Using *Jason*, $\mathcal{M}$OISE$^+$, and CArtAgO to Develop a Team of Cowboys

Jomi F. Hübner[1], Rafael H. Bordini[2], Gustavo Pacianotto Gouveia[3], Ricardo Hahn Pereira[3], Gauthier Picard[4], Michele Piunti[5], and Jaime S. Sichman[3]

[1] Federal University of Santa Catarina, Brazil
jomi@das.ufsc.br

[2] Federal University of Rio Grande do Sul, Brazil
r.bordini@inf.ufrgs.br

[3] University of São Paulo, Brazil
{jaime.sichman,ricardo.pereira1,gustavo.gouveia}@poli.usp.br

[4] École des Mines de Saint-Étienne, France
picard@emse.fr

[5] Università di Bologna, Italy
michele.piunti@istc.cnr.it

1 Introduction

This paper gives an overview of a multi-agent system simulating a team of cowboys to compete in the Multi-Agent Programming Contest 2009. This edition of the contest uses a "Cows and Herders" scenario, similar to the 2009 contest but now extended with fences that require cooperation an coordination to be opened. In the previous contests we tested and improved *Jason* and its integration with other tools, in particular the organisational platform provided by $\mathcal{M}$OISE$^+$. *Jason* [2] is an interpreter for an agent-oriented programming language that extends AgentSpeak(L) [6]. The language is inspired by the BDI architecture [7], therefore based on notions such as beliefs, goals, plans, intentions, etc. $\mathcal{M}$OISE$^+$ is an organisational framework [5] that includes: (i) a language used to program the organisation of the MAS with concepts such as groups, roles, missions, global goals; and (ii) a platform that provides the necessary services for the agents to manage and operate within organisations.

The participation in the last contests has contributed to our experience both in programming agents with *Jason* and in using BDI concepts. In the 2006 contest, the focus was on creating agent plans [1], which resulted in rather reactive agents. In the 2007 contest, the focus was on (declarative) goals [3], leading to more pro-active, goal-directed agents. In the 2008 contest, the focus was on the definition of the organisation of the MAS, leading to more social-aware agents [4]; instead of communication only (as in previous years), roles, groups, and common goals were also considered in the last edition of the multi-agent programming contest.

This year, we were motivated to continue to improve and evaluate the integration of *Jason* with other technologies. Besides agents and organisation, we had hoped to

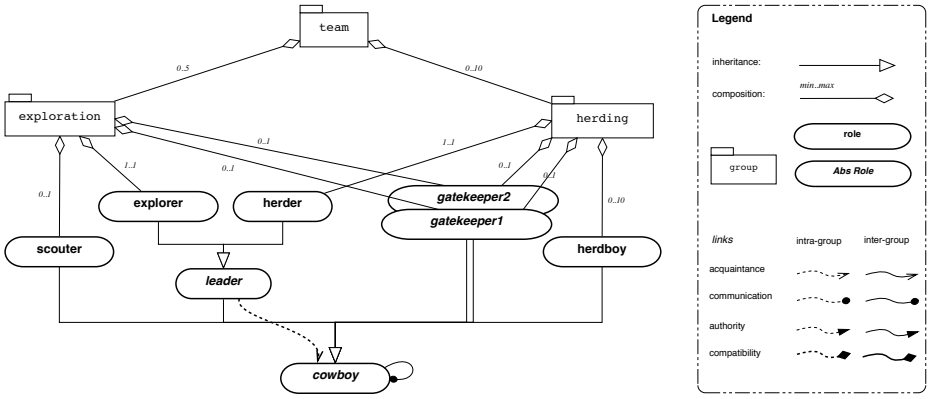

Fig. 1. The Structural Specification of the Organisation.

also use *artifacts* that could help the agents in shared tasks [9]. Artifacts provide mechanisms to externalise functions that currently are implemented as internal actions in *Jason*. The system would therefore be developed in three dimensions: agents (using declarative goals), organisation (using groups, roles, and shared goals), and artifacts (using external, coordinating operations). Our objective in participating in this contest was originally twofold: (*i*) to continue to test and improve *Jason* and its integration with other tools ($\mathcal{M}$OISE$^+$ and CArtAgO); (*ii*) evaluate the use of artifacts in the development of the team. Due to lack of time, we had to drop the use of artifacts in the implemented team for this edition of the agent contest, and have left it for future work (hopefully for the next edition).

2 System Analysis and Design

It is clear, from the description of the scenario, the importance of cowboys working as a coordinated team. It would be very difficult for a cowboy alone to herd a group of cows. As in the previous edition, we adopted a strategy strongly tied to the notion of group of agents where issues such as spatial formation, membership, and coordination would be emphasised.

The overall analysis of the team is the same used in the previous contest, since the scenario is very similar; we refer the reader to [4] as in the space available we can only discuss the main additions to team developed for the last edition of the agent contest. The organisational structure of the team is specified in Fig. 1 using the $\mathcal{M}$OISE$^+$ notation. Compared to the previous edition, the structural specification now has two new roles, called gatekeeper1 and gatekeeper2. This scenario requires two agents to cooperate to open the fence to allow their team members and cows to pass, and they also need to coordinate their action, as discussed below.

The two new roles created are used to handle the new feature of the scenario for this edition of the competition of fences that agents and cows need to pass through. They are the key roles in the new $\mathcal{M}$OISE$^+$ scheme called Pass-Fence (see Figure 2) which is

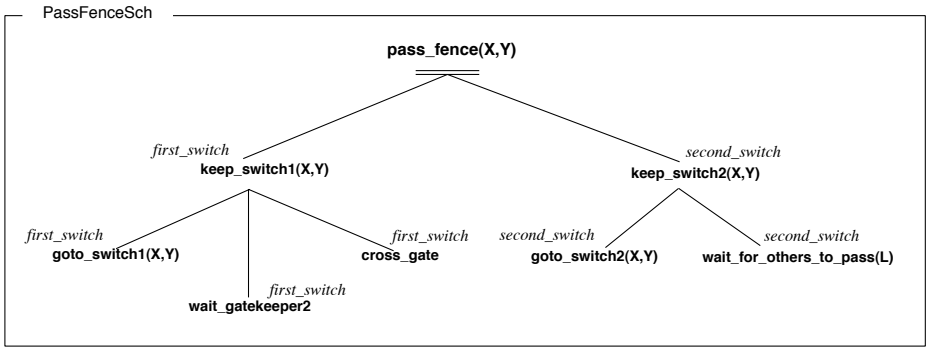

Fig. 2. The Functional Specification for the Pass-Fence Scheme.

used when a group of agents need to pass through a gate with a closed fence. When an exploring or herding group perceives a fence in their chosen path, the agents playing these two special roles within the groups will know the goals they have to do to ensure the group passes safe through the gate. The agent playing the gatekeeper1 role is sent to the position where the first switch (the one on the side where the agents currently are) can be activated. This allows the agent playing the gatekeeper2 role to go through the gate and position itself where the second switch can be activated (i.e., on the other side of the fence). When all agents of the group have passed through the gate, the scheme is finished. Table 1 briefly presents the goals that agents are obliged to achieve when playing one of the new roles (remember that we are not presenting here the part of of our solution that was already described in [4]); the goals are part of the *first_switch* and *second_switch* missions as shown in Figure 2.

There is a further complication with the fences in this scenario which is when two groups of the team need to cross the same gate. To handle this, before creating an instance of the `pass_fence` scheme, the second gatekeeper will always check with all team members (through communication) whether another group already has an active instance of such scheme, and if so, instead of creating another instance, the second gatekeeper will contact its counterpart in the group that is already in the process of passing that gate. The currently acting gatekeeper will then wait for all agents in *both* groups to pass through the gate and only then terminate the scheme. When the scheme

Table 1. The New Organisational Goals of the Team.

Role	Goal	Goal Description
gatekeeper1	`goto_switch1(X,Y)`	position itself where the switch can be activated[1]
	`wait_gatekeeper2`	keep on activating the first switch until the other gatekeeper has reached its destination
	`pass_fence`	once the second gatekeeper is at its position, this agent can already go and join the rest of its group
gatekeeper2	`goto_switch2(X,Y)`	position itself where the switch at the other side of the fence can be activated
	`wait_for_other_to_pass`	this agent is the one that needs to wait until all team members, in any groups, who wanted to pass that fence at that time, have done so

terminates, the acting second gatekeeper joins the group again, who go on to resume whatever they were doing (either exploring or herding).

Although we have some global constraints over the agents' behaviour (based on the roles they are playing), they are *autonomous* to decide how to achieve the goals assigned to them. While *coordination* and *team work* are managed by the $\mathcal{M}$OISE$^+$ tools, the *autonomy* and *pro-activeness* are facilitated by the BDI architecture of our agents implemented in *Jason*. Regarding *communication* (required, for example, for the share_seen_cows goal), we use speech-act based communication available in *Jason*.

3 Software Architecture

We initially planned to use artifacts to encapsulate two functions in our solution: integration with the contest simulator and maintenance of a shared view of the scenario. These were the two artifacts we wanted to implement. The first artifact would replace the customised *Jason* agent architecture we used to interface our agents with the simulator. In that new version, each agent would have access to the InterfaceArtifact that would provide, as observable properties, the current perception provided by the simulator to the agent, and, as an operation, the capability to send the actions of the agent back to the contest simulator. This artifact would also responsible for encapsulating all network issues, like reconnection, login, failure handling, etc.

The second artifact would be the PathArtifact. The motivation for this artifact is to have a shared representation of the scenario instead of each agent having its own representation. In the previous contest edition, we used broadcast messages: for each seen cow/obstacle, a message is broadcast so that all other team members can update their representation. This is a quite expensive solution in terms of communication. With this new artifact, for each seen cow/obstacle, an operation is triggered in the PathArtifact to update the scenario state. This operation may be triggered either by the agents or directly by the InterfaceArtifact. Other useful operations such as '`find_path`' would be implemented in this artifact so that the agents do not need to internally keep a representation of the world. The implementation and deployment of the artifacts was to be done with the CArtAgO platform [8].

4 Agent Team Strategy

1. Navigation algorithms. *As in previous teams, we use the A* algorithm to find paths and avoid obstacles.*
2. Describe the team coordination strategy (if any). *The coordination is based on shared global goals and global plans as defined in $\mathcal{M}$OISE$^+$.*
3. Does your team strategy use some distributed optimisation technique w.r.t., e.g., minimising distances walked by the agents? *In general, no, but in future work negotiation techniques might be used to find out good global solutions. At the individual level, A* finds optimal paths.*
4. Describe and discuss the information exchanged (and shared) in the agent team. *The more information (specially obstacles and fences) about the scenario is available for A*, the better it performs. So when an agent perceives an obstacle or a fence, it communicates that information to all team members.*

5. Describe the communication strategy in the agent team. Can you estimate the communication complexity in your approach? *We have not yet formally defined the communication protocols.*
6. Did your system do some background processing? By background processing we understand some computation which happened while agents of the team were *idle*. *No.*
7. Possibly discuss additional technical details of your system such as failure/crash recovery and alike. *We associate an "angel" to each agent; the angel checks if the agent is blocked/crashed and then tries to solve the problem automatically.*

5 Conclusion

Due to lack of time, we have not been able to implement the planned integration with CArtAgO. This would have made some parts of the implementation of our team (e.g., the sharing of spatial information) more elegant. The added feature of "fences" in the latest scenario of the agent competition lead to significant extra complexity in the scenario. However, our final solution remains compact and elegant because the high-level code at the organisational and agent levels remain essentially the same with the addition of only two extra roles and five new goals that agents playing those roles are required to achieve. It remains future work to implement the use of artifacts and make a thorough evaluation of the overall approach combining three of the most prominent agent development techniques.

References

1. R. H. Bordini, J. F. Hübner, and D. M. Tralamazza. Using *Jason* to implement a team of gold miners. In *CLIMA VII*, volume 4371 of *LNCS*, pages 304–313. Springer, 2007.
2. R. H. Bordini, J. F. Hübner, and M. Wooldrige. *Programming Multi-Agent Systems in AgentSpeak using Jason*. John Wiley & Sons, 2007.
3. J. F. Hübner and R. H. Bordini. Developing a team of gold miners using *Jason*. In *ProMAS 07*, volume 4908 of *LNCS*, pages 241–245. Springer, 2008.
4. J. F. Hübner, R. H. Bordini, and G. Picard. Using *Jason* and MOISE+ to develop a team of cowboys. In *ProMAS 08*, volume 5442 of *LNAI*, pages 238–242. Springer, 2009.
5. J. F. Hübner, J. S. Sichman, and O. Boissier. Developing organised multi-agent systems using the MOISE+ model: Programming issues at the system and agent levels. *International Journal of Agent-Oriented Software Engineering*, 1(3/4):370–395, 2007.
6. A. S. Rao. AgentSpeak(L): BDI agents speak out in a logical computable language. In *MAAMAW'96*, number 1038 in LNAI, pages 42–55, Springer, 1996.
7. A. S. Rao and M. P. Georgeff. BDI agents: from theory to practice. In *ICMAS'95*, pages 312–319. AAAI Pess, 1995.
8. A. Ricci, M. Viroli, and A. Omicini. CArtAgO: A framework for prototyping artifact-based environments in MAS. In *E4MAS 2006*, volume 4389 of *LNAI*, pages 67–86. Springer, 2007.
9. M. Viroli, T. Holvoet, A. Ricci, K. Schelfthout, and F. Zambonelli. Infrastructures for the environment of multiagent systems. *Autonomous Agents and Multi-Agent Systems*, 14(1):49–60, July 2007.