Rete Algorithm Applied to Robotic Soccer

M. Palomo[1], F.J. Martín-Mateos[2], and J.A. Alonso[2]

[1] Department of Computer Languages and Systems, University of Cádiz,
Escuela Superior de Ingeniería, C/ Chile, s/n. 11003 Cádiz, Spain
manuel.palomo@uca.es

[2] Computational Logic Group,
Dept. of Computer Science and Artificial Intelligence, University of Seville,
E.T.S.I. Informática, Avda. Reina Mercedes, s/n. 41012 Sevilla, Spain
{fmartin, jalonso}@cs.us.es

Abstract. This article is a first approach to the use of *Rete* algorithm to design a team of robotic soccer playing agents for *Robocup Soccer Server*. Rete algorithm is widely used to design rule-based expert systems. Robocup Soccer Server is a system that simulates 2D robotic soccer matches. The paper presents an architecture based on *CM United* team architecture for Robocup Soccer Server simulation system. It generalizes the low-level information received by the agent as high-level soccer concepts. This way it can take advantage of expert system design techniques.

1 Introduction

Robotic soccer is one of the most interesting examples of multi-agent systems. In this environment, agents must be able to perform as a team to get a common long-term goal. They have to manage themselves in a real-time, non-deterministic, partially-known world. All this facing a team whose goal is the opposite (both teams can't fulfil their goals at the same time.) These features are common to other problems like hospital or factory maintenance, search and rescue missions, multi-spacecraft space missions, etc.

This paper is focused on simulated robotic soccer. The simulation system allows to deal with high-level problems instead of spending time with low-level details. Scientists can apply automated reasoning techniques (off-line and on-line.) It also offers the advantages of software utilities: recording matches (as video or log files) and replaying them later, playing matches in any moment just with a computer (if the implementations of the teams are available), building teams with exactly the same resources, etc. Anyway, under some constraints, direct implementation of simulated soccer algorithms in real robots is possible [2].

2 Foundations

This paper is a first approach that shows how expert system design techniques (in particular, *Rete* algorithm) can be used to design a soccer playing agent for *Robocup Soccer Server*.

2.1 Expert Systems

Our proposal is based on the use of Rete algorithm [1] to design a rule-based expert system. Expert systems design is a branch of artificial intelligence specialized in the development of systems simulating expert human decision skills. In particular, rule-based expert systems are composed of a set of facts (knowledge base) and a set of if-then statements (rules.) Facts represent information known (or believed) about the world. And rules describe how that information could change. The set of rules are fixed during all the execution of an expert system, but facts usually change. Every rule has two parts: the antecedent (a set of conditions about facts) and the consequent (a set of actions that modify the knowledge base adding or deleting facts.) One of the key features of rule-based expert systems is the possibility of increasing the knowledge (adding new rules) without losing the information in the knowledge base.

Expert systems work in an infinite loop. In each cycle they check the antecedent of every rule. If all the conditions of one of them are true then the rule is activated and placed in the *agenda*. The agenda is the list of all rules which have their conditions satisfied and have not been executed yet. After checking all the rules, one of the activated ones (that are in the agenda) is selected to be triggered. The selection method is called *conflict resolution strategy*, and depends on the implementation of the system. There are several conflict resolution strategies, such as assigning a priority to every rule, choosing the most recent activated, the least frequently triggered, etc. Each one of these strategies has both advantages and drawbacks.

In every cycle of simulation the system checks the rules seeking for new activations, as the triggered rule could have changed the facts in the knowledge base. A direct implementation of an algorithm checking all the conditions of every rules would be too inefficient ($O(R \cdot F^P)$ being R the number of rules, F the number of rules in the knowledge base and P the average of conditions per rule.)

Rete algorithm takes advantage of two facts (as pointed in [5]). First, most of the facts in an expert system don't change from one cycle to another. Thus, it doesn't check all the rules every cycle, but it remembers past facts test results, so only new facts are tested. And second, as several rules can share part of their antecedents a network is created to minimize the number of tests to be made in each cycle. Figure 1 shows an example of optimization of two rules. These advantages produce a more efficient algorithm on the average $O(R \cdot F \cdot P)$.

2.2 Robocup Soccer Server

Robocup Soccer Server [3] is a system that simulates robotic soccer matches in a 2D field. It's supported by the *Robocup Federation* [4] and used in all its competitions.

The code is distributed under the GNU GPL license, and it works with a client-server architecture. In each match there is one server simulating everything concerning the match, and 22 clients (11 for each team), each one controlling

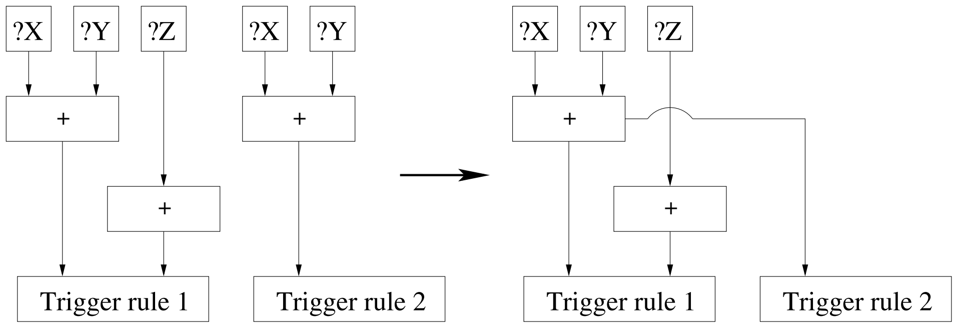

Fig. 1. Rule optimization creating a network sharing facts and an inner condition

one player. Clients are autonomous, and they can only communicate with mates through the server, sending messages in a standard language through an UDP socket. So clients can be programmed in any hardware, any operating system and any programming language as long as it implements UDP socket support. The server implements a low bandwidth, unreliable, communication channel.

The server simulates the world in steps of 100 ms. It accepts one action (like kick or turn) from each client every step, and simulates them all (applying real physics and soccer rules like decay, off-side rule, etc.) The simulation is non-deterministic and all the actions are affected by some noise.

Part of the result of this simulation is sent to every client in each cycle. That information is of three kinds: aural (what a player hears), visual (what a player sees) and physical (what a player knows about himself, like stamina, speed and so.) Every player only receives the information it senses: messages heard, objects seen and physical information.

3 Proposed Architecture

Our proposal is based on the architecture presented by Peter Stone in [2] for *CM United* team. It, basically, uses some static information common to all the players in a team (called *locker-room agreements*) and the information received from the world (partial information) to update the internal state of the agent (Fig. 2.) With those internal beliefs a directed acyclic graph (known as *external behaviors*) selects the action to be made (an example is shown in Fig. 3.)

We propose two modifications to this model:

3.1 Generalization of the Information

In the CM United architecture all the information received from the world is stored as low-level facts (like positions of players, speed of the ball, etc.) But it can be generalized as high-level information relative to objects in the field, in such a way that small changes in the world won't lead to changes in the

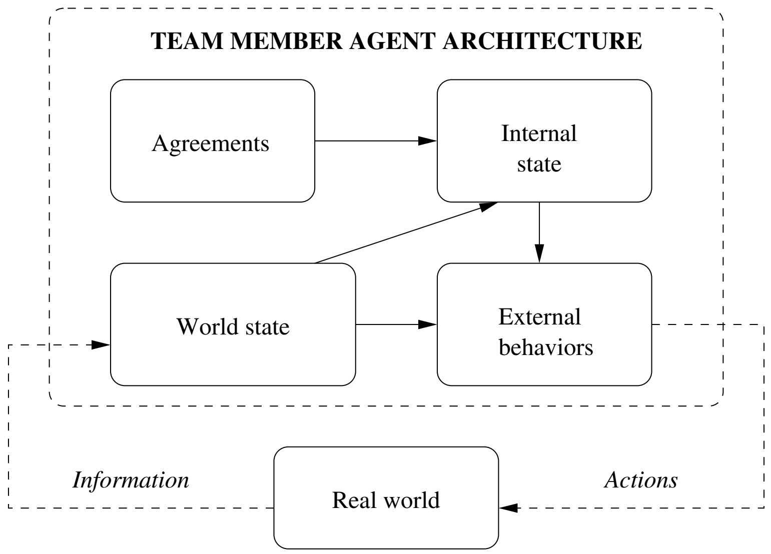

Fig. 2. Simplified internal architecture of an agent

high-level perception of it. Of course, this generalization must be according to soccer concepts: for example, a set of positions of players can be generalized in an "off-side" fact.

This generalization allows the agent to focus on soccer concepts instead of coordinates in a field. And these concepts will only change when an important event (from a soccer point of view) happens.

3.2 Use of a Rule-Based Expert System

The second modification proposed consists in replacing the directed acyclic graph that chooses the action to be performed in the CM United architecture by a set of rules managed with Rete algorithm. In this case the antecedents of the rules must only check high-level properties (so the algorithm won't have to be re-testing a lot of rules every cycle), and the consequent (that indicates the action to be made) uses low-level facts in order to calculate the best action according to accurate information. According to the off-side example, if "off-side" fact exists in the knowledge base, rules as "pass forward" can't be activated. And a slight movement of the player to another off-side position won't lead to the activation of any of those rules. But if the movement avoids off-side, the rules could be activated.

The most suitable conflict resolution strategy is priority-based: assigning a priority value to every rule and selected the activated rule with the highest priority. In the case that several activated rules have the highest priority value, a random selection is performed. This way the agents will be able to take good

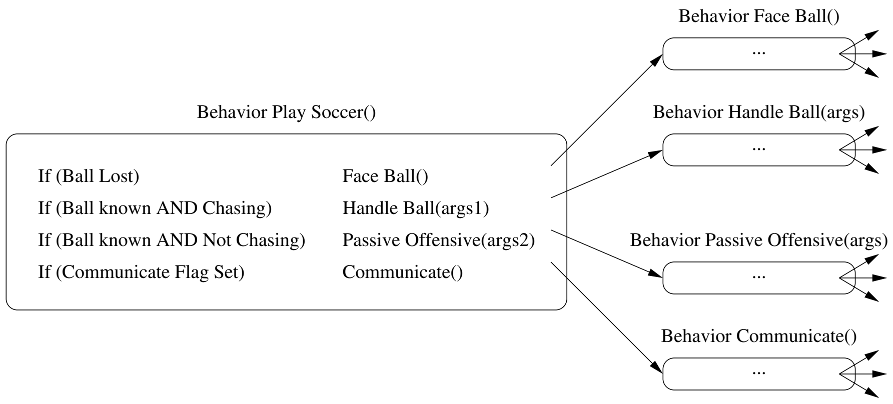

Fig. 3. Example of external behaviors as implemented in [2]

actions (as they have higher priority.) But they won't behave always the same, as in the same circumstances they could take different rules with the same priority.

3.3 Advantages

The modified architecture provides some very interesting advantages:

1. The team can be easily debugged. With a log file of a match (and some help from an engineer) a human expert could check if the concepts generalized in each moment and the actions taken are right, and improve the team. This task is not something obvious in other mathematical approaches to the design of playing agents.
2. Increasing or modifying the system knowledge is as simple as adding or editing rules.
3. Rete algorithm is widely used (there are several free implementations available) and has proved to be very efficient.
4. As Rete minimizes the number of tests to be made in each cycle, the number of rules could be high.
5. Our proposal defines an architecture, not a team. Different concept generalization and different rules design absolutely different teams with absolutely different behavior.

4 Conclusions and Future Work

Robocup Soccer Server is a very interesting test-bed for soccer playing multi-agent systems. This paper is a first approach that shows how expert system design techniques can be applied to design a soccer playing agent for Robocup Soccer Server. One of the main contributions is the generalization of the low-level information received as soccer concepts. With those soccer concepts an expert human could program the agent with simple if-then rules. The second contribution is the introduction of Rete algorithm in the architecture used by the CM United team. That way the rules could be easily processed by the agent.

Anyway there are several details that have to be precised to finish this work:

The use of communication protocols is very important for two reasons. First agents can help each other sharing information known about the world, and second, they need to synchronize periodically (in case they dynamically change the tactic of the team or the roles played by each agent in the tactic.) So the communication protocol should be defined carefully. There are three possibilities to do it: agents can use it on their own automatically (without human control), it can be controlled only by rules, or an hybrid method could be implemented.

The inclusion of *set-plays* can be very interesting. A set-play is a multi-agent plan fired by some condition. Their main advantage is that during their execution agents know where mates must be, so they can act faster and more accurately. An interface to define set-plays would consist of three parts: definition of the activation conditions, specification of the actions to be taken during its execution and definition of the conditions to cease it.

Robocup Soccer Server allows the use of an on-line coach in each team during the match to assist players. It's a privileged agent that receives all the information of the environment and can send messages to players periodically. It works as a soccer coach: analyzing the game and the opponent, and sending information to players. It could be implemented in lots of different ways, as long as it send the players interesting facts for them to play better.

Finally we are developing a batch file to test teams. It will be used to test our final architecture (implementing different sets of rules and generalizations). The batch file will program as many matches as desired versus some of the top-level teams in previous *Robocup World Championships* and will collect the results.

References

1. Forgy, C. Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem. *The Journal of Artificial Intelligence*, Vol. 19, 1982, pp. 17-37.
2. Stone, P. *Layered Learning in Multi-agent Systems*. PhD thesis, School of Computer Science, Carnegie Mellon University, 1998.
3. Robocup team. Soccer Server System. *http://sserver.sourceforge.net.*
4. Robocup Federation home page. *http://www.robocup.org.*
5. Giarratano, J. & Riley, G. *Expert systems: principles and programming*, third edition. PWS Publishing Company, 1998.