RoboCup BDI Agent

Jeonghwan Lee, Ziyad Rabeh, Rui Chen

ABSTRACT

In this report, we are introducing three models used for Jason agents based on BDI Architecture and their implementations in RoboCup project, to illustrate how those agents could perform practical reasoning.

Keywords: BDI Architecture, AgentSpeak, Jason Programming Language

I. INTRODUCTION

In order to implement RoboCup agents using BDI architecture, we used agent programming language Jason, which provides abstract interpretation of BDI model. In the implementation, the agent makes decisions based on the interactions between Jason BDI engine and RoboCup environment.

BDI Architecture

BDI is proposed to represent the theory of practical reasoning, which includes three attitudes: beliefs, desires and intentions. Beliefs are characteristics from the environment and are updated after each action. Desires represent the goals to be achieved. Intentions are the current chosen plan to execute [1].

AgentSpeak and Jason

AgentSpeak is an abstract framework for BDI architecture. It provides elegant notations for agent programming. The formalisation of the ideas behind the BDI architecture is expressed using modal logics.

Jason is a Java-based interpreter that implements the operational semantics of AgentSpeak [2].

II. ARCHITECTURE

In order to use only Jason BDI engine, we implemented our own agent class called “BDIAgent” where the Jason infrastructures are not used. The BDIAgent class is based on “SimpleJasonAgent” which extends “AgArch” class in demos of Jason library to be used by the Jason engine without all Jason IDE. The class functions as an overall agent architecture for the AgentSpeak interpreter, i.e., it sends agent’s perception from the environment to the interpreter and get the agent actions resulted from the AgentSpeak reasoning cycles [3].

Fig. Process Diagram from perception to action

III. DESIGN & IMPLEMENTATION

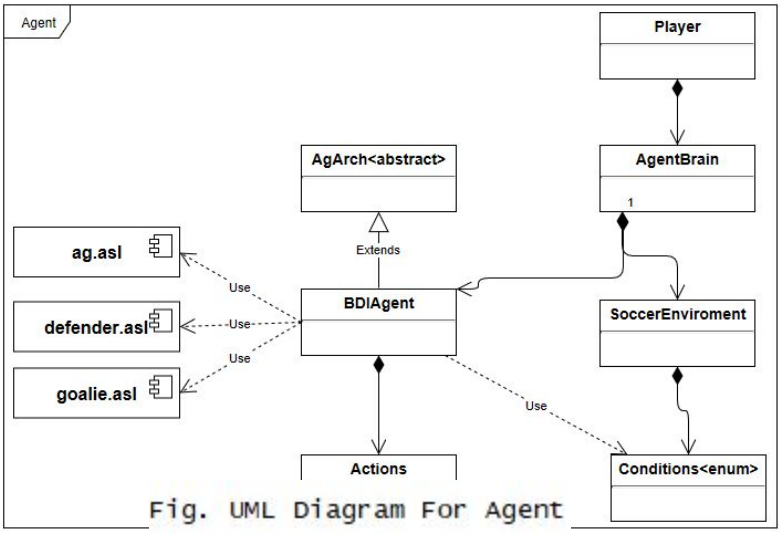

Fig. UML Diagram For Agent

We designed and implemented three models for RoboCup agents based on their roles: Attacker(ag.asl), Defender(defender.asl) and Goalie(goalie.asl). BDIAgent extends abstract class AgArch and initialize the agent from asl files. It gets the conditions calculated from SoccerEnviroment and uses them to add perceptions. Then BDIAgent generates actions and sends them to AgentBrain. Then AgentBrain would execute the actions. Below is the diagram of the agent.
The field is divided into three areas, one area for each role. Each player will stay in his own area if he cannot see the ball or the ball is too far away.

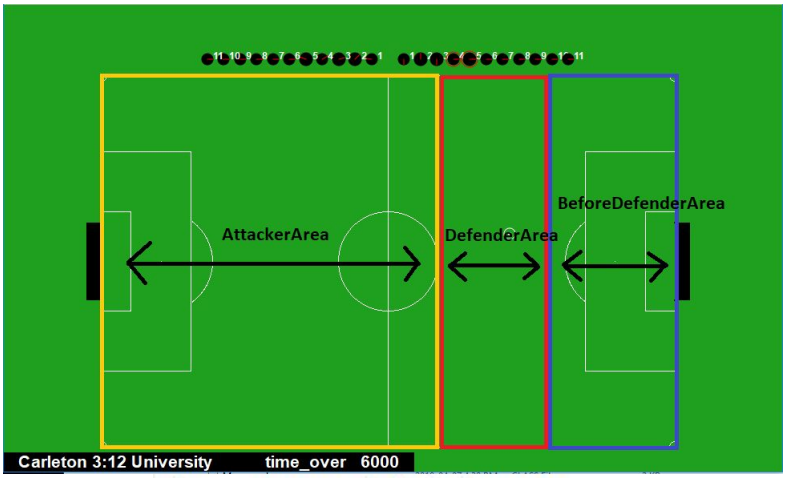

Fig. Soccer Field Diagram

Attacker

The attacker's main goal is to kick the ball and score. Only the closest attacker to the ball would try to kick it, the second attacker just jogs and waits in case the attack fails and there is counter attack.

```
//if you are in defense area and close to the ball,
//| join the defense and  try to kick the ball.
+!backToAttackerArea
    :   closeToBall & beforeDefenderArea
    <-  dash_to_ball;
        -closeToBall;
        -canSeeBall;
        -beforeDefenderArea;
        !move.
```

Defender

The defender works as the first line of defence, he tries to kick any incoming ball so that the opposite team cannot score. If the ball is too far away and the defender is in the defender area, he just observes and waits for any incoming attack. In case the defender cannot see the ball or the ball is too far away and he is not in his area, the defender's goal becomes to go back to his area.

```
// If the ball is away and you are not in the defender area,
//then go back to defender area.
+!findBall
        :   (awayFromBall | farAwayFromBall) & not inDefenderArea
        <-  observe;
            -farAwayFromBall;
            -awayFromBall;
            -canSeeBall;
            !backToDefenderArea.
```

```
// If you found the ball and its in medium range,
//dash and try to kick it.
+!findBall
        :   awayFromBall | closeToBall
        <-  dash_to_ball;
            -awayFromBall;
            - closeToBall;
```

Goalie

The goal defender is the final line of defence for the team. His job is to stay next to his team's goal and to observe until there is an attack and then he will try to kick and clear the ball. In case the goalie cannot see the ball or the ball is too far away and he is not in his area, the goalie's goal becomes to go back to his post next to his own goal. If the goalie kicked the ball but the ball was blocked by opposite team and it was still close to him, he will try to pursue it and kick it again.

IV. TEST & EVALUATION

We implemented agents based on their roles and successfully tested via multiple simulations if each agent works properly. In order to evaluate if the agent makes a decision appropriately on BDI architecture, we faced a problem that showed the agent was not able to choose the most rational decision based on only his perceptions. The scenario below illustrates the problem:

A goalie kicked the ball outside the penalty area and was going back to his own goal. However, if the own goal and the opposite goal become invisible on the way back and the goalie does not know his position on the field, he is in trouble of choosing his action only based on perceptions. We added the belief called "goingToOwnGoal" to his belief-base.

```
180
181 +!backToOwnGoal
182     :   not nextToOwnGoal & not awayFromOwnGoal & goingToOwnGoal
183     <-  turn;
184         !backToOwnGoal.
```

As the above code, in order to achieve "backToOwnGoal", the goalie was able to choose "turn" action to see the own goal because he believed he was going to own goal although his perceptions were not enough.

V. CONCLUSION

In this project we implemented five robocup players in which each belonged to one of three roles. Each role has separate BDI model that it uses to reason about its environment perceptions in order to generate new beliefs. Based on current beliefs and desires, the players deliberate and generate appropriate intentions.

Using the added power of the BDI architecture, we were able to create more complex behaviours and players that performed very well and were able to win all the matches against regular reactive players .

REFERENCES

[1] Baris Tekin Tezel, Moharram Challenger, & Geylani Kardas.A metamodel for jason BDI agents. doi:10.4230/OASIcs.SLATE.2016.8

[2] Hans-Dieter Burkhard, Markus Hannebauer, & Jan Wendler. (1998). Belief-desire-intention deliberation in artificial soccer; doi:https://doi.org/10.1609/aimag.v19i3.1397

[3] Jason FAQs. Retrieved from http://jason.sourceforge.net/faq/#_is_it_possible_to_use_only_the_jason_bdi_engine