

Towards a League-Independent Qualitative Soccer Theory for RoboCup ^{*}

Frank Dylla¹, Alexander Ferrein², Gerhard Lakemeyer², Jan Murray³,
Oliver Obst³, Thomas Röfer¹, Frieder Stolzenburg⁴, Ubbo Visser¹, and
Thomas Wagner¹

¹ Center for Computing Technologies (TZI), Universität Bremen, D-28359 Bremen
{dylla, roefer, visser, twagner}@tzi.de

² Computer Science Department, RWTH Aachen, D-52056 Aachen
{gerhard, ferrein}@cs.rwth-aachen.de

³ Universität Koblenz-Landau, AI Research Group, D-56070 Koblenz
{murray, fruit}@uni-koblenz.de

⁴ Hochschule Harz, Automation and Computer Sciences Department
D-38855 Wernigerode, fstolzenburg@hs-harz.de

Abstract. The paper discusses a top-down approach to model 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 RoboCup soccer leagues, i.e. for different hardware platforms. We investigate if and how soccer theory can be formalized such that specification and execution is possible. The advantage is clear: theory abstracts from hardware and from specific situations in leagues. We introduce basic primitives compliant with the terminology known in soccer theory, discuss an example on an abstract level and formalize it. We then consider aspects of different RoboCup leagues in a case study and examine how examples can be instantiated in three different leagues.

1 Motivation

Thinking about the goal of the RoboCup community “to beat the human soccer champion by the year 2050” we start thinking about the human way of playing soccer. Talking to real experts in that field revealed that strategy and tactics play a major part in the game. But a computer scientist is more intrigued by available methods and restrictions that do exist for various reasons (e.g. expressivity of languages used). The following question arises: *Can we apply soccer theory to the RoboCup domain in a way that the majority of the leagues would benefit?* The motivation of this paper is therefore to take an adequate soccer theory book and examine its formalization.

Success in modern soccer games largely depends on the physical and tactical abilities of single players and on the overall strategy that coordinates team behavior whose

^{*} 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 Council (DFG).

goal is to sustain the strength of the individual players and to restrict the abilities of the opponents. Additionally, the use of an appropriate tactic is the foundation for coordinated team behavior. A big advantage of this approach is that the outcome can be applied to more than one RoboCup league. We go further and argue that it is possible to have a team of robots from different institutions that are able to play soccer together.

The paper is organized as follows: We motivated our approach in Sect. 1, introduce basic primitives compliant with the terminology known in soccer theory. We discuss an example on an abstract level formalizing it with Golog as specification language in Sect. 2. We then consider aspects of different RoboCup leagues in a case study and examine how examples can be instantiated in three different leagues in Sect. 3. We discuss our approach in Sect. 4.

2 World Modeling for the Soccer Domain

This section contains a description of how modern soccer knowledge is organized. Nowadays there are many textbooks on soccer theory. Here, we focus on Lucchesi's book [8], because it concentrates on the presentation of tactics (and not on training lessons). We derive basic primitives from [8] and formally specify some soccer tactics.

2.1 The Organization of Soccer Knowledge

According to [8], we interpret a soccer strategy as a tuple $str = \langle RD, CBP \rangle$. With RD as a set of *role descriptions* that describe the overall required abilities of each player position in relation to CBP , the set of *complex behavior patterns* is associated with the strategy. Given the strategy str , the associated role description $rd \in RD$ can be described by the defense tactics task, the offense tactics task, the tactical abilities, and the physical skills. Although soccer strategies in current literature [8, 10] are not as highly structured as strategies for American football, they provide sufficient structure to build up a top-level ontology with respect to specialization and aggregation. According to [8], the offensive phase can be structured into four sub-phases: *gaining ball possession*, *building up play*, *final touch* and *shooting*. In general, there are two ways to build up the play: either we introduce the phase in a counter-attack manner, fast and direct with a long pass or deliberately by a diagonal pass or a deep pass followed by a back pass. In the sequel, we will concentrate on the building-up phase.

2.2 Basic Primitives

Following the lines of [8], we distinguish between *role* (back, midfield, forward) and *side* (left, center, 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. center forward) can be interpreted as *type* of a (human or robotic) soccer player or as *position* (region or point) on the soccer field. Therefore, we basically have nine different positions, as illustrated in Fig. 1(a).

The notions player type and position can be seen as instances or specializations of the notion of an abstract *address*, usually associated with its (actual) coordinates or a

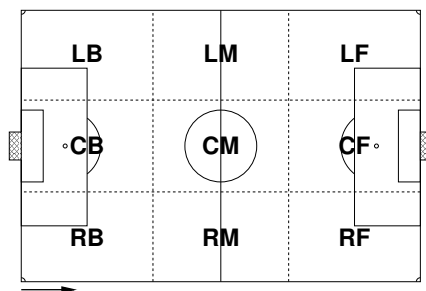
region on the soccer field. Also the ball (strictly speaking, its position) is an address, i.e. the parameter or goal of a test or operation of a soccer player (agent). A movable *object* in the context of soccer may be a player or the ball. An object is in a current *state*, which includes besides other data the current speed or view direction.

Although not explicitly mentioned, a *model of behavior* 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 Fig. 1(b).

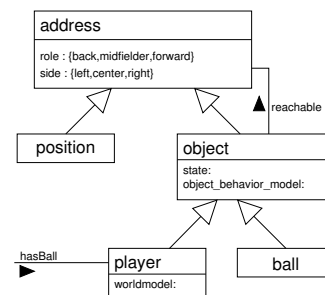
In [8, p. ii] only few symbols are introduced that are used throughout the many diagrams in that book: players (in many cases only the team-mates, not the opponents are shown), the ball, passing, movement of the player receiving the ball, and dribbling. Conceptually, all symbols correspond to *actions*, 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 *address*. *goto(player[LF], region[CF])* e.g. means that the left forward player moves in front of the opponent goal.

Although in most cases this is not explicitly mentioned in [8], actions require that certain prerequisites are satisfied, when they are performed. Since our approach aims at a very abstract and universal (league-independent) formalization of soccer, we restrict ourselves to only two tests: possession of ball and reachability. Each of them can be seen as *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. We will go into further details in Sect. 2.5.



(a) Tactical regions on the field.



(b) Class hierarchy for soccer derived from [8].

Fig. 1. Tactical regions and address hierarchy derived from [8]. The field is divided into three rows (corresponding to player roles) back (B), midfield (M), and forward (F) and three lanes (sides): left (L), center (C), right (R). An address may be one of the nine regions or player types.

2.3 Towards a Formal Specification of Soccer Tactics

For specifying soccer moves we use the logic-based programming language Golog [6]. Golog is a language for reasoning about actions and change and is based on the situation calculus [12]. Properties of the world are described by fluents, functions and relations with a situation term as their last arguments. The way actions change fluents is specified in terms of so-called *successor state axioms*, which also provide a solution to the frame problem. Together with action precondition axioms, axioms for the initial situation, a few foundational axioms and a domain closure and unique names assumption these form the basic action theories [12]. Golog uses basic action theories to define the meaning of primitive actions. In addition it provides familiar control structures like sequence, if-then-else, or procedures to specify complex action patterns. Recent extensions dealing with concurrency, continuous change and time [2, 5] make the language suitable for the soccer domain.

While Golog has been and is used to implement soccer agents [3], we use it here merely as a specification language, because it comes equipped with a formal semantics. As we will see, the language allows a fairly natural representation of typical play situations. 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.

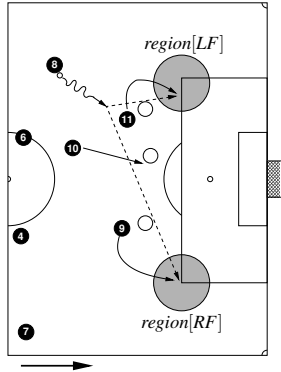
2.4 Example

Fig. 2(a) depicts a possible move for a counter-attack. There, player movements are represented by arrows (\rightarrow or \curvearrowright), passes are indicated by dashed arrows ($--\rightarrow$), and squiggly arrows (\rightsquigarrow) stand for dribbling. Before we are able to formalize the whole manoeuvre, we have to think about what *passing* means exactly. As in several action calculi, we introduce *constraints* associated with this action. A pass from player p to p' requires that beforehand p is in ball possession and 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 [8, p. 27], three different types of passes are mentioned that can be formalized by additional constraints: long pass with $p.role = B \wedge p'.role = F$, diagonal pass with $p.side \neq p'.side$, and deep pass with $p.role < p'.role$ where we assume that the roles (which can also be understood as rows in Fig. 1(a)) are ordered.

In Fig. 2(a), player 8 just captured the ball from the opponent team, dribbles toward the goal while the forwards (player 9 and player 11) revolve the opponent defense 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 center. The white circles represent the opponents.⁵

The counter-attack can be specified with Golog as shown in Fig. 2(b). The program is from the view of player 8, that is, all actions and tests are performed by this player.

⁵ In the original figure (diagram 21 in [8]) there are no opponent players as well as no dedicated regions; we inserted them here for illustration purposes.



(a) Extended Diagram 21 from [8].

```

proc counterattack_21
  intercept;
  startDribble(region[CF]);
  waitFor(reachable(p11, region[RF]) ∨
    reachable(p9, region[LF]) ∨
    ∃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]);
endproc

```

(b) The specification in Golog.

Fig. 2. Counter-attack example.

Player 8 gains the ball with an intercept action. He dribbles toward the center (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. We omit further technical details and refer to [5].

The next step in the presented sequence is a $waitFor$ construct. Its meaning is that no further actions are initiated until one of the conditions becomes true, i.e. player 11 or 9 are able to receive a pass in their respective region or an opponent tackles player 8, i.e., an opponent can intercept the ball (go-reachability). It is perhaps worth mentioning that during the blocking of the $waitFor$ the dribbling of player 8 continues and sensor inputs are processed to update the relation $reachable$, which is discussed in more detail in Sect. 2.5. See [5] for details of how sensor updates can be formalized in Golog.

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 neither player 9 nor player 11 can receive the pass as this would be the matter of another soccer move procedure. The counter-attack programs for the other players can be specified similarly.

2.5 Reachability

For our theory, reachability is central. As our theory aims at being a general one for different soccer leagues, we do not have a specific reachability relation. Building a specific reachability relation is dependent on the league and even within a league, it depends on abilities of single robots or agents. However, the different reachability relations share some properties independent of the league. In general, we can distinguish three different reachability relations:

1. 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 $\neg hasBall(p)$
2. 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 $hasBall(p)$
3. 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 $hasBall(p)$

We are aware of the fact that we need as precise world knowledge as possible, e.g. current positions and speed, for determining the future ball possession like above. Additionally we need assumptions on future behaviors, e.g. the ball path after being kicked. While for team-mates we know the agent's internal structure we may conclude possible future actions with high probability. About opponent agents a lot less is known and therefore predictions are more uncertain. The uncertainty of world data is quite different over the leagues. In the simulation league world data is quite reliable while in the four-legged league, e.g. position estimations are not very accurate.

Many different implementations of reachability can be thought of for the different leagues. The use of Voronoi diagrams and their dual, the Delaunay triangulation (see e.g. [1]) has been proven useful in the past, especially in the simulation league. Here only direct neighboring players, team-mates and opponents, are connected. Note that a direct approach with Voronoi diagrams is only one possibility for implementing reachability. It will only be applicable for robotic soccer, if all agents more or less have the same physical abilities in each region on the soccer field.

3 The RoboCup as Case Study

So far, we have only presented a very abstract way of describing team-play and cooperative moves in soccer. We investigated the reachability relation, that forms a central part of the theory, and discussed some of the underlying models and assumptions, as well as the simplifications we made, but nothing has been said about the concrete problems that arise when one tries to actually carry out the specified moves. Thus, in this section we will discuss possible ways of realizing the abstract specifications in the mid-size, simulation, and legged league.

Mid-Size League. The design of robots in the mid-size league underlies only few restrictions like the maximum size of robots. As the robots are fully autonomous, one central problem is the perception of the environment and dealing with actuators like ball kicking devices. Therefore, many problems in this league rather deal with low-level problems, e.g. vision or ball handling, than with high-level aspects (team-play). Concerning the primitive actions as in the example in Sect. 2.4, goto, pass, and dribbling facilities are needed.

On the other hand, it was shown in [3] that Golog with extensions like decision-theoretic planning or probabilistic projection can be applied in the mid-size league in the *RoboCup* (2003) and is really competitive and, thus, it should be possible to adapt the moves described in [8].

Simulation League. From a technical point of view, the simulation league is suited best for carrying out the tactics presented in [8]. First of all, this is the only league where teams of 11 players play against each other. So the number of players that is needed for making the presented moves is given. In addition, the skills of the players are developed well enough in this league, too, such that team-play can easily be realized.

As dribbling is an expensive and potentially unsafe behavior in the simulation league, and passing is preferred, we focus on describing possibilities of implementing *pass reachability* (see Sect. 2.5). The reachability of a pass partner is usually determined by checking a cone from the player with the ball towards a potential pass recipient. If this cone is free of opponent players, the recipient is reachable with a pass. A possible implementation is presented in [13].

Sony Four-Legged Robot League. As in the mid-size league, the four players are too few for carrying out tactical diagrams such as the ones in [8], and, moreover, with the current field size there is no need for passing the ball.

One method used in this league by the GermanTeam [14] to describe robot behavior is the *Extensible Agent Behavior Specification Language (XABSL)* [7]. It describes behavior in the form of a hierarchy of state machines, so-called *options*, using XML. In each option, the current *state* defines which sub-option or which *basic behavior* (some pre-coded routine such as *pass*) is active. So at each point in time, a path from the root option through several levels of other options to a basic behavior is active. This path changes whenever the current state of an option is changed to another one based on a decision tree. In principle, the behavior of player 8 in diagram 21 in [8] (counter-attack) can be modeled in XABSL. As the XML description would be out of proportion for this paper, we refer the reader to the long version [4].

4 Conclusions

As mentioned in Sect. 2.5, the concept of reachability is important. *Reachable* can be based on some qualitative information such as distance (e.g., near, intermediate, far away) or orientation (e.g., front-left, right, back-right). Examples of how to use qualitative spatio-temporal knowledge and reasoning for the RoboCup can be found in [9, 15, 16].

The intention of our investigation was to apply soccer theory as stated in soccer expert books to the RoboCup soccer domain. Our motivation was the significance of strategies and tactics in real soccer games. The main goal was to figure out whether we would be able to find an abstract level of formalization that enables us to bring benefit to multiple soccer leagues in RoboCup. We have chosen two examples for counter-attacks and used Golog as specification language. Please note that Golog is only one example for a specification language.

The biggest lesson learned 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. A prominent example is the concept of *reachability*, which is used in our examples. It turned out that the definition plays a crucial part in the implementation.

Our case study revealed that there is a lot of work to be done. There are still many problems concerning the low-level skills such as receiving the ball in the mid-size league. The simulation league has an excellent platform for this kind of experiments. The move has been implemented prototypically for the simulation league teams of both, RoboLog Koblenz⁶ and Allemaniacs Aachen as a proof of concept. We will carry out systematic experiments in the future. The behavior of the German Team in the Sony legged league is modeled in XABSL as mentioned in the previous section. It turned out that the abstract behavior could be modeled and therefore implemented.

References

1. F. Aurenhammer and R. Klein. Voronoi diagrams. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, chapter 5, pages 201–290. North-Holland, 2000.
2. G. De Giacomo, Y. Lésperance, and H. J. Levesque. ConGolog, A concurrent programming language based on situation calculus. *Artificial Intelligence*, 121(1–2):109–169, 2000.
3. F. Dylla, A. Ferrein, and G. Lakemeyer. Specifying multirobot coordination in ICPGolog – from simulation towards real robots. In *Proc. of the Workshop on Issues in Designing Physical Agents for Dynamic Real-Time Environments: World modeling, planning, learning, and communicating (IJCAI 03)*, 2003.
4. F. Dylla, A. Ferrein, G. Lakemeyer, J. Murray, O. Obst, T. Röfer, F. Stolzenburg, U. Visser, and T. Wagner. Towards a league-independent qualitative soccer theory for RoboCup. Technical report, 2004. Forthcoming.
5. H. Grosskreutz and G. Lakemeyer. On-line execution of cc-Golog plans. In *Proc. of IJCAI-01*, 2001.
6. H. J. Levesque, R. Reiter, Y. Lesperance, F. Lin, and R. B. Scherl. GOLOG: A logic programming language for dynamic domains. *Journal of Logic Programming*, 31(1-3), 1997.
7. M. Löttsch, J. Bach, H.-D. Burkhard, and M. Jüngel. Designing agent behavior with the extensible agent behavior specification language XABSL. In Polani et al. [11].
8. M. Lucchesi. *Coaching the 3-4-1-2 and 4-2-3-1*. Reedswain Publishing, 2001.
9. A. Miene, U. Visser, and O. Herzog. Recognition and prediction of motion situations based on a qualitative motion description. In Polani et al. [11].
10. B. Peitersen and J. Bangsbo. *Soccer Systems & Strategies*. Human Kinetics, 2000.
11. D. Polani, B. Browning, A. Bonarini, and K. Yoshida, editors. *RoboCup 2003: Robot Soccer World Cup VII*, LNAI Series. Springer, Berlin, Heidelberg, New York, 2004.
12. R. Reiter. *Knowledge in Action*. MIT Press, 2001.
13. B. Riedel. Developing similarity measures for comparing game situations in RoboCup. Diplomarbeit, Knowledge Based Systems Group, RWTH Aachen, Aachen, Germany, 2002.
14. T. Röfer, I. Dahm, U. Düffert, J. Hoffmann, M. Jüngel, M. Kallnik, M. Löttsch, M. Risler, M. Stelzer, and J. Ziegler. Germanteam 2003. In Polani et al. [11].
15. F. Stolzenburg, O. Obst, and J. Murray. Qualitative velocity and ball interception. In M. Jarke, J. Köhler, and G. Lakemeyer, editors, *KI-2002: Advances in Artificial Intelligence – Proceedings of the 25th Annual German Conference on Artificial Intelligence*, LNAI 2479, pages 283–298, Aachen, 2002. Springer, Berlin, Heidelberg, New York.
16. T. Wagner, O. Herzog, and U. Visser. Egocentric qualitative spatial knowledge representation for physical robots. In C. Schlenoff and M. Uschold, editors, *AAAI Spring Symposium*, Stanford, CA, 2004. AAAI Press. To appear.

⁶ Thanks to Heni Ben Amor for the implementation.