FCFoo - a Short Description

Fredrik Heintz, frehe@ida.liu.se

Institute of Computer Science at Linköping University

Abstract. This paper describes FCFoo, a RoboCup team developed at Linköping Institute of Technology.We use a reactive-deliberate architecture composed of four different layers. The layers represent different levels of abstraction and deliberation. The decision-making is based on decision-trees, finite automatas, and roles. As a result of the team development we have created a C++ library, designed for educational use, called RoboSoc, with extensive low and middle level functionality.

1 Introduction

This paper is a short description of our RoboCup team FCFoo. The agents are built on a layered reactive-deliberative architecture. The four layers describes the agent on different levels of abstraction and deliberation. The lowest level is mainly reactive while the others are more deliberate. The teamwork is based on finite automatas and roles. A role is a set of attributes describing some of the behaviour of a player. The decision-making uses decision-trees to classify the situation and select the appropriate skill to perform. The other two layers are used to calculate the actual command to be sent to the server.

The agent architecture and the basic design are inspired by the champions of RoboCup'98, CMUnited [3, 4]. The idea of using decision-trees and roles is inspired from Silvia Coradeschi et al [1, 2].

As a result of the team development we have created a C++ library called RoboSoc. The library is designed for educational use. Therefor we have added not only the standard functionality like information processing and containers for the different objects, but also primitive actions and complete skills. Examples of primitive actions are turn to absolute direction 45 degrees and kick the ball as close as possible to <x,y> with one kick.

The structure of the paper is as follows: In section 2 we describe the agent architecture. Section 3 is devoted to teamwork and section 4 is about the decision-making process of the agent. We conclude with a short discussion on possible improvements in section 5.

2 Agent architecture

The aim of our agent architecture was to create a robust system that is able to reason on different levels of abstraction. This makes it possible to reason about whether the attacker should intercept the ball or make himself easy to pass when the team is attacking, not what sequence of dashes and turns to do. At another level it is possible to reason about what sequence of dashes and turns to do to actually intercept the ball. Those levels can work in parallel independently of each other. The architecture should also take care of all the low-level problems like process synchronisation among the different parts of the agent and communication with other processes and the soccer server.

We decided to use a layered reactive-deliberative architecture since our agents needs both to react to sudden changes in the environment, like an opponent popping up in front of us when we have the ball, and to do more advanced planning, like where shall I go to assist my teammates as much as possible. It also enables us to use different abstraction levels in each layer. Figure 1 shows an overview of the architecture. The arrows indicates between what units communication takes place. The different parts are described below.

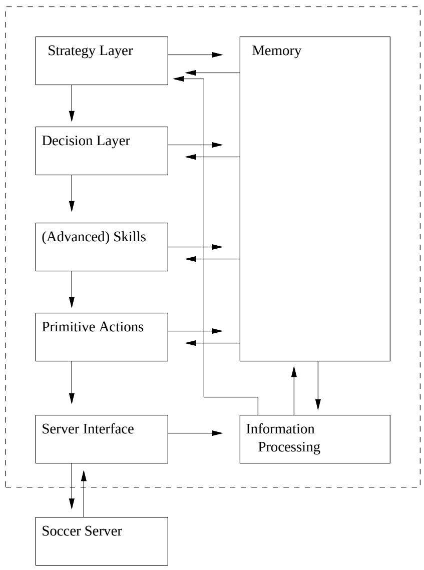

Fig. 1. The agent architecture.

The **server interface** takes care of the communication with the server. Both sending commands and receiving sensor input.

The **information processor** processes the sensor data that come from the server and updates the memory with the new information.

The **memory** keeps track of the information that the agent needs to know. The most important data structures are those that keep track of the agent, the players (both opponents and teammates), and the ball. The information is not only dependent on sensor inputs but it is also updated each cycle of the simulation and when the agent acts in the world.

The **strategy layer** is responsible for the team level planning. This layer is identical in all agents. The output of this layer is the behaviour that the agent believes the team should do in the current situation.

The **decision layer** is responsible for the execution of the behaviour decided by the strategy layer. This is how the individual agent plans to help the team. The output is a specific skill the agent wants to perform.

The **skills layer** is the backbone of the agent. It contains the different skills the agent can execute. They are among others dribble, score, pass, make passable, and clear ball. The skills are developed in cooperation with the other team from Linköping, the Headless Chickens. The output of this layer is a primitive action.

The **primitive actions layer** calculates the parameters of the primitive actions that the agent can perform in the server. The output is a complete command ready to be sent to the server.

3 Teamwork

Our teamwork is based on roles. Each player is given an initial role. The role define where he should position himself on the field, when to take the freekicks and so on. The roles can be assigned and changed during the game. The roles are mainly used when making a decision on what to do next.

Limited communication is used mainly to share vital information, like information about objects, roles, and the internal state of the sender.

3.1 Roles

A role is a set of attributes that defines the behaviour of a player. A player can have only one role. But since it is possible to define new roles as combinations of roles it is actually possible for a player to have several roles. Problems with this approach are among others that one have to define a specific role for each role-type. Attributes can have different values for each individual having that role and therefore specialisation is supported.

Examples of attributes are: freekick-area, home-position, and home-area. They define where the player with this role should be and when he should do the freekicks. Examples of roles are: goalkeeper, defender, midfielder and attacker. Examples of specialisations are: left inner midfielder and right outer defender.

4 Decision making

The decision making of the agent is based on decision-trees (DT) and finite automatas (FA). The automatas are used in the strategy layer to decide what DT to use. The automata describes when to change states, and DT, and when to communicate with the other players. The triggers used by the FA are referee-calls, messages from other players and the position of the ball. Figure 2 shows

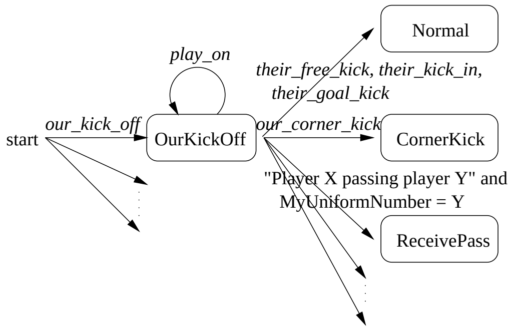

Fig. 2. Part of the finite automata used by an agent making a decision.

a part of the FA used by the agents. The italic words on the arrows are referee-calls, the words inside quotes are messages from other players and the words inside the boxes are the name of the DT to use when in that state.

The DTs are used in the decision layer to classify the situation and decide what skill to execute. When the decision layer has found a skill to perform it sends that information to the skills layer. The skills layer calculates what primitive action to do based on the current position and internal state of the agent. The primitive action layer takes the action with its parameters and calculates the actual command to be sent to the server.

5 Further work

The further work will be focused on the RoboSoc library. The first public release will be in august, but already the team is completely built on it. The team itself also needs much improvements when it comes to higher level reasoning. The decision making could be vastly improved by adding for example learning and refined finite automatas.

References

1. Silvia Coradeschi and Lars Karlsson. A Role-Based Decision-Mechanism for Teams of Reactive and Coordinating Agents. In Hiroaki Kitano, editor, *RoboCup-97: Robot Soccer World Cup I*. Springer Verlag, Berlin 1998.
2. Silvia Coradeschi and Paul Scerri. A User Oriented System for Developing Behavior Based Agents. In Minoru Asada and Hiroaki Kitano, editors, *RoboCup-98: Robot Soccer World Cup II*. Springer Verlag, Berlin, 1999.
3. Peter Stone and Manuela Veloso. The CMUnited-97 simulator team. In Hiroaki Kitano, editor, *RoboCup-97: Robot Soccer World Cup I*, pages 387-397. Springer Verlag, Berlin 1998.
4. Peter Stone, Manuela Veloso, and Patrick Riley. The CMUnited-98 Champion Simulator Team. In Minoru Asada and Hiroaki Kitano, editors, *RoboCup-98: Robot Soccer World Cup II*. Springer Verlag, Berlin, 1999.