), 5 and 3:

```
SO 6 5 3
```

Upon reading this message, these players position themselves in the desired positions. Since the desired pass can be accomplished, the *coach* will announce, as follows, that the desired next step is nr. 1, which in this case is the only available option:

```
O 1
```

Please cite this article in press as: Mota L et al. Multi-robot coordination using Setplays in the middle-size and simulation leagues. Mechatronics (2010), doi:10.1016/j.mechatronics.2010.05.005

This will allow the *striker* to rotate towards the *shooter* (Fig. 12b). After the players reach their positions, another step progress is announced by the *coach*:

```
1 2
```

The *striker* will then pass the ball to the *shooter* (see Fig. 12c). This robot will accomplish the 'receiveBall' action by waiting for the ball and, when it comes close, moving back to avoid it to reflect (Fig. 12d). After catching the ball, the *coach* will evaluate if it considers a shot at goal possible, in which case it will post a new step change message:

```
2 3
```

This triggers the preparation (see Fig. 12e) for a shot at goal through a kick behaviour, which in turn is visible on Fig. 12f. In case the shot is possible, the Setplay will be finished, with the corresponding message being written on the RTDB by the *coach*, where `-1` stands for an inexistent state:

```
3 -1
```

6. 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 in the middle-size league example, 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.

Its generality is also beyond doubt, since the same framework, without any kind of change in its core, has been applied to two very different RoboCup setups. This is clearly a positive point 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

```
(setplay :name playOnCambadaLeft
  :comment (Only works on left side because of the positioning of ball.
      Pass can only be done to shooter)
  :players (list (player :team our :number 6) (playerRole :roleName striker)
      (playerRole :roleName shooter))
  :steps (seq
    (step :id 0 :abortTime 5
      :participants (list (player :team our :number 6) (at striker (pt ball))
          (at shooter (pt :x 0 :y 4.5)))
      :condition  (and (playm play_on) (bpos :region (regNamed :name mid_left))
                   (bpos :region (regNamed :name their_back))
                   (bowner :players (player :team our :number 0)))
      :leadPlayer (player :team our :number 6)
      :transitions (list (nextStep :id 1
          :condition  (and (canPassPl :from striker :to shooter)
              (bowner :players striker))
          :directives (list
            (do :players striker :actions (attentionTo :object shooter))))))
    (step :id 1 :abortTime 5
      :participants (list (player :team our :number 6) (at striker (pt ball))
          (at shooter (pt :x 0 :y 4.5)))
      :condition  (and (playm play_on)
           (ppos :players shooter :region (arc :center (pt :x 0 :y 4.5)
             :radius_large 1)))
      :leadPlayer (player :team our :number 6)
      :transitions (list
        (nextStep :id 2 :condition (canPassPl :from striker :to shooter)
          :directives (list (do :players striker :actions (bto :players shooter))
            (do :players shooter :actions (receiveBall))))))
    (step :id 2 :abortTime 3
      :participants (list (player :team our :number 6) striker
          (at shooter (pt ball)))
      :condition (and (playm play_on) (bowner :players shooter))
      :leadPlayer (player :team our :number 6)
      :transitions (list (nextStep :id 3
          :condition (and (playm play_on) (canShoot :players shooter))
          :directives (list (do :players shooter :actions (shoot))))))
    (step :id 3 :abortTime 1
      :participants (list (player :team our :number 6) striker shooter )
      :condition (playm ag_our) :leadPlayer (player :team our :number 6)
      :transitions (list (finish)))))
```

Fig. 11. Play-on Setplay definition.

Please cite this article in press as: Mota L et al. Multi-robot coordination using Setplays in the middle-size and simulation leagues. Mechatronics (2010), doi:10.1016/j.mechatronics.2010.05.005

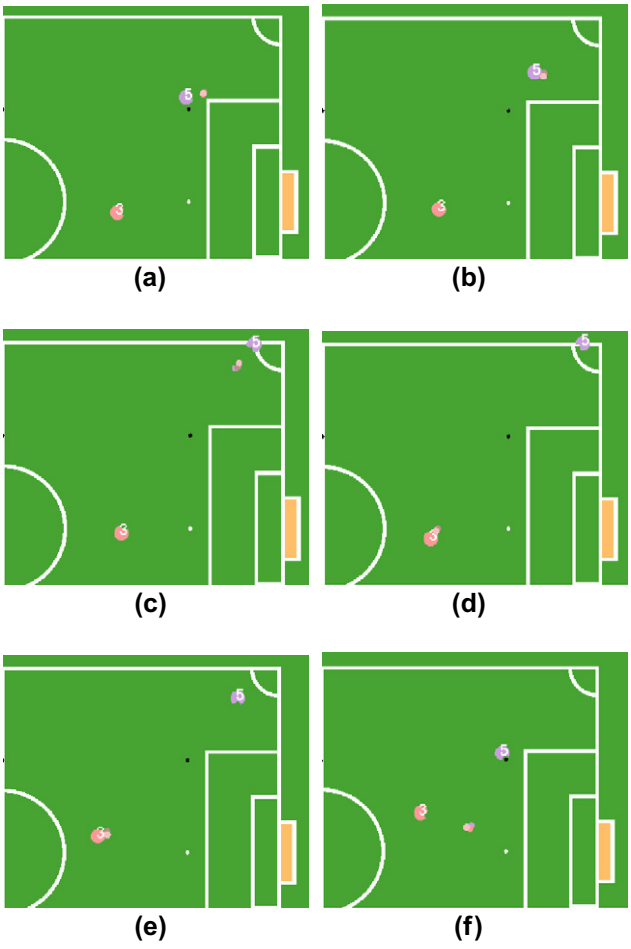

Fig. 12. Play-on Setplay execution steps.

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.

To deal with languages like the one presented in this article, the community is presently used to XML-based syntaxes. Though the proposed syntax has an expressiveness comparable to, or even higher than, XML, this is not a strict limitation. Nevertheless, it would be fairly easy to create a translator from an XML syntax to the present s-expression syntax, through style sheets or some similar technology. Other possibility would be the development of an XML parser from scratch. Such options would make the interaction with the framework more appealing to users familiar with XML.

Setplays have, in the simulation league, been mainly applied in situations where the startup conditions are very clear, i.e., situations signalled by the referee and therefore known to all players. This will be further investigated in the near future: how to startup Setplays in play-on situations, and how to deal with coordination in these more complex situations, when there is no time to prepare the Setplay, extending the work already done at this level in the middle-size league.

Another line of research will be the application of machine learning techniques 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.

References

[1] Aangenent W, de Best J, Bukkems B, Kanters F, Meessen K, Willems J et al. TechUnited eindhoven team description 2009. In: CD proceedings of RoboCup 2009 symposium; 2009.
[2] Chen M, Foroughi E, Heintz F, Kapetanakis S, Kostiadis K, Kummeneje J et al. Users manual: RoboCup soccer server manual for soccer server version 7.07 and later; 2003. <http://sourceforge.net/projects/sserver/>.
[3] Corrente G. Arquitectura de controlo/coordenação de uma equipa de futebol robótico. Master's thesis, Universidade de Aveiro; 2008.
[4] Kitano H, Asada M, Kuniyoshi Y, Noda I, Osawa E, Matsubara H. Robocup: a challenge problem for AI. AI Mag 1997;18(1):73–85.
[5] Lange S, Müller C, Welker S. Tribots: soccer strategy. RoboCup workshop Kassel 2008; 2008. <http://www.ni.uos.de/fileadmin/user_upload/tribots/Research/Kooperation.pdf>.
[6] Lau N, Lopes LS, Corrente G. CAMBADA: information sharing and team coordination. In: Eighth conference on autonomous robot systems and competitions, Universidade de Aveiro, Aveiro, Portugal; 2008. p. 27–32.
[7] Lau N, Lopes LS, Corrente G, Filipe N. Multi-robot team coordination through roles, positionings and coordinated procedures. In: 2009 IEEE/RSJ international conference on intelligent robots and systems – IROS 2009, St. Louis, USA; October 2009.
[8] Lau N, Reis LP. Coordination methodologies developed for FC Portugal 3D 2006 team. In: 10th Robocup 2006 symposium CD, Bremen, Germany; 2006.
[9] Lau N, Reis LP. FC Portugal – high-level CoordinationMethodologies in soccer robotics. Vienna, Austria: ItechEducationandPublishing; 2007. p. 598.
[10] Lauer M, Hafner R, Lange S, Riedmiller M. Cognitive concepts in autonomous soccer playing robots. Cognitive Syst Res. doi:10.1016/j.cogsys.2009.12.003.
[11] Lopes, R., Mota, L., Reis, L.P., Lau, N., Playmaker: Graphical Definition of Formations and Setplays, Workshop em Sistemas Inteligentes e Aplicações - 5ª Conferência Ibérica de Sistemas e Tecnologias de Informação (CISTI'2010), <http://www.aisti.eu/cisti2010/index.php?option=com_content&view=article&id=66&Itemid=70&lang=pt>.
[12] McMillen C, Veloso M. Distributed, play-based role assignment for robot teams in dynamic environments. In: 8th International symposium on distributed autonomous robotic systems (DARS 2006), Minnesota, USA; 2006.
[13] Mota L, Reis LP. Setplays: achieving coordination by the appropriate use of arbitrary pre-defined flexible plans and inter-robot communication. In: Winfield AFT, Redi J, editors. First international conference on robot communication and coordination (ROBOCOMM 2007). ACM international conference proceeding series. IEEE, vol. 318; 2007.
[14] MSL Technical Committee. Middle size robot league rules and regulations for 2009. Tech. rep., RoboCup Federation; 2008.
[15] Noda I, Matsubara H, Hiraki K, Frank I. Soccer server: a tool for research on multiagent systems. Appl Artif Intell 1998;12(2–3):233–50.
[16] Reis LP, Lau N. FC Portugal team description: Robocup 2000 simulation league champion. In: Stone P, Balch T, Kraetzschmar G, editors. RoboCup-2000: Robot Soccer World Cup IV. LNAI, vol. 2019. Springer; 2000. p. 29–40.
[17] Reis LP, Lau N. Coach unilang – a standard language for coaching a (robo) soccer team. In: RoboCup-2001: Robot Soccer World Cup V. LNAI, vol. 2377. Springer Verlag; 2002. p. 183–92.
[18] Reis LP, Lau N, Oliveira E. Situation based strategic positioning for coordinating a simulated robosoccer team. In: Hannebauer M, Wendler J, Pagello E, editors. Balancing react and social deliberation in MAS. LNAI, vol. 2103. Springer; 2001. p. 175–97.
[19] Risler M. Behavior control for single and multiple autonomous agents based on hierarchical finite state machines. Ph.D. thesis, Technische Universität Darmstadt; 2009. <http://tuprints.ulb.tu-darmstadt.de/2046>.
[20] Risler M, von Stryk O. 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.
[21] Röfer T, Brose J, Carls E, Carstens J, Goehring D, Juengel M et al. 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.
[22] Ros R, de Màntaras RL, Arcos JL, Veloso M. Team playing behavior in robot soccer: a case-based reasoning approach. LNCS, vol. 4626. Berlin/Heidelberg: Springer; 2007. p. 46–60.
[23] Sequeira RP. Strategic coordination of CAMBADA robotic soccer team. Master's thesis, Universidade de Aveiro; 2009.
[24] Stone P. Layered learning in multi-agent systems. Ph.D. thesis, CMU; 2008.
[25] Stone P, Veloso M. Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork. Artif Intell 1999;110(2):241–73.
[26] Zweigle O, Käppeler U-P, Rajaie H, Hüssermann K, Lafrenz R, Tamke A et al. CoPS Stuttgart team description 2008. In: Visser U, Ribeiro F, Ohashi T, Dellaert F, editors. RoboCup 2008 symposium CD; 2008.
[27] Zweigle O, Lafrenz R, Buchheim T, Käppeler U-P, Rajaie H, Schreiber F et al. Cooperative agent behavior based on special interaction nets. In: Intelligent autonomous systems, vol. 9: IAS-9; 2006.

Please cite this article in press as: Mota L et al. Multi-robot coordination using Setplays in the middle-size and simulation leagues. Mechatronics (2010), doi:10.1016/j.mechatronics.2010.05.005