Team Description VirtualWerder2002

David Bernau, Lukas Brachmann, Waleri Enns, Sebastian Huebner,
Magnus Kolweyh, Alexander Silber-Mankowsky, Ubbo Visser (Team Leader),
Jan Weidner, and Bernd Wollny

TZI - Center for Computing Technologies, University of Bremen, Germany
eMail: grp-vw2002@tzi.de
WWW home page: http://www.virtualwerder.org

Abstract. Virtual Werder 2002 is a complete new team, which has been developed from scratch. The team is characterized by three main features: a new world model based on own sensors, a new I/O-synchronisation module and a deliberation module.

1 Introduction

The team *VirtualWerder2002* is a part of the students project *RoboCup*, which also consists of teams that participate in both the Small-Sized League and the Sony Legged League. It consists of eight students of Computer Science at the University of Bremen. The intention of the team is to qualify for *Robocup 2002* in Fukuoka and to be a worthy successor of team Virtual Werder which took part in the events in Melbourne and Seattle.

The key issues of the team are threefold: (a) the world model, (b) a new I/O synchronisation module, and (c) a deliberation module. The world model is based on our own sensor model and plays a central role in our teams architecture. The I/O synchronisation module is divided into two threads, one for the decision-making process and one for the soccer server synchronisation. The deliberation module is based on some ideas from the BDI architecture [3] and from planning methods [1]. Furthermore, we were slightly influenced by the AT Humboldt 99 team [2].

One of the main issues of this years team is the fundamental change of the basic client. While Virtual Werder was based upon the CMU-99 [4]client over the last couple of years, we now have decided to develop an own team from scratch. To date, some features of our team are still not implemented yet. However, we are positive about the ongoing development of our team with regards to the Fukuoka tournament since there is enough time.

2 Architecture

Our architecture consists of, roughly divided, the world model module, I/O-synchronization module and the deliberation module.

2.1 World model

The world model is the central aspect in our team architecture, which uses no given low-level library like the most teams that integrate the CMUnited basic skills. Instead, the world model is completely based on our own sensor model. It acts as storage for all kinds of state information and gets updated with new incoming sensor data. The world model also consists of "old" information, i.e. information about player positions from former cycles. Every agent constructs a world model for itself, which contains the view of the player about his surroundings. Typically, this includes basic information such as positions of team mates, opponents, the agent itself, and the ball. In addition, the player integrates these sensor data and generates predicates, which represent assumptions about concrete game situations, e.g. *nextToBall*, *atGoodPointToScore*, ... Other architecture modules can access the agents' beliefs of the world data using these predicates. Predicates based on actual sensor information contain an uncertainty factor of 1, i.e. this information is believed to be true. Other predicates that are based on "old" sensor information have an uncertainty factor, which decreases with the rate of actuality. This means that the older an information is the more uncertain it is.

2.2 I/O-Synchronisation

Our agent is split into two threads: one makes all the decisions that lead to an intelligent behaviour (*reasoning thread*), the other controls the synchronisation with the soccer server (*synchro thread*). The threads are connected via two queues that contain the timestamped server messages and the commands to be sent.

In order to spare as much computing power as possible for the reasoning thread, the synchro thread sleeps most of the time. Whenever a new simulation cycle starts with a received sense body message, absolute points in time are calculated, at which the commands have to be sent. The offsets are taken from the configuration file and should regard server timing and network lag.

The sleep in between is realised with a call to *select()* on the connecting socket. In case a new server message arrives, the thread awakes, adds the message to the receive queue, signals the reasoning thread and returns to sleep. Otherwise a timeout occurs, the thread takes the command on top of the queue and sends it to the server. Because of limitations of minimal sleep times when using select(), intervals with a length shorter than a given threshold are spend with busy waiting. This threshold can be controlled via the configuration file to suit the requirements of the operating system and the hardware.

2.3 Deliberation

We have adopted some ideas of the BDI-approach [3] for the design of our deliberation module. In addition, general planning ideas [1] have also been adopted. Furthermore, we were slightly influenced by the team AT Humboldt 99 [2].

In order to avoid misunderstandings we will first clarify some terms used in the descriptions stated below.

Predicate: A function returning a probability value (double from the range [0..1]). In the planning layer this value is converted to {false, true} according to certain rules.
Action: A capability of the agent on the tactical level. An action is used as the smallest performance element in the planning layer and is characterized by having pre- and post conditions and a fixed interface, which allows the planner to deal with all the actions in the same way. An action is more abstract than a skill and is generally implemented using them.
Skill: A capability of the agent on the technical level, such as *kick ball*, *pass ball*, *go to location*, ...
Condition: A set of predicates, linked together with logical operators.
Desire: A condition extended by a priority.
Plan: A plan consists of a sequence of actions.

At execution time the agent tries to fulfill the desires specified at compilation time. In detail, the decision process runs as follows:

Step 1: building the state tree
The root state in this tree is created according to the current situation which is taken from the updated world model using the predicates. Then, a whole tree of states is created by applying certain actions to the state currently observed. The applicability of a certain action to a certain state can be decided on basis of the pre-condition of this action. The building of the tree terminates at a fixed depth.

Step 2: determining the desires
In the second step the created tree is filtered for desired states. A state is desired if it logically includes at least one desire.

Step 3: determining the intention
After all, the state is chosen from the desired states, which has the highest utility according to an utility function. The chosen state is then called the intention and is used to create a plan. This plan is a sequence of actions leading to that state.

The created plan is fulfilled by executing all the actions included one after another. In the case of a failure, replanning is started. The decision whether the execution of the whole plan succeeded or failed is determined using the post-conditions of the actions and some other additional conditions.

2.4 Skills

The agent of the team VirtualWerder2002 is able to choose between different kind of various *skills*. The provided skills are structured in an hierarchy nearly conform to the soccer-agents world and architecture. This hierarchy of skills is organized as described as follows.

The main intention of the team VirtualWerder2002 in reference to the implementation of skills was to offer the soccer-agent "real" skills that represent a skill layer, which is more abstract than the basic skills. These skills describe a pool of main skills the agent would choose from e.g. *go to location*, *stop ball*, *goal kick*. The team of VirtualWerder2002 tried to built up skills needed in different situations of a soccer game and specific to the players. Related to this attempt the agent is given a pool of useful skills. These skills often consist of sequences of more atomic skills according to the underlying basic skills. These basic skills also represent a skill layer in a way of abstract parameterised soccer server commands.

3 Conclusion and future work

Although it is possible to find an appropriate and effective hand-coded implementation for a certain skill (say dribbling) it is much more effective and much less time-consuming (when the needed infrastructure is given) to let the agent learn that skill. The second point becomes the more important the more changes the RoboCup simulation league committee decides to introduce to the soccer simulation from year to year. Therefore, it is one of the main goals of our team to develop a powerful and scalable online learning framework. This framework, which is already partially implemented, will utilise an asynchronous, model-based variant of value iteration and will make use of an off-line coach. The benefit of model-free Q-Learning algorithm will be examined. Once the work on that framework is completed, the implemented skills used by VirtualWerder2002 will be incrementally replaced by learned ones.

Furthermore, we intend to improve the decision process by extending the use of uncertain informations, sensed in past soccer server cycles. Because this could lead to choose a non-optimal action, the use of it should be limited to non-critical situations.

References

1. M. E. Bratman, D. J. Israel, and M. E. Pollack. Plans and resource-bounded practical reasoning. *Computational Intelligence*, 4(4):349–355, 1988.
2. H.-D. Burkhardt, J. Wendler, T. Meinert, H. Myritz, and G. Sander. At-humboldt in robocup-99. In M. Veloso, E. Pagello, and H. Kitano, editors, *RoboCup-99: Robot Soccer World Cup III*, volume 1604 of *LNAI*, pages 358–363, Melbourne, AUS, 1999. Springer Verlag.
3. A. S. Rao and M. P. Georgeff. Bdi-agents: From theory to practice. In V.Lesser, editor, *ICMAS*, pages 312–319. MIT Press, 1995.
4. Peter Stone, Patrick Riley, and Manuela Veloso. The cmunited-99 champion simulator team. In Manuela Veloso, Enrico Pagello, and Hiroaki Kitano, editors, *RoboCup-99: Robot Soccer World Cup III*, volume 1856 of *LNAI*, pages 35–48. Springer Verlag, Melbourne, AUS, 2000.