

Developing a Team of Gold Miners Using *Jason* (a preliminary design)

Jomi F. Hübner¹ and Rafael H. Bordini²

¹ FURB, Brazil

² University of Durham, UK

1 Introduction

This document describes an overview of a multi-agent system formed by a team of gold miners to compete in the Multi-Agent Programming Contest 2007 (the Gold Miners scenario). The objective is to test and improve *Jason*, the interpreter for an agent programming language used to implement the MAS. *Jason* [2, 4] is an agent platform based on an extension of an agent-oriented programming language called AgentSpeak(L) [6]. The language is inspired by the BDI architecture [7], hence based on notions such as goals, plans, beliefs, intentions, etc.

2 System Analysis and Design

The analysis and design of the system is based on our previous team that entered the CLIMA Contest in 2006 [3]. There are two kinds of agents in the team: miners and leader. Miners are the agents that interact with the contest simulator and the leader helps the coordination of some activities.

2.1 Leader

The leader helps the miners to coordinate themselves in two situations. It initially allocates miners to quadrants of the scenario based on their starting location; they will thus look for gold in different places. A simple “quadrant allocation protocol” is used in this case (Figure 1). The second situation of coordination is the negotiation started when a miner sees a piece of gold and is not able to collect it (because its container is full). This miner broadcasts the gold location to other miners who then send bids to the leader. The leader chooses the best offer and allocate the corresponding agent to collect that piece of gold (Figure 2). The protocol also states that whenever some agent decides to go to some gold location, it should announce it to others (so they can reconsider their intentions). Similarly, they should announce when a piece of gold is collected.

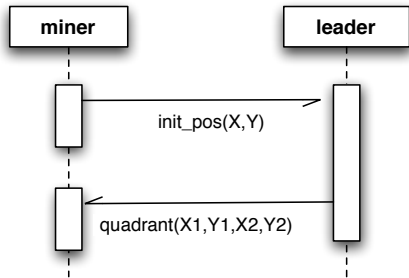


Fig. 1. Quadrant Allocation Protocol.

2.2 Miners

All miners have the same individual goals.³ These goals are mutually exclusive (Figure 3) and the conditions for adopting and dropping them are:

find gold : search for gold in the environment. This goal is the initial goal of the agent and is adopted when there is nothing else to do. The drop conditions are: the miner sees a piece of gold or some gold was allocated to it. We plan to use two strategies to achieve this goal. The first is to systematically (rather than randomly) scan for gold in the miner's quadrant. The second is to always go to the near location least visited. The agents might change their strategy during the simulation.

pick gold : go to the location of some piece of gold and pick it up. This goal is adopted when the agent has space in its container and either sees gold or is allocated to one by the leader. The drop conditions are: another agent has picked that gold or has committed to it; or the miner sees gold that is nearer than that one it is going to collect.

go to depot : go to depot to drop all carried pieces of gold there. This goal is adopted when the miner is carrying at least one piece of gold and the cost of going to catch another known gold is higher than first go to depot and after collect that gold. The evaluation of this cost should consider the fatigue of the miner. The drop conditions is: to see a piece of gold when having space to pick it up.

One of the existing software engineering methodologies which is particularly suitable for BDI agents is the Prometheus methodology [5]. Figures 4 and 5 are depicted in the notation of that methodology to briefly give an idea of the overall system and the miner agent specification, respectively.

³ Shared or global goals are not explicitly represented or handled by *Jason*. If we have time, we plan to use the MOISE^+ organisational model that allows the specification of social goals and the coordination required to achieve them.

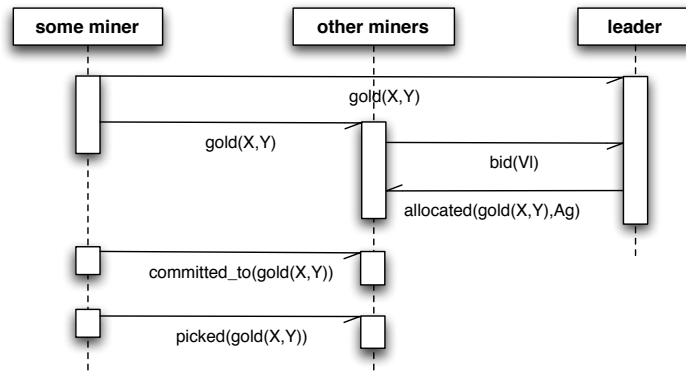


Fig. 2. Gold Allocation Protocol.

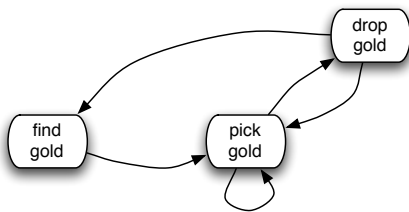


Fig. 3. Goal Transition.

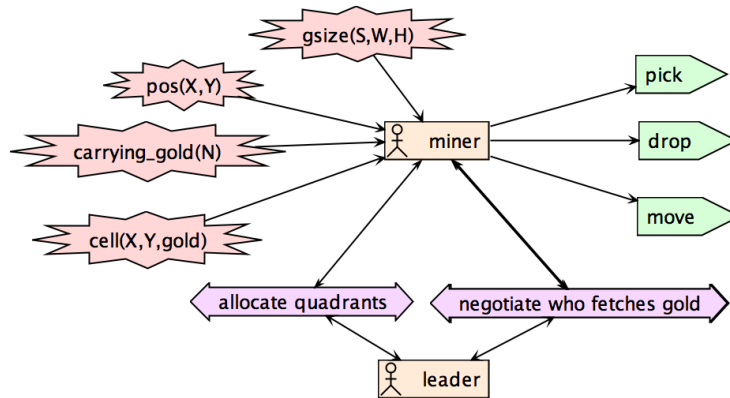


Fig. 4. System Overview Diagram.

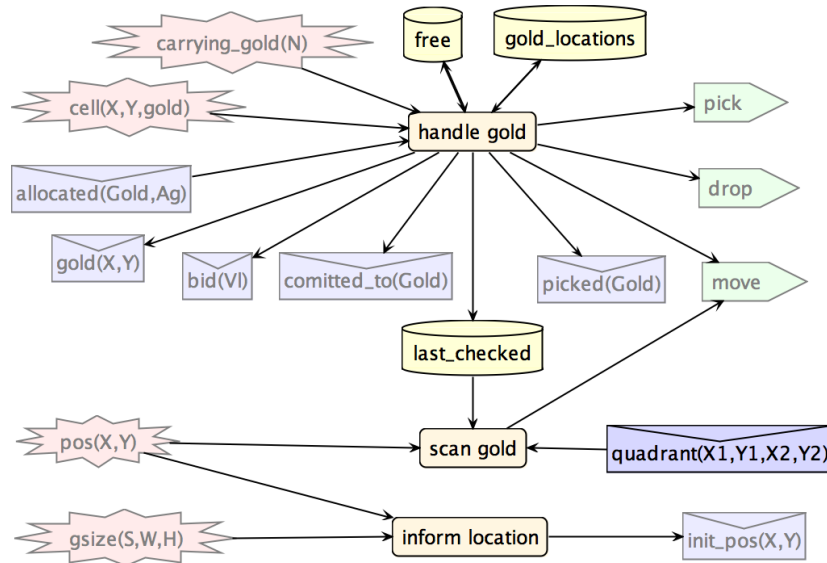


Fig. 5. Miner Agent Overview Diagram.

3 Software Architecture

To implement our agent team, two features of *Jason* were specially useful: architecture customisation and internal actions (see Figure 6). A customisation of the agent architecture is used to interface between the agent and its environment. The environment for the Agent Contest is implemented in a remote server that simulates the mining field, sending perception to the agents and receiving requests for action execution. Therefore, when an agent attempts to perceive the environment, the customised architecture sends to the agent the information provided by the simulator, and when the agent chooses an action to be performed, the architecture sends the action execution request to the simulator. This architecture customisation also allow us to easily change the contest simulator to our local simulator by simply choosing another architecture; using a local simulator makes testing much easier.

Although most of the agent code will be written in AgentSpeak, some parts are to be implemented in Java, in this case because we aim to use legacy code. In particular, we already had a Java implementation of the A* search algorithm, which we plan to use to find paths and calculate distances in the various scenarios of the competition. This algorithm was made accessible to the agents by means of *internal actions*. The more information (specially obstacles) about the scenario is available for A*, the better it performs. So when an agent sees an obstacle, it broadcasts this information to all agents so that they can update their world model accordingly (unlike in [3], we will not use shared memory for obstacle information).

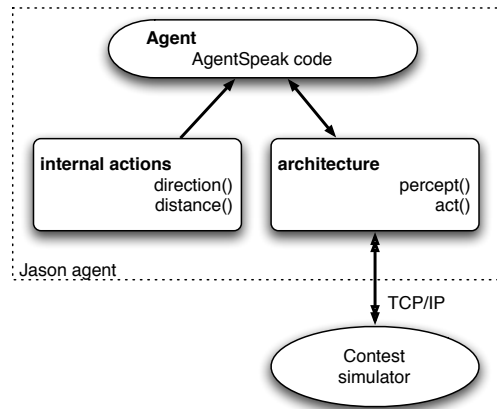


Fig. 6. Agent Architecture.

4 Final Remarks

Clearly, this design is still preliminary and needs to be much more carefully considered (including the Prometheus diagrams) before the actual competition. The various supporting techniques we might need to use within the BDI agents (e.g., *MOISE*⁺) are, as mentioned above, not completely decided yet. Only after we experiment with the *Jason* team in Agent Contest simulations we will be able to make final decisions about which techniques will work best. The general strategy outlined above may need to be changed too as we start experimenting with the *Jason* team.

References

1. R. H. Bordini, M. Dastani, J. Dix, and A. El Fallah Seghrouchni, editors. *Multi-Agent Programming: Languages, Platforms and Applications*. Springer-Verlag, 2005.
2. R. H. Bordini, J. F. Hübner, et al. *Jason: A Java-based Interpreter for an Extended version of AgentSpeak*, manual, version 0.9.5, Feb 2005. <http://jason.sourceforge.net/>.
3. R. H. Bordini, J. F. Hübner, and D. M. Tralamazza. Using *Jason* to implement a team of gold miners. In *Proc. of CLIMA VII*, LNAI 4371, pp. 304–313. Springer-Verlag, 2006.
4. R. H. Bordini, J. F. Hübner, and R. Vieira. *Jason* and the golden fleece of agent-oriented programming. In Bordini et al. [1], chapter 1, pp. 3–37.
5. L. Padgham and M. Winikoff. *Developing Intelligent Agent Systems: A Practical Guide*. John Wiley and Sons, 2004.
6. A. S. Rao. AgentSpeak(L): BDI agents speak out in a logical computable language. In *Proc. of MAAMAW'96*, LNAI 1038, pp. 42–55, London, 1996. Springer-Verlag.
7. A. S. Rao and M. P. Georgeff. BDI agents: From theory to practice. In *Proc. of ICMAS'95*. AAAI Press / MIT Press, 1995.