An Agent Team Based on FLUX for the ProMAS Contest 2007

Stephan Schiffel, Michael Thielscher, Doan Thu Trang

Dresden University of Technology
Dresden, Germany
{stephan.schiffel,mit}@inf.tu-dresden.de
tieuyen@gmail.com

Abstract. FLUX is a constraint logic programming system based on a general calculus for reasoning about actions. FLUX supports the development of agents that base their decisions on their own knowledge state and update this state in accordance with a declarative specification of their primitive actions and sensing capabilities. This is the second time we participate in the Multi-Agent Programming Contest with a team of FLUX agents, and in this paper we describe an improved system architecture for competing in the Gold Mining Domain.

1 Introduction

Intelligent agents have the ability to generate actions based on their own knowledge about the environment that they inhabit. Since last year, the Multi-Agent Programming Contest provides the research community with an opportunity to apply and compare different approaches and methodologies for the design of intelligent agents. This is the second time we participate in the contest with a team of FLUX agents, and in this paper we describe an improved system architecture for competing in the Gold Mining Domain.

FLUX [1] is a constraint logic programming system based on a general calculus for reasoning about actions. It supports the development of agents that base their decisions on their own knowledge state and update this state in accordance with a declarative specification of their primitive actions and sensing capabilities. A FLUX agent is a logic program consisting of three parts. A general *kernel* provides the basic reasoning facilities by means of an encoding of the foundational axioms of the action formalism known as fluent calculus [2]. The domain-specific *background theory* is used to maintain the internal knowledge state of an agent. It consists of a declarative specification of the actions and sensing capabilities of an individual agent. Finally, the *strategy* part of a FLUX program guides the behavior of the agent. The quality of a team of agents is crucially dependent on the quality of the strategy of each individual agent and how these work together.

This paper is organized as follows. Following this introductory section, we give an overview of the System Design, where we show how the three parts of

each FLUX agent are constructed, including the strategy of the whole team as well as of each individual agent. Thereafter, we give details about our software architecture, describing the tools and environment that are being used for the FLUX agent team with which we participate in the Multi-Agent Programming Contest 2007.

2 System Analysis and Design

As described above, an agent developed using the FLUX framework is a logic program consisting of three modules: the fundamental reasoning facilities based on the fluent calculus, the specification of the effects of actions, and the strategy. Since the first part is application-independent and is therefore provided by the general FLUX system, developing a FLUX agent amounts to programming the latter two modules. In what follows, we give an overview of these modules of the FLUX agents for the Gold Mining Domain.

FLUX Agent Team Our FLUX agent team consists of six agents and a leader. The role of the leader is to help the other agents in sharing information about the environment and to coordinate the other agents. To reduce communication complexity, the agents do not communicate directly with each other. Instead, the leader collects and distributes all new information among the agents.

Each agent has intentions that change over time based on sensor information and executed actions. The next action of an agent depends on the current intentions of that agent and the current state of the world.

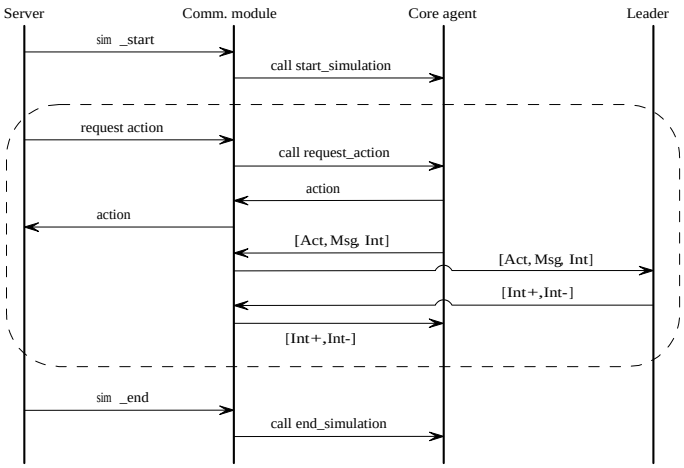

Fig. 1. Messages within Flux Agents

In order for the agents to cooperate, after an agent decides on its next action it sends new information it got and its current intention to the leader. In return the leader sends information gathered by the other agents to the agent. Additionally the leader might request the agent to change its intentions for coordinating the agents of the team. Figure 1 shows how the agents of the FLUX Team exchange messages with each other.

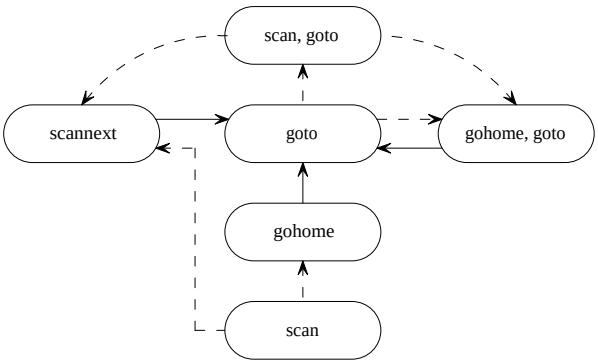

Fig. 2. Intentions of agents

Figure 2 shows how intentions of agents are changed. The solid arrows indicate the modifications of intentions that are decided by the leader, and the broken arrows mean that the transitions are done by the agents themselves. The main intentions of an agent are to scan an area of the grid, to go to some location either for scanning the area around it or for collecting a gold item, or to go home, i.e., go to the depot. The intentions can change when gold is picked up or dropped or an area was scanned completely. The intentions are sometimes changed upon request of the leader in order to assign areas to the agents for exploration or to resolve conflicts between the agents' intentions.

Knowledge Update Depending on the type, each member of the FLUX agent team behaves differently when updating its own knowledge about the environment. A knowledge update for the leader is triggered whenever an agent sends new information. The information that the leader receives contains all new information that the agent learned as well as the agent's intentions and action. Based on this, the leader updates its current knowledge of the grid as well as of the states of the agents. A knowledge update for an agent, on the other hand, is done whenever it executed an action. The knowledge state of an agent is updated using a FLUX implementation of a so-called Knowledge Update Axiom in the fluent calculus. To deal with the nondeterministic nature of actions due to random action failures it is sufficient to check whether the position of the agent and the number of nuggets it carries differs from the expectation. The Knowl-

edge Update Axiom incorporates all sensor information about the contents of the cells surrounding the agent into its state.

3 Software Architecture

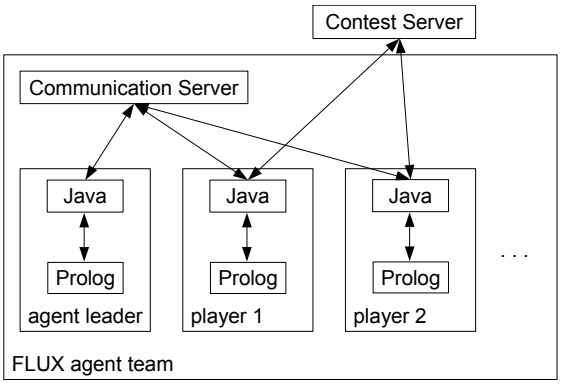

Fig. 3. Software Architecture

Each agent of the team of agents consists of two processes communicating via streams. One process runs a Java program responsible for communicating with the contest server and the other agents. The other process executes the actual strategy of the agent. The latter is implemented in (ECLiPSe)-Prolog using the FLUX framework. The agents of a team communicate with asynchronous messages using a simple self-implemented communication framework based on sockets. This architecture has a couple of advantages. It allows the individual agents to run on different computers across a network. All the other agents remain functional if one the agents crashes. Only failure of the leader agent will result in a less efficient strategy because of the coordination of the agents is missing. The system can be used easily for different numbers of agents.

4 Agent Team Strategy

The goal of the team is to collect as much gold as possible in a match. However, given the complexity and nondeterministic nature of the domain, it is difficult to come up with a plan for each agent of the team which maximizes the overall score of the team taking the unpredictable and widely hidden activities of the opponent team into account. Therefore, the agents mostly act greedily. Competing interests of exploration and predictable and fast traveling are only rudimentarily incorporated into the path planning algorithm. Conflicts between the agents of our team are resolved in two ways. First, the leader coordinates the agents by

assigning areas to the agents for exploration. Second, small scale conflicts such as when several agents try to get into the same cell, are resolved using fixed priorities of the agents without the direct help of the leader. In order for the agents to be able to cooperate, it is necessary that the individual goals and intentions of the agents are communicated between each other. To keep the communication complexity low, there is no peer-to-peer communication between the agents. Instead all information is collected and distributed by the leader agent. Apart from the communication with the simulation server each agent (except for the leader) sends and retrieves just one message per step. The obvious weak point in this setup is the leader agent. The complexity of the leader agent and computation power of the computer also limit the maximal number of agents that can be added to the system.

5 Discussion

Our successful participation in the ProMAS Contest 2007 has shown that FLUX - originally designed as a single agent framework - can be used as a basis for true multi-agent systems. The only weak spot we discovered was that FLUX lacks an integrated mechanism for communication between the agents.

The contest provides a useful testbed for multi-agent systems. However, it doesn't cover all aspects of uncertainty which agents normally have to face in an environment. In particular, the nondeterministic nature of the actions is easily resolved using the sensor information in the next step of the simulation. As a consequence, several important features of agent programming systems, like the support for nondeterministic actions, were not covered by the ProMAS Contest.

6 Conclusion

We have given an overview of an approach to the design of intelligent agents that participate in the ProMAS Contest 2007. In comparison with our contribution to the previous competition [3], the FLUX team has been significantly improved in the way the agents communicate with each other in order not to just follow their individual strategy but to also build a joint strategy for the entire team. The behavior of each agent is written in Prolog while the communication module has been implemented in Java. Thanks to the interface between Java and Prolog supported by ECLiPSe Prolog, single agents developed using the FLUX methodology can be easily joined in a team for multi-agent settings.

References

1. Thielscher, M.: FLUX: A logic programming method for reasoning agents. Theory and Practice of Logic Programming **5** (2005) 533–565
2. Thielscher, M.: From situation calculus to fluent calculus: State update axioms as a solution to the inferential frame problem. Artificial Intelligence **111** (1999) 277–299
3. Schiffel, S., Thielscher, M.: Multi-agent FLUX for the gold mining domain (system description). Volume 4371 of LNCS., Hakodate, Japan, Springer (2006)