Towards a League-Independent Qualitative Soccer Theory for RoboCup

Frank Dylla, Alexander Ferrein, Gerhard Lakemeyer, Jan Murray, Oliver Obst, Thomas Röfer, Frieder Stolzenburg, Ubbo Visser, Thomas Wagner

6/2004

Universität
Koblenz-
Landau

Fachbereich
Informatik

unifko

Fachberichte
INFORMATIK

Towards a League-Independent Qualitative Soccer Theory for RoboCup [*]

Frank Dylla[1], Alexander Ferrein[2], Gerhard Lakemeyer[2],
Jan Murray[3], Oliver Obst[3], Thomas Röfer[1],
Frieder Stolzenburg[4], Ubbo Visser[1], Thomas Wagner[1]

[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

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

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. On the one hand, talking to real experts in that field revealed that strategy and tactics play a major part in the game. But on the other hand, computer scientists are 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 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 teams or 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 the context of related approaches for (qualitative) world modeling in Sect. 4 and draw conclusions, giving an outlook about further research issues in Sect. 5.

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 [9], because it concentrates on the presentation of tactics (and not on training lessons). We derive basic primitives from [9] and formally specify some soccer tactics.

2.1 The Organization of Soccer Knowledge

In contrast to ball games like American football, which is strongly based on a more or less static taxonomy of patterns of behavior, soccer strategies are described in a more abstract fashion, e.g. by a set of examples that sums up the *key idea* of a tactical pattern. As a result, individual players in soccer have a much higher degree of flexibility in choosing an appropriate instantiation of a given pattern of tactical behavior.

According to [9], 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
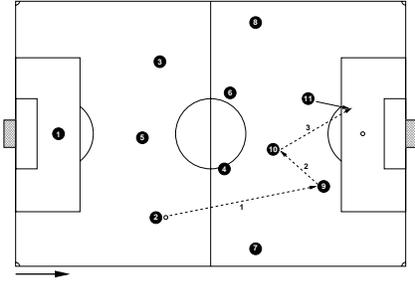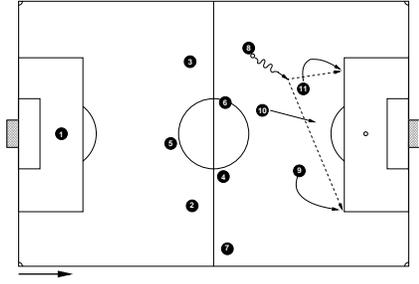
(a) Diagram 4 from [9].   (b) Diagram 21 from [9].

Figure 1: Two tactical diagrams from [9]. 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 ($--\rightarrow$), and squiggly arrows ($\rightsquigarrow$) stand for dribbling. Opponents are not shown.

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 [9, 13, 19] 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 [9], 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 (either by a deep pass of the intercepting player, or by a short pass to a team-mate who afterwards runs downfield and plays a deep pass, or by a diagonal run to a free side before playing the deep 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. Fig. 1 shows two example diagrams from [9].

2.2 Basic Primitives

Following the lines of [9], 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. 2(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

(a) Tactical regions on the field.

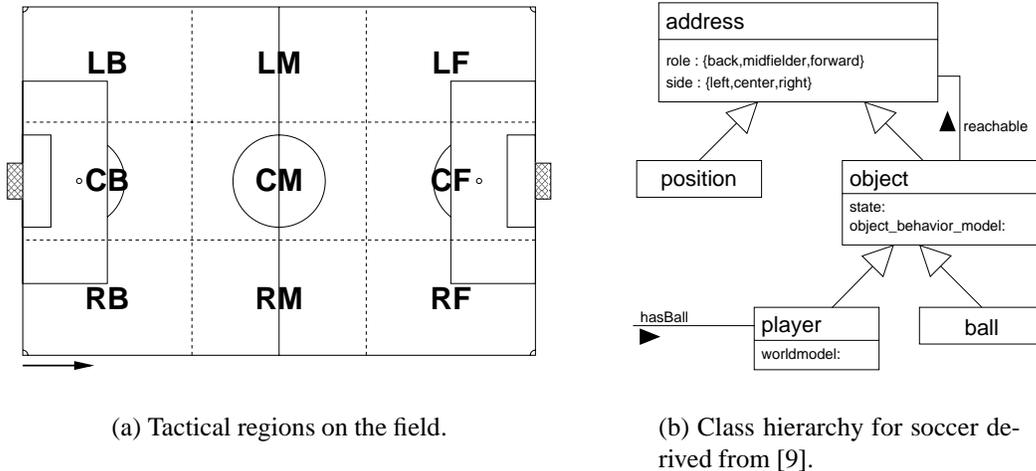(b) Class hierarchy for soccer derived from [9].

Figure 2: Tactical regions and address hierarchy derived from [9]. 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.

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. 2(b).

In [9, 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 [9], 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.

2.3 Towards a Formal Specification of Soccer Tactics

For specifying soccer moves we use the logic-based programming language Golog [7]. Golog is a language for reasoning about actions and change and is based on the situation calculus [15]. 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 [15]. 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 [3, 6] make the language suitable for the soccer domain.

While Golog has been and is used to implement soccer agents [4], 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. 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 soccer domain axiomatization we give successor state axioms for the *ballPos* function (ball position) and *hasBall* fluent as examples. There we assume, that the ball position only changes if we pass the ball to a team-mate or if we dribble with the ball.

$$ballPos(do(a, s)) = b \equiv \exists player \exists region$$
$$\Big((a = goto(player, region) \wedge ballPos(s) = b)$$
$$\vee \big((a = pass(player, region) \vee a = dribble(player, region)) \wedge b = region \big) \Big)$$

A player is in ball possession if his position is the same as the ball position. 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.

$$hasBall(player, do(a, s)) \equiv \exists region$$
$$\Big((a = goto(player, region) \wedge ballPos(s) = region)$$
$$\vee \big(hasBall(player, s) \wedge \neg \exists region\, a = pass(player, region) \big) \Big)$$

2.4 Example

Let us now try to formalize the examples of building up play from Fig. 1, starting with Fig. 1(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 center midfielder 10, who can make a pass to forward 11, who cuts in deep downfield, as written in [9, p. 29]. Four agents that are team-mates are actively involved in this manœuvre: back player $p_2 = player[B]$ (whose side need not to be specified), the center 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. 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 [9, 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. 2(a)) are ordered. With these definitions and constraints for passing, the tactics in Fig. 1(a) can be described as a sequence of actions in a straightforward manner:

proc *build-up-play*_4 $\Big(\big(pass(p_2, p_9); \ pass(p_9, p_{10}) \big) \| goto(p_{11}, r) \Big); \ pass(p_{10}, r)$ **endproc**

Here, 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 [9], 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. We stated this example in Golog notation from a logical point of view. Of course, other representations are possible, e.g. with statecharts [12, 11] or more procedurally, which we will do next with Fig. 1(b).

Fig. 3(a) depicts a possible move for a counter-attack. There, 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. In the original figure (diagram 21 in [9], see also Fig. 1(b)), there are no opponent players as well as no dedicated regions; we inserted them here for illustration purposes.

The counter-attack can be specified with Golog as shown in Fig. 3(b). The program is from the 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 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

6
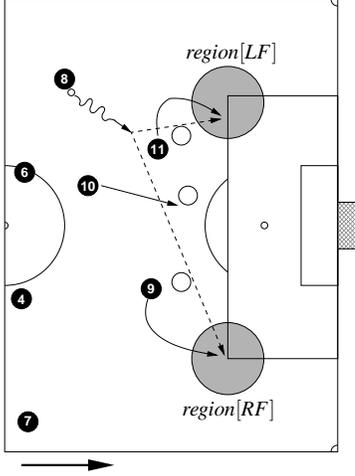
(a) Extended diagram 21 from [9].

```
proc counterattack_21
    intercept;
    startDribble(region[CF]);
    waitFor(reachable(p_11, region[RF])∨
            reachable(p_9, region[RF])∨
            ∃x.Opponent(x) ∧ Tackles(x));
    endDribble;
    if reachable(p_11, region[LF])
      then pass(region[LF]);
        else if reachable(p_9, region[RF])
                then pass(region[RF]);
endproc
```

(b) The specification in Golog.

Figure 3: Counter-attack example.

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. Additionally in the formalism of the situation calculus, a (linear) model of the action with specific agent abilities as parameters, e.g. starting point and average velocity, is started in the background. Therefore the agent can keep track of time and distance passed by while dribbling. We omit further technical details and refer to [6].

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 (which means go-reachability, see Sect. 2.5). 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*. See [6] 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. Since our theory aims at being a general one for different soccer leagues, we only have an abstract 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

7

independent of the league. 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 $\neg hasBall(p)$

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 $hasBall(p)$

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. Assumptions on action facilities will not only vary heavily between the leagues. Sometimes absolutely different assumptions have to be made from team to team within a specific league. The mid-size league is a good example for this with all the different hardware. At this point we want to emphasize that we are able to formalize the mentioned aspects within the specification language Golog. However, we refer to [5] for more details.

2.6 A Possible Implementation for Reachability

Many different implementations of reachability can be thought of for the different leagues. Due to the uncertainties in real play, it is hard to determine a set of parameters that are sufficiently expressing the reachability relation on one hand, and efficiently computable on the other. From our point of view, reachability can be defined via a graph structure with players as nodes and certain weights and values derived from agent abilities attached to the edges. Such values *might* be the length of the passway, which leads us to Voronoi diagrams. A Voronoi diagram is the partitioning of a plane with n points into n convex polygons such that each polygon contains exactly one point and every point in the given polygon is closer to its central point than any other.

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

(a) Initial situation.
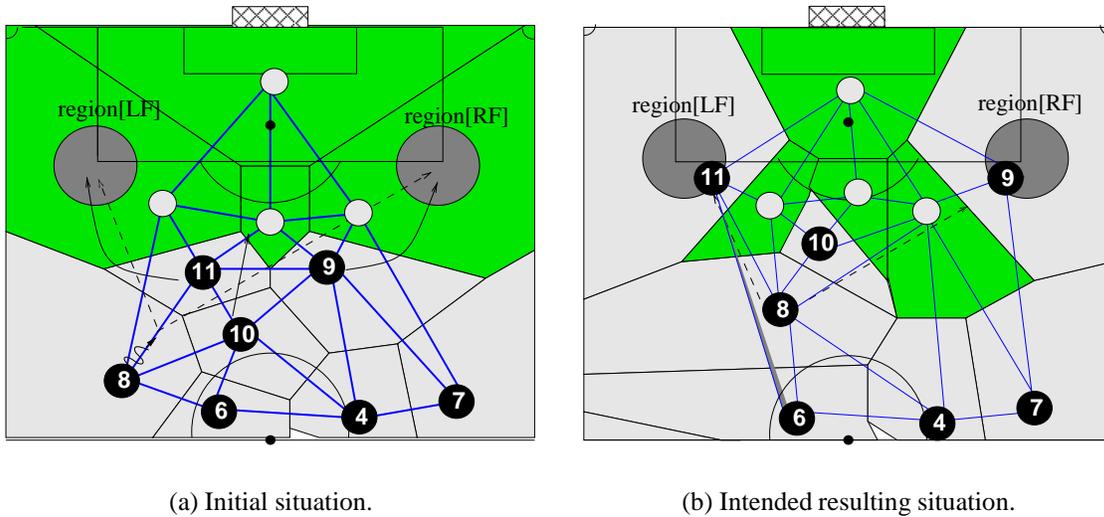
(b) Intended resulting situation.

Figure 4: Delaunay triangulation (blue) and Voronoi regions (green/gray) for the counter-attack example.

region on the soccer field. In other cases, refined or completely different approaches should be used.

In Sect. 2.2, the reachability relation is needed to express spatial relationships between the players and the ball. It was already used in the specification of the counter-attack example (Fig. 3) to support spatio-temporal aspects of that move. In this section we want to continue the counter-attack example to illustrate how the relationship *reachable* could be expressed. There exist lots of complex calculi addressing this problem. Here, however, we want to present only one possible approach which is expressive enough for our purpose and not costly in terms of computational complexity as this relationship is central to the spatial aspects and therefore must be computed quite often in a particular implementation.

We want to make use of Voronoi diagrams and their dual, the Delaunay triangulation (see e.g. [1]), as they separate the field into non-intersecting regions and we get a connection graph between the players. Fig. 4 depicts the Delaunay triangulation and the Voronoi regions for the positions of the players in the counter-attack example. The blue lines represent the triangulation, the gray and green shaded regions correspond to the Voronoi regions of the attacking team and the defending team, respectively. Fig. 4(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 here.

The Voronoi diagram gives us information about which position on the field is closer to which player. In Fig 4(a) we draw the conclusion that the opponent defense controls the goal area, whereas after the successful counter-attack the defense line is penetrated. The triangulation gives information 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. The go and dribble reachability is also covered by these diagrams. In Fig. 4(a) player 9 and player 11, respectively, can test if their target regions are occupied by

9

opponents and they can also test the distance of the defenders to their particular region. With a similar argument the dribble reachability can be expressed. Player 8 can dribble the ball as long as no opponent is in a distance where it can tackle player 8.

With this we can define our *reachable* relation as a connection between vertices in the Delaunay triangulation. Note again that this approach 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. League-specific problems and stumbling blocks that make it difficult or even impossible to carry out the plays from [9] are addressed, as well as potential solutions to them.

3.1 Mid-Size League

In the mid-size league competitions teams play on fields sized about 10 m × 5 m with four fully autonomous robots per team. The additional usage of one external computer for coordination purposes is allowed. 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.

While goto and kicking abilities are becoming more and more reliable for many teams, only few teams show good dribbling abilities, but e.g. the team *Fusion* during the *RoboCup World Championship* 2003 (RC03). Performance of ball control while receiving a pass (which can be understood as intercept) is even worse, though prototypes of ball stopping devices were presented at RC03. Therefore, a secure pass is merely possible with the robot hardware at the moment. Combining the small field size and the poor ball handling abilities executing coordinated team-play as described in [9] is rather questionable. Certainly, only simpler moves than the ones presented here could be applied to the mid-size league which also must be adapted to the three field players available. The RoboCup committee is concerned about this problem and, in this year's competition rules, the field size and number of players will be changed. We have to wait for first experiences to draw further conclusions how this will affect the team-play.

On the other hand, it was shown in [4] 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. It should therefore be possible to apply some of the moves described in [9] to the mid-size league, possibly in an adapted and simpler form than those presented in Sect. 2.4. Offensive team plays can only be modeled with at least three robots due to the team size. Because of the teams being designed quite differently, it is hard to make any general predictions about physical capabilities, e.g. the maximum turning velocity.

3.2 Simulation League

From a technical point of view, the simulation league is suited best for carrying out the tactics presented in [9]. 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. The abilities of players may differ, e.g. their stamina or maximal speed, such that a player can outrun his opponent, which is important for the success of many plays. 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 [16]. In Fig. 5 a visualization together with its parameters is presented. The intention is playing a pass from player *A* to player *B*. If at least one opponent (e.g. no. 3) is located within the inner blue cone a pass would be rather insecure. If the closest opponent is within the light blue zone the of chance playing a successful pass is much higher. If no player is situated in the colored cones (only player 1 being present) a pass will be definitely successful, omitting any sort of ball handling errors at this point. The required parameters, shown in 5(b), have to be determined by means of features of the game, e.g. maximum speed of ball and players. This model has been shown a good compromise between computational complexity and adequacy.

One of the drawbacks of the simulation league is that passes and shots can easily be intercepted, because of the restriction to 2 dimensions and the simple physics of the simulation which allow for very precise prediction of future ball position and velocity. This makes it difficult to realize team-plays like in Fig. 1(a) that contain long passes.

But this will change when the current 2D simulator is replaced by a physical 3D simulator. Once the simulation is full 3D it is possible to give longer passes for two reasons. First, the movement of the ball cannot be predicted as easily, and second, as the ball may be lifted into the air, longer passes cannot be intercepted in the middle of the trajectory. Reachability of a team-mate can then be established by testing regions around the kicker and the recipient of the pass for opponent players.
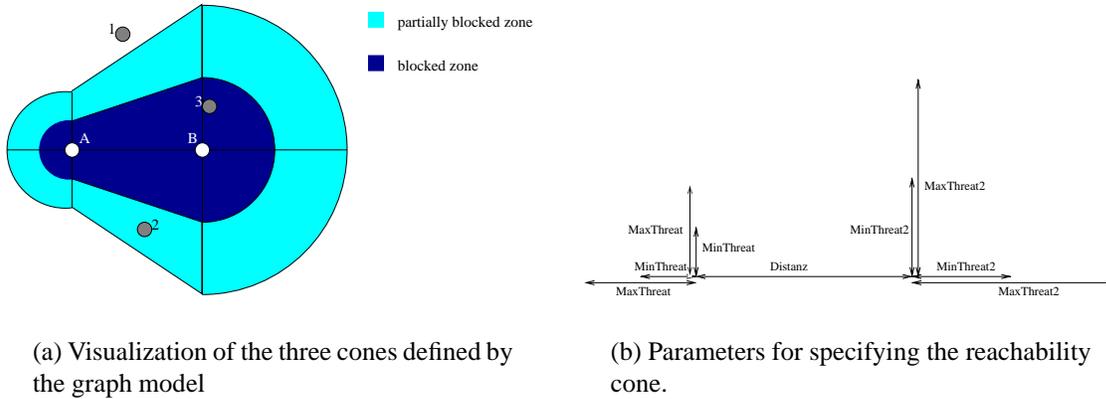
(a) Visualization of the three cones defined by the graph model

(b) Parameters for specifying the reachability cone.

Figure 5: Visualization and important parameters for modeling pass reachability with the graph model taken from [16].

3.3 Sony Four-Legged Robot League

In the Sony four-legged robot league, teams consisting of four Sony Aibo robots each play on a field of 4.2 m × 2.7 m. The robots operate fully autonomously, i.e. there is no external control, neither by humans nor by computers, and they can communicate with each other via a wireless network. As in the mid-size league, the four players are too few for carrying out tactical diagrams such as the ones in [9]. In addition, the field is very small. In the previous years, teams that only dribble the ball were quite successful, so with the current ratio between robot speed, ball speed, and field size, there seems to be no real need for passing the ball. However, in 2003 first ball stopping behaviors were implemented and a few passes were seen. The field size can be expected to be increased for 2005, but the number of robots will almost certainly not.

One method used in this league by the GermanTeam [17] to describe robot behavior is the *Extensible Agent Behavior Specification Language* (*XABSL*) [8]. It describes behaviors 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 [9] (counter-attack) can be modeled in XABSL.

The option *counter-attack-21* only describes the dribbling and the decision whether to pass to the left or the right region (both in state *dribble*). Everything else (detecting that player 8 just got the ball from an opponent and whether he is tackled or even lost the ball) has to be described in higher option in the hierarchy, because the actions to perform in these cases are not specified. If player 8 is able to pass the ball to one of his team-mates, the option will reach one of its final states (*pass left* or *pass right*), in which it will stay until it is not active anymore, i.e. the higher option has activated another sub-option. The next time it is reactivated, *counter-attack-21* will start in its initial state *dribble* again.
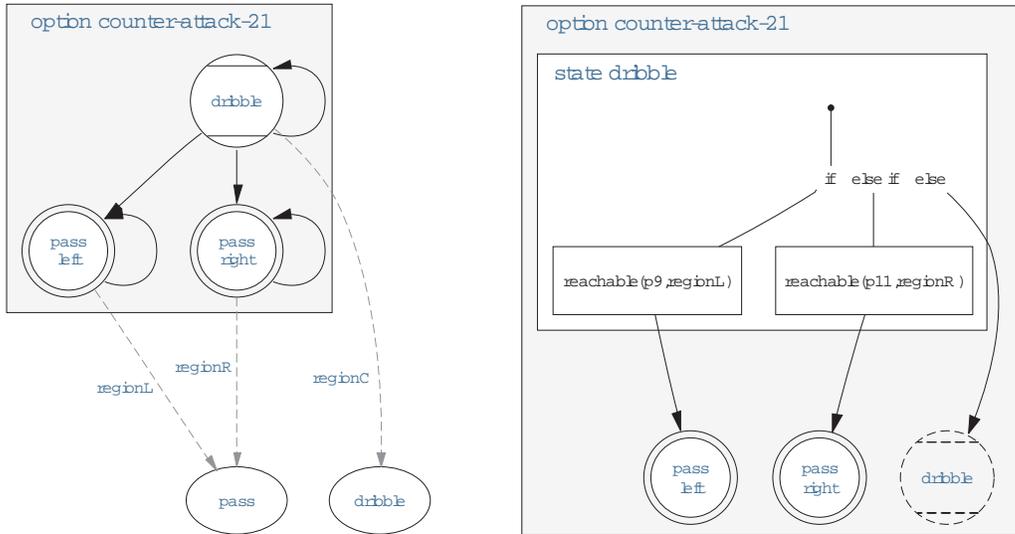
(a) The option *counter-attack-21*.   (b) The state *dribble*.

Figure 6: Modeling the behavior of player 8 shown in Fig. 3 in XABSL. The initial state is depicted as a circle with lines, final states as double circles, basic behaviors as ellipses.

4 Discussion

As mentioned in Sect. 2.5, the concept of reachability is important. Our approach makes use of Voronoi diagrams, which is one way of dealing with the problem of defining the concept of reachability. This way, however, is an *implicit* approach, i.e. our reachable relation is a connection of vertices in the Delaunay triangulation. Other approaches are also possible.

If we talk about regions that can be reached we obviously deal with space. We therefore would like to discuss an *explicit* approach: qualitative spatial modeling. When dealing with spatial world modeling in the context of real world agents one has to distinguish between several aspects how to model the data. Roughly two categories can be distinguished: *(a) frame of references*, and *(b) level of precision*.

4.1 Frame of Reference

A Frame of Reference (FoR) is a set of assumptions or conditions which describe how data have to be understood in the given context. A FoR always has a well-defined origin. From a psychological view the term FoR refers to the association between stimuli from the external environment and the internal qualities of an individual.

First we have to distinguish between the intrinsic aspects, i.e. properties that are purely based in the agent itself (e.g., the agent's front or its energy level), and extrinsic aspects, i.e. based on the agents interaction with the world, e.g. the agent's speed.

13

A second distinction is between an allocentric FoR and an egocentric one. An allocentric view can be compared with data from a bird's view, e.g. turning from direction north to east, while an egocentric frame is based in the agent itself, e.g. turning right. If the relation between the allocentric origin and the individuum is given, the representations can be transformed into each other.

Quite comparable is the differentiation between an absolute and a relative frame of reference. In the first case all measurements are based on a static origin, e.g. the middle point or the agent's speed in m/s, while in the latter this base is dynamic, e.g. a position estimation or speed relative to another agent on the field.

Based on the agents' facility to exchange data we must categorize global and local data as well. While local data is based on the perception of a single agent, global data is based on the perception of several agents. In the presence of faulty or noisy data, methods for dealing with have to be specified, otherwise the global view is corrupted.

4.2 Level of Precision

The second category deals with the level of precision and abstraction of data. An important question to all autonomous agents is how much we can rely on the measurements or results given by sensors or high-order modules, e.g. the degree of uncertainty of range data is higher for sonar sensors than for laser sensors. While quantitative data is purely metric qualitative spatial representations try to abstract from metric details to make predictions about spatial relations [2].

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). Player 11 in Fig. 1(b) would thus be able to decide whether he can carry out the action. This however does not include the temporal information that might also by needed. Player 11 might be able to reach the region but he might not be there in time. Therefore, we have to bear in mind that temporal information has to be considered.

Examples of how to use qualitative spatio-temporal knowledge and reasoning for the RoboCup can be found in [10, 18, 20]. [10] shows how spatio-temporal relations can be used to recognize and predict motion situations. [18] makes use of a qualitative concept of velocity. [20] propose an egocentric qualitative spatial knowledge representation for physical robots based on panoramas.

5 Conclusions

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. We used Voronoi diagrams as a first idea and they operate fine, but processing time will be a problem since the calculation takes a while on real robots.

A promising alternative is to use qualitative knowledge in order to calculate the information we need (e.g., free regions, reachability, etc.). A big advantage of qualitative spatial and temporal knowledge is the processing time. The agents' world model can be updated quite quickly and some of the information can be in a qualitative manner anyhow. It is, for instance, not necessary for a player to know where exactly the point-to-play is (in terms of (x, y)-positions). The agent just needs to know the region, e.g., *front-left*, where he has to play the ball into.

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 Koblenz[*] 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.

Future work includes the experiments in both the mid-size and the simulation league in order to get decent results for our approach. Our current experiments are rather encouraging.

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] Anthony G. Cohn and Shyamanta M. Hazarika. Qualitative spatial representation and reasoning: An overview. *Fundamenta Informaticae*, 46(1-2):1–29, 2001.

[3] Giuseppe De Giacomo, Yves Lésperance, and Hector J. Levesque. ConGolog, A concurrent programming language based on situation calculus. *Artificial Intelligence*, 121(1–2):109–169, 2000.

[4] 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 Design-*

---

[*]Thanks to Heni Ben Amor for the implementation.

ing Physical Agents for Dynamic Real-Time Environments: World modeling, planning, learning, and communicating (IJCAI 03), 2003.

[5] Henrik Grosskreutz. *Towards More Realistic Logic-Based Robot Controllers in the GOLOG Framework*. PhD thesis, RWTH Aachen, 2002.

[6] Henrik Grosskreutz and Gerhard Lakemeyer. On-line execution of cc-Golog plans. In *Proc. of IJCAI-01*, 2001.

[7] Hector J. Levesque, Raymond Reiter, Yves Lesperance, Fangzhen Lin, and Richard B. Scherl. GOLOG: A logic programming language for dynamic domains. *Journal of Logic Programming*, 31(1-3), 1997.

[8] M. Lötzsch, J. Bach, H.-D. Burkhard, and M. Jüngel. Designing agent behavior with the extensible agent behavior specification language XABSL. In Polani et al. [14].

[9] Massimo Lucchesi. *Coaching the 3-4-1-2 and 4-2-3-1*. Reedswain Publishing, 2001.

[10] Andrea Miene, Ubbo Visser, and Otthein Herzog. Recognition and prediction of motion situations based on a qualitative motion description. In Polani et al. [14].

[11] Jan Murray. Specifying agents with UML statecharts and StatEdit. In Polani et al. [14].

[12] Jan Murray, Oliver Obst, and Frieder Stolzenburg. Towards a logical approach for soccer agents engineering. In Peter Stone, Tucker Balch, and Gerhard Kraetzschmar, editors, *RoboCup 2000: Robot Soccer World Cup IV*, LNAI 2019, pages 199–208. Springer, Berlin, Heidelberg, New York, 2001.

[13] Birger Peitersen and Jens Bangsbo. *Soccer Systems & Strategies*. Human Kinetics, 2000.

[14] Daniel Polani, Brett Browning, Andrea Bonarini, and Kazuo Yoshida, editors. *RoboCup 2003: Robot Soccer World Cup VII*, LNAI Series. Springer, Berlin, Heidelberg, New York, 2004.

[15] R. Reiter. *Knowledge in Action*. MIT Press, 2001.

[16] B. Riedel. Developing similarity measures for comparing game situations in RoboCup. Diplomarbeit, Knowledge Based Systems Group, RWTH Aachen, Aachen, Germany, 2002.

[17] T. Röfer, I. Dahm, U. Düffert, J. Hoffmann, M. Jüngel, M. Kallnik, M. Lötzsch, M. Risler, M. Stelzer, and J. Ziegler. Germanteam 2003. In Polani et al. [14].

[18] Frieder Stolzenburg, Oliver Obst, and Jan Murray. Qualitative velocity and ball interception. In Matthias Jarke, Jana Köhler, and Gerhard 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.

[19] Giovanni Trapattoni and Emilio Cecchini. *Konzeption und Entwicklung der Taktik im Fußball*. Onli Verlag, 1999. in German.

[20] Thomas Wagner, Otthein Herzog, and Ubbo Visser. Egocentric qualitative spatial knowledge representation for physical robots. In Craig Schlenoff and Mike Uschold, editors, *AAAI Spring Symposium*, Stanford, CA, 2004. AAAI Press. To appear.

Available Research Reports (since 1999):

2004

6/2004 *Frank Dylla, Alexander Ferrein, Gerhard Lakemeyer, Jan Murray, Oliver Obst, Thomas Röfer, Frieder Stolzenburg, Ubbo Visser, Thomas Wagner.* Towards a League-Independent Qualitative Soccer Theory for RoboCup.

5/2004 *Peter Baumgartner, Ulrich Furbach, Margret Groß-Hardt, Thomas Kleemann.* Model Based Deduction for Database Schema Reasoning.

4/2004 *Lutz Priese.* A Note on Recognizable Sets of Unranked and Unordered Trees.

3/2004 *Lutz Priese.* Petri Net DAG Languages and Regular Tree Languages with Synchronization.

2/2004 *Ulrich Furbach, Margret Groß-Hardt, Bernd Thomas, Tobias Weller, Alexander Wolf.* Issues Management: Erkennen und Beherrschen von kommunikativen Risiken und Chancen.

1/2004 *Andreas Winter, Carlo Simon.* Exchanging Business Process Models with GXL.

2003

18/2003 *Kurt Lautenbach.* Duality of Marked Place/Transition Nets.

17/2003 *Frieder Stolzenburg, Jan Murray, Karsten Sturm.* Multiagent Matching Algorithms With and Without Coach.

16/2003 *Peter Baumgartner, Paul A. Cairns, Michael Kohlhase, Erica Melis (Eds.).* Knowledge Representation and Automated Reasoning for E-Learning Systems.

15/2003 *Peter Baumgartner, Ulrich Furbach, Margret Gross-Hardt, Thomas Kleemann, Christoph Wernhard.* KRHyper Inside — Model Based Deduction in Applications.

14/2003 *Christoph Wernhard.* System Description: KRHyper.

13/2003 *Peter Baumgartner, Ulrich Furbach, Margret Gross-Hardt, Alex Sinner.* 'Living Book' :- 'Deduction', 'Slicing', 'Interaction'..

12/2003 *Heni Ben Amor, Oliver Obst, Jan Murray.* Fast, Neat and Under Control: Inverse Steering Behaviors for Physical Autonomous Agents.

11/2003 *Gerd Beuster, Thomas Kleemann, Bernd Thomas.* MIA - A Multi-Agent Location Based Information Systems for Mobile Users in 3G Networks.

10/2003 *Gerd Beuster, Ulrich Furbach, Margret Groß-Hardt, Bernd Thomas.* Automatic Classification for the Identification of Relationships in a Metadata Repository.

9/2003 *Nicholas Kushmerick, Bernd Thomas.* Adaptive information extraction: Core technologies for information agents.

8/2003 *Bernd Thomas.* Bottom-Up Learning of Logic Programs for Information Extraction from Hypertext Documents.

7/2003 *Ulrich Furbach.* AI - A Multiple Book Review.

6/2003 *Peter Baumgartner, Ulrich Furbach, Margret Groß-Hardt.* Living Books.

5/2003 *Oliver Obst.* Using Model-Based Diagnosis to Build Hypotheses about Spatial Environments.

4/2003 *Daniel Lohmann, Jürgen Ebert.* A Generalization of the Hyperspace Approach Using Meta-Models.

3/2003 *Marco Kögler, Oliver Obst.* Simulation League: The Next Generation.

2/2003 *Peter Baumgartner, Margret Groß-Hardt, Alex Sinner.* Living Book – Deduction, Slicing and Interaction.

1/2003 *Peter Baumgartner, Cesare Tinelli.* The Model Evolution Calculus.

2002

12/2002 *Kurt Lautenbach.* Logical Reasoning and Petri Nets.

11/2002 *Margret Groß-Hardt.* Processing of Concept Based Queries for XML Data.

10/2002 *Hanno Binder, Jérôme Diebold, Tobias Feldmann, Andreas Kern, David Polock, Dennis Reif, Stephan Schmidt, Frank Schmitt, Dieter Zöbel.* Fahrassistenzsystem zur Unterstützung beim Rückwärtsfahren mit einachsigen Gespannen.

9/2002 *Jürgen Ebert, Bernt Kullbach, Franz Lehner.* 4. Workshop Software Reengineering (Bad Honnef, 29./30. April 2002).

8/2002 *Richard C. Holt, Andreas Winter, Jingwei Wu.* Towards a Common Query Language for Reverse Engineering.

7/2002 *Jürgen Ebert, Bernt Kullbach, Volker Riediger, Andreas Winter.* GUPRO – Generic Understanding of Programs, An Overview.

6/2002 *Margret Groß-Hardt.* Concept based querying of semistructured data.

5/2002 *Anna Simon, Marianne Valerius.* User Requirements – Lessons Learned from a Computer Science Course.

4/2002 *Frieder Stolzenburg, Oliver Obst, Jan Murray.* Qualitative Velocity and Ball Interception.

3/2002 *Peter Baumgartner.* A First-Order Logic Davis-Putnam-Logemann-Loveland Procedure.

2/2002 *Peter Baumgartner, Ulrich Furbach.* Automated Deduction Techniques for the Management of Personalized Documents.

1/2002 *Jürgen Ebert, Bernt Kullbach, Franz Lehner.* 3. Workshop Software Reengineering (Bad Honnef, 10./11. Mai 2001).

2001

13/2001 *Annette Pook.* Schlussbericht "FUN - Funkunterrichtsnetzwerk".

12/2001 *Toshiaki Arai, Frieder Stolzenburg.* Multiagent Systems Specification by UML Statecharts Aiming at Intelligent Manufacturing.

11/2001 *Kurt Lautenbach.* Reproducibility of the Empty Marking.

10/2001 *Jan Murray.* Specifying Agents with UML in Robotic Soccer.

9/2001 *Andreas Winter.* Exchanging Graphs with GXL.

8/2001 *Marianne Valerius, Anna Simon.* Slicing Book Technology — eine neue Technik für eine neue Lehre?.

7/2001 *Bernt Kullbach, Volker Riediger.* Folding: An Approach to Enable Program Understanding of Preprocessed Languages.

6/2001 *Frieder Stolzenburg.* From the Specification of Multiagent Systems by Statecharts to their Formal Analysis by Model Checking.

5/2001 *Oliver Obst.* Specifying Rational Agents with Statecharts and Utility Functions.

4/2001 *Torsten Gipp, Jürgen Ebert.* Conceptual Modelling and Web Site Generation using Graph Technology.

3/2001 *Carlos I. Chesñevar, Jürgen Dix, Frieder Stolzenburg, Guillermo R. Simari.* Relating Defeasible and Normal Logic Programming through Transformation Properties.

2/2001 *Carola Lange, Harry M. Sneed, Andreas Winter.* Applying GUPRO to GEOS – A Case Study.

1/2001 *Pascal von Hutten, Stephan Philippi.* Modelling a concurrent ray-tracing algorithm using object-oriented Petri-Nets.

2000

8/2000 *Jürgen Ebert, Bernt Kullbach, Franz Lehner (Hrsg.).* 2. Workshop Software Reengineering (Bad Honnef, 11./12. Mai 2000).

7/2000 *Stephan Philippi.* AWPN 2000 - 7. Workshop Algorithmen und Werkzeuge für Petrinetze, Koblenz, 02.-03. Oktober 2000 .

6/2000 *Jan Murray, Oliver Obst, Frieder Stolzenburg.* Towards a Logical Approach for Soccer Agents Engineering.

5/2000 *Peter Baumgartner, Hantao Zhang (Eds.).* FTP 2000 – Third International Workshop on First-Order Theorem Proving, St Andrews, Scotland, July 2000.

4/2000 *Frieder Stolzenburg, Alejandro J. García, Carlos I. Chesñevar, Guillermo R. Simari.* Introducing Generalized Specificity in Logic Programming.

3/2000 *Ingar Uhe, Manfred Rosendahl.* Specification of Symbols and Implementation of Their Constraints in JKogge.

2/2000 *Peter Baumgartner, Fabio Massacci.* The Taming of the (X)OR.

1/2000 *Richard C. Holt, Andreas Winter, Andy Schürr.* GXL: Towards a Standard Exchange Format.

1999

10/99 *Jürgen Ebert, Luuk Groenewegen, Roger Süttenbach.* A Formalization of SOCCA.

9/99 *Hassan Diab, Ulrich Furbach, Hassan Tabbara.* On the Use of Fuzzy Techniques in Cache Memory Managament.

8/99 *Jens Woch, Friedbert Widmann.* Implementation of a Schema-TAG-Parser.

7/99 *Jürgen Ebert, and Bernt Kullbach, Franz Lehner (Hrsg.).* Workshop Software-Reengineering (Bad Honnef, 27./28. Mai 1999).

6/99 *Peter Baumgartner, Michael Kühn.* Abductive Coreference by Model Construction.

5/99 *Jürgen Ebert, Bernt Kullbach, Andreas Winter.* GraX – An Interchange Format for Reengineering Tools.

4/99 *Frieder Stolzenburg, Oliver Obst, Jan Murray, Björn Bremer.* Spatial Agents Implemented in a Logical Expressible Language.

3/99 *Kurt Lautenbach, Carlo Simon.* Erweiterte Zeitstempelnetze zur Modellierung hybrider Systeme.

2/99 *Frieder Stolzenburg.* Loop-Detection in Hyper-Tableaux by Powerful Model Generation.

1/99 *Peter Baumgartner, J.D. Horton, Bruce Spencer.* Merge Path Improvements for Minimal Model Hyper Tableaux.