N95-19766

Reflexive Reasoning for Distributed Real-Time Systems

David Goldstein
Computer Science Department
North Carolina Agricultural & TechnicalState University
Greeensboro, NC 27411
USA
Voice: (910) 334-7245
Fax: (910) 334-7244
goldstn@garfield.ncat.edu

Abstract

This paper discusses the implementation and use of reflexive reasoning in real-time, distributed knowledge-based applications. Recently there has been a great deal of interest in agent-oriented systems. Implementing such systems implies a mechanism for sharing knowledge, goals and other state information among the agents. Our techniques facilitate an agent examining both state information about other agents and the parameters of the knowledge-based system shell implementing its reasoning algorithms. The shell implementing the reasoning is the Distributed Artificial Intelligence Toolkit, which is a derivative of CLIPS.

Introduction

There has been a great deal of recent interest in multi-agent systems largely due to the increasing cost-effectiveness of utilizing distributed systems; in just the national CLIPS conference six papers appear which seem to discuss developing multi-agent systems. Further, although we strenuously try to avoid incorporating domain-specific knowledge in systems, real-time applications have an obvious need to understand the relationships between their processing requirements, available system resources, deadlines which must be met, and their environment. Hence, our programming methodology has been that individual agents need not necessarily be cognizant of any system information, but rather can communicate their own informational requirements, can sense their state in the system, and modify the internal processing parameters of the system (e.g., for load balancing) as the application demands. We allow the agents to sense and affect their processing environment so that they can intelligently reason about and affect the execution of applications.

Implementation of Reflexive Reasoning

We have previously documented the characteristics of our tool, the Distributed Artificial Intelligence Toolkit [1][2]. The tool provides extensions to CLIPS for fault-tolerant, distributed, real-time reasoning. Many of these characteristics are controllable by individual reasoning agents so that when insufficient processing time is available, for example, processor fault-tolerance may need to sacrificed. The control of such characteristics is provided numerous predicates. Correspondingly, the agents can sense the current settings of the environment through a state description of the inference engine contained in the fact base (and which can be pattern-matched).

Calls to such predicates, as well as numerous 'C' functions implemented to provide additional functionality, were used to implement the Agent Manipulation Language (AML). AML (Table 1) provides the functionality to manipulate, assign tasks to, and teach agents. The functions used to implement AML include those providing fault-tolerance, for transmitting facts, template and objects, and those mimicking

user-interface functionality; the data assistants of our architecture(Figure 1) actually interface to the user, evoking functionality from the reasoning agents [1].

build_agent(*processor*)	Creates an agent
teach_agent(*agent name, set of rules as text*)	Sends a ruleset to agent for reasoning
agent_unlearn(*agent name, set of rules as text*)	Causes agent to remove ruleset from its processing
agent_learn(*filename*)	Have agent read a rulebase from *filename*
die(*agent name*)	Kill the agent
wait(*agent name*)	Have the agent suspend reasoning
continue(*agent name*)	Have the agent continue reasoning

Table 1: Agent Manipulation Language (AML)

The last big issue is how the environment is sensed and affected at a low level. These processes are accomplished by intercepting and interpreting the individual elements of facts and templates before they are actually asserted. This kind of functionality allows agents to be minimally required to affect other agents; agents can know and affect each other (on the same or other machines) as much or as little as they desire. Formally proprietary information, this is now being divulged because implementing the code for the parsing of such information has been deemed too difficult for students, even graduate students, to maintain.

Reflexive Reasoning in Distributed Real-time Systems

Consider a real-time robotics application. The application consists of path planners, task planners, sensor and effector managers, motion control modules, etc. For the planning modules we typically would want to employ fault-tolerance, but for many of the other modules we would want very fast update rates. Hence, we might initially turn off fault-tolerance for, turn on interruptable reasoning for, and reduce the granularity of reasoning for the machines controllers, sensor managers, motion control modules, etc.

Certain planners and motion control modules would probably would probably require more resources than others. Modules noting short times between actual completion of tasks and the tasks' actual deadlines can evoke operating system functionality, via the data assistants, to determine processors with "excess" computing power. The over burdened agents could then create new agents, advise them to learn a set of rules, and off-load some of their work to newly instantiated agents. Hence, as the system executes, overworked agents could instigate load balancing.

Agents can also use reflexive reasoning in less subtle manners; any agent can request to see the fact and object base of any other agent. Agents can also request to know the goals of interest of other agents (or rather, which agents are interested in what goals). Hence, agents can also reason about the reason about the reasoning being performed by other agents.

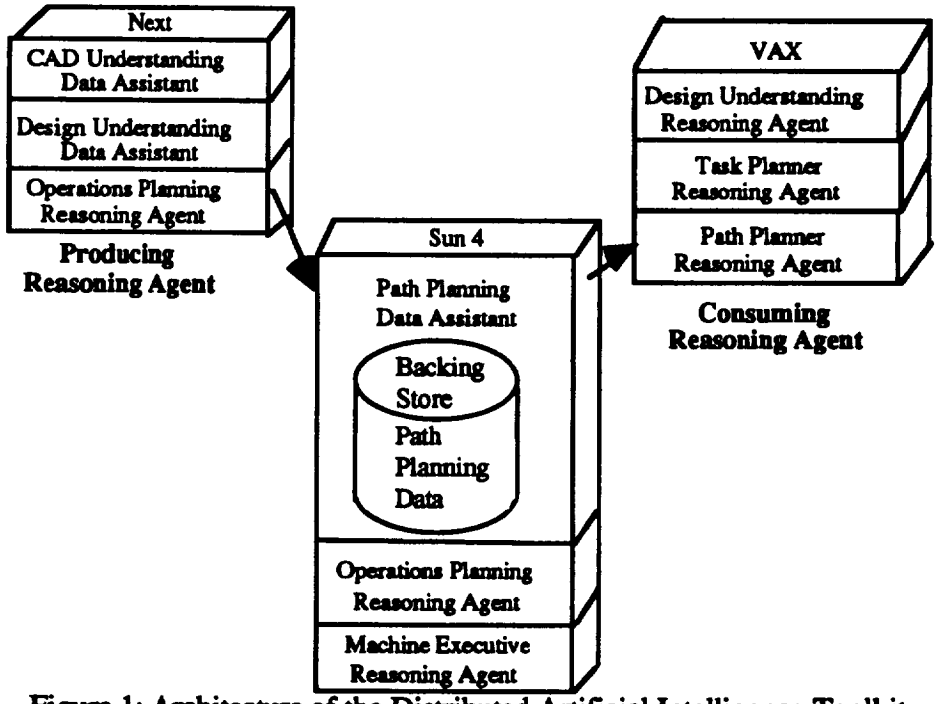

Figure 1: Architecture of the Distributed Artificial Intelligence Toolkit

Conclusion

We have briefly discussed implementing and using reflexive reasoning in distributed, real-time applications. Reflexive reasoning provides reasoning agents in distributed systems to analyze and modify the reasoning processes of other agents. We have found reflexive reasoning an effective tool for facilitating control of real-time, multi-agent systems. Our implementation of reflexive reasoning has been hierarchical, building up an agent manipulation language from predicates describing and affecting the reasoning process. These predicates have been, in turn, implemented from low-level functions written in 'C'.

References

1. Goldstein, David, "The Distributed Artificial Intelligence Toolkit", AI EXPERT, Miller-Freeman Publishing, January, 1994, pp 30-34.

2. Goldstein, David, "Extensions to the Parallel Real-time Artificial Intelligence System (PRAIS) for Fault-tolerant Heterogeneous Cycle-stealing Reasoning", in Proceedings of the 2nd Annual CLIPS ('C' Language Integrated Production System) Conference, NASA Johnson, Houston, September 1991.