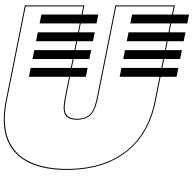

UNIVERSITÄT
KOBLENZ · LANDAU

Spatial Agents Implemented in a Logical Expressible Language

Frieder Stolzenburg, Oliver Obst,
Jan Murray, Björn Bremer

4/99

Fachberichte
INFORMATIK

Universität Koblenz-Landau
Institut für Informatik, Rheinau 1, D-56075 Koblenz

E-mail: researchreports@infko.uni-koblenz.de,
WWW: http://www.uni-koblenz.de/fb4/

Spatial Agents Implemented in a Logical Expressible Language

Frieder Stolzenburg, Oliver Obst, Jan Murray, Björn Bremer

E-mail: {stolzen,frvit,murray,moddy}@uni-koblenz.de

Abstract

In this paper, we present a multi-layered architecture for spatial and temporal agents. The focus is laid on the declarativity of the approach, which makes agent scripts expressive and well understandable. They can be realized as (constraint) logic programs. The logical description language is able to express actions or plans for one and more autonomous and cooperating agents for the RoboCup (Simulator League). The system architecture hosts constraint technology for qualitative spatial reasoning, but quantitative data is taken into account, too. The basic (hardware) layer processes the agent's sensor information. An interface transfers this low-level data into a logical representation. It provides facilities to access the preprocessed data and supplies several basic skills. The second layer performs (qualitative) spatial reasoning. On top of this, the third layer enables more complex skills such as passing, offside-detection etc. At last, the fourth layer establishes acting as a team both by emergent and explicit cooperation. Logic and deduction provide a clean means to specify and also to implement teamwork behavior.

1 Introduction

Naturally, tasks to be solved by a team of autonomous agents are many-sided and complex. In order to achieve a goal, a single agent has to use a set of complementary subtasks. On the one hand, some of these actions can be performed in a purely reactive manner, meeting real-time requirements. On the other hand, tasks may require a certain amount of planning and reasoning. So, we were led to the idea of combining both the advantages of procedural and logic programming and decided on a hybrid system with a layered architecture.

Using a layered architecture has several advantages. Firstly, it allows one to build a conceptionally clear architecture of (at least partially) independent modules. We can distinguish between urgent tasks that have to be performed in real-time (so-to-speak subconscious patterns of behavior), and others that require some time for careful planning. These are the more cognitive tasks.

1.1 Implementing Agents in Logic

In contrast to other approaches that provide an architecture for (multi-)agent systems (see e.g. [Rao, 1996; Jung, 1999]), we use different logical and deductive formalisms not only as a specification language but also as an implementation language. Also widespread is the use of a Belief-Desire-Intention (BDI) architecture (see e.g. [Burkhard *et al.*, 1998]), which has been originally specified by means of modal logics. A first-order axiomatization has been proposed for this kind of architecture only recently [Rao, 1996]. However, it seems that it is not actually used as implementation language there.

We will now describe our system architecture and show how different deductive processes—including constraint solving—can be used for the RoboCup [Noda, 1995]. The system combines the BDI approach with a multi-layered architecture, allowing multiple agents to perform collective actions. Nevertheless, each agent is autonomous and can be implemented in a manner similar to (Constraint) Logic Programs (CLP) [Jaffar and Maher, 1994]. The major goals of the RoboLog project, undertaken at the University of Koblenz, Germany, are the following:

- A flexible, modular system architecture should be established, meeting the various requirements for RoboCup agents. For example, on the one hand, agents have to be able to react in real-time. But on the other hand, it is also desirable that more complex behavior of agents can be programmed easily in a declarative manner.

- It should be possible to handle different representation formats of knowledge about the environment. Information may be quantitative or qualitative, pictorial or propositional in nature. Therefore, we propose a deductive framework, that is expressible in plain first-order logic (possibly plus constraint technology components), that integrates axiomatic approaches in geometry, spatial constraint theories and numerical sensor data.

- Agents not only should be able to act autonomously on their own, but also to cooperate with other agents. For this, we develop a multi-agent script language for the specification of collective actions or intended plans that are applicable in a certain situation. These scripts can be translated into logic programs in a straightforward manner.

1.2 Outline of the Approach

In the following, we discuss our layered system architecture and the functionality of the respective layers. Fig. 1 shows the complete architecture of RoboLog. The lowest layer—the RoboLog kernel, which is implemented in C++—essentially is the interface between the SoccerServer [Corten *et al.*, 1999] and Prolog, since all other layers are implemented in this logic programming language.

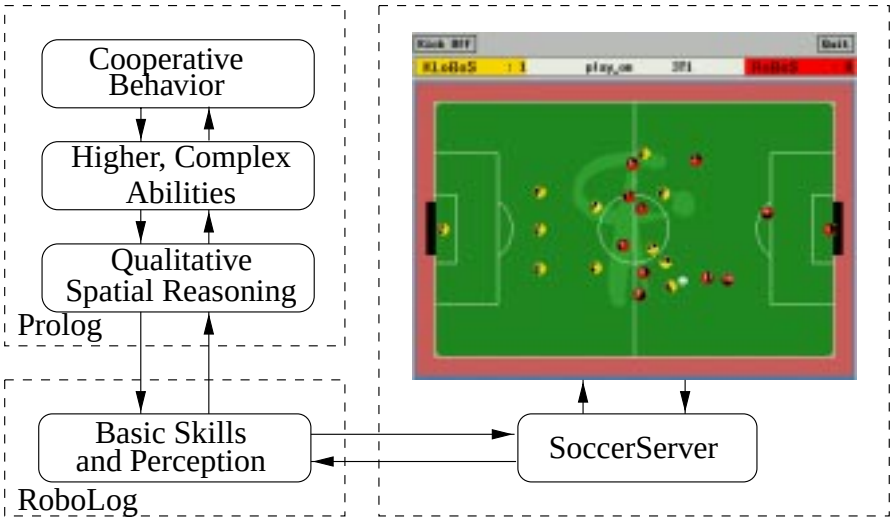

Figure 1: System Architecture of RoboLog.

The basic layer hosts reactive behavior. It is implemented in the RoboLog Prolog extension [Obst, 1998; Obst, 1999]. This extension is an enhanced RoboCup SoccerServer interface for ECLiPSe-Prolog [ECLiPSe, 1998]. Time critical and computational expensive tasks are handled within the RoboLog module, as well as the exchange of data. The module provides the atomic SoccerServer commands and some more complex actions. Hence already at this level, logic (programming) formalisms are available. They are used for running, turning, kicking and—last but not least—sensing the environment. Also position determination is settled in this layer (see Sect. 2.1). It also provides more specific facilities. In the soccer scenario this means, for instance, dribbling and ball interception. For these actions, (almost) no spatial cognition is required.

Spatial cognition is the contents of the second layer. For example, players have to recognize when passing the ball is possible or a player is offside. Many approaches (see e.g. [Clementini *et al.*, 1997; Freksa and Zimmermann, 1992]) propose purely qualitative reasoning, i.e. disregarding quantitative information after it has been transferred into a qualitative representation. But this may be too inexact and too vague. Since we use logic as connecting formalism in all layers, we can access low-level data at all levels of abstraction. This implies, reasoning can be as exact as required. We will present our approach in more detail in Sect. 3.

The last two layers host complex situations, possibly requiring teamwork, i.e. single or multi-agent plans. Nevertheless, the question remains whether teamwork should be invoked explicitly by communication or whether it is sufficient and more robust just to have implicit teamwork. The first kind is more flexible, even if different agents have to work together. The current implementation implicitly exploits knowledge on other implementation of agents. With the exception of the goalkeeper, they are clones of each other. Cooperative behavior may be required even if the implementation details

are different or not known. The problem is then, what communication language can be used in this case.

A general approach for the exchange of knowledge between agents is the Knowledge Query and Manipulation Language (KQML) [Labrou and Finin, 1997]. If the domain of application is restricted, KQML may be too general. But it allows reliable communication between agents, even if their internal architecture is quite different or unknown for the other agent. Instead, we communicate Prolog predicates directly. The advantage of this approach is, that no meta-logical interpretation of received information is necessary. A disadvantage is that for a successful communication the agents have to know each other's internal structure exactly.

2 Basic Abilities and Actions (Layer 1)

The lowest layer in our system architecture handles basic skills and perception of the environment. The basic skills may be actions that can be performed immediately by the agent, e.g. turning around, dashing, kicking the ball etc. In addition, we will allow more complex actions in this layer, that do not need (qualitative) spatial reasoning, so-to-speak subconscious processes, such as dribbling with the ball and, to a certain extent, even ball interception. In our current implementation, these complex actions are not yet a part of the RoboLog interface, but are written as Prolog modules.

Depending on the hardware used, perception of the environment, including self and object localization is a complex task, requiring more or less processing. In the simplest case, perception just means reading off the data from one of the agent's sensors. Usually, the data is digitized, in order to make it tractable for computer programs. Note that we aim at having a (first-order) logic presentation for each agent, which combines the advantages of being declarative and efficient to a certain extent, because in addition we make use of constraint technology. The logical description language we are going to introduce allows agent programs (scripts) to be written and interpreted in a manner similar to CLP.

Following the lines of [Rao, 1996], we distinguish two classes of predicates: ACTIONS a and PERCEPTIONS p. When executed successfully, a perception predicate p returns the requested data. We will assume, that this data is quantitative, i.e. some arguments of the predicate are real numbers. For example, a perception predicate p may return the distance to a certain landmark, measured in meters and given as a real number. The main matter of an action a is its side-effect, i.e. the performed action. Nevertheless, an action predicate (except the primitive actions of the SoccerServer) also is assigned a truth value, depending on the success or failure of the action. Note that the truth value for all predicates is dependent on the actual time t, when the action or request for data is executed. In summary, the RoboLog interface provides the following functionality:

- For each agent, it requests the sensor data from the SoccerServer. By this, the agents' knowledge bases are updated periodically. If some requested information

about a certain object is currently not available (because it is not visible at the moment), the most recent information can be used instead. Each agent stores information about objects it has seen within the last 100 simulation time steps. So we can think of it as the agent's memory or recollection.

- This low-level data is processed in such a way that more complex and more precise information becomes available, such as global position information (see also Sect. 2.1) or direct relations between objects with or without reference to the actual agent. The relation $is\_left(Obj_1, Obj_2)$, e.g., depends on the relative position of the agent, whereas $is\_between(Obj_1, Obj_2, Obj_3)$ is an agent independent property.

- The passing of time can be modeled in several ways with RoboLog. It provides various means for creating snapshots of the world and defining an event-driven calculus upon them, keeping track of the actual simulator step time or just ignoring real-time at all.

- Last but not least, Prolog predicates are provided that can be used to request the current status of sensor information on demand. The data should be synchronized with the SoccerServer, before an agent's action is initiated.

2.1 Position Determination

An important piece of information for an agent is to know its own position. Therefore, the RoboLog system provides an extensive library that makes precise object localization possible. Let us now give a brief overview on the whole procedure, which is able to work even when only few or inconsistent information is given. First of all, an agent has an egocentric view of the world. The actual sensor data provide more or less precise information about the positions of other agents, landmark points and border lines. Naturally, they are given relative to the position and orientation of the agent as polar coordinates. This is the *internal* coordinate system. Note that each agent has an obvious orientation, given by its face or body direction. However, this relative position information often is not sufficient for complex tasks.

In addition to the egocentric view of each agent, a scenario often yields another frame of reference with absolute coordinates, which may be given by a global map of the environment. This is the *external* coordinate system. In the RoboCup scenario, it is given by the geometry of the playing field. Converting position information into a geocentric view, usually with Cartesian coordinates, has several advantages. Firstly, knowing the absolute position allows the agent to identify its own location with respect to other objects on the map, even if they are not visible at the moment. Secondly, an absolute frame of reference is helpful in order to communicate with other agents in situations where cooperative actions are appropriate.

The RoboLog system hosts several different procedures for self and object localization wrt. an external coordinate system. The first method, that is introduced in [Betke

and Gurvits, 1997], requires (only) three or more directions to visible landmarks relative to the orientation of the agent to be known. Provided that at least three of them and the position of the agent neither form a circle nor lie on a straight line, the absolute position and orientation of the agent can be computed with a time complexity that is only linear in the number of landmarks. If the corresponding equation system in complex numbers is over-determined, and the data is noisy, the procedure estimates the position applying the least squares method. Without reference to a third landmark, the actual position in question lies on a segment of a circle. This is shown in Fig. 2.

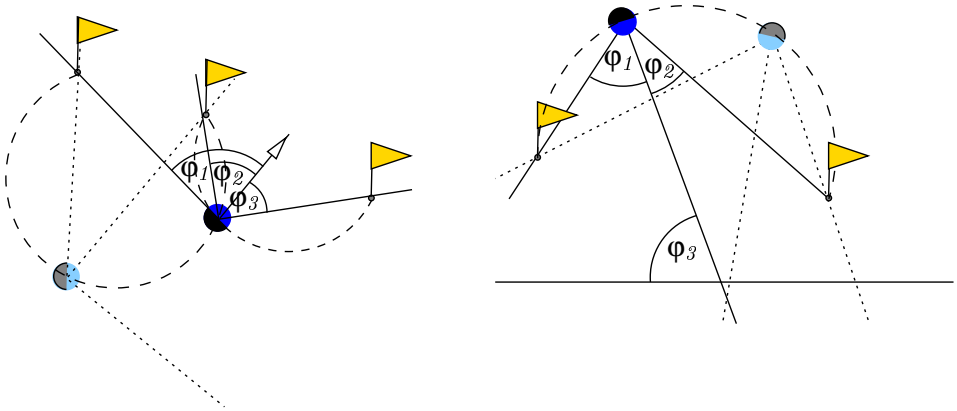

Figure 2: Self-localization by angle information.

However, we also need procedures that are able to work, if only limited sensor data is available. If less than three landmarks or border lines are visible, other procedures for position determination are required. There are two cases where the position of the agent is uniquely determined even then. Firstly, knowing the distances and angles to two landmarks is sufficient. Secondly, the position can be determined if the relative position of one landmark and of a point on a border line where the center line of the view axis of the agent crosses this line is given, provided that both points are not identical. In addition, by keeping track of the movements of the agent, it is also possible to estimate the current position. All these methods are implemented in the RoboLog module, in order to lay a solid quantitative basis for the qualitative reasoning in the higher layers.

2.2 Basic Skills

Agents have to be able to move in their environment without collision. This is a basic requirement for many practical robot multi-agent systems. In the RoboCup scenario agents should also be able to handle the ball. This means they must be able to run and kick to a certain position, dribble with the ball etc. Another important task is ball interception. For this, an agent has to recognize and compute the ball trajectory in advance, compute and go to the point where ball interception is possible, and stop the

ball. This is a macro task, which could be executed in a certain situation without any qualitative reasoning.

A large set of low-level abilities for the RoboCup scenario is stated in [Stone *et al.*, 1999]. There, kicking, goal-tending and—as a sub-task—getting sight of the ball among others are considered as part of the low-level architecture of an agent. Of course, such tasks may require deep computation. However, only quantitative data is made use of for these actions. This is the reason why it is reasonable to classify these actions as basic skills. Nevertheless, more complex actions will require (deductive) reasoning. That is the contents of the next layer (see Sect. 3).

In our system, the following basic skills (among others) are implemented (see also [Murray, 1999] that also describes special skills of the goalkeeper):

- The agents can search for the ball, taking into account their knowledge about the last time the ball was seen.
- Dashing and kicking to a certain position, regarding the agent's condition and avoiding obstacles is possible and (based upon these skills) also dribbling.
- Extrapolating the ball trajectory to a given time in the future enables the agents to intercept opponent passes and block shots.

3 Qualitative Spatial Reasoning (Layer 2)

During a match, a human soccer player will enter a lot of different situations, in which he has to decide what to do. In most of the cases, he will decide regarding former experience, i.e. comparing his situation to situations he already handled before. Hence, if we want to build a client, we have to provide the client with some situations and connected actions. Recognizing a situation the client knows, it can use former experiences to plan actions. Only in rare cases the client's situation will fit exactly to a stored situation pattern (identified by its set of preconditions), so we have to parameterize and abstract the patterns. The similarity between two situations can be used as a heuristic to determine, how far associated actions should be executed or modified.

What we need in order to identify situations is the abstraction of quantitative data onto a qualitative level. Therefore, we have another class of predicates—in addition to the classes mentioned in Sect. 2—, namely QUALITIES q. Qualitative predicates are defined upon the quantitative perceptions via logical rules and constraints, e.g. the *in-front-of* relation (1). But it may also be the case that there are qualitative predicates or relations based on each other. In the latter case we speak of purely qualitative predicates or reasoning, e.g. the relation *left_and_in-front-of* can be reduced to the qualitative predicates *left* and *in-front-of* (2).

$$\textit{in-front-of} \leftarrow \textit{Dist} > 0\,. \tag{1}$$

$$\textit{left} \leftarrow \textit{Dir} < 0\,.$$

$$\textit{left\_in-front-of} \leftarrow \textit{left} \wedge \textit{in-front-of}\,. \tag{2}$$

For example, concerning the distance of an agent to the ball in the RoboCup scenario only a few (qualitative) aspects are interesting. Thus, in RoboLog we only distinguish few distances: *close* (the ball is in the kickable area), *near* (the agent is able to detect much detail by its sensors), *short* (maximal shooting distance), *far away* (sensor data become unreliable from this distance), *remote* (out of reach). Quantitative distance intervals can be mapped to qualities. Concerning the other direction, chosen plan schemes must be instantiated with quantitative data for the actual execution. A related work is presented in [Clementini *et al.*, 1997]. There, reasoning on the qualitative level (alone) is provided. Fig. 3 illustrates the correspondence between quantitative and qualitative distances.

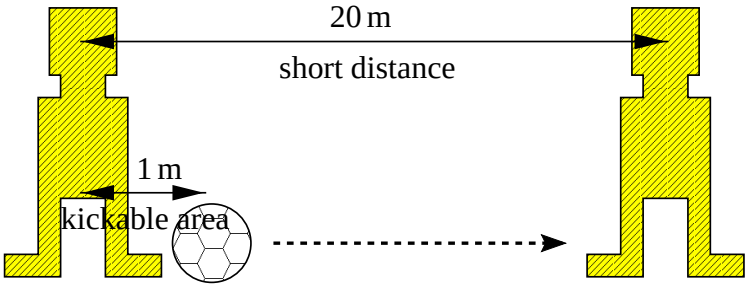

Figure 3: Distances – quantitative and qualitative.

3.1 Constraint Reasoning

In the literature, many approaches for qualitative spatial reasoning are proposed. Most of them rely on the Region Connection Calculus (RCC), see e.g. [Randel *et al.*, 1992; Bennet *et al.*, 1997; Renz and Nebel, 1998]. On the one hand, the advantage of qualitative information certainly is that seemingly complex situations can be reduced to a few patterns of situations, and concentration on the relevant portion of information is possible. On the other hand, a qualitative description may be a too rough approximation of the reality, such that reasoning on a purely qualitative level may become too vague [Clementini *et al.*, 1997; Freksa and Zimmermann, 1992]. So the question remains, how can we make use of both quantitative and qualitative information.

In most cases, if sensor data is available, it is a good idea to make use of the quantitative data by just abstracting it to a qualitative level. Only in some cases, when no more precise quantitative information is available, purely qualitative reasoning is necessary. More precise knowledge should be preferred. So, we combine real-time quantitative reasoning with qualitative spatial reasoning, that can be implemented as a constraint system (in the formal sense) [Renz and Nebel, 1998] and integrated in a more general deductive framework for constraint logic programming (CLP).

The process of spatial reasoning has to be seen in the context of its purpose, that is laying the basis for what action should be performed next. There are (at least) two

decision problems in this context:

- If there are different sources of information (e.g. numerical sensor data, derived qualitative knowledge or conclusions thereof), there must be some control mechanism for deciding how the requested information should be obtained. In our current implementation, quantitative data is preferred: it is simply converted into a qualitative presentation. There are only very rare cases where purely qualitative reasoning is performed. This could mean applying the transitivity rule to topological relations such as *between*.

- In addition, it may be difficult to decide what should be done next in a situation where we have several options (e.g. dribbling, passing, kicking). In the current implementation, we simply make use of the backtracking facilities of Prolog for this purpose. However, it might be a good idea to employ defeasible reasoning in this decision process [Dix *et al.*, 1999].

3.2 An Axiomatic Approach

We are also investigating the problem of modeling certain situations as patterns by means of logic programs and the full first-order theorem proving system Protein [Baumgartner and Furbach, 1994]. For example, passing the ball is possible in a situation where one player has the ball, another player can be reached and there is no player (of the opposite team) in between. We modeled these situations on top of the logical relations *left*, *right* and *between*. Since we use logic, the properties of the qualities have to be axiomatized. Two possibilities come into mind: we can model *between* on top of general geometric axioms [Borsuk and Szmielew, 1960], or use *collinear* as basic concept [Eschenbach and Kulik, 1997]. We believe that it is more natural to use (an ordered version of) *between* as base relation, since we can assume that the sensor data provides information about order anyway. In addition, the order information may be required for planning certain actions in detail.

However, for axiomatic approaches in general, there is one problem: how can the negative information be deduced, e.g. if we want to know that there is no opponent in between. With Prolog alone this is not possible: the built-in negation as failure sometimes causes problems if used in complex queries. So we were led to use full first-order logic with the Protein theorem prover [Baumgartner and Furbach, 1994]. As example for this, let us consider the problem of determining whether passing is possible. This could be checked by the following logical rule with negation in the rule body:

$$Passing \leftarrow \neg\exists Opp : Between(Me, Opp, Partner)$$

The intended meaning of this rule is as follows: passing is possible, if there is no opponent between the agent and one of its partners. The question is: how should negation ($\neg$) and existential quantification ($\exists$) be interpreted? Protein provides classical negation as usual in first-order theorem proving. Existential quantification causes problems,

if treated by Skolemization, i.e. replacing existentially quantified variables by new constant or function symbols, because then we have potentially infinitely many players.

Since we need real-time behavior, we just considered the finite domain of players visible for the agent in our implementation. This is closed world or constraint domain reasoning. By this, we get a complete and terminating system. Possibly, more sophisticated kinds of non-monotonic negation can be used here in this context of decision-finding. Note that, currently, this component is not yet integrated into the actual RoboLog Koblenz implementation, but has been used for axiomatizing situations (see [Bremer, 1999]).

4 Higher Abilities (Layers 3 and 4)

Many tasks require deeper reasoning, which can be expressed within a BDI agent architecture [Bratman, 1987; Rao, 1996]. In our context, a BELIEF b is a qualitative predicate q, its negation $\neg q$ or a conjunction of beliefs $b_1 \wedge b_2$. A GOAL g is either an *achievement* goal $!q$ or a *test* goal $?q$, where q is a qualitative predicate. A DESIRE (or event) d is a goal or an action, indexed by a list of agents—the actors—, which must satisfy the desire by performing some actions.

Now we can build rules for a certain SITUATION in form of scripts, written $d : b - i$, where d is a desire, b is a belief (identifying the precondition of the situation), and i is the INTENTION (or, strictly speaking, the intended plan). The intended plan is an acyclic graph of desires with a designated start node. Its edges are labeled with actors which must be a subset of the actors in d. Edges outgoing from test goals are labeled in addition with *yes* or *no* and possibly a time-out delay. Consider now all possible subgraphs wrt. edges for a certain actor. It is required that this is a tree with the start node as root, where binary branching is only allowed after test nodes. These subgraphs represent the ROLE for the respective actor. An achievement goal has to be performed actively by the indexed actors, while non-actors wait for the achievement until a certain time-limit. If the time-limit is exceeded or an external interruption occurs (e.g. a referee message in the RoboCup scenario), then the agent has to return to a *default plan*, which must be applicable without precondition.

4.1 An Example: Double Passing

Let us now consider an example for a collective action of agents, namely double passing. There are two actors in this situation: actor 1 kicks the ball to actor 2, then actor 1 runs towards the goal, and expects a pass from actor 2. This is illustrated in Fig. 4. In order to initiate such an action, two agents simultaneously have to recognize their respective roles in the current situation in their belief state. The belief b for a double passing situation can be described as follows: 1 and 2 are nearest neighbors belonging to the same team, and player 1 has the ball. Player 2 must be clear, whereas an opponent is near to 1 such that 1 cannot dribble straight on. The intended plan i

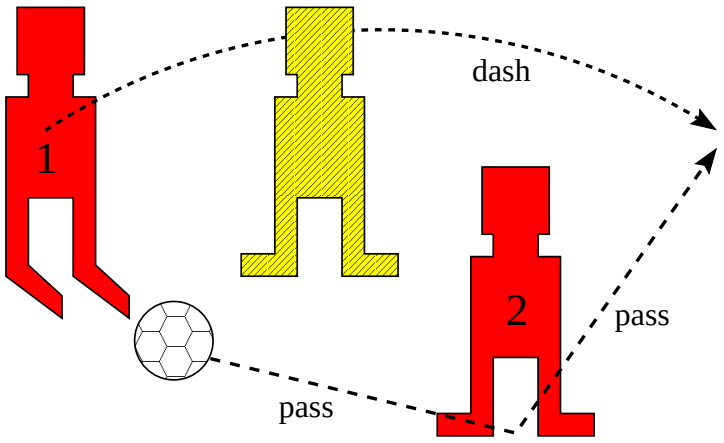

Figure 4: Double Passing Situation.

is then, that 1 passes the ball to 2 at first, then 1 runs towards the goal, and finally 2 passes the ball to 1. The respective rule can be expressed as shown in Fig. 5.

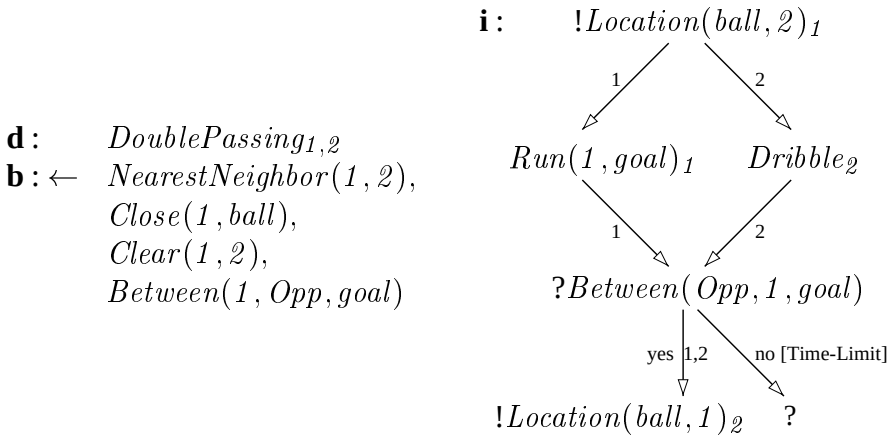

Figure 5: Double Passing Script.

While experimenting with an implementation of double passing, we noticed that the main problem is that both actors simultaneously have to recognize their role, because one of the agents possibly does not see the other agent. In this context, communication (i.e. telling the other agent one's desire) helps a lot. A cooperating partner could tell its coordinates or even its whole own belief state.

We made similar experiences with an even simpler kind of action, namely simple passing. The main problem there is that an agent may not see partners behind itself where the agent can pass to. Therefore again, we make use of communication, in order to initiate this complex action. This increases the probability of successful actions and

decreases the complexity of deciding the next action. Nevertheless, each agent should remain as autonomous and robust as possible.

4.2 Translating Rules into CLP

We may distinguish several types of plans: basic plans with only one actor and complex plans where there are more than one actors. The former plans implement higher abilities (layer 3), while the latter realize teamwork (layer 4). Each BDI script can be translated into a CLP rule in a straightforward manner. For each achievement or test goal we introduce new symbols: $\underline{!P}$ and $\underline{?P}$. For each rule some default recipes are introduced:

$$\begin{aligned} P(x_1,\ldots,x_n) &\leftarrow \underline{!P}(x_1,\ldots,x_n). \\ \underline{?P}(x_1,\ldots,x_n) &\leftarrow P(x_1,\ldots,x_n). \end{aligned}$$

The former and external events update predicates; this is the main difference to CLP. An approach that can handle external events and concurrency is *ConGolog* [De Giacomo *et al.*, 1997]. The qualitative reasoning is performed in a constraint module, which may e.g. realize the RCC [Randel *et al.*, 1992]. Of course, the changes over time have to be taken into account. For each situation and for each role in it, a BDI script can be translated directly into a logic program rule, possibly with concurrent constraints (belief conditions):

$$d \leftarrow b \wedge i$$

The reader may have noticed that a situation with n roles corresponds to n CLP rules. These rules are identical wrt. their heads d. The preconditions b for the actions are also very similar; they only differ in their actor role. The last (but not least) part i is really different, because each actor plays a different role in the respective situation. For example, the instantiated plans for both actors of the double passing rule (see Fig. 5) are as follows:

$\underline{Role\ 1}$	$\underline{Role\ 2}$
$!Location(ball, 2)$	$?Location(ball, Self)$
$Run(Self, goal)$	$Dribble$
$?Between(Opp, Self, goal)$	$?Between(Opp, 1, goal)$
$?Location(ball, Self)$	$!Location(ball, 1)$

Recall that achievement goals are converted into test goals for non-actors. In addition, the control sequence for giving up after some time-limit is not shown here. Clearly, the translation into several CLP rules increases the time complexity for deciding which action or role therein is performed next. This problem can at least be partially overcome by communicating the next action directly to partners. In fact, we do this in our implementation by sending calls to Prolog predicates. But nevertheless,

robustness of the whole system (of agents) has to be guaranteed in the case of failing actions or failing communication.

5 Summary

We presented a logical description language for multi-agent systems, following the lines of [Rao, 1996]. The implementation language can be understood as a generalization of CLP. Both quantitative and qualitative spatial reasoning can be built-in. Purely qualitative reasoning and efficient calculi for this (see e.g. [Renz and Nebel, 1998]) are especially useful in contexts where no precise quantitative information is available.

The RoboLog system provides a clean means for programming soccer agents declaratively. The RoboLog Koblenz players were implemented by a team of 3 to 5 people. We conducted several test games with different scores on our local network—a 100 MBit Ethernet. The results of some successful games are shown in Fig. 6.

RoboLog Koblenz (on 3 Pentium II Linux-PCs and 2 Sun Sparc Ultra-1, 143 MHz, all 64 MB main memory)	*Linköping Lizards* (on 1 Sun Ultra-Enterprise with 14 336 MHz processors, 3 GB main memory)	**6:1**
RoboLog Koblenz (on 1 Sun Ultra-Enterprise with 14 336 MHz processors, 3 GB main memory)	*AT Humboldt 97* (on 1 Sun Sparc 5, 110 MHz, 64 MB main memory)	**2:0**

Figure 6: Successful soccer simulation games.

But there are still some problems to solve. Apart from several small questions, we identified two main problems, which often have a huge effect on the players' behavior:

- The execution of a collective action often fails because of the imperfect implementation of the low level facilities. Although all active agents recognize the situation and their special role correctly, the intended plan may fail, e.g. because a pass from agent 1 does not reach agent 2. To minimize such faults, we do not only improve the low-level skills implemented so far, but also work on adapting the skills of the *CMUnited Simulator Team*, especially the predictive locally optimal skills (PLOS) [Stone *et al.*, 1999], to RoboLog.

- Another point is that only few situations like double passing are implemented yet. So the agents very frequently have to stick to their default plans, because situations arise which are unclear for the agents. But this problem can be solved by axiomatizing more situations and providing the according scripts to the agents.

Further work should concentrate on the real-time requirements in exceptional situations and the concurrency of different mechanisms for information acquisition. One area of research is how far logical mechanisms can be used within the lower levels of

our approach. Deduction could be used to build a more complete view of the agent's world. Each time an agent gets new information, a set of logical rules rebuilds the agent's spatial database. The application of these techniques to real robots is one of the next steps of our research activities. In addition, the robustness of the decision process can be improved by means of defeasible reasoning. The specification of a communication language and the use of any-time reasoning formalisms should also be investigated.

References

[Baumgartner and Furbach, 1994] Peter Baumgartner and Ulrich Furbach. PROTEIN: A PROver with a Theory Extension INterface. In Alan Bundy, editor, *Proceedings of the 12th International Conference on Automated Deduction*, LNAI 814, pages 769–773, Nancy, 1994. Springer, Berlin, Heidelberg, New York.

[Bennet *et al.*, 1997] Brandon Bennet, Anthony G. Cohn, and Amar Isli. Combining Multiple Representations in a Spatial Reasoning System. In *Proceedings of the 9th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'97)*, pages 314–322, Newport Beach, CA, 1997.

[Betke and Gurvits, 1997] Margrit Betke and Leonid Gurvits. Mobile Robot Localization using Landmarks. *IEEE Transactions on Robotics and Automation*, 13(2):251–263, April 1997.

[Borsuk and Szmielew, 1960] Karol Borsuk and Wanda Szmielew. *Foundations of Geometry*. North-Holland, Amsterdam, 1960.

[Bratman, 1987] Michael E. Bratman. *Intention, Plans, and Practical Reason*. Harvard University Press, Cambridge, MA, London, England, 1987.

[Bremer, 1999] Björn Bremer. Erkennung von Paß- und Abseitssituationen mit räumlichem Schließen. Studienarbeit, Fachbereich Informatik, Universität Koblenz, 1999.

[Burkhard *et al.*, 1998] Hans-Dieter Burkhard, Markus Hannebauer, and Jan Wendler. Belief–Desire–Intention – Deliberation in Artificial Soccer. *AI Magazine*, pages 87–93, 1998.

[Clementini *et al.*, 1997] Eliseo Clementini, Paolino di Felice, and Daniel Hernández. Qualitative representation of positional information. *Artificial Intelligence*, 95(2):317–356, 1997.

[Corten *et al.*, 1999] Emiel Corten, Klaus Dorer, Fredrik Heintz, Kostas Kostiadis, Johan Kummeneje, Helmut Myritz, Itsuki Noda, Jukka Riekki, Patrick Riley, Peter

Stone, and Tralvex Yeap. *Soccerserver Manual*, 5th edition, May 1999. For Soccerserver Version 5.00 and later.

[De Giacomo *et al.*, 1997] Giuseppe De Giacomo, Yves Lespérance, and Hector J. Levesque. Reasoning about concurrent execution, prioritized interrupts, and exogenous actions in the situation calculus. In Martha E. Pollack, editor, *Proceedings of the 15th International Joint Conference on Artificial Intelligence*, pages 1221–1226, Nagoya, Japan, 1997. Volume 2.

[Dix *et al.*, 1999] Jürgen Dix, Frieder Stolzenburg, Guillermo R. Simari, and Pablo R. Fillottrani. Automating Defeasible Reasoning with Logic Programming (DeReLoP). In *Proceedings of the 2nd German Argentinian Workshop on Information Technology*, Königswinter, 1999. To appear.

[ECLiPSe, 1998] International Computers Limited and IC-Parc. *ECLiPSe User Manual / Extensions User Manual – Release 4.0*, 1998. Two volumes.

[Eschenbach and Kulik, 1997] Carola Eschenbach and Lars Kulik. An Axiomatic Approach to the Spatial Relations Underlying *Left-Right* and *in Front of–Behind*. In Günther Görz and Steffen Hölldobler, editors, *KI-97: Advances in Artificial Intelligence — Proceedings of the 21st Annual German Conference on Artificial Intelligence*, LNAI 1303, pages 207–218, Freiburg, 1997. Springer, Berlin, Heidelberg, New York.

[Freksa and Zimmermann, 1992] Christian Freksa and Kai Zimmermann. On the Utilization of Spatial Structures for Cognitively Plausible and Efficient Reasoning. In *Proceedings of the IEEE International Conference on Systems*, Chicago, 1992.

[Jaffar and Maher, 1994] Joxan Jaffar and Michael J. Maher. Constraint Logic Programming: a Survey. *Journal of Logic Programming*, 19/20:503–581, 1994.

[Jung, 1999] Christoph G. Jung. Layered and Resource-Adapting Agents in the RoboCup Simulation. In M. Asada and Hiroaki Kitano, editors, *RoboCup–98: Robot Soccer WorldCup II*, LNAI. Springer, Berlin, Heidelberg, New York, 1999. To appear.

[Labrou and Finin, 1997] Yannis Labrou and Tim Finin. A Proposal for a new KQML Specification. Technical Report TR CS-97-03, Computer Science and Electrical Engineering Department, University of Maryland Baltimore County, Baltimore, MD 21250, February 1997.

[Murray, 1999] Jan Murray. My Goal Is My Castle — Die höheren Fähigkeiten eines RoboCup-Agenten am Beispiel des Torwarts. Studienarbeit S 564, Fachbereich Informatik, Universität Koblenz, 1999.

[Noda, 1995] Itsuki Noda. Soccer Server: a simulator for RoboCup. In *JSAI AI-Symposium*, 1995.

[Obst, 1998] Oliver Obst. *RoboLog – An ECLiPSe-Prolog SoccerServer interface: Users manual*, March 1998.

[Obst, 1999] Oliver Obst. RoboLog: Eine deduktive Schnittstelle zum RoboCup Soccer Server. Diplomarbeit D 488, Fachbereich Informatik, Universität Koblenz, 1999.

[Randel *et al.*, 1992] D. A. Randel, Z. Cui, and Anthony G. Cohn. A spatial logic based on regions and connections. In *Proceedings of the third Int. Conf. on Knowledge Representation and Reasoning*, pages 165–176, San Mateo, 1992. Morgan Kaufmann.

[Rao, 1996] Anand S. Rao. AgentsSpeak(L): BDI Agents speak out in a logical computable language. In Walter van de Velde and John W. Perrame, editors, *Agents Breaking Away – 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, LNAI 1038, pages 42–55, Berlin, Heidelberg, New York, 1996. Springer.

[Renz and Nebel, 1998] Jochen Renz and Bernhard Nebel. Efficient Methods for Qualitative Spatial Reasoning. In Henri Prade, editor, *Proceedings of the 13th European Conference on Artificial Intelligence*. John Wiley & Sons, 1998.

[Stone *et al.*, 1999] Peter Stone, Manuela Veloso, and Patrick Riley. The CMUnited-98 Champion Simulator Team. In M. Asada and Hiroaki Kitano, editors, *RoboCup–98: Robot Soccer WorldCup II*, LNAI. Springer, Berlin, Heidelberg, New York, 1999. To appear.

Available Research Reports (since 1997):

1999

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.

1998

24/98 *Jürgen Ebert, Roger Süttenbach, Ingar Uhe.* Meta-CASE Worldwide.

23/98 *Peter Baumgartner, Norbert Eisinger, Ulrich Furbach.* A Confluent Connection Calculus.

22/98 *Bernt Kullbach, Andreas Winter.* Querying as an Enabling Technology in Software Reengineering.

21/98 *Jürgen Dix, V.S. Subrahmanian, George Pick.* Meta-Agent Programs.

20/98 *Jürgen Dix, Ulrich Furbach, Ilkka Niemelä .* Nonmonotonic Reasoning: Towards Efficient Calculi and Implementations.

19/98 *Jürgen Dix, Steffen Hölldobler.* Inference Mechanisms in Knowledge-Based Systems: Theory and Applications (Proceedings of WS at KI '98).

17/98 *Stefan Brass, Jürgen Dix, Teodor C. Przymusinski.* Super Logic Programs.

16/98 *Jürgen Dix.* The Logic Programming Paradigm.

15/98 *Stefan Brass, Jürgen Dix, Burkhard Freitag, Ulrich Zukowski.* Transformation-Based Bottom-Up Computation of the Well-Founded Model.

14/98 *Manfred Kamp.* GReQL – Eine Anfragesprache für das GUPRO–Repository – Sprachbeschreibung (Version 1.2).

12/98 *Peter Dahm, Jürgen Ebert, Angelika Franzke, Manfred Kamp, Andreas Winter.* TGraphen und EER-Schemata – formale Grundlagen.

11/98 *Peter Dahm, Friedbert Widmann.* Das Graphenlabor.

10/98 *Jörg Jooss, Thomas Marx.* Workflow Modeling according to WfMC.

9/98 *Dieter Zöbel.* Schedulability criteria for age constraint processes in hard real-time systems.

8/98 *Wenjin Lu, Ulrich Furbach.* Disjunctive logic program = Horn Program + Control program.

7/98 *Andreas Schmid.* Solution for the counting to infinity problem of distance vector routing.

6/98 *Ulrich Furbach, Michael Kühn, Frieder Stolzenburg.* Model-Guided Proof Debugging.

5/98 *Peter Baumgartner, Dorothea Schäfer.* Model Elimination with Simplification and its Application to Software Verification.

4/98 *Bernt Kullbach, Andreas Winter, Peter Dahm, Jürgen Ebert.* Program Comprehension in Multi-Language Systems.

3/98 *Jürgen Dix, Jorge Lobo.* Logic Programming and Nonmonotonic Reasoning.

2/98 *Hans-Michael Hanisch, Kurt Lautenbach, Carlo Simon, Jan Thieme.* Zeitstempelnetze in technischen Anwendungen.

1/98 *Manfred Kamp.* Managing a Multi-File, Multi-Language Software Repository for Program Comprehension Tools — A Generic Approach.

1997

32/97 *Peter Baumgartner.* Hyper Tableaux — The Next Generation.

31/97 *Jens Woch.* A component-based and abstractivistic Agent Architecture for the modelling of MAS in the Social Sciences.

30/97 *Marcel Bresink.* A Software Test-Bed for Global Illumination Research.

29/97 *Marcel Bresink.* Deutschsprachige Terminologie des Radiosity-Verfahrens.

28/97 *Jürgen Ebert, Bernt Kullbach, Andreas Panse.* The Extract-Transform-Rewrite Cycle - A Step towards MetaCARE.

27/97 *Jose Arrazola, Jürgen Dix, Mauricio Osorio.* Confluent Rewriting Systems for Logic Programming Semantics.

26/97 *Lutz Priese.* A Note on Nondeterministic Reversible Computations.

25/97 *Stephan Philippi.* System modelling using Object-Oriented Pr/T-Nets.

24/97 *Lutz Priese, Yurii Rogojine, Maurice Margenstern.* Finite H-Systems with 3 Test Tubes are not Predictable.

23/97 *Peter Baumgartner (Hrsg.).* Jahrestreffen der GI-Fachgruppe 1.2.1 'Deduktionssysteme' — Kurzfassungen der Vorträge.

22/97 *Jens M. Felderhoff, Thomas Marx.* Erkennung semantischer Integritätsbedingungen in Datenbankanwendungen.

21/97 *Angelika Franzke.* Specifying Object Oriented Systems using GDMO, ZEST and SDL'92.

20/97 *Angelika Franzke.* Recommendations for an Improvement of GDMO.

19/97 *Jürgen Dix, Luís Moniz Pereira, Teodor Przymusinski.* Logic Programming and Knowledge Representation (LPKR '97) (Proceedings of the ILPS '97 Postconference Workshop).

18/97 *Lutz Priese, Harro Wimmel.* A Uniform Approach to True-Concurrency and Interleaving Semantics for Petri Nets.

17/97 *Ulrich Furbach (Ed.).* IJCAI-97 Workshop on Model Based Automated Reasoning.

16/97 *Jürgen Dix, Frieder Stolzenburg.* A Framework to Incorporate Non-Monotonic Reasoning into Constraint Logic Programming.

15/97 *Carlo Simon, Hanno Ridder, Thomas Marx.* The Petri Net Tools Neptun and Poseidon.

14/97 *Juha-Pekka Tolvanen, Andreas Winter (Eds.).* CAiSE'97 — 4th Doctoral Consortium on Advanced Information Systems Engineering, Barcelona, June 16-17, 1997, Proceedings.

13/97 *Jürgen Ebert, Roger Süttenbach.* An OMT Metamodel.

12/97 *Stefan Brass, Jürgen Dix, Teodor Przymusinski.* Super Logic Programs.

11/97 *Jürgen Dix, Mauricio Osorio.* Towards Well-Behaved Semantics Suitable for Aggregation.

10/97 *Chandrabose Aravindan, Peter Baumgartner.* A Rational and Efficient Algorithm for View Deletion in Databases.

9/97 *Wolfgang Albrecht, Dieter Zöbel.* Integrating Fixed Priority and Static Scheduling to Maintain External Consistency.

8/97 *Jürgen Ebert, Alexander Fronk.* Operational Semantics of Visual Notations.

7/97 *Thomas Marx.* APRIL - Visualisierung der Anforderungen.

6/97 *Jürgen Ebert, Manfred Kamp, Andreas Winter.* A Generic System to Support Multi-Level Understanding of Heterogeneous Software.

5/97 *Roger Süttenbach, Jürgen Ebert.* A Booch Metamodel.

4/97 *Jürgen Dix, Luis Pereira, Teodor Przymusinski.* Prolegomena to Logic Programming for Non-Monotonic Reasoning.

3/97 *Angelika Franzke.* GRAL 2.0: A Reference Manual.

2/97 *Ulrich Furbach.* A View to Automated Reasoning in Artificial Intelligence.

1/97 *Chandrabose Aravindan, Jürgen Dix, Ilkka Niemelä .* DisLoP: A Research Project on Disjunctive Logic Programming.