Approaching a formal soccer theory from behaviour specifications in robotic soccer*

F. Dylla[1], A. Ferrein[2], G. Lakemeyer[2], J. Murray[3], O. Obst[4], T. Röfer[5], S. Schiffer[2], F. Stolzenburg[6], U. Visser[7] & Th. Wagner[7]
[1]*SFB/TR 8 Spatial Cognition, Universität Bremen, Germany.*
[2]*Knowledge-based Systems Group, RWTH Aachen University, Germany.*
[3]*AI Research Group, Universität Koblenz-Landau, Germany.*
[4]*School of Electrical Engineering and Computer Science, The University of Newcastle, Australia.*
[5]*Safe and Secure Cognitive Systems Group, DFKI Lab Bremen, Germany.*
[6]*Automation and Computer Sciences Department, Hochschule Harz, Germany.*
[7]*Center for Computing Technologies (TZI), Universität Bremen, Germany.*

Abstract

This chapter discusses a top-down approach to modelling soccer knowledge, as it can be found in soccer theory books. The goal is to model soccer strategies and tactics in a way that they are usable for multiple robotic soccer leagues in the RoboCup. We investigate if and how soccer theory can be formalized such that specification and execution are possible. The advantage is clear: theory abstracts from hardware and from specific situations in different leagues. We introduce basic primitives compliant with the terminology known in soccer theory, discuss an example on an abstract level and formalize it. The formalization of soccer presented here is appealing. It goes beyond the behaviour specification of soccer playing robots. For sports science a unified formal soccer theory might help to better understand and to formulate basic concepts in soccer. The possibility of the formalization to develop computer programs, which allow to simulate and to reason about soccer moves, might also take sports science a step further.

* This research has been carried out within the special research program DFG-SPP 1125 *Cooperative Teams of Mobile Robots in Dynamic Environments* and the Transregional Collaborative Research Center SFB/TR 8 on *Spatial Cognition*. Both research programs are funded by the German Research Foundation (DFG). A short and preliminary predecessor of this paper appeared as [1].

WIT Transactions on State of the Art in Science and Engineering, Vol 32, © 2008 WIT Press
www.witpress.com, ISSN 1755-8336 (on-line)
doi:10.2495/978-1-84564-064-4/06

1 Robotics and soccer

In 1997, the first *RoboCup*, the international world championship in robotic soccer was held. The event was part of the International Joint Conference on Artificial Intelligence (IJCAI), and set a new benchmark: the goal of RoboCup is to foster artificial intelligence (AI) and robotics research so that by 2050 a team of autonomous humanoid robots can be built that will be able to win against the human soccer world champion [2]. Similar to the goal of beating the human world champion in chess with a computer, in RoboCup the journey is the reward.

The initiators of RoboCup have chosen soccer as a test bed and common research platform because it covers a wide range of problems from robotics to AI: from energy supply, creating robust robots, over vision and sensor fusion to coordination, cooperative multiagent systems, behaviour programming, machine learning, strategy acquisition and many more issues have to be solved until it will be possible to reach this goal. Soccer is an interesting research problem because it is a multiagent domain where agents have to cooperate with their team-mates and deal with adversarial agents in real-time simultaneously.

Moreover, during the first years of research in robotic soccer it has already turned out that soccer playing robots can also be interesting by themselves, because matches between robots or computer programs can be quite entertaining and exciting – not just for the developers of the respective teams.

1.1 RoboCup leagues

To place the emphasis on different aspects, a number of leagues have been introduced in the RoboCup. In this section, we will briefly discuss these different leagues and their particular properties.

Simulation league. The 2D simulation league, one of the first leagues in RoboCup, concentrates on research in multiagent systems (architectures and coordination mechanisms). Two teams of eleven agents compete in a virtual soccer match in a real-time, but highly abstracted discrete time simulation. A simulation server, called Soccer Server [3], receives the action commands from the agents. Based on these commands it updates the state of the world and dispatches current sensory information to each agent in the next simulation cycle. The simulator also controls the game play. An automated referee judges offsides, throw-ins and counts the goals. The frame of reference for sensory inputs and agent commands is egocentric, i.e. the positions of all visible objects are given as distance and direction to the respective agent. Besides the visual information, the Soccer Server also sends aural messages to the player, i.e. a player can send 10 bytes per simulation cycle, and within a close range around the agent the message may be heard by other players. By this an unreliable low-bandwidth communication among players can be realized, which most teams use for exchanging parts of the agents' local world models. Agents can settle actions by sending one of five basic actions back to the server. These actions are *dash*, *kick*, *turn*, *catch* (for the goal keeper) and *tackle*. Communication is handled with the help of an

extra *say* action. In recent years a more realistic but still abstract 3D simulation has been added which will soon replace the 2D simulation. This slowly evolves into a simulation of humanoid robots.

Small-size league. The Small-size league is a robotic league. Five small wheeled robots play on a field of the size of a table tennis board with a golf ball. As the robots are too small to carry sensors on-board, a ceiling camera is installed above the field. The camera images are sent to each team. Vision processing extracts the relevant information from the images. To alleviate the recognition, each player has a special colour coding on top. With these information the actions that the robots should perform are calculated by a computer off the field. The actions are sent back to the robots via radio. Thus, the league is partly autonomous. The research focus here is mainly on image processing and decision making. In contrast to the other leagues mentioned, player behaviour can be derived from a global, allocentric world model.

Middle-size league. Here, two teams of up to five fully autonomous wheeled robots compete on a field of the size 8×12 m. The robots may have a maximal size of 50×50 cm and the height may not exceed 80 cm. The research focus of the Middle-size league is on robotics, decision making, sensor and actuator systems and the integration of software and hardware. Especially in this league it turns out that the whole system, hardware as well as software, must form one unit. Only completely well integrated systems are competitive.

Four-legged league. While in the Small-size and the Middle-size league the hardware is developed by the participating teams and this development is part of the research, the Four-legged league aims at developing robot control software on a common platform. The robots here are Aibo dog robots from Sony. The different developments can well be compared as they all work on the same platform. The capabilities of the robots are limited. It has only a very small camera resolution, the sensor values of the joints in the legs of the dog are bad. Another problem in this league regarding the hardware platform is that Sony does not provide too many information about the hardware such that several controllers had to be reverse-engineered in order to learn how they work. Finally the quadruped walk and therefore, changing horizon, pose difficulties in this league. Another remarkable thing in this league is that code development is highly distributed. Within the *German Team* [4] several German universities work on a common code base. While their respective teams compete against one another in national events, the most promising approaches are integrated in a national team that participates quite successfully in the RoboCup world cups. Clearly, this is supported by the shared hardware. Unfortunately, this league will come to an end, as Sony stopped the production of Aibo robots in 2006.

Humanoid league. The ultimate goal of the RoboCup initiative is to play (robotic) soccer with humanoid robots. Of course, research on human-like robots must be conducted in order to achieve this goal. The humanoid league exists for three years now and has already made remarkable progress since then. In the beginning, the competitions were only so-called technical challenges, where the teams showed the capabilities of their robots. Today, there are already soccer

matches two-on-two. There exist two different sub-leagues based on the sizes of robots, the so-called Kid-size league and the Teen-size league. In the Kid-size league robots with a height from 30 to 60 centimetre compete while a typical Teen-size robot measures between 65 and 130 centimetre, although in special cases robots up to 180 cm may participate in this league.

Non-soccer leagues. In order to address robotic and AI related problems not covered by the soccer-playing robots scenario, additional RoboCup competitions have been created: the RoboCup@Home and the RoboCup Rescue leagues. In the former, robots should fulfil helper tasks in human home environments. In the latter, autonomous agents and robots have to solve large and small scale rescue tasks, namely coordinating rescue teams in a (simulated) earthquake scenario and rescuing entombed people from an urban disaster area.

1.2 Challenges in robotic soccer

Participating in RoboCup, one faces a variety of problems in order to enable the robots to play soccer. Some of the problems are purely related to robotics: one has to deal with sensors and actuators, making the robot run. Other aspects are related to software design accounting for the real-time aspects of the soccer domain. The software system must be designed in such a way that sensor information coming in with high frequency can be processed promptly. Upon these data the robot must decide on appropriate actions like *kicking* or *dribbling* and compute the corresponding actuator commands. Further problems like navigation, collision avoidance and localization have to be solved. Beyond these problems it comes to the question how the processing from sensor inputs to actuator output is designed. State of the art for soccer robots are so-called reactive systems. Usually the robots make up a model of the accessible world from their sensor data. Such a world model contains data like the own position on the field, the position of the ball and the positions of opponents perceived. In a reactive system, a particular configuration of the world state, i.e. a configuration of world model variables, are mapped to an action the robot can take. In reactive systems, the occurrence of certain variable values is mapped directly onto a specific action without considering what happened in the past. These actions are rather complex actuator signal sequences. For an intercept action, for example, the system has to take into account the position of the ball, its velocity and the relative direction to it to set appropriate motor commands such that the ball is finally in the gripper of the robot.

At this stage of development the designer of such a robot system has to think about how the behaviour of the robot should be set up. As laid out before, one possibility is to directly couple sensors and actuators (possibly via a world model representation). The system designer now has to think about the abilities the robots have and she/he has to think about how soccer is played. With a soccer robot and its restricted abilities, currently, the behaviours are quite simple: usually, one robot captures the ball, dribbles towards the opponent goal and tries to score.

1.3 Learning from human soccer

Keeping in mind the RoboCup vision, only utilizing such simple behaviour patterns is not sufficient. One has to think about how humans play soccer. What are the different moves, strategies and tactics? Clearly, a humanoid robotic system must be able to perform moves that are as sophisticated as the ones used by human soccer teams in order to be competitive.

That is why we started investigating human soccer theory to learn more about the core of soccer in [1]. The aim of this joint research with background in different RoboCup leagues was to come up with a general theory of soccer for robots. We approached building this theory by investigating existing soccer literature to work out formal aspects of soccer. Particularly, we looked at a textbook by Lucchesi [5]. In this book soccer moves are described by diagrams containing important player positions. Several actions like *dribble* or *move* are depicted with arrows pointing to other positions on the pitch. In most of the diagrams the positions of opponent players are left out. This is due to the abstract character of the representation. To be able to understand the idea of the moves a common soccer ontology is assumed. In general, humans are very good in understanding such abstract qualitative representations. In brief, qualitative knowledge is obtained by comparing features within the object domain rather than by measuring them in terms of some artificial external scale. Thus, qualitative knowledge is relative knowledge where the reference entity is a single value rather than a whole set of categories [6]. Though, for a robot these representations do not help at first. The diagrams have to be translated into a formal, mathematical description of the scene before a robot could perhaps make use thereof. It starts with fixing an ontology for soccer defining, for example, what it means for a player to be a defender or an attacker. Furthermore, the spatial relations used within the descriptions have to be defined. The questions to be answered here are, for example, in which situation is a soccer move applicable, or when can a pass be played to a team-mate and finally, how can the sequence of actions of the soccer move be formally described.

1.4 Overview of the rest

This chapter is about the ongoing work to formalize soccer theory for the behaviour specification of soccer robots. The future direction we pose here is to come to a formal description of soccer. Beyond robotic soccer it could help to better understand soccer, to be able to better analyse soccer, and to simulate soccer moves with a computer program.

The chapter is organized as follows. In Section 2 we sketch the specification language Readylog and motivate spatial relations we use for formalizing soccer theory. In Section 3 we introduce soccer moves as given by Lucchesi in [5], deriving the soccer ontology and the building blocks for soccer in terms of actions and their preconditions. We also present a robotic soccer example and show how elements of soccer theory can be adapted to soccer robots. Here, we

also sketch the qualitative world model we defined in order to describe important regions on the soccer pitch. We conclude with a discussion of the related work and a perspective outlook on this work (Section 4).

2 Theoretical background

In this section we briefly introduce the language *Readylog* which we use for the behaviour specification and programming of our soccer robots. This language is a formalism for combining robot programming with planning. It is also very well suited to formalize the soccer domain. We start with an introduction to the situation calculus, that is the formalism which Readylog is based on. Then we give an overview of Readylog and the different programming constructs which are available. We leave out the formal definition of the language constructs. It is important to note that a formal semantics for the language exists, i.e. the execution of programs of Readylog is not dependent on a particular implementation, which makes it very well suited to formalize soccer theory. In the rest of this section, we explain spatial relations that are needed for a formalization of soccer theory.

2.1 Situation calculus

The situation calculus [7, 8] is a logical second-order language proposed by John McCarthy in 1963. It allows us to reason about actions and change. The world evolves from an initial situation due to primitive actions. Possible world histories are represented by sequences of actions. The situation calculus distinguishes three different entities: *actions*, *situations* and domain dependent *objects*. There exists a special binary function $do(a,s)$ which denotes the situation that arises after performing action a in situation s. The constant s_0 denotes the initial situation, i.e. the situation where no actions have occurred yet.

The state of the world is characterized by relations and functions with a situation term as their last argument. They are called *relational* and *functional* fluents. As an example consider a robot lifting an object. The fluent $holding(s)$ describes whether the robot has lifted the object and holds it in its gripper. In the initial situation s_0 the robot has not picked up the object, thus $holding(s_0)$ is false. Now the robot performs the action $pickup(object)$. The situation describing the new state of the world is $s_1 = (do(pickup(object),s_0))$. The effect of the action can be described in terms of the fluent *holding*. It should hold that after performing the pickup action the fluent

$$holding(do(pickup(object), s_0)) \equiv true,$$

i.e. the robot has lifted the object and holds the object in its gripper. What is needed to make this true is a so-called effect axiom which describes the effects of the action *pickup*. A possible effect of this action is

$$holding(object, do(pickup(object, s))).$$

This means that if the robot performs the action pickup, it follows that in the successor situation $do(pickup(object, s))$ the fluent $holding(object)$ becomes true. What is further needed is a set of axioms which describe when an action is possible. For our pickup action a precondition could be that the box must not be too heavy, or expressed formally

$$Poss(pickup(object), s) \equiv \sim heavy(object, s).$$

Poss is a predicate which denotes the possibility to execute an action in a particular situation. Summarizing the basic ingredients of the situation calculus, one specifies a formal theory defining fluents, actions, their preconditions and their effects. Now, one can reason, for example, if at some state of the world after performing a sequence of actions, particular properties of the world are true or false. With this formalism one can easily simulate sequences of actions and find out how the world looks like afterwards. We only briefly sketched the situation calculus. For a thorough discussion about the situation calculus we refer to the textbook written by Reiter [8].

2.2 Readylog

Readylog [9, 10] is a variant of Golog [11] which is based on Reiter's variant of the situation calculus [7, 8] as described above. We will very briefly sketch the constructs of Readylog in the following. The original *Golog* evolved to an expressive language over the recent years. It not only has imperative control constructs such as loops, conditionals and recursive procedures, but also less standard constructs like the nondeterministic choice of actions. Extensions exist for dealing with continuous change [12] and concurrency [13], allowing for exogenous and sensing actions [14] and probabilistic projections into the future [15] or decision-theoretic planning [16] which employs Markov Decision Processes (MDPs).

Readylog integrates these extensions in one agent programming framework [9, 10]. For specifying the behaviour of an agent or robot the following constructs exist:

1. sequence: $(a; b)$
2. nondeterministic choice between actions: $(a|b)$
3. nondeterministic choice between action arguments: *pickBest*
4. MDP solving: $solve(p, h)$,
   p is a Golog program, h is the MDP's solution horizon
5. test actions: $?(c)$
6. event-interrupt: $waitFor(c)$
7. conditionals: $if(c, a_1, a_2)$
8. loops: $while(c, a_1)$
9. condition-bounded execution: $withCtrl(c, a_1)$
10. concurrent execution of programs: $(p_1 \parallel p_2)$

11. probabilistic actions: $prob(val_{prob}, a_1, a_2)$
12. probabilistic (offline) projection: $pproj(c, a_1)$
13. procedures: $proc(name(parameters), body)$

To encode the behaviour one has to give a domain axiomatization including the actions the robot can perform together with their effects, and the fluents which describe the properties of the world, e.g. the ball position. Examples of domain descriptions for the soccer domain can be found in [17, 18].

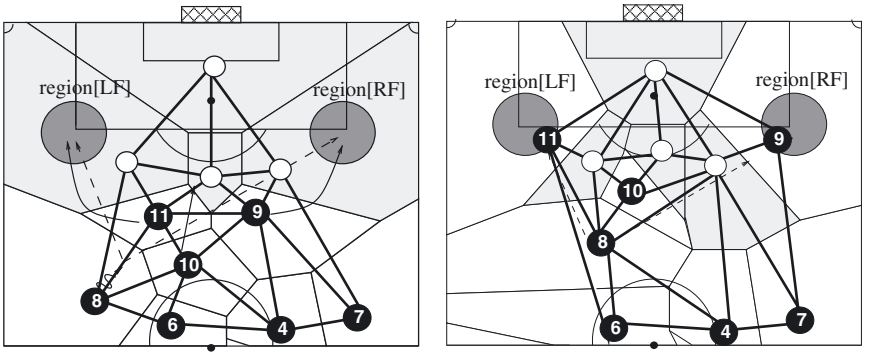
2.3 Spatial relations

In contrast to formal descriptions of (soccer) knowledge by means of mathematical equations or the situation calculus and their derivatives, human representations of soccer are as such qualitative. Distances between players on the pitch are, of course, never quantitatively represented or perceived by human players. A player would never express: I have to play the pass when my team-mate has a distance of exactly 2.35 metre and she is at an angle of 48° to me. This is one of the problems when trying to transfer the human soccer theory to soccer robots. Adequate human-readable models have to be found in order to be able to transfer human expert knowledge to a robot. In the following, we list some basic spatial categories.

Distance and orientation. The basic notion needed for describing soccer moves is a notion of space and distance. The approach described in [19] is based on [20, 21]. We started with an egocentric relation of distance. One defines a metric for IR^1, builds equivalence classes for ranges of IR^1 and assigns constants like *near* or *far* to each class. To build the orientation relation we build equivalence classes over ranges of angles. Each sector is assigned an orientation like with a compass rose. We use eight different orientations like *front*, *front-right*, *right* and so on. With these models for distance and orientation the robot is able to describe objects in a qualitative egocentric fashion like: the ball is located in the front-left direction at a medium distance.

Tactical regions. What is further needed to describe the soccer scenario, is a qualitative notion of strategic positions in a global frame of reference. In soccer, there are several strategic roles a player can have like *defence*, *midfield* and *offence*. Further, one can distinguish between three sides of the pitch as in [5]. The play can be on the left or right side or in the centre of the field. For our qualitative world model we defined five zones ranging from *farFront*, which is a zone directly in front of the opponent goal, to *farBack* which includes the own goal, and the sides *left*, *middle*, *right*. Again, we refer to [19] for details.

Reachability of regions or objects. To express reachability mathematically, one needs a model which takes into account the amount of free space available between the positions of team-mates or opponents. One possible model is to use Voronoi diagrams and their dual, the Delaunay triangulation as they separate the field into non-intersecting regions and we get a connection graph between the players. (A Voronoi diagram $V(S)$ of a set S of n point sites is the partitioning



(a) Initial situation (b) Intended resulting situation

Figure 1: Delaunay triangulation (bold lines) and Voronoi regions (white/grey) for the counter-attack example.

of a plane with n points into n convex polygons such that each polygon contains exactly one point $p \in S$ and every point in the given polygon is closer to p than any other point $p' \in S$. For a more detailed account on Voronoi diagrams and their dual, the Delaunay triangulation DT(S), see e.g. [22].) Figure 1 depicts the Delaunay triangulation and the Voronoi regions for the positions of the players in a counter-attack situation. The intention of this move is that after player 8 captured the ball, it dribbles towards the centre of the goal area until player 9 or 11 become reachable for a pass at the (in Figure 1) marked regions LF and RF, respectively. The bold lines represent the triangulation, the white and grey regions correspond to the Voronoi regions of the attacking team and the defending team, respectively. Figure 1(b) shows the diagram for the intended situation after player 9 and player 11 have taken their positions in their regions near the goal area. Note that we ignore offside for simplicity here.

The Voronoi diagram gives us information about which position on the field is closest to which player. In Figure 1(a) we draw the conclusion that the opponent defence controls the goal area, whereas after the successful counter-attack this line of defence is penetrated. The triangulation yields a conservative estimate about which player can receive a secure pass. In this particular example there is no connection between player 8 and player 9 and this resembles our intuition that the pass is not secure. In Figure 1(a) player 9 and player 11, respectively, can test if their target regions are occupied by opponents and they can also test the distance of the defenders to their particular region. Player 8 can dribble the ball as long as no opponent is in a distance where it can tackle player 8.

3 Formalizing soccer strategies

In this section we describe our approach to formalize soccer strategies for soccer robots [1]. The idea was to derive the basic behaviour patterns of soccer as

described in [5] and adapt them to the different soccer leagues in RoboCup. The formalization of the soccer moves was done in Readylog which is also very suited for this task because of its formal semantics. In [1] we also started a case study how to apply the soccer moves to the different leagues. We leave out this part here, concentrating on the qualitative aspects of the formalization and refer to [1] for further details.

In the following, we first describe the basic ontology for soccer strategies and (related with this) the basic actions of a soccer player, before we derive the basic qualitative predicates needed to formalize soccer moves. Further, we give an example specification of a soccer move which can almost directly be encoded as Readylog program that is executable on a soccer robot.

3.1 The organization of soccer knowledge

Among modern soccer publications, Lucchesi's book [5] is one of the most interesting ones because it concentrates on strategic aspects of soccer rather than on training lessons. Soccer strategies in literature (e.g. [5, 23]) are not as highly structured as, say, strategies for American football. Though, they are structured enough to build a top-level ontology for it. According to [5] there are two phases in a soccer game: (1) the defensive phase and (2) the offensive phase. In the defensive phase the ultimate goal of the team is to *prevent the opponent from scoring a goal* and to *gain ball possession* again. When the second sub-goal of this phase is fulfilled the game enters the offensive phase. Here, a controlled *build-up of the play* has to be performed. In general, there are two ways to build up the play: either we introduce this phase in a counter-attack manner, i.e. fast and direct with a long pass, or deliberately by a diagonal pass or by a deep pass followed by a back pass. The taxonomy for soccer strategy is depicted in Figure 2, where 3-4-1-2 stands for the basic tactical setup of the team. The pattern 3-4-1-2 means that the team is playing with three defenders, four midfielders, one offensive midfielder and two forwards.

In the following, we will concentrate on the building-up phase for an illustration of how to derive basic behaviour patterns for soccer play. Figure 3 shows two example diagrams from [5]. The goal of the attacking team after building up the play is to *create a scoring opportunity*. The final move then is to try to *score a goal*. In soccer, there exist several strategic groups, each having a particular task in fulfilling the strategy just mentioned. The defence has to prevent the opponent team from scoring and must build up the play. The midfielders work to create a scoring opportunity and the offence has to score the goal.

Accordingly, a soccer strategy can be defined as a tuple $str = \langle RD, CBP \rangle$. Here, RD is a set of *role descriptions* that describe the overall abilities required of each player position in relation to CBP, the set of *complex behaviour patterns* is associated with the strategy. Given the strategy str, the associated role description $rd \in RD$ can be described by the defence tactics task, the offence tactics task, the tactical abilities and the physical skills.

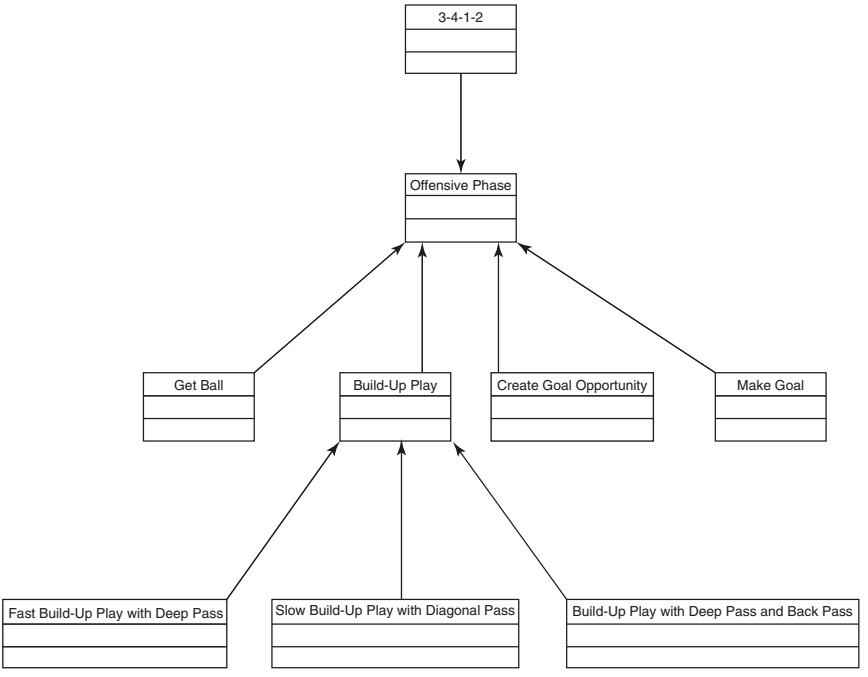
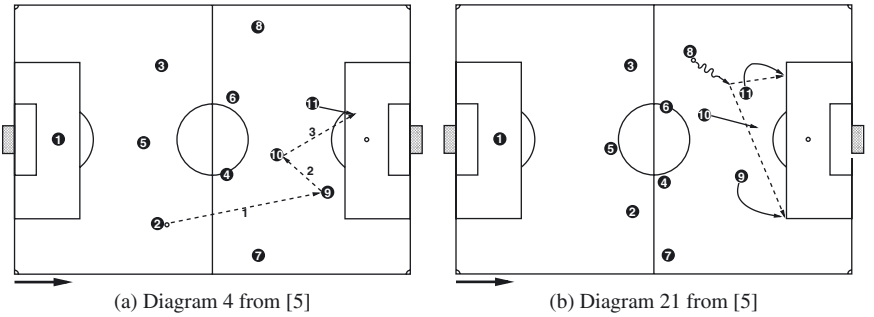


Figure 2: Top-level ontology according to [5].



(a) Diagram 4 from [5]

(b) Diagram 21 from [5]

Figure 3: Two tactical diagrams from [5]. The bold arrow next to the field indicates direction of play. Player movements are represented by arrows ($\rightarrow$ or $\curvearrowright$), passes are indicated by dashed arrows ($\dashrightarrow$), and squiggly arrows ($\rightsquigarrow$) stand for dribbling. Opponents are not shown.

3.2 An example: build-up play

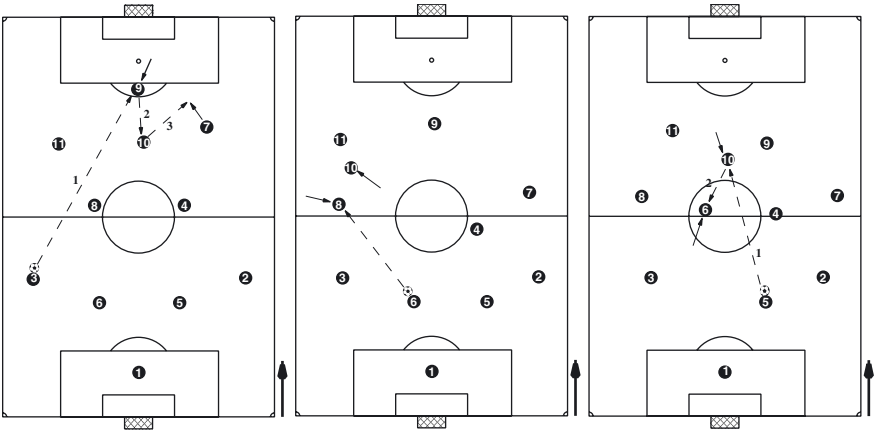
In this phase of the game, the team's objective is to take the ball towards the opponent goal in order to establish a setting which allows for creating a scoring opportunity. The ball has to be taken from the defensive players to the offensive players. There are a number of ways to build up the play:

Build-up play immediately with long pass. The long pass enables the team to take the ball up-field towards the opposing goal very quickly. There is an immediate reversal of play and the risk of losing the ball near one's own penalty area is very low. However, the long pass is difficult to receive, so the opponent may be able to steal the ball more easily. Moreover, as there is not much time, the team cannot move forward in a coordinated way.

Build-up play deliberately with diagonal pass. The diagonal pass allows for a coordinated way to move forward with the ball and it is easy to receive. The time to get close to the opponent goal is longer than with the long pass. Thus, chances of losing the ball in a dangerous area are higher.

Build-up play deliberately with deep pass and subsequent back pass. This way of building up play requires very good timing as it involves three players who have to move in a coordinated way. If such a move is carried out successfully it allows the team to move forward up-field without great risks if they lose the ball. We depict one exemplary tactical move for each of the three patterns to build up play mentioned in Figure 4.

The decision which pattern to choose certainly depends on the opposing team as well as on the particular situation. That is to say, when playing against an



(a) A long pass (b) A diagonal pass (c) A deep pass and a subsequent back pass

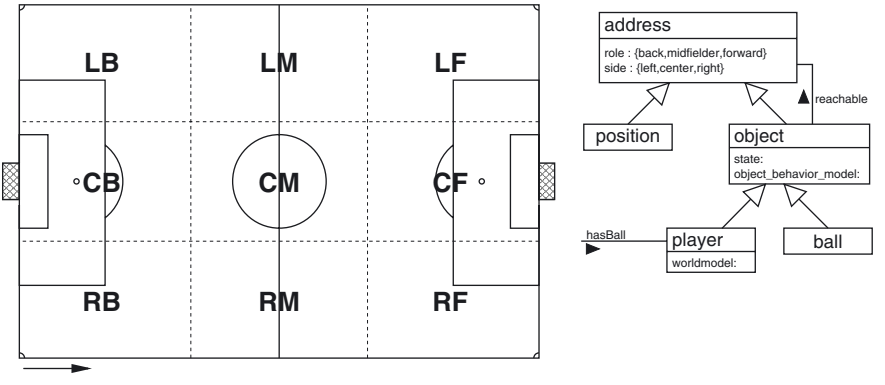
Figure 4: Three different ways to build up a play. Dashed lines represent pass ways and solid lines denote a player movement. The bold arrow next to the field indicates the direction of play.

opposing team which has many players in the midfield one would perhaps favour to build up a play with a long pass whereas with a team leaving lots of space uncovered in the midfield one would prefer the deep pass.

Basically, all the possibilities mentioned above are meant to take the ball up-field and establish a more offensive setting for the team while remaining in possession of the ball. The ball is either taken forward from the defence to the midfield section or in the case of the long pass directly to the offence section. Both can be done through the centre of the playing field or by using the wings of the pitch. Depending on how the play was built up there are several ways to create a scoring opportunity.

3.3 Basic primitives

From the example of the previous section one gets quite a good idea which behaviour patterns are needed for a soccer formalization. Following the lines of [5], we distinguish between *role* (back, midfield, forward) and *side* (left, centre, right) in soccer. This distinction is more or less independent from the pattern of play (e.g. 3-4-1-2 or 4-2-3-1). The combination of role and side (e.g. centre forward) can be interpreted as *type* of a (human or robotic) soccer player or as *position* (region or point) on the soccer field. In the latter case, there is a certain *zone* corresponding to the three roles back, midfield and forward. Altogether, this leaves us with basically nine different positions, as illustrated in Figure 5(a).



(a) Tactical regions on the field

(b) Class hierarchy for soccer dervied from [5]

Figure 5: Tactical regions and abstract position hierarchy derived from [5]. The field is divided into three rows (corresponding to player roles): back (B), midfield (M), and forward (F), and three lanes (sides): left (L), centre (C), right (R). An address may be one of the nine regions or player types.

The notions, player type and position, can be seen as instances or specializations of the notion of an *abstract position* or *address* for short usually associated with its (actual) coordinates or a region on the soccer field. Also the position of the ball is abstract, i.e. the parameter or goal of a test or operation of a soccer player (agent). A movable *object* in the context of soccer may be a player or the ball. An object is in a current *state*, which besides other data includes information such as the current speed or the view direction. Although not explicitly mentioned, a *model of behaviour* is assigned to every object, e.g. average or maximum speed or, as a special case, a deceleration rate for the ball. Additionally, every player needs to hold data about other agents' states. We abstract this by the term *world model*. All this is summarized in the class diagram in Figure 5(b).

In [5, p. ii] only few symbols are introduced that are used throughout the many diagrams in that book: players (in many cases only the team-mates, not the opponents are shown), the ball, passing, movement of the player receiving the ball and dribbling. Conceptually, all symbols correspond to *actions*, which we abbreviate as *pass* , *goto* and *dribble* . Since all actions are drawn as arrows starting at some player, naturally two arguments can be assumed: *player* and *abstract position*. *goto*(*player*[*LF*],*region*[*CF*]) means for instance that the left forward player moves in front of the opponent goal.

Although in most cases this is not explicitly mentioned in [5], actions require that certain prerequisites are satisfied, in order to be applicable. Since our approach aims at a very abstract and universal formalization of soccer, we restrict ourselves, for instance in case of a pass, to only two tests: possession of ball and reachability. Each of them can be seen as a *predicate* with several arguments: *hasBall* has the argument *player* (the ball owner); *reachable* has two arguments, namely an object and an address.

A pass e.g. presupposes reachability, i.e. it should be guaranteed that the ball reaches the team-mate. Clearly, the implementation of the reachability test is heavily dependent of the respective soccer league and its (physical) laws. Therefore, at this point, we only give a very general and abstract definition: Object *o* can reach an address *a* iff *o* can move to *a* and after that the ball is not in possession of the opponent team. This also covers the case of going to a position where the ball will be intercepted.

The primitive actions we consider here are *goto*(*player,region*), *pass*(*player*, *region*) and *dribble*(*player*, *region*). Further we need the action *intercept* which is a complex action built from the primitive ones. The arguments of the actions are *player* and *region* denoting that the particular player should go to, pass or dribble the ball to the given position. For describing the properties of the world on the soccer field we need the fluents *reachable* and *hasBall*(*player*) among others. The precondition axioms for the actions are:

$$Poss(pass(player, region), s) \equiv hasBall(player)$$
$$Poss(dribble(player, region), s) \equiv hasBall(player)$$
$$Poss(goto(player, region), s) \equiv true$$

For our formalization of soccer, reachability is central. Reachability strongly depends on the physical abilities of the robot. Besides the physical abilities the reachability relation has some independent properties. In general, we can distinguish three different reachability relations:

go-reachability: a player p not being in ball possession will reach an address a on the field before any other player: $reachable_{go}(p, a)$ with prerequisite $hasBall(p)$

dribble-reachability: a player p being in ball possession is able to dribble towards address a with high probability of still being in ball possession afterwards: $reachable_{dribble}(p, a)$ with prerequisite $hasBall(p)$

pass-reachability: a player p being in ball possession is able to pass the ball b towards address a with high probability of a team-mate being in ball possession afterwards: $reachable_{pass}(b, a)$ with prerequisite $hasBall(p)$

With the Voronoi model presented in Section 2.3, we can define our *reachable* relation as a connection between vertices in the Delaunay triangulation. Note that this approach is only one possibility for implementing reachability. The practical experiences made in robotic soccer show that this model is useful as a mathematical description of all three kinds of reachability.

3.4 Deriving the specification of soccer tactics

For our soccer domain axiomatization, we give successor state axioms for the *ballPos* function (ball position) and *hasBall* fluent as examples. We assume, that the ball position changes only if we pass the ball to a team-mate or dribble with the ball.

$$\begin{aligned} ballPos(do(a, s)) = b \equiv \exists player \exists region \\ ((a = goto(player, region) \wedge ballPos(s) = b) \\ \vee((a = pass(player, region) \vee a = dribble(player, region)) \wedge b = region)) \end{aligned}$$

A player is in possession of the ball if its position is the same as the position of the ball. Of course, the player should be located in a certain area around the ball, but for ease of presentation we leave this out. If the player passes the ball to another position, the fluent value becomes false.

$$\begin{aligned} hasBall(player, do(a, s)) \equiv \exists region \\ ((a = goto(player, region) \wedge ballPos(s) = region) \\ \vee(hasBall(player, s) \wedge \neg\exists region\ a = pass(player, region))) \end{aligned}$$

Please note the difference between effect axioms and successor state axioms. In Section. 2.1 we introduced effect axioms to describe the effects of actions. A successor state axiom is defined for each fluent and defines all possibilities how the value of this fluent is changed by any action (cf. [24]).

With these basic actions and their effects we can easily formalize examples of building up play from Figure 3, starting with Figure 3(a) showing a long pass as first action. There, back player 2 makes a long pass to forward 9, who then passes back to the centre midfielder 10, who can make a pass to forward 11, who cuts in deep down-field, as written in [5, p. 29]. Four agents that are team-mates are actively involved in this manœuvre: back player $p_2 = player[B]$ (whose side needs not to be specified), the centre midfielder $p_{10} = player[CM]$, and two forwards $p_9 = player[xF]$ and $p_{11} = player[yF]$ on different sides, i.e. $x \neq y$.

Before we are able to formalize the whole manœuvre, we have to think about what passing means exactly. A pass from player p to p' requires that p is in ball possession and that the ball can be passed to p', i.e. the logical conjunction $hasBall(p) \wedge reachable(ball, p')$. Afterwards p' is in ball possession, i.e. $hasBall(p')$. In [5, p. 27], three different types of passes are mentioned. They can be formalized by additional constraints:

1. long pass with $p.role = B \wedge p'.role = F$,
2. diagonal pass with $p.side \neq p'.side$, and
3. deep pass with $p.role < p'.role$ where we assume that the roles (which can also be understood as rows in Figure 5(a)) are ordered.

With these definitions and constraints for passing, the tactics in Figure 3(a) can be described by the following program in a straightforward manner:

proc build-up-play
(pass(p_2, p_9); pass(p_9, p_{10})||goto(p_{11}, r));pass(p_{10}, r)
endproc

Algorithm 1: Build-up-play algorithm.

Recall that subsequent actions (sequences) are marked with semicolon; concurrent, i.e. parallel actions are separated by the symbol ||. In addition, $r = region[CF]$ denotes the region in front of the opponent goal. Since we take the allocentric view from the diagrams in [5], we may have parallel actions of different agents (e.g. player 11 running in front of the opponent goal, while player 9 or 10 initiates the pass). Clearly, this has to be turned into an implementation for each agent.

As another example, consider the move depicted in Figure 6 which is a possible move for a counter-attack. There, player 8 just captured the ball from the opponent team and dribbles toward the goal while the forwards (player 9 and player 11) revolve the opponent defence in order to get a scoring opportunity from both corners of the penalty area while player 10 starts a red herring by running to the centre. The white circles represent opponent players. In the original Figure [5, diagram 21] (see Figure 3(b)), there are neither opponent players nor dedicated regions; we inserted them here for illustration purposes.

proc counterattack_21
intercept;
startDribble (*region*[CF]);
waitFor(*reachable*(p_{11}, *region*[*LF*])) $\vee$

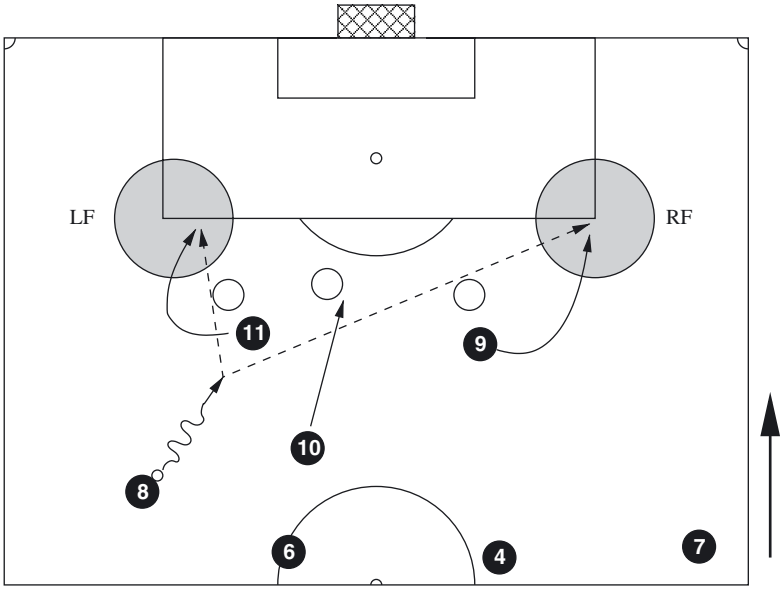


Figure 6: Extended diagram 21 from [5].

```
        reachable (p9, region[RF]) ∨
        ∃x.opponent(x) ∧ Tackles(x));
    endDribble;
    if reachable(p11,region[LF]) then
        pass(region[LF]);
    else if reachable(p9,region[RF]) then
        pass(region[RF]);
    endif
endproc
```

Algorithm 2: Counter-attack program.

Algorithm 2 is from the point of view of player 8, that is, all actions and tests are performed by this player. Player 8 gains the ball with an intercept action. He dribbles towards the centre (denoted by *region*[CF]) until either player 11 or player 9 is able to receive the pass or an opponent forces player 8 to do another action (which is not specified in this example). In the specification above, we use the action pair *startDribble* and *endDribble* instead of a single *dribble* action accounting for temporal aspects of that action. Splitting the dribble action into initiation and termination is a form of implicit concurrency since other actions can be performed while dribbling.

The next step in the presented sequence is a *waitFor* construct. It is used to specify that no further actions are initiated until one of the conditions becomes true, i.e. players 11 or 9 are able to receive a pass in their respective regions or an opponent tackles player 8, i.e. an opponent could probably intercept the ball.

Note that during the blocking of the *waitFor*, the dribbling of player 8 continues and sensor inputs are processed to update the relation *reachable*.

Finally, in the conditional we have to test which condition became true to choose the appropriate pass. Note that we do not choose an action in the case of both player 9 and player 11 cannot receive the pass as this would be the matter of another soccer move procedure. The counter-attack programs for other players participating in this move can be specified similarly.

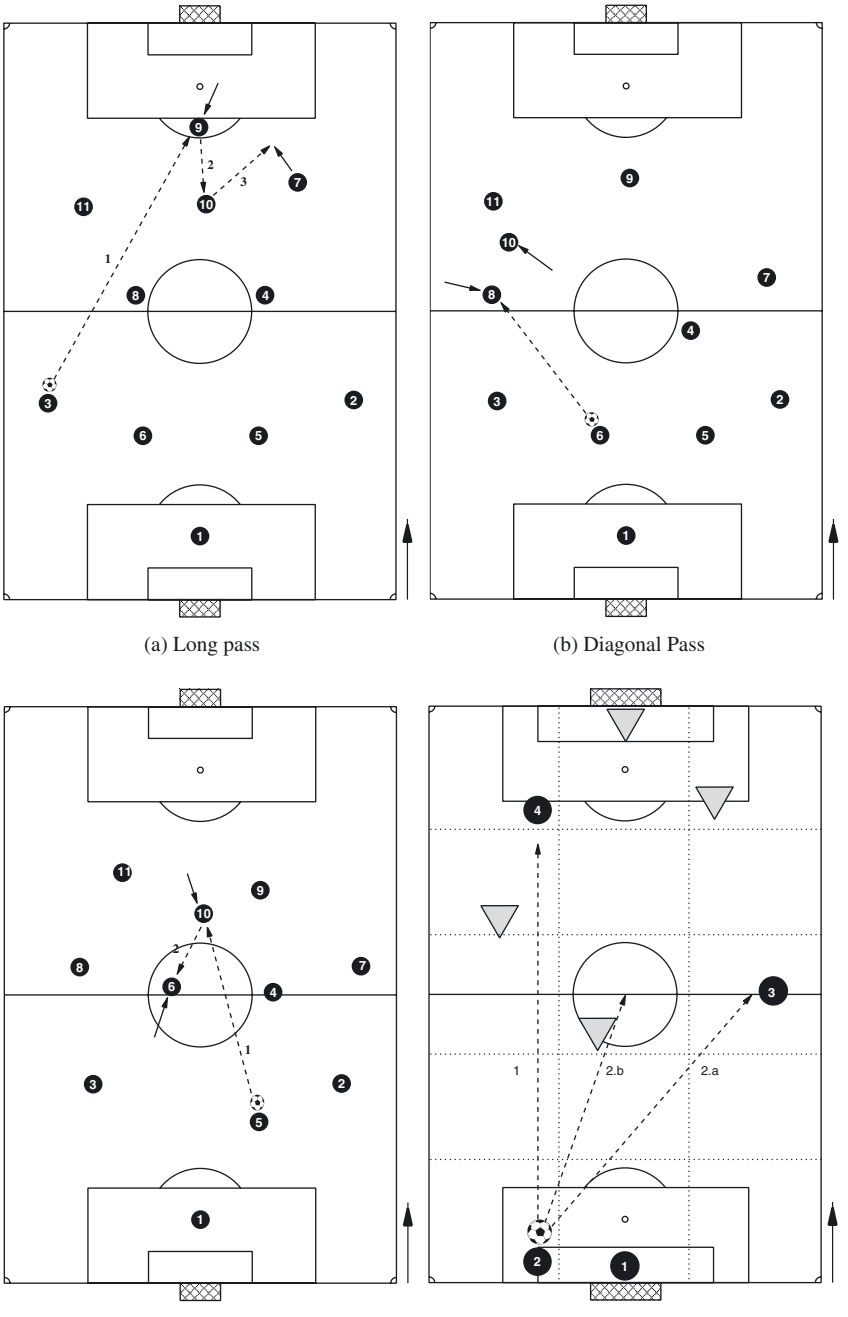
3.5 An example move on a robot

We now specify the soccer move build up play in Readylog and we show that our qualitative world model supports the specification. We adapted three possible ways to build up a play as discussed in [5] (see also Section 2.3).

The first way to build up play is with a long pass (Figure 7(a)). We immediately notice that the term long is one of the coarse, qualitative notions we need to establish in order to adapt human soccer theory for our autonomous soccer agents. We could also formulate this as passing the ball from a back position to a front position on the playing field. The second way to build up play is with a diagonal pass as depicted in Figure 7(b). This time, the term diagonal is of qualitative nature. Diagonal means passing to the side being opposite to the current one. Figure 7(c) shows the last possibility to build up play which is with a deep pass (dashed line labelled with 1) followed by a subsequent back pass (dashed line labelled with 2). The term deep is used to denote the space behind or in between a group of opponent players. The endpoint of such a pass has to be the most free position available in-between or behind the group of opponents.

We now try to adapt as much of these descriptions as possible by integrating their most essential parts into one pattern. All three possibilities have in common that the ball is located in the back part on the pitch. According to a role ontology it is a player currently having a defensive role which is about to initiate the pattern to build up play. We already characterized the possibility of a long pass as bringing the ball to the front part of the pitch. Therefore, in this case the agent chooses to pass to a team-mate located in the attacking zone. For the two other possibilities the destination of the pass is the midfield. The agent can either make a diagonal pass, that is the case if the target position is on the opposite side of the field, or it can simply pass to a free area on the same side or in the centre of the pitch. To illustrate our adaptation of the build up play patterns for the Middle-size league we depicted a diagram similar to the ones in [5] in Figure 7(d).

```
proc build_up_play_defender,
    if haveBall(ownNumber) then
        getFreeSide(offense,FreeSide);
        getPassPartner(offense,FreeSide,PassPartner);
        solve (
          if ¬isKickable(ownNumber ) then
              interceptBall
```



(a) Long pass (b) Diagonal Pass

(c) Back pass (d) Adaptation for the MSL

Figure 7: Example for the *build-up play* move.

```
            else
                if isPassReachable(ownNumber, PassPartner) then
                    passTo(ownNumber, PassPartner)
                endif
            endif
            | pickBest (bestSide,{leftSide, middleSide, rightSide}
                dribbleTo(ownNumber, middleZone, bestSide)
                | kickTo(ownNumber, middleZone, bestSide))
                /* end of pickBest */
            | interceptBall; kickTo(ownNumber, middleZone, middleSide),
        3, func_Reward) /* end solve with horizon 3 */
    else
        interceptBall
    endif
endproc
```

Algorithm 3: The build-up play program for the defender.

Algorithm 3 shows a program in our action language Readylog capturing the above example. Note that this specification contains several qualitative elements such as middleZone, leftSide and offence as well as qualitative predicates such as isPassReachable. With our qualitative world model we are able to simply transfer the qualitative notions from the specification in [5]. Moreover, the use of qualitative terms and predicates makes the program applicable in many game situations.

In the first part of the procedure the agent assigns the most free side in the offensive zone to the variable *FreeSide*. It further picks a team-mate in the offense that may receive a pass. Then the agent starts deliberating with the solve statement. Using decision-theoretic planning it decides which of the options separated by the | to take. The first possibility is to check whether the ball is kickable. If it is not kickable the agent needs to intercept the ball. If the pass partner which was determined initially is reachable with a pass the agent is supposed to carry out the *pass* action. The second possibiblity is to use the *pickBest* statement to choose the best argument for the subsequent action. That is to say, the agent decides to call the action *pass_to* with the best value taken from the list of arguments specified with the *pickBest* statement. This list of positions corresponds to the set of possibilities to build up a play we depicted in Figure 7(d). It can choose here whether to dribble or to kick to the position chosen by *pickBest*. The last possibility available is to intercept the ball and directly kick it towards the centre of the field. Further details and case studies on formalizations can be found in [1, 19].

4 Discussion

In this chapter, we presented an approach towards a formal specification of soccer. For the behaviour specification of our soccer robots we started thinking if it could be helpful to review the soccer literature in order to learn more about how soccer is specified. What showed up was that, on the one hand, there exists

work on soccer strategies and tactics and, on the other hand, that these strategies are not formalized in a way which is appropriate for soccer robots.

As soccer is a structured game, a top-level ontology can be extracted quite easily. Also does the soccer literature mention only few basic primitives like *pass* or *run*. With the appropriate preconditions for these actions one can define a basic action theory for the soccer domain. For our soccer robot application it is possible to define spatial relations like distance or reachability in a qualitative fashion. One has to remark that our approach works for soccer playing robots. Trying to adapt the results to human soccer play in order to gain more insights about human soccer strategies and tactics it becomes obvious that especially the spatial relations have to be adapted, too. One has to find appropriate models which reflect the mobility of human soccer players and which model the possibility to play headers and high cross passes.

4.1 Related works

The benefit of mathematics for detailed analysis and improvement of sports was described formally before, e.g. in [25]. These approaches concentrate in many cases on the optimization of single tactics, as e.g. on downwind sailing in [26]. Other approaches on formalizing sports exist. For example, in [27] tactical patterns for water polo are defined. Similar to our approach basic action primitives are identified and tactical formations are described formally. There is also work in the field of automatic commentating and analysing sports.

Automated commentator systems. Specifically related to the RoboCup domain there are three groups which developed systems for automatic real-time commentary. Their systems together won the scientific award at RoboCup 98 [28]. An overview on the three systems can be found in [29]. Of particular interest are the two systems Rocco and Mike since both try to model and identify certain aspects of soccer.

Rocco evolved from the Soccer system [30] which generates natural language descriptions of a soccer game. It works on information provided by the RoboCup soccer simulation server. Based on elementary events (like *kick* or *catch*) and geometric data provided by the Soccer Server , Rocco performs a high-level scene analysis by an incremental event-recognition. Declarative concepts of events represent a priori knowledge about typical occurrences in a scene. These concepts are organized in an abstract hierarchy, grounded on specialisation and temporal decomposition. There exists a simple recognition automaton for each concept.

Mike also uses the information provided by the Soccer Server as its input. Mike consists of six analysis modules running concurrently that post propositions of information gathered to a pool. One of the six analysis modules also makes use of Voronoi diagrams. Their application enables Mike to determine the defensive areas covered by a player as well as to assess the overall positioning. Players are considered to be free if they are positioned as close as possible to a Voronoi vertex of the diagram of the opposing team. Furthermore, triangular shapes in a Voronoi diagram indicate a tight formation since the average shape

of a Voronoi area is hexagonal. These two observations seem to provide further qualitative insight from a tactical point of view.

Real-time analysis tools. In a recent work [31], Beetz *et al* overview the FIPM system, a real-time analysis tool for soccer games. Based on position data of the players and the ball they interpret common soccer concepts. In [31] they report on first results drawn from data from the RoboCup simulation league. They use first-order interval temporal logic to represent events or situations. Their model consists of five layers comprising a motion, situation, action and tactical layer. On the situation layer they identify concepts like *ScoringOpportunity*. With data mining techniques they assess the conditions for such situations. On the action layer they distinguish between several kind of models. The observation model for example classifies shots to belong to a dribbling or a pass, the predictive model use decision-tree learning to form rules for predicting the success rates of goal shots. They also provide, for example, information about the physical abilities of players based on the distances the player covers during a match and also tactical patterns of a team can be derived. These information are especially useful for soccer coaches.

Miene *et al* [32] report on successful experiments on detecting and predicting offside positions based on data also from the simulation league. They developed an algorithm for rule-based motion interpretation. The rules are given as background knowledge in first-order logic. The input data are first temporally segmented based on thresholds and monotonicity criteria. Then the segmented motion data are mapped into qualitative classes like *no motion*, or *slow*. They use logical representations to model game situations like a player being in an offside position. They are able to detect offside positions successfully and can also predict if a player risks to run into an offside trap. Important to note is that they do not regard static situations but analyse the motion data. This covers especially the dynamic aspects of soccer.

Formalizing teamwork and strategies. One approach for specifying teamwork among others is the *communicative multiagent team decision problem* (COM-MTDP) model [33]. Communication is explicit in this framework and uncertainty can be expressed by probabilities. A reward function allows analysing the optimality of behaviour, which could be done with model checking or theorem proving techniques for Golog or similar specifications in our context. However, the main focus of this paper is on the question *what* should be evaluated, and the answer is dependent on the chosen domain. Therefore, if we want to evaluate the behaviour of multiagent systems for the RoboCup scenario, we need primitives for describing the behaviour of robots in this domain as done here.

In a recent paper [34], the four-legged league champion team NUbots 2006 was tested against more aggressive and more defensive strategies. The results indicate that global team tactics should be considered in conjunction with a team's style of play. A set of metrics was developed which may enable a future robot soccer team to observe, reason and modify its global strategy to suit that of an opposing team.

McMillen and Veloso [35] present a framework for distributed, play-based role assignments. The proposed framework allows to specify several team play strategies on a very high level. In addition, some applicability conditions for each strategy must be given, e.g. second half of the game and the team is winning. Finally, roles (goalkeeper, defender etc.) must be listed. In [35] a successful case study with Aibo robots in the four-legged league is also reported. However, no clear formal semantics of the calculus is stated which we get for free in related approaches e.g. with state machines [36] or the situation calculus as presented in this paper.

4.2 Final remarks

Finally, it will be interesting to see if, in the future, a formal approach to tactics cannot only be useful for computer scientists trying to specify a system but also, this work as a starting point, for sport scientists to build formal models of their respective sports discipline. Future work in this direction for soccer has to concentrate on appropriate models for the mobility of human players to find a unified theoretical model for soccer. We believe that such formal approaches in general will help sport scientists to better understand and to analyse their sport disciplines.

The biggest lesson we learnt is that we *are* able to formalize soccer theory on an abstract level. This might not be surprising, however, some of the concepts real soccer experts use are quite fuzzy and therefore difficult to define and implement. One of the challenges of Computer Science in Sport is to get researchers from sport and computer science together to find a common language. This will help both sides to bring research further.

References

[1] Dylla, F., Ferrein, A., Lakemeyer, G., Murray, J., Obst, O., Röfer, T., Stolzenburg, F., Visser, U. & Wagner, T., Towards a league-independent qualitative soccer theory for RoboCup. *RoboCup 2004: Robot World Cup VIII*, Springer: Berlin, 2005.

[2] Kitano, H., Kuniyoshi, Y., Noda, I., Asada, M., Matsubara, H. & Osawa, E.I., RoboCup: a challenge problem for AI. *AI Magazine*, **18**, pp. 73–85, 1997.

[3] Noda, I., Matsubara, H., Hiraki, K. & Frank, I., Soccer server: a tool for research on multi-agent systems. *Applied Artificial Intelligence*, **12(2)**, pp. 233–250, 1997.

[4] GermanTeam Homepage. Website: www.germanteam.org, 2006.

[5] Lucchesi, M., *Coaching the 3-4-1-2 and 4-2-3-1*. Reedswain Publishing: Spring City, PA, 2001.

[6] Freksa, C., Using orientation information for qualitative spatial reasoning. *Theories and Methods of Spatio-Temporal Reasoning in Geographic Space*, eds. A.U. Frank, I. Campari & U. Formentini, Springer: Berlin, 1992.

[7] McCarthy, J., Situations, actions and causal laws. Technical report, Stanford University, 1963.

[8] Reiter, R., On knowledge-based programming with sensing in the situation calculus. *ACM Transactions on Computational Logic (TOCL)*, **2(4)**, pp. 433–457, 2001.

[9] Ferrein, A., Fritz, C. & Lakemeyer, G., On-line decision-theoretic golog for unpredictable domains. *Proc. of 27th German Conf. on Artificial Intelligence*, Springer: Berlin, pp. 322–336, 2004.
[10] Ferrein, A., Fritz, C. & Lakemeyer, G., Using golog for deliberation and team coordination in robotic soccer. *KI Künstliche Intelligenz*, **1**, pp. 24–29, 2005.
[11] Levesque, H.J., Reiter, R., Lesperance, Y., Lin, F. & Scherl, R.B., GOLOG: A logic programming language for dynamic domains. *Journal of Logic Programming*, **31(1–3)**, pp. 59–83, 1997.
[12] Grosskreutz, H. & Lakemeyer, G., cc-Golog – An action language with continous change. *Logic Journal of the IGPL*, **11(2)**, pp. 179–221, 2003.
[13] De Giacomo, G., Lésperance, Y. & Levesque, H.J., ConGolog, A concurrent programming language based on situation calculus. *Artificial Intelligence*, **121(1–2)**, pp. 109–169, 2000.
[14] De Giacomo, G. & Levesque, H., An incremental interpreter for high-level programs with sensing. *Logical Foundation for Cognitive Agents: Contributions in Honor of Ray Reiter*, eds. H.J. Levesque & F. Pirri, Springer: Berlin, pp. 86–102, 1999.
[15] Grosskreutz, H., Probabilistic Projection and Belief Update in the pGOLOG. *The 2nd International Cognitive Robotics Workshop, 14th European Conference on Artificial Intelligence*, pp. 34–41, Berlin, Germany, 2000.
[16] Boutilier, C., Reiter, R. Soutchanski, M. & Thrun, S., Decision-theoretic, high-level agent programming in the situation calculus. *Proceedings of the 7th Conference on Artificial Intelligence (AAAI-00) and of the 12th Conference on Innovative Applications of Artificial Intelligence (IAAI-00)*, AAAI Press: Menlo Park, CA, pp. 355–362, 2000.
[17] Dylla, F., Ferrein, A. & Lakemeyer, G., Acting and Deliberating using Golog in Robotic Soccer – A Hybrid Architecture. *Proceedings of the 3rd International Cognitive Robotics Workshop*, AAAI Press: Menlo Park, CA, 2002.
[18] Dylla, F., Ferrein, A. & Lakemeyer, G., Specifying multirobot coordination in ICPGolog – from simulation towards real robots. *AOS-4 at IJCAI-03*, pp. 19–28, 2003.
[19] Schiffer, S., Ferrein, A. & Lakemeyer, G., Qualitative world models for soccer robots. *Qualitative Constraint Calculi, Workshop at KI 2006, Bremen*, eds. S. Wölfl & T. Mossakowski, pp. 3–14, 2006.
[20] Hernandez, D., Clementini, E. & Felice, P.D., Qualitative distances. *Spatial Information Theory: a theoretical basis for GIS*, eds. W. Kuhn & A. Frank, Springer-Verlag, number 988 in LNCS, pp. 45–58, 1995.
[21] Clementini, E., Felice, P.D. & Hernandez, D., Qualitative representation of positional information. *Artificial Intelligence*, **95(2)**, pp. 317–356, 1997.
[22] Aurenhammer, F. & Klein, R., Voronoi diagrams. *Handbook of Computational Geometry*, eds. J.R. Sack & J. Urrutia, Elsevier Science Publishers: Amsterdam, 2000.
[23] Peitersen, B. & Bangsbo, J., *Soccer Systems & Strategies*. Human Kinetics: Champaign, IL, 2000.
[24] Reiter, R., The frame problem in the situation calculus: a simple solution (sometimes) and a completeness result for goal regression. *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy*, ed. V. Lifschitz, Academic Press: San Diego, pp. 359–380, 1991.
[25] Patrick, J., The marriage of mathematics and computer technologies for sport improvement. *First Mathematics and Computers in Sport Conference*, Bond University: Australia, 1992.

[26] Tonkes, E., Optimising downwind sailing. *Fifth Mathematics and Computers in Sport Conference*, University of Technology: Sydney, 2000.
[27] Pavicic, L., Some possibilities for the formal definition of a water polo game. *Sport und Informatik II*, ed. J. Perl, Sport und Buch Strauss, 1990.
[28] Asada, M. & Kitano, H. (eds.), *RoboCup-98: Robot Soccer World Cup II*, Springer-Verlag: London, 1999.
[29] André, E., Binsted, K., Tanaka-Ishii, K., Luke, S., Herzog, G. & Rist, T., Three RoboCup simulation league commentator systems. *AI Magazine*, **21(1)**, pp. 57–66, 2000.
[30] André, E., Herzog, G. & Rist, T., The simultaneous interpretation of real world image sequences and their natural language description: The system Soccer, *Proceedings of the 8th European Conference on Artificial Intelligence (ECAI-88), Munich*, ed. Y. Kodratoff, John Wiley and Sons, Inc: Chicester, England, pp. 449–454, 1988.
[31] Beetz, M., Kirchlechner, B. & Lames, M., Computerized real-time analysis of football games. *IEEE Pervasive Computing*, **4(3)**, pp. 33–39, 2005.
[32] Miene, A., Visser, U. & Herzog, O., Recognition and prediction of motion situations based on a qualitative motion description. *RoboCup 2003: Robot Soccer World Cup VII*, eds. D. Polani, B. Browning, A. Bonarini & K. Yoshida, Springer Verlag, vol. 3020 of *Lecture Notes in Computer Science*, 2003.
[33] Pynadath, D.V. & Tambe, M., Multiagent teamwork: analyzing the optimality and complexity of key theories and models. *Proc. of the 1st Int. Joint Conf. on Autonomous Agents & Multi-Agent Systems*, vol. 2, eds. C. Castelfranchi & W.L. Johnson, ACM Press: Bologna, pp. 873–880, 2002.
[34] Quinlan, M.J., Obst, O. & Chalup, S.K., Towards autonomous strategy decisions in the robocup four-legged league. *Proc. of the 7th IJCAI Int. Workshop on Nonmontonic Reasoning, Action and Change*, eds. A. Karol, P. Peppas & M.A. Williams, AAAI Press: Menlo Park, CA, 2007.
[35] McMillen, C. & Veloso, M., Distributed, play-based role assignment for robot teams in dynamic environments. *Proc. of 8th Int. Symposium on Distributed Autonomous Robotic Systems (DARS)*, Minneapolis, 2006.
[36] Murray, J., Obst, O. & Stolzenburg, F., Towards a logical approach for soccer agents engineering. *RoboCup 2000: Robot Soccer World Cup IV*, eds. P. Stone, T. Balch & G. Kraetzschmar, Springer: Berlin, Heidelberg, New York, LNAI 2019, pp. 199–208, 2001.