ROBOCUP WITH JASON BDI AGENT

Pang Hongbo, Busa Akshay Dheerubhai, Kshirsagar, Jay, Patel Yash Subhashkumar

Department of Systems and Computer Engineering, Carleton University

ABSTRACT

BDI-based reasoning is implemented in Robocup Simulation by leveraging the Jason BDI engine. Jason uses Agent speak (L) to pass on beliefs, desires, and intentions into its engine. Such ASL files are used to provide the logic for individual types of players in soccer such as goalie, defender, and attacker. We have developed one such team consisting of one goalie, two defenders, and two attackers. Krislet sends the perceptions which are then parsed by Jason's agent to derive intentions. Even though agents have distinct plan sets based on individual beliefs they can be drafted in such a way as to make them work collaboratively.

1. INTRODUCTION

This paper states creating a Robocup agent using the BDI model to play soccer in Krislet using the AgentSpeak(L) language Jason. Jason is an open-source AgentSpeak language which can be used with high-level programming languages like java. The paper discusses the utilisation of Jason to create a Robocup agent by adding the belief in the agent's system. Based on the belief, the agent will decide its intentions based on the BDI model and act according to the intention. We have built three different types of agents based on their model. Each of them has different intentions based on their belief and acts as an attacker, defender and goalie of soccer. In this paper, we will go through the methodology behind it, the validation of the agents and a guide on how to run the code.

2. BDI-KRISLET MODEL

Our aim of integrating Jason BDI[2] with RoboCup simulation[1] is sufficed by adapting krislet as an interface. Krislet provides us with the current state of the environment and it is then mapped with attributed perceptions. A method is defined by extending Jason's AgArch class to pass current perceptions into BDI engine for action extraction.

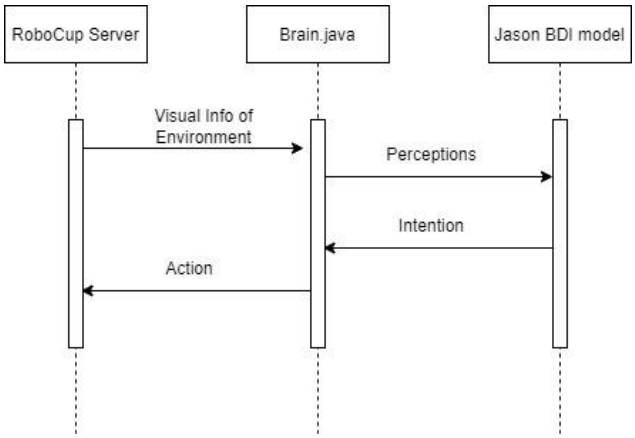

Figure 1

Krislet's brain.java file is a point of connection between the Jason agent and the RoboCup server.

Visual info from servers is mapped as perceptions such as ball_visible, self_goal_visible, etc. These perceptions and additional goals such as !find_ball, and !keep_goal are added to the agent's current belief space. And finally, to execute the derived intentions we have tagged them as dash, kick, find_ball, defend_goal, etc.

3. TEAM STRUCTURE

We've created a team structure of five players, performing three different roles. In team, We've one goalies, two defenders and two attackers. The tasks of attacker will be to find enemy goal and score as much as can and also be in their attacker zone only. The defenders task is to be in defending zone and defend the goal and kick the ball to enemy side when ball reaches near them. Goalie's task will be to stop the goal scoring from goalie's area. Below figure shows the initial positioning of the team in the field. Additionally, we've also defined football ground into three zones, that are Attacker zone (a_zone), Defender zone (d_zone), Goalie zone (g_zone) that can be utilized by different players while finding intentions.

3.1. Attacker

We've two attackers that are placed on the field as shown in the Figure-X. The main objective of the

Figure 2

attacker is to score goal. For that it'll kick the ball closer to the enemy goalpost.

The attacker agent will begin by locating the ball first by turning. Once the ball is located, it'll then turn to the ball, so it is facing the ball. After that, it'll rush to the ball by dash. Once the agent is near ball, the agent will locate the enemy goal post and kick the ball towards that. And again finally goes into finding the ball again.

Additionally, Attacker will only be in the attacker zone only. While chasing the ball, if attacker enters the defenders zone, then it'll stop there and will start observing ball. It'll start chasing the ball again once the ball is going towards enemy goal.

3.2. Defender

This agent works as the first line of defense and its primary goal is it stops the ball from approaching the self-goal post. The defender's logic starts by moving into a strategic position in the field. Once the position is set it will look for the ball to act on it. Unless the ball is in the desired range defender will just monitor the ball. If the ball is in the defending range then it will kick towards the opponent's goal. In a scenario if the ball is not visible and defender is not in zone then it will go to its position.

3.3. Goalie

The goalie is the last line of the defence. The goalie's responsibility in this project is to stop the opposing team from scoring within the goalie's area. The goalie agent starts in front of the team's goal (-50, 0), with a 5-unit difference in the x-axis to the flag c. It will not be allowed to pass the goalie zone, represented by a red semicircle, as Figure 3 describes. With a radius or 18, the goalie agent will check its distance to the top center flag (flag c|t), center flag (flag c) and bottom center flag (flag c|t). If the distance from the agent to the top center flag is less than 32 or 43 less than the top center flag and bottom center flag, it will return to the original position. The original position is defined as the semicircle's edge with a radius of 3 from the flag g|I. The goalie agent observes the ball constantly, and it will dash to the ball if the distance to the ball is less than 23, representing the yellow semicircle in Figure 3.

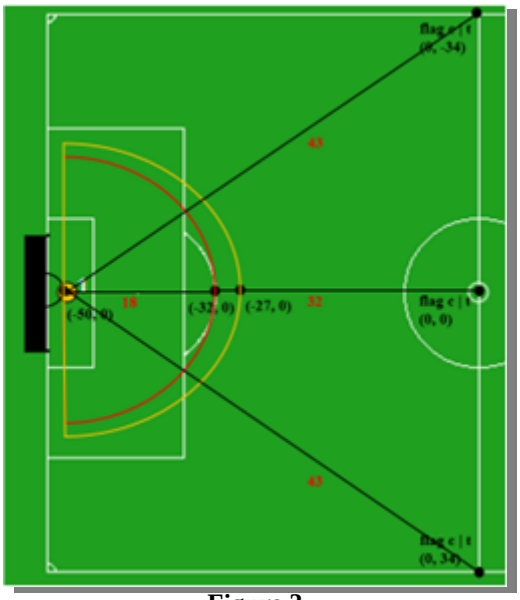

Figure 3

Initially, the goalie's goal is to keep the goal since it is placed in front of the goal before kicking off. It stays in the initial position until the ball enters its defending zone, and the agent will then dash toward the ball to kick the ball. If the goalie agent passes the goalie zone when approaching, it will head back to the original position. In the meantime, if the goalie sees the ball heading back, it will also kick the ball backward to defend the goal. Kicking will happen when the agent has a distance of less than one to the ball, while the goalie agent will choose which direction to kick based on current perceptions. If the agent can observe the enemy's goal, it will kick in that direction. If the agent only observes the goal for its team, it will kick backward. If it cannot observe both, it will kick to the side to avoid the opponent from scoring.

4. MODEL VALIDATION

The first level of validation is done on the plans written in the *.asl files of each type of agent. We wrote a testing file with the main() function, which takes hardcoded belief and, based on the belief using the BDI agent, We apply it to get the intention. Then, we compare

this actual intention with the intention written in the plan. We have verified all the plans this way.

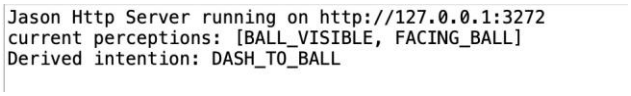

Figure 4

The next level of testing we have done is integration testing. We needed to integrate the BDI model with Krislet's Brain.java and use the agent to build a RoboCup player based on the type given for the agent (attacker, defender, goalie). This integration testing has been done by creating a bat file for two attackers, two defenders and one goalie and then running the bat file to see whether the players are moving according to the model we have built.

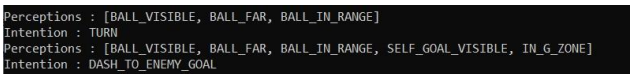

Figure 5

4. RUN CODE LOCALLY

Below are the steps to run the Robocup agent locally.

1. First, Dowload the zip file from the submission and extract it.
2. Run Robocup Server executable file and then monitor file.
3. To run a match between Krislet and our team please run MyTeam.bat file from Krislet Directory to run our team's agent.
4. And run TeamStart.bat file from Krislet-Original Directory to run opponent team.
5. To start the match, do kickoff from monitor.

Whole code can be found at github repo.

5. CONCLUSION

To conclude, The Robocup agent can be built for the non-deterministic environment using the AgentSpeak(L) language Jason and BDI model. The agent uses its sensors to scan the environment and add the beliefs to its model. Using this belief, the agent will make a rational decision on which plant to execute, written in the *.asl file and generating the intention as an outcome. The agent will perform the action mapped to its intention using the intended outcome. We specified multiple agent specifications based on the types of soccer players, attacker, defender and goalie. They have different plans for the same beliefs and perform different actions based on the model. Overall, this paper demonstrated using the BDI model using the AgentSpeak language Jason to design various Robocup players.

6. REFERENCES

[1] Robocup federation official website, Nov. 2021. [Online]. Available: http://www.robocup.org/.

[2] Jason. [Online]. Available: http://jason.sourceforge.net/wp/

[3] B. Esfandiari, Software agent course. [Online]. Available: https://www.sce.carleton.ca/cgi-babak/agentcourse.cgi