YowAI2007 Team Description

Hiroki Asakawa[1], Masafumi Ueda[1], Yasuyuki Yamazaki[2], and Ikuo Takeuchi[1]

[1] Graduate School of Information Science and Technology,
The University of Tokyo,
Bunkyo, Tokyo 113-8656, Japan
ask@nue.ci.i.u-tokyo.ac.jp
[2] Dept. of Computer Science,
The University of Electro-Communications,
Chofu, Tokyo 182-8585, Japan

Abstract. In this paper, we propose three techniques named short shout, strategic shout and semi-reflection. Shouts are one of the means to achieve cooperation human players use in real soccer. We have achieved short term and reflective cooperation by using short shouts. Team plays are cooperation which involve more than two players and some long term tactics. To achieve team play, we developed a human-like agent communication "Strategic Shouts". Strategic shouts can spread the team plays objectives and adjust each agent's priority of behavior, so that a sort of strategic team play emerges. We implemented the semi-reflection function on RoboCup soccer agents. We developed agents which have the ability of self-monitoring, quick self-correction and malfunction report that demands programmers to do debugging. The semi-reflection function improved the agent performance and reduced developer's burden of agent development.

1 Introduction

The target of YowAI2007 is realizing "human-like agents". A human-like agent is an agent which makes its own action decision and cooperation with other agents by a similar way as a human player takes in actual soccer. To achieve this goal, we introduce three ways; short shout, strategic shout and semi-reflection.

2 Short Shout

In real human soccer, communications among players do not include precise numbers. They use eye contact and short shouts that are bare minimum information they need. For example one says "Agare" (which means "move forward" in Japanese) when he wants to tell "Move forward the defense line". One of the reasons why they don't say it fully is that it is impossible to say it in a flash, and it is hard to understand in a wink. YowAI soccer agents use this short shout not only for short tactical terms, but for sharing information without using any cipher or data compression. By communicating player's tactical move, a shout

gives the receiver a hint what he should do next. To send strategic information by short shouts one has to abstract the situation relevant to the strategy. To understand abstract information represented by a short shout the receiver has to attach the meaning of the short shout to what situation he understands. Therefore more intelligence is needed than communicating accurate numerical values. Using short shouts in RCSS has been one of the answers to achieve team cooperation. YowAI agents cooperate with each other by "short shout" like humans.

2.1 Short Term Tactical Shout

We have achieved short term and reflective cooperation by using "short term tactical shout". A player can progress the tactical situation of the team, and give the addressed player a hint to the tactical judgment and action selection. The judgment or action selection is highly recommended. The following example is a short term tactical shout.

2.2 Mae

When one controls the ball and there is a space in front of a particular teammate to pass, he will say "Mae". The actions the sender and the receiver (agent number 10) should do will be the following.

sender Calculate where to pass, kick the ball, and send message "`(say "Mae 10")`".
receiver Run forward to prepare for receiving the pass.

"Mae" will bring the ball forward and make an advantage in attacking (Figure 1).

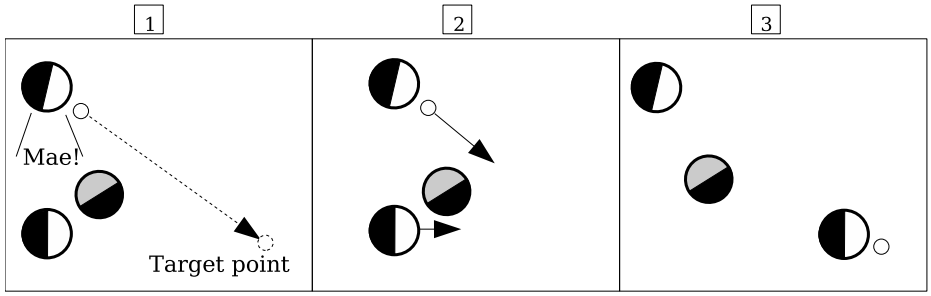

Fig. 1. Mae

2.3 Gotzan

When one has a wider and easier shooting range than the teammate which controls the ball at that time, he will say "Gotzan". The teammate that controls the ball will pass the ball to the sender of "Gotzan" if he can't shoot. "Gotzan" will make better shoot chance and raise the goal rate (Figure 2).

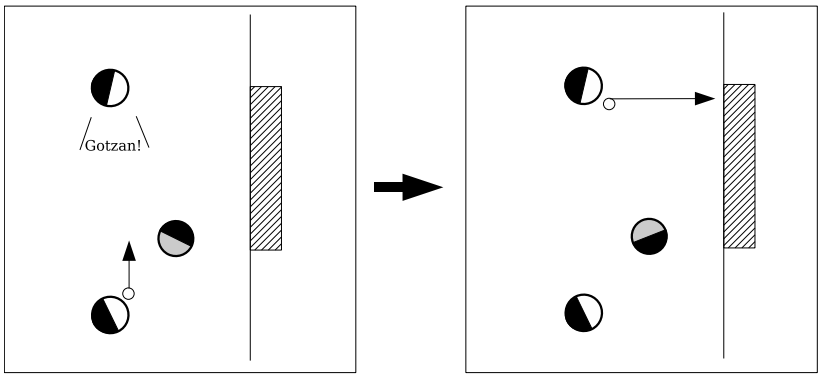

Fig. 2. Gotzan

3 Strategic Shouts

3.1 Strategic Team Play

As shown in examples in sections 2.2 and 2.3, shouts did achieve cooperation but were more like instantaneous pair wise combinations of related players than team plays or team plans. The shout effects the members that are related to the ball which are only two: the sender and one receiver. On the other hand, team play is a part of team planning that includes not only combination chains but also formationing. Team plays can effect not only the members near the ball but also the members far from the ball. These make team plays an advanced cooperation than those cooperation achieved by short shouts.

Strategic team play is a team play whose target is to make an advantageous strategic situation. To achieve strategic team play we developed "strategic shout".

3.2 Strategic Shourts

There are two important points in strategic shouts; a strategic shout effects more than one teammates, and one can use short shouts during the combination chains in strategic team play. We have already developed combination plays by short shouts and therefore we use strategic shouts to spread the target of the team play to some or entire teammates. The teammates involved in the team play will make the best decision to success the team play moves. The following section describes an example of strategic shouts.

3.3 Back2Toward

The target of a strategic shout named "Back2Toward" is to break through the opponent's defense line. Against top teams high skill defenses, an advanced offense skill has been needed. One answer to this is Back2Toward. By passing the

ball back once to a player that can see the opponent's defense line exactly than the passer, it will make it easier to break through the opponent's defense line. The moves related to the ball are as follows.

1. If the ball controller thinks that it is hard to carry the ball forward and cannot find a good pass course, he will say "Back2" (Figure 3 -1).
2. Let a teammate be "teammate A". If teammate A thinks that there is a pass course from the ball controller to him and there is a chance to pass the ball to a teammate more forward than where the ball is located, he will say "Toward N". N is the number of a particular teammate he thinks the best teammate to pass the ball to at that time.
3. The ball controller will pass the ball to teammate A.
   During the pass, teammate N will move to make a clear space in front of him (Figure 3 -2).
4. After teammate A gets the ball, he will first check if he can pass it in front of the player N. If he can, he will pass it to him. If cannot, he has to find a better teammate to pass.
5. When passing the sender might use short shouts such as "Mae"(Figure 3 -3).

During the movements above, other teammates have to help the cooperation if they can. They may get theirselves marked by opponents on purpose to make the pass course clear, or make clear spaces in front of them to be deputies of the teammate N (Figure 4). These movements of teammates will push up the success rate of Back2Toward and make it not a chain of combination but a strategic team play.

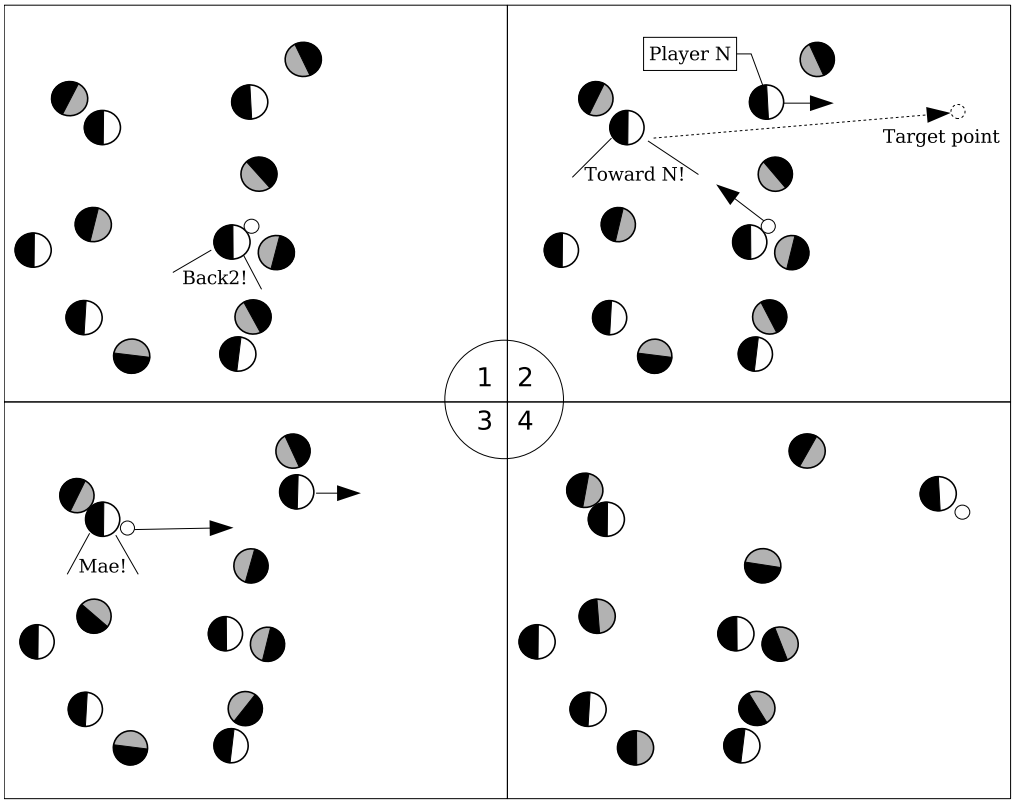

Fig. 3. Back2Toward; Only players related to ball

4 Semi-reflection

4.1 Reflection

A reflection is one of effective ways to construct flexible systems on dynamic environments. Reflection is a mechanism that can sense self status and change itself. A system that has a reflection mechanism can modify its function and reform itself by referencing and modifying inside information and coded instructions.

We propose semi-reflection in order to apply the concept of reflection to RoboCup Soccer. We build a system which has the ability of self-monitoring, quick self-correction and malfunction report.

4.2 Semi-reflection

Under semi-reflection, we don't change program itself which triggered malfunctions but detect suspicious actions and unusual behavior of agents and do quick self-correction. A semi-reflection system logs the information of action modifications and environment when it works. Developers can modify the program code

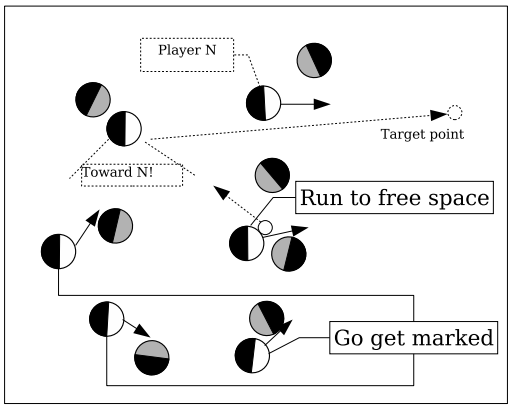

Fig. 4. Non related players' movement from Figure 3 -2

to see that logs. Because a modification is quick and impromptu, we call it semi-reflection. On complex environment, it is hard to construct reflection system that modifies program codes in order to respond to unexpected circumstances. Since an agent must respond quickly to urgent problem, it choose an action from simple and reliable alternative actions.

4.3 Self-monitoring agents

On our study, we implement malfunction detection and problem solving function as a self-monitoring agent. Figure 5 shows self-monitoring agent and main agent. Self-monitoring agent and main agent are in the same agent but they act as a different agent. That means self-monitoring agent sees itself from the third person.

Self-monitoring agent gathers information about the main agent's world model, logs of actions and history of objectives. It evaluates the world model, actions and objectives of the main agent. When it detects a malfunction, it commands modification of the world model and alternative action to the main agent. At the same time, it logs that information and informs it to the developer.

4.4 Observation of main agent

Our previous version of YowAI had following problems.

- Malfunctions on unexpected occasions.
- Meaningless repeat of same action
- Inconsistent actions

Those problems caused by lack of meta-cognitions of agent. For example, an agent doesn't have cognition of what it knows and what it doesn't know or what its objective is and why it does current action. To solve this problem, we make self-monitoring agent that observes agent's world model, actions and objectives.

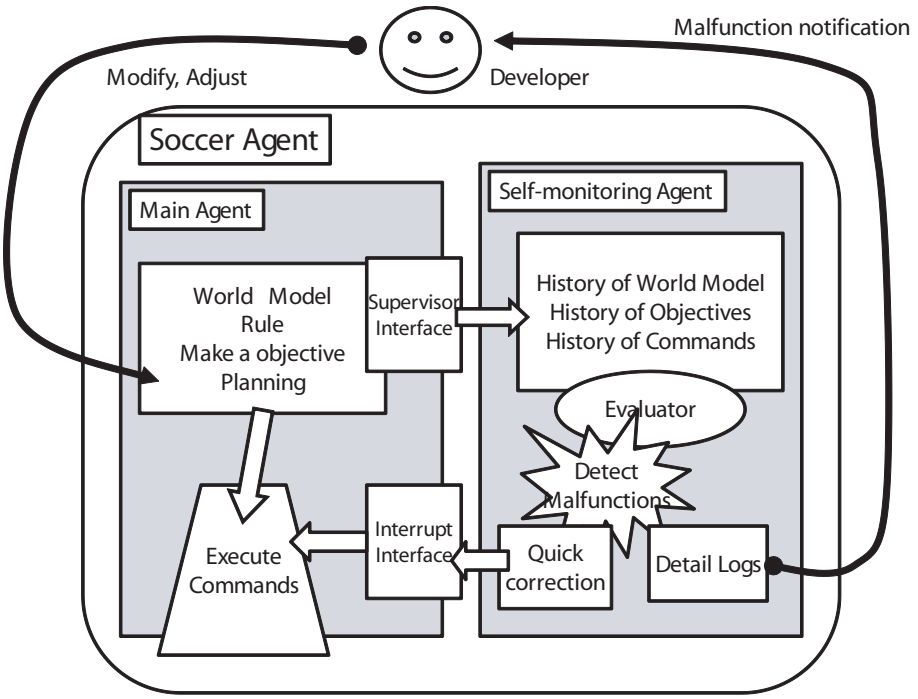

Fig. 5. The framework of semi-reflection

4.5 Modification of action

The main agent makes a plan and executes a command to achieve a plan on each cycle. An agent make a plan based on its world model, but sometimes it fail to choose an action or a plan because of incomplete information and dynamic environment. Especially because an action needs more accurate information of environment, it is hard to choose a appropriate action. We focused on following actions that we inspect.

- Apparently unnatural actions or plans for soccer rules.
- Meaningless repeats of same action
- Inconsistent changing of plan.

It may be feasible to prevent those problems by main agent planning. But when you edit your huge agent program codes, it costs a large amount of labor and it introduces greater complexity into structure of agent. Consequently, when you split a self-monitoring function as module, your program is more clear and easy to maintain. Besides, an agent can inspect problems more objectively because self-monitoring agent evaluates using different logic from the main agent.

Self-monitoring agent detects malfunctions through checking a history of main agent's actions. A history of actions is a log of executed actions and

agent's plans. When self-monitoring agent detects malfunctions, it interrupts the main agent's behavior decision and force to change its actions to alternative action which self-monitoring agent chose. Described above, self-monitoring agent doesn't modify program code itself, but it commands a specific action.

4.6 Detection of malfunctions

Self-monitoring agent detect malfunctions using *validness* which is calculated by evaluating whether main agent's action is feasible for current plan and whether its plan is appropriate for soccer rule. We introduced *requirement* to define the minimum requirement for main agent's actions and plans. Self-monitoring agent calculates *validness* by comparing agent's actions and plans to *requirement*. When *validness* is 0, that is assumed malfunctions. More and more close to *requirement*, it is feasible action and plan.

We show an example to calculate *validness* of pass objective. Minimum requirements of pass are following.

Rule the target of pass is not over offside line.
Knowledge the target of pass is in the soccer field.
Strategy the target of pass is not too close to self position (not in its kickable area).

When a pass doesn't fulfill above requirements, *validness* is 0. The *requirement* for pass is following.

requirement 1 distance between the position of ball and the target of pass is about 1/3 of the soccer field.
requirement 2 The direction of target of pass is the direction of opponent goal.

We calculate *validness* by comparing above *requirement*s and main agent's actions. For example, when a agent which has a ball and it is going to pass to a teammate which is at backside of the passer, *validness* is calculated following. We defined the distance between the position of ball and the target of pass as 25.5m, the angle of between the direction of the target of pass and the direction of opponent goal as 170 degrees, and BestPassDist as 17.5m (1/3 of the soccer field).

$$
\begin{aligned}
\text{BestPassDist} &= 17.5 \\
\text{MaxPassDir} &= 180.0 \\
\text{DiffDist} &= min(25.0, 2 \times \text{BestPassDist}) - \text{BestPassDist} \\
\text{Requirement} &= 100 \times (1 - \text{DiffDist}/\text{BestPassDist}) \\
&= 57 \\
\text{Requirement} &= 100 \times (1 - min(170.0, \text{MaxPassDir})/\text{MaxPassDir}) \\
&= 6
\end{aligned}
$$

$$\begin{aligned}\text{Validness} &= (57 + 6)/2 \\ &= 32\end{aligned}$$

4.7 Detection of unusualness repeats

Not every repeat is malfunction. Dribbles are repeats of kicks and dashes. Described in the above section, *validness* is calculated by comparing *requirement*s and actual actions. When a single low *validness* action is detected, you can't always assume it as malfunction. But when actions which have low *validness* are repeated, you can assume them as malfunctions. For example, when an agent is going to pass, it is natural for that agent to kick the ball to near itself in order to bring the ball to best position to pass. But if that cycle is repeated, that agent can't pass the ball forever and can't achieve its objective.

In order to detect repeats of action which has low *validness*, self-monitoring agent logs action which has low *validness*. Self-monitoring agent evaluates validity of action considering the history of action which has low *validness*. We used three types of history; short term, midterm and long term. Self-monitoring agent can detect malfunctions which repeat in variety of periods. We defined short term as 15 cycles, midterm as 300 cycles and long term as 6000 cycles.

4.8 Detection of inconsistent actions

When an agent's objective varies with each cycle and it goes hither and thither without being able to achieve its objective, we assume it as inconsistent actions. It is inevitable to a certain degree for an agent to change its objective in dynamic environment and with incomplete information. Consequently, using history of objectives, we detect inconsistent actions as following.

- Objective itself is changed many times in short term.
- The direction of target of objective is changed many times.

For example, the change of objective itself is a occasion when an agent chose a shot in a certain cycle but in the next cycle it chose dribble, and in the next cycle it chose pass. Similarly, changing the direction of target of objective is a occasion when an agent chose a dribble forward in a certain cycle but in the next cycle it chose dribble backward, and in the next cycle it chose dribble right hand.

4.9 Quick self-correction

We used simple alternative actions and plans. When an agent is going to do invalid actions, self-monitoring agent modified them to more appropriate predefined actions or plans.

Predefined actions and plans are not the best actions and plans, but when an agent is going to be wrong, it is more important to make an agent to choose

another action. Because quick self-correction is ad hoc modification, it is better to choose simple and reasonable actions in short computational time than to use more power to look for optimum actions.

5 Summary

In this paper, we introduced our thoughts for RoboCup Soccer agent development. We aims to realize more human-like agent on RoboCup Soccer.

We use human-like agent communication to achieve team plays. When agents use human-like communications, they can receive hint of requirements and decisions by teammates. We developed " Strategic Shouts "as a human-like agent communication to achieve team plays. By using strategic shouts, agents can share the objectives of team plays and make decision on what to do to help success team plays.

We introduce the concept of semi-reflection to RoboCup Soccer. An agent which uses semi-reflection can work more stably on unexpected occasions. We split an agent to main agent and self-monitoring agent so that self-monitoring agent can observe its situation more objectively.

Currently we are interested in team play without any communication. Humans can do team plays thinking other players' attentions. Humans see other players actions and read other players' attentions. We will construct other players' models in an agent and realize non-communication team plays.