

# Towards Deduction in RoboCup

Oliver Obst · Jan C. Murray · Frieder Stolzenburg · Björn Bremer  
Universität Koblenz · Institut für Informatik  
Rheinau 1 · D-56075 Koblenz · Germany  
{frvit,murray,stolzen,moddy}@informatik.uni-koblenz.de

We are currently working on designing clients for Robocup-Simulation-League in Prolog and C++. We feel that logic and deduction is an appropriate approach for this task. A single player is part of a team and he has to deduce information on his situation. For example, players have to recognize when passing the ball is possible or a player is offside.

But, almost naturally, tasks to be solved by a team of autonomous agents are many-sided and complex. To achieve a goal a single agent has to use a set of complementary subtasks.

## 1 Robolog as an Extension of Prolog

Some of these subtasks consist of solving numerical equations to enable a player to handle tasks like dribbling or actually passing the ball. On the other hand, we have to derive new information from a given set of facts. So we were lead to the idea to combine both the advantages of procedural and logic programming and decided on a hybrid system. As a result, we implemented the Robolog Prolog extension. This extension is an enhanced RoboCup soccerserver interface for ECLiPSe-Prolog [3]. Time critical and computational expensive tasks are handled within the Robolog module, as well as the exchange of data. The developer of a soccer client accesses the calculated data via Prolog predicates. The module provides the atomic soccerserver commands and some more complex actions.

Our C++-Module simulates the player's nerves in so far as it is the reasoners connection to the soccer-server. It supplies the reasoner with sensory input and initiates actions.

The Prolog engine handles the player's reasoning and planing, i.e. to a certain degree it models a real soccer player's reasoning. However, it is difficult to say, in how far cognitive actions that are done on a subconscious level by humans, e.g. calculating the amount of kickpower needed to pass (or stop) the ball, should or could be added to the "nerves-layer" of our clients.

## 2 The Basic Machinery in C++

We have implemented some small sample clients (e.g. a dribbler client) to illustrate the interaction between C++ and Prolog. They show the pros and cons of our approach: on the one hand, we have a kind of "unconscious" processing of sensory input (unconscious in a way like humans "process" visual information, for example). On the other hand, it is difficult to decide what to put into the interface part and what to leave for the Prolog part.

## 3 Behaving Correctly in Situations

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 to his experience, i.e. compare 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 this "experiences" to plan its actions. Only in rare cases the clients situation will fit exactly to a stored pattern, so we have to parameterize the patterns. The similarity between two situations can be used as a heuristic to determine, how far associated actions should be executed or modified.

Currently, we are investigating the problem of modeling certain situations as patterns by means of logic programs and our PROTEIN system [1]. For example, a situation where passing the ball is possible is a situation where one player has the ball, and 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*. See also [2].

A situation pattern is realized as a logical description of spatial - and, to a degree - temporal relations. Robolog and Prolog predicates will be used to define these relations. To ensure robustness of our approach, further methods can be used to aid the decision process, should a match not be close enough.

## References

- [1] P. Baumgartner and U. Furbach. PROTEIN: A PROver with a Theory Extension INterface. In A. Bundy, editor, *Proceedings of the 12th International Conference on Automated Deduction*, LNAI 814, pages 769–773, Nancy, 1994. Springer, Berlin, Heidelberg, New York.
- [2] C. Eschenbach and L. Kulik. An axiomatic approach to the spatial relations underlying *Left-Right* and *in Front of-Behind*. In G. Görz and S. 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.
- [3] International Computers Limited and IC-Parc. *ECLiPSe User Manual / Extensions User Manual – Release 3.6*, 1997. Two volumes.