Puppets:
Development on Evolutional Action Architecture

Hiroki Shimora[1], Tomohiro Odaka[2], and Hisakazu Ogura[2]

[1] Graduate School of Engineering, Fukui University, Fukui-shi 910–8507 Japan
`shimora@rook.fuis.fukui-u.ac.jp`
`http://bishop.fuis.fukui-u.ac.jp/%7Eshimora/RoboCup/`
[2] Faculty of engineering, Fukui University, Fukui-shi 910–8507 Japan
`odaka@rook.fuis.fukui-u.ac.jp, ogura@nqueen.fuis.fukui-u.ac.jp`

Abstract. The RoboCup soccer system is one of very complex systems. Owing to it's complexity, agent creator employed for building various algorithms, tuning parameter combinations or other trivial matters. In the past, most part of such works are made up manually. To automate the works, we propose "Evolutional Action Architecture". In this architecture, agent creator builds agent by making new actions and composing them. Parameters in each action are controlled in evolutional system and thay are evolved automatically. This architecture helps agent creator to build intellectual agent.

1 Introduction

The RoboCup soccer system is one of very complex systems. It's state space has large dimension and relationship of each state is non-deterministic because of each agent's autonomy. Creating programs for such complex system requires much man-power. Agent creator employed for building various algorithms, tuning parameter combinations or other trivial matters. In the past, most part of such works are made up manually. There are many thresholds, many predicates, many functions and many other parameters. In addition, these have complicated relationships each other.

2 Overview

We propose "Evolutional Action Architecture" for easily agent creation and more intellectual agent creation.

Figure 1 shows overview of this architecture. Details about "Agent Environment" and "Action" are described in section 3.

This architecture has an "Evolutional Server" outside of agent. "Action" has zero or more specific parameters or structured programs which is not known what value or program is the best. We call this "Problem". "Action" has "Problem"s depends on itself. "Evolutional Server" tries to find the bests using evolutional methods that "Evolutional Server" has. "Problem" is independent of specific evolutional methods. One of the evolutional methods might be Genetic Algorithm, Neural Network or tuning by human.

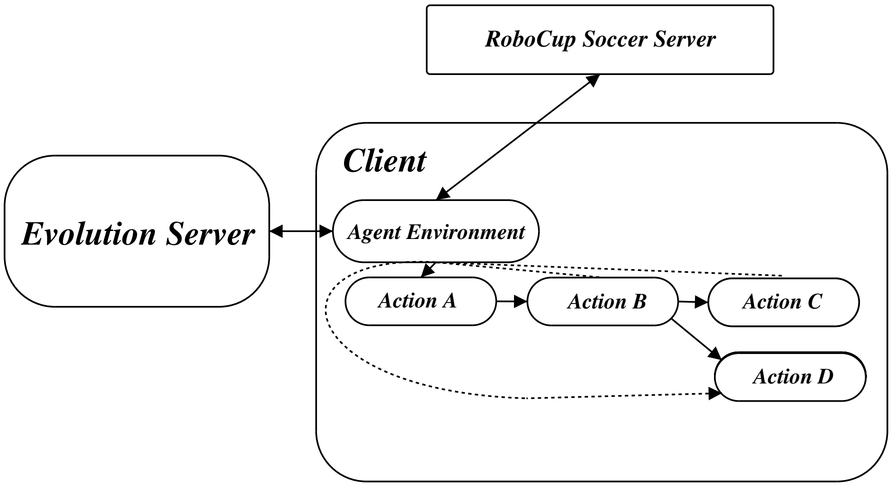

Fig. 1. Overview

3 "Agent Environment" and "Action"

Each player in Team Puppets is based on RoboCup soccer library system created from scratch. This system implements "Evolutional Action Architecture". This architecture consists of one "Agent Environment" and many "Action"s.

"Agent Environment" provides low level management which includes inter server-client communication, step controlling, field recognition updating, debug log management, debug viewer interaction, and so on. Agent creator have to know how to get field information from "Agent Environment". But agent creator does not have to know more about "Agent Environment".

"Action" is a basic component in "Evolutional Action Architecture". Agent creator builds agent by making new "Action"s and composing them. "Action" is provided as an abstract class. Agent creator makes new specialized concrete "Action"s by inheriting. Each concrete "Action" manages decision making to

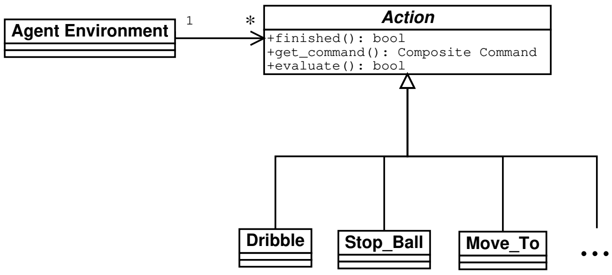

Fig. 2. Action Class Diagram

perform individual feature. (e.g. stop ball, dribble, shoot, find teammate) Concrete "Action" can create and use new "Action" objects to perform in itself.

"Action" has following methods. Agent creator can override these methods.

- finished
- get_command
- evaluate

"finished" method returns a boolean value. True means this "Action" does not have any more commands. False means this "Action" have command in this simulation step. If false returned by "finished", "Action" is ready for generate composite command by receiving "get_command" message. Composite command is a set of commands server executable in one step. (e.g. "(dash 100.0) + (turn_neck 30.0) + (change_view narrow high)") For example, Move_To "Action" returns true if agent reaches target point, otherwise false. Finished "Action" is managed in "Action Environment" and the "Action" will receive "evaluate" message every simulation step. "evaluate" method investigates result and returns more "evaluate" is needed or not. Generally evaluation is not determinable at "finished" step. It usually takes a while to determine the evaluation.

In "Evolutional Action Architecture", each action will evolute automatically by evolutional methods which system provides or agent creator makes.

Therefore this architecture helps agent creator to build intellectual agent.

4 Future Works

Now in our system, "Evolutional Action Architecture" is implemented partially. More works has needed to complete all of the architecture. By RoboCup 2002 Fukuoka, we'll finish all implementation.