A Direct Approach to Robot Soccer Agents: Description for the Team Mainz Rolling Brains Simulation League of RoboCup '98

Daniel Polani, Stefan Weber, and Thomas Uthmann

Institut für Informatik, Johannes Gutenberg-Universität,
D-55099 Mainz, Germany
{polani,stefanw,uthmann}@informatik.uni-mainz.de
http://www.informatik.uni-mainz.de/PERSONEN/{Polani,Uthmann}.html

Abstract. In the team described in this paper we realize a direct approach to soccer agents for the simulation league of the RoboCup '98-tournament. Its backbone is formed by a detailed *world model*. Based on information which is reconstructed on the world model level, the rule-based decision levels chose a relevant action. The architecture for the goalie is different from the regular players, introducing heterogeneousness into the team, which combines the advantages of the different control strategies.

1 Introduction

The challenge of constructing intelligent agents capable of coordinated operation, real-time response, handling uncertain information and more has been a main motivation for setting up the RoboCup challenge. A tournament element as propagated in [1–5] introduces the character of a "co-evolutionary" development process on the level of concepts. With a tournament in mind, concepts being developed are subject to a "selection pressure" towards exploitation of "concept space" in contrast to the prevalent exploration character of pure research. The ability to measure the performance of a given concept in objective numbers (i.e. goals) introduces a tradeoff requirement between generality of the approach (allowing flexibility and extensibility) and utilization of specialized knowledge (allowing exploitation of model properties for performance).

To be able to evaluate the importance of accurate modeling versus higher generalizability attempts of creating agents must include both directions of research. In our approach we choose to place more emphasis on a *direct* model which tries to utilize accurately the knowledge available and to restrict tunable (learnable) parameters to domains where precise modeling is not possible or practical.

Furthermore we concentrate on having a world model which is as precise as possible. The world model level factorizes out the inconsistencies of the sensor data and merges them as well as possible, thereby serving as a well-founded basis

M. Asada and H. Kitano (Eds.): RoboCup-98, LNAI 1604, pp. 390-395, 1999.

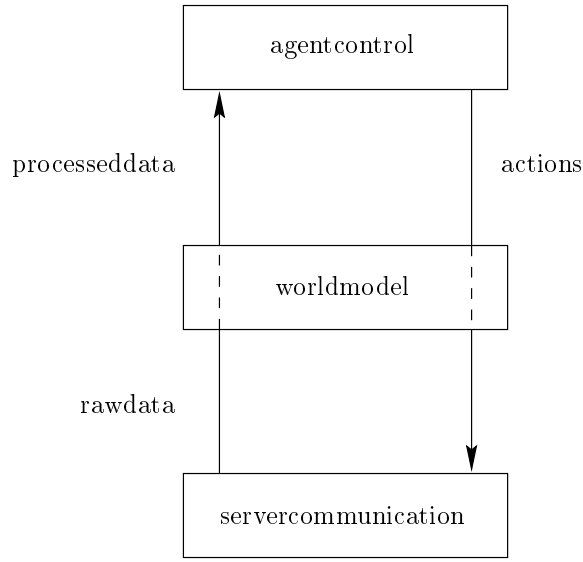

Fig. 1. General architecture of the soccer clients

for higher-level decisions and actions. For instance, it serves to reconstruct the time synchronization with the server. The particular importance of developing techniques for world model design is to be seen in conjunction with sensor fusion techniques for sensor data in real robots. These considerations warranted the particular emphasis which we put on the world model. This document describes some aspects of our MAINZ ROLLING BRAINS team, which participated at RoboCup '98 in the simulation league.

2 Agent Architecture

2.1 General Architecture and Communication Interface

The general architecture of our soccer agents is depicted in Fig. 1. The agents are realized in an object-oriented approach using C++ as language. Communication between agent and soccer server takes place via the *server communication interface*. The data are preprocessed by the *world model* level before they can be utilized by the control level. The actions performed by the control level are not sent directly to the server; instead their expected influence on the world is tracked by the world model and only then they are forwarded to the server.

The communication interface consists of several classes which are responsible for opening the communication to the server and maintaining it. They parse the information sent by the server and convert it into data structures that can be handled by the higher level classes. These raw data are never seen in this form by the *agent control*. Instead they are filtered and processed by the *world model*

level which provides the agent control information not directly obtained from the server.

2.2 World Model

World Model Data Structures World model information can be e.g. position and orientation of the agent itself, estimated speed of the agent and of other agents as well as of the ball, the relative and absolute positions of objects seen, estimated positions of objects not seen and more. The position of the current agent is calculated in a straightforward way from the flags and lines seen, other objects are computed relative to it. Object movement is currently calculated from object positions in two successive time steps if they are to far away to use the velocities given by the server. Further quantities modeled are stamina and effort.

Every data in the world model also possesses a separate validity, which is dropping continuously while the object is not directly detected via the server, and reset to full, whenever the data is being refreshed from the server.

By thus processing the data the world model class provides a framework, in which the more complex decision levels can transparently access the relevant information about the world. The world models construction always assumes that there are "real" quantities of position and velocities behind the values provided by the server and that just their accuracy to which they are known to the agent is not perfect, instead of assuming a fuzzy concept of positions and other quantities as a priori structure. The control level is then free to interpret those quantities as fuzzy ones. In particular, it is a deliberate design choice not to introduce interpretation of higher quality already on the world model level.

In the current implementation the actions dispatched by the control level are only logged by the world model to be able to simulate future developments. They are not modified further, therefore, in a way, the path of the actions sent through the world model level by the control level is much "shorter" than that of the raw data arriving from the communication level.

Consistency of World Model Information An important feature of our world model is the fact that it keeps track how consistent its data are with the data received from the server. Due to the connectionless protocol used in server communication it is not possible for the world model to have a precise and consistent picture of the actions that reached the server and were performed at a certain time step. In other words, it is not known at exactly which time step an action dispatched by the agent control is going to be performed.

To perform an action at the right time step can be vital: imagine a goalie that has to intercept an incoming ball. If for instance, client or network load is high, the last observed sensor data can be several time steps older than the state currently processed by the server. To be able to handle such situations, our world model keeps a time slice of (in our case typically 20) time steps around the estimated "current" server time in memory. On request of the control level the world model provides a snapshot for a given time step inside this time slice.

The data in this snapshot can be generated by different ways. At fixed time intervals the sensor update thread (realized by timer interrupts) fills in the sensor

data for the time step given by the data time stamp. If a snapshot is requested for a time step no later than that of the most recent sensor data, the original sensor data are used if present, or, if not, the missing data are interpolated from neighboring snapshots. If, however, the time step requested for the snapshot lies in the future of the most recent server data, the estimated future world state is calculated, taking into account the actions assumed to be still in the action queue. In the presence of high network or client load it can prove an advantage instead of the snapshot containing the most recent data to use the snapshot of a time step delayed to the future for the control decision.

The world model further provides a measure determining how well the world state as calculated matches the world state as obtained by the sensor update. By this the reliability of the world model forecasts is quantified and can be used as basis for decision. Using these features of the world model our agents attained a high degree of robustness. During informal games with other teams at RoboCup '98 we found that the ROLLING BRAINS agents usually were less sensitive to network or workstation load than agents from other groups. We attribute this strength in particular to the elaborate world model management described above.

3 Agent Control

Agent control structure is represented in Fig. 2. We have a large submodule which is responsible for handling general game situations as occurring during regular play. For special situations like free kick, goal kick and similar we have other submodules which take responsibility in such a case. Delegation of responsibility is not performed by syntactic separation, i.e. separation into structural different modules, but are handled on a semantic level, i.e. in principle in the same context as handling two different regular game situations. The advantage is that it is possible to handle special situations using the same level of deliberative models as regular game, allowing non-"local" strategic and tactic considerations (e.g. delaying a free kick to gain stamina or to reduce the opportunity for the opponent to obtain a goal in the remaining time). An structurally separate consideration of the different situations would have had the advantage of easier design handling, but we were interested to evaluate the merits of the unified approach.

Our current implementation realizes direct situation-based actions, i.e. the current situation as represented in the world model is mapped directly to an action, including actions which include estimates of future development. Our agents therefore act essentially as reactive ones.

The control level strategies we were and still are working on include a fuzzy-rule based and a hierarchy-rule based mechanism. The fuzzy rule approach is currently using a flat set of rules which activate in parallel the different possible actions of the agent (like *kick*, *dash* and *turn*). The degree to which an action is activated is determined by the fuzzy inference. All actions activated beyond a certain threshold are performed (according to a predefined order).

The hierarchical rule concept operates using rules consisting of two parts: the first part is the *condition* for firing the rule, the second part is an action to be performed for a *terminal rule* or subrules to be evaluated recursively for a *meta-rule* (Fig. 3).

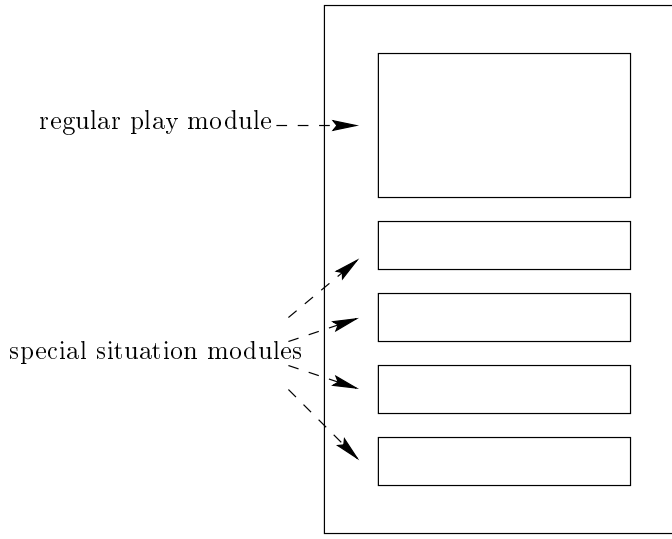

Fig. 2. Structure of agent control level

One always evaluates a set of rules, beginning from an initial starting set of rules. The condition for each rule is evaluated giving a priority for the given rule. The rule with the highest priority from the set is selected. If a terminal rule, the corresponding action (which can be an action sequence) is performed, if a meta-rule, the same rule activation algorithm is recursively applied to the set of sub-rules. Thus it is possible to implement conditions triggering higher level behavior and lower level reactive behavior in the same framework.

The selection of relevant rules and conditions in both approaches is part of the knowledge engineering task involved in the creation of the agents. Tuning the priorities for the different rules is partly expert knowledge, but mainly parameters that have to undergo adaptation for the rules to be of any use. We are attempting to approach their optimization also using Evolutionary Algorithms and related methods.

The two different approaches (fuzzy and hierarchical) have both been realized in the team. The regular players have been designed using the fuzzy representation. The goalie was a completely independent development using the hierarchical approach. It was introduced into the team in a late stage when it became clear that the original version of the goalie using the fuzzy approach was not adequate for its task. Thus the Mainz Rolling Brains indeed realized a heterogeneous team and did not only consist of clones of the same agent, whose actions only would differ by initial conditions or its position in the game. The combination of the strengths of the two different approaches is a good paradigm for a true multi-agent system where the different agent architectures complement each other to attain a common goal.

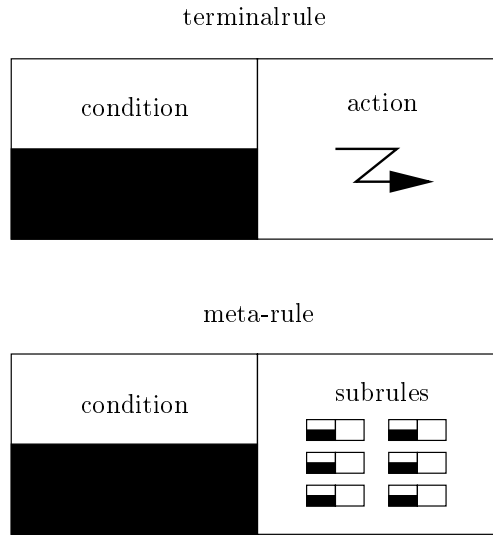

Fig. 3. Rule concept

Acknowledgements

We wish to mention here all the members of the team participating at the development of the MAINZ ROLLING BRAINS agents, namely (in alphabetical order) Christian Bauer, Marc Hellwig, Michael Junges, Oliver Labs, Achim Liese, Christian Meyer, Ralf Schmitt, and Frank Schulz. Furthermore we would like to thank Hans-Dieter Burkhard and the members of his team for their encouragement and helpful discussions, Hans-Jürgen Schröder for finding a sponsor for the project. We also gratefully acknowledge the support by the Herdt-Verlag.

References

1. Hiroaki Kitano. RoboCup: The robot world cup initiative. In *IJCAI-95 Workshop on Entertainment and AI/Alife*, August 1995.
2. Hitoshi Matsubara, Itsuki Noda, and Kazuo Hiraki. Learning of cooperative actions in multi-agent systems: a case study of pass in soccer. In *AAAI-96 Spring Symposium on Adaptation, Coevolution and Learning in Multi-Agent Systems*, pages 63–67, Mar 1996.
3. Peter Mössinger, Daniel Polani, René Spalt, and Thomas Uthmann. XRAPTOR – A synthetic Multi-Agent Environment for Evaluation of Adaptive Control Mechanisms. In F. Breitenecker and I. Husinsky, editors, *EUROSIM '95*. Elsevier, 1995.
4. Peter Mössinger, Daniel Polani, René Spalt, and Thomas Uthmann. A virtual testbed for analysis and design of sensorimotoric aspects of agent control. *Simulation Practice and Theory*, 5(7-8):671–687, 1997.
5. Itsuki Noda. Soccer server: a simulator for RoboCup. In *JSAI AI-Symposium 95*, Dec 1995.