19951-3347

N95-19763

34079
P-11

DAI-CLIPS: Distributed, Asynchronous, Interacting CLIPS

Denis Gagné & Alain Garant

Groupe de Recherche en Intelligence Artificielle Distribuée

Collège militaire royal de Saint-Jean

Richelain, (Québec)

Canada, J0J 1R0

dgagne@cmr.ca

Abstract

DAI-CLIPS is a distributed computational environment within which each CLIPS is an active independent computational entity with the ability to communicate freely with other CLIPS. Furthermore, new CLIPS can be created, others can be deleted or modify their expertise, all dynamically in an asynchronous and independent fashion during execution. The participating CLIPS are distributed over a network of heterogeneous processors taking full advantage of the available processing power. We present the general framework encompassing DAI-CLIPS and discuss some of its advantages and potential applications.

1 Introduction

Scenarios to be solved by Artificial Intelligence (AI) applications rapidly increase in complexity. If we are to even allude to the enormously difficult endeavor that represents Artificial Intelligence, very flexible, robust and powerful computational systems are going to be needed. These programs, and the processing architecture supporting them, will have to be able to cope with a very wide range of dynamic external demands that are simply unpredictable.

Conditions vary greatly across tasks instances and constantly dictate different accomplishment strategies and usage of disparate sources of expertise. To be effective in such situations, fusion must take place at many levels (information, expertise, processes,...). Maximum degree of openness and flexibility is required of AI systems. Both hardware architecture and software solutions must allow for scalable performance and scalable functionalities. This, in return, will allow the matching of AI systems to the needs of the situation as well as gaining access to the latest technological advances.

We believe that complex problems can best be solved via a pendemonium of smaller agents. Each agent specializes in a different narrow aspect of cognition or field of knowledge [Min85, Ten88]. Emphasis is thus placed on data and application parallelism.

PRECEDING PAGE BLANK NOT FILMED

344

The computational environment presented herein integrates the simplicity, elegance and expressiveness of the ACTOR computational model [Hew73, Agh86] with the exceptional processing power of heterogeneous distributed and parallel architectures [Des93]. Expressiveness is increased by providing for disjunct or even disparate sources of expertise to cohabitate rather than trying to integrate them. The interfacing or fusion required is achieved via message passing enabling asynchronous exchange of knowledge (facts, rules) among agents. Agents in the present context are effectively extended CLIPS. CLIPS is an expert system tool that was developed by NASA at the Software Technology Branch of the Johnson Space Center [Gia94].

DAI-CLIPS is a distributed computational environment within which each extended CLIPS is an active independent computational entity with the ability to communicate freely with other CLIPS. Furthermore, new CLIPS can be created, others can be deleted or modify their expertise, all dynamically in a totally asynchronous and independent fashion during execution.

The remainder of this text is structured as follows. Section 2 highlights the key characteristic of the Actor computational model that influenced DAI-CLIPS and briefly describes the system layer that constitutes the foundation of DAI-CLIPS. In section 3, we provide a description of the conceptual framework offered by DAI-CLIPS by outlining its global functionalities. In section 4, we provide some details about the architecture and available functions. Section 5 brushes a quick picture of a few potential areas of research and development that could benefit from such computational environment. Finally, sections 6 and 7 provides a discussion and our conclusions on the subject.

2 The Actor Model & CLAP

In this section we highlight some of the main characteristics of the Actor model that influenced and characterizes DAI-CLIPS. We then present a brief description of CLAP [Des93, Gag93, Gag94], a system layer based on the Actor Model, that constitute the foundation of DAI-CLIPS.

2.1 The Actor Model

A detailed description of the Actor Model can be found in [Agh86, Hew77]. We will only discuss here a few of the more salient characteristics of the model:

Distributed. The Actor model consists of numerous independent computational entities (actors). The actors process information concurrently, which permits the overall system to handle the simultaneous arrival of information from different outside sources.

Asynchronous. New information may arrive at any time, requiring actors to operate asynchronously. Also, actors can be physically separated where distance prohibits them from acting synchronously. Each actor has a mailbox (buffer) where messages can be stored while waiting to be processed.

Interactive. The Actor model is characterized by a continuous information exchange through message passing, subject to unanticipated communication from the outside.

Thus, an actor is basically an active independent computational entity communicating freely with other actors.

There are at least two different ways to look at the Actor model. Viewed as a system, it is comprised of two parts: the actors and a system layer (operating/management system). From such a point of view the actors are the only available computational entities. The system layer is responsible to manage, create and destroy actors as required or requested. The system layer is also responsible for ensuring message passing among actors.

Viewed as a computational entity, an actor also comprises two parts: a script, that defines the behaviors of the actor upon receipt of a message; and a finite set of acquaintances which are the other actors known to the actor.

2.2 CLAP

The above viewpoint duality is preserved in CLAP[1]. CLAP is an implementation of an extension of the actor model that can execute on a distributed heterogeneous network of processors. The present version of CLAP can execute over a network of SUN SPARC workstations and Alex Informatique AVX parallel machines which are transputer based distributed memory machines [Des93]. A port to HP and SGI workstations is in progress.

CLAP is an object-oriented programming environment that implements the following concepts of the Actor model: the notion of actor, behaviors, mailbox, and parallelism at the actor level. Further, CLAP offers the extension to the model of intra-actor parallelism.

Generally, CLAP applications will consist of many programs distributed over available processors executing as a task under the control of the CLAP run time environment. In CLAP, each actor is a member of a given task. It is up to the programmer to determine how many actors there will be for any given task (although, a large number of actors in a single task could mean the loss of potential parallelism in the application.) A scheduler controls the execution of processes inside the tasks. Each task possesses a message server that handles message reception for the actors in the task. Inter-processor message transmissions are handled via RPC servers. XDR filters and type information are utilized for the encoding and decoding of these messages. The CLAP environment is implemented in C++.

3 The Conceptual Framework

DAI-CLIPS is a distributed computational environment within which each CLIPS has been extended to become an active independent computational entity with the ability to communicate freely with other extended CLIPS. Furthermore, new extended CLIPS can be created, others can be deleted or modify their expertise, all dynamically in a totally asynchronous and independent fashion during execution.

The *desirata* behind DAI-CLIPS is to produce a flexible development tool that captures the essence of the "aggregate of micro agents" thesis supported by many in the study of Computational Intelligence and Cybernetic [Hew73, Min85, Ten88]. The underlying thesis being to have "micro agents", in our case complete CLIPS, specialized in different very

[1] C++ Library for Actor Programming.

narrow aspects of cognition or fields of knowledge[2]. As they go about their tasks, these micro-agents confer with each other and form coalitions producing collated, revised enhanced views of the raw data they take in. These coalitions and their mechanism implement various cognitive processes leading to the successful resolution of the problem.

3.1 D.A.I.

The three highlighted characteristics of the Actor model in the previous section, namely distributed, asynchronous and interactive, are at the basis of the conceptual framework for DAI-CLIPS. The augmented CLIPS participating in the environment are completely encapsulated and independent allowing their distribution at both the software and hardware level. Meaning that not only can the CLIPS execute in parallel but they can also be physically distributed over the network of available processors. Our present version of DAI-CLIPS can have participating CLIPS distributed over a network of SPARC workstations and/or the nodes of a transputer based distributed memory parallel machine. The interaction among the CLIPS is asynchronous (synchronicity can be imposed when required). The interchange of knowledge between these extended CLIPS can involve exchanging facts, rules, and any other CLIPS data object or functionality.

3.2 Cooperation

The DAI-CLIPS environment is conducive of cooperation among a set of independent CLIPS. We regard as cooperation any exchange of knowledge among CLIPS whether productive or not.

There is *a priori* no pre-defined notion of an organizational structure among the CLIPS in DAI-CLIPS. Any desired type of organization (*e.g.* hierarchy, free market, assembly line, task force, *etc.*) can be achieved by providing each CLIPS the appropriate knowledge of the structure and the mechanism or protocol to achieve it.

The broad definition of cooperation and the inexistence of pre-defined organizational structures in DAI-CLIPS were conscious initial choices. We wanted to maintain the highest flexibility possible for the environment in this first incarnation. We are contemplating the introduction of mechanisms to DAI-CLIPS to ease the elaboration of specific types of organizations based on the premise of groups or aggregates. The aim of these efforts is to capture the recursive notion of *agency*.

3.3 Dynamic Creation

A powerful capability of DAI-CLIPS is the possibility of dynamically generating or destroying participating CLIPS at run time. The generation of new CLIPS can involve introducing a new expertise or simply cloning an existing participant. When generating a new CLIPS, one can specify which expertise the CLIPS is to possess by indicating the appropriate knowledge base(s) to be loaded in the CLIPS at creation. This functionality has enormous potentials that we have yet to completely explore.

[2] Expert Systems excel under these domain constraints.

4 The Architecture

The general framework encompassing DAI-CLIPS can be viewed as four layers: the hardware layer, the system layer, the agent layer, and the application layer (see fig1). The hardware layer consists of a set of nodes (available processors) on the network (SPARCs and transputers). The system layer (CLAP) is responsible to manage, create and destroy the processes required or requested from the above agent layer as well as managing inter-agent communications. The agent layer provides a series of functionalities to implement various cognitive processes via coalitions and organization mechanisms for a series of specialized micro-agents (DAI-CLIPS). Finally, the application layer provides interfacing services and captures and implements the user's conceptualization of the targeted domain. Such conceptualization usually involves the universe of discourse or set of objects presumed in the domain, a set of functions on the universe of discourse, and a set of relations on the universe of discourse [Gen87].

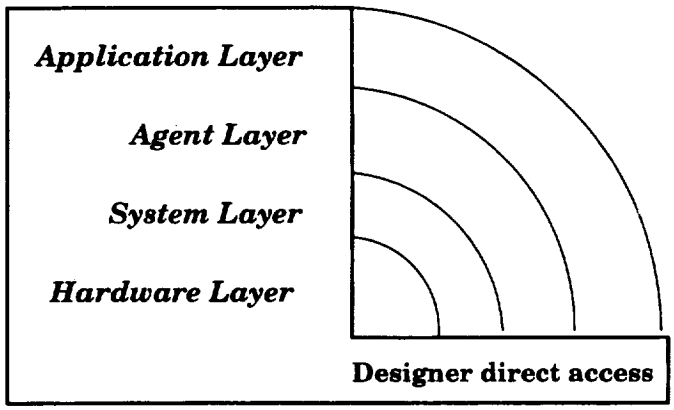

Figure 1: The conceptual framework.

Within this general framework, an application designer can directly access and manipulate any of the four layers of the environment. This provides the designer with the flexibility of manipulating objects at the level of abstraction he is more at ease with. For example, a more advanced application designer could seek efficiency in his particular application by manipulating objects all the way down to the system layer level, where someone else may be quite content of the functionalities provided at the top layer.

4.1 Design

In the present version of the environment, each augmented CLIPS is associated with a CLAP actor. These actors are loaded on different available processing nodes according to the load of the nodes. To each augmented CLIPS is connected an interface which provides access to the individual standard command loops of the CLIPS. An initial knowledge base is loaded in each CLIPS (see fig2). Note that it is possible for two CLIPS to be uploaded with the

same initial knowledge base or for a CLIPS to upload a supplementary knowledge base at run time.

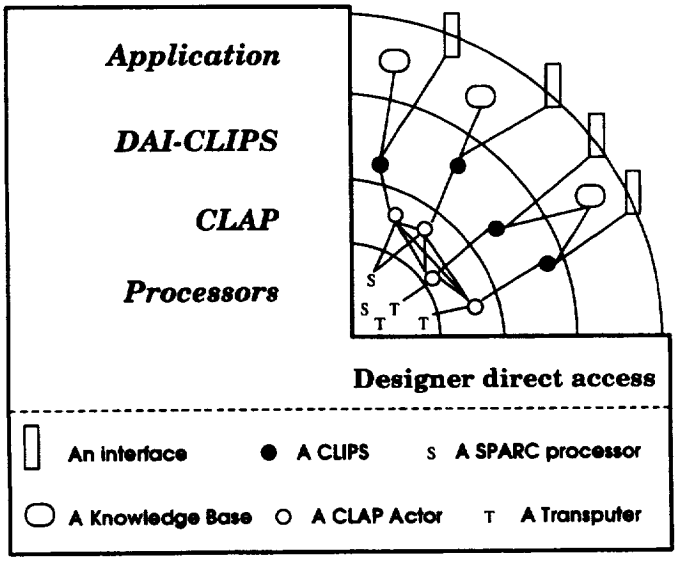

Figure 2: Surrounding environment of DAI-CLIPS.

4.2 Implementation

In this section we enumerate some of the functions specific to DAI-CLIPS and describe their respective use and functionality. The list is not exhaustive[3], rather the intent here is to present some of the main functions which can be used directly by the user.

create-agent

Purpose: Creates a named agent without expertise.
Synopsis: `(create-agent <string-or-symbol-agent-name>)`
Behavior: The agent `<string-or-symbol-agent-name>` is created.

destroy-agent

Purpose: Destroys a named agent.
Synopsis: `(destroy-agent <string-or-symbol-agent-name>)`
Behavior: The agent `<string-or-symbol-agent-name>` is destroyed.

send-fact-agent

Purpose: Asserts a run-time fact in a named agent.
Synopsis: `(send-fact-agent <string-or-symbol-agent-name> <string>)`
Behavior: The fact `<string>` is asserted in the agent `<string-or-symbol-agent-name>` who then executes.

[3]Due to restricted space in this article.

send-deffact-agent

Purpose: Defines a persistent fact in a named agent.

Synopsis: `(send-deffact-agent <string-or-symbol-agent-name> <symbol-deffact-name>)`

Behavior: The fact `<symbol-deffact-name>` is permanently asserted in the agent `<string-or-symbol-agent-name>` who then executes.

send-defglobal-agent

Purpose: Defines a global variable in a named agent.

Synopsis: `(send-defglobal-agent <string-or-symbol-agent-name> <symbol-defglobal-name>)`

Behavior: The global variable `<symbol-defglobal-name>` is defined in the agent `<string-or-symbol-agent-name>` who then executes.

send-defrule-agent

Purpose: Defines a rule in a named agent.

Synopsis: `(send-defrule-agent <string-or-symbol-agent-name> <symbol-defrule-name>)`

Behavior: The rule `<symbol-defrule-name>` is added in the expertise of agent `<string-or-symbol-agent-name>` who then executes.

send-deftemplate-agent

Purpose: Defines a template in a named agent.

Synopsis: `(send-deftemplate-agent <string-or-symbol-agent-name> <symbol-deftemplate-name>)`

Behavior: The template `<symbol-deftemplate-name>` is added in the expertise of agent `<string-or-symbol-agent-name>` who then executes.

load-for-agent

Purpose: Send a message to a named agent ordering him to load a specific expertise from a named file.

Synopsis: `(load-for-agent <string-or-symbol-agent-name> <file-name>)`

Behavior: The agent `<string-or-symbol-agent-name>` possesses the expertise specified in `<file-name>`.

5 Potential Areas of Applications

DAI-CLIPS provides an environment with a varying number of autonomous knowledge based systems (expert-systems) that can exchange knowledge asynchronously. Such organizations of interconnected and independent computational systems are what Hewit calls **Open Systems**[4] [Hew85]. We thus refer to DAI-CLIPS as an *Open Knowledge Based Environment* or *Open KBE* for short. The potential areas of research and development that could benefit from such a computational environment are considerable.

[4]The term "open system" being an overloaded term, we specifically refer to Hewit's definition of open systems within the context of this article.

5.1 Distributed Artificial Intelligence

The first such area that comes to mind is that of Distributed Artificial Intelligence (DAI) [Bon88, Gas89, Huh87]. The facilities available in DAI-CLIPS to support interaction between "intelligent" agents make it a flexible tool for DAI applications and research. In comparison with some existing DAI test beds, DAI-CLIPS: does not impose a specific control architecture such as the blackboard in GBB [Cor86]; does not restrict agents to a specific set of operators as in TRUCKWORLD; and is not a domain specific simulator as in PHOENIX[Han93]. The most closely related work is SOCIAL CLIPS [Adl91]. The major difference with SOCIAL CLIPS is DAI-CLIPS' dynamic creation and destruction of participating CLIPS at run time.

There are *a priori* no predefined domain of application for DAI-CLIPS. A designer is free to specialize his agents in the domain of his/her choice. Further, the agents can be heterogeneous in their speciality (expertise) within a single application. The only imposed commonality in DAI-CLIPS is the use of the extended CLIPS shell. The added power provided by the dynamic creation/destruction of agents within DAI-CLIPS is the source of the potential area of application proposed in the next section.

5.2 Evolutionary Computing

The principle behind evolutionary computing is that of population control [Koz93]. That is ensuring that the population is not allowed to grow indefinitely by selectively curtailing it. This population control is carried out by creative and destructive processes guided by natural selection principles. The destructive process examines the current generation (population) and curtails it by destroying its weakest members (usually those with the lowest values from some predefined fitness measure). The creative process introduces a new generation created from the survivors of the destructive process. The expected result is that of a better fit or optimum population.

Given DAI-CLIPS capability of dynamically creating and destroying participating CLIPS at run-time, one can begin to explore the potential of coarse grain evolutionary computing. That is, applying evolutionary computing principles to a population of "agents" or expert systems in order to optain a population of expert systems that selectively better perform on a global task in accordance with some selected fitness measure. The creative and destructive processes could be carried out by two independent agents. One agent evaluating the agents of the population and destroying those that do not perform as expected (destructive process), another, either bringing together the expertise of two fit agents into a newly created expert or simply cloning a fit agent (creative process). We will refer to this approach as a *disembodied genetic mechanism.* Alternatively, the agents of a population could themselves possess "genetic knowledge" that would lead to self-evaluation. Based on the knowledge of its own fitness, an agent could then decide to terminate operations or to seek an appropriate agent for procreation (via mutation, crossover, *etc.*). This *embodied genetic mechanism* could take place based on some pre-determined evolution cycle. Note that both the embodied and disembodied genetic mechanisms can take place continuously and in totally asynchronous fashion.

6 Discussion

Open KBEs such as the one presented herein have considerable advantages:

- they allow independent systems to cooperate in solving problems;
- they allow disparate participant systems to share expertise;
- they allow for the presence of incoherent information among participant systems, no need for global consistency;
- they provide for participant systems to work in parallel on common problems;
- participant systems can be distributed physically to make ultimate use of the available processing power;
- asynchronous communication ensures very remote chances of deadlock;
- fault tolerance is easy to implement via system redundancy;
- participant systems can be developed and implemented independently and modularly;
- participant system are reusable in other applications;

and many others.

By choice, DAI-CLIPS has one limitation with respect to Open KBE: the participant systems are limited to CLIPS based systems. In fact, the general framework encompassing DAI-CLIPS can easily be extended to allow heterogeneous applications (*e.g.* other ES shells, Data Bases, Procedural applications) to participate through the use of a common formal language for the interchange of knowledge among disparate computer programs such as *Knowledge Interface Format* (KIF) and the use of a common message format and message-handling protocol such as the *Knowledge Query and Manipulation Language* (KQML). The use and adherance to these two upcoming standards from the DARPA Knowledge Sharing Initiative can assure that any incompatibility in the paticipant systems' underlying models for representing data, knowledge and commands can be ironed out to attain the desired higher level of openness.

DAI-CLIPS and its encompassing environment will be the source of more research and enhancements. We are presently putting the final touch to a second version of DAI-CLIPS that implements the notion of multiple behaviors from the Actor model. That is, the capability of an agent to change its behavior in order to process the next message. Effectively, a CLIPS shell will possess different expertise and will "context switch" to make use of the appropriate knowledge to process the received message.

7 Conclusion

We introduced DAI-CLIPS, a distributed computational environment within which each CLIPS is an active independent computational entity communicating freely with other CLIPS. Open KBEs such as this one have many advantages, in particular, they allow for

scalable performance and scalable functionalities at both the hardware and the software level. The potential applications of such environments are considerable. The unique power of dynamic creation and destruction of DAI-CLIPS could lead to new forms of "intelligent" evolutionary systems.

8 Acknowledgments

The authors would like to thank Jocelyn Desbiens and the members of the Centre de Recherche en Informatique Distribuée (CRID) for their constant support in the implementation of DAI-CLIPS. We also want to thank Alain Dubreuil and André Trudel for comments on an earlier version of this article.

References

[Adl91] Adler, R., "Integrating CLIPS Applications into Heterogeneous Distributed Systems.", In *Proceedings of the Second CLIPS Conference*, NASA Conference Publication 10085, 1991.

[Agh86] Agha, G.A., *Actors: A Model of Concurrent Computation in Distributed Systems*, Cambridge, Massachusetts: MIT Press, 1986.

[Bon88] Bond, A. & Gasser, L. (Eds), *Readings in Distributed Artificial Intelligence.*, Los Altos, California: Morgan Kaufmann, 1988.

[Cor86] Corkill, D., Gallagher, K. & Murray, K.,"GBB: A Generic Blackboard Development System." In *Proceedings of AAAI-86*, 1986.

[Des93] Desbiens, J., Toulouse, M. & Gagné, D.,"CLAP: Une implantation du modèle Acteur sur réseau hétérogène"., In *Proceedings of the 1993 DND Workshop on Knowledge Based Systems/Robotics.* Ottawa, Ontario, 1993. (In French)

[Gag93] Gagné, D., Nault, G., Garant, A. & Desbiens, J.,"Aurora: A Multi-Agent Prototype Modelling Crew Interpersonal Communication Network"., In *Proceedings of the 1993 DND Workshop on Knowledge Based Systems/Robotics.* Ottawa, Ontario, 1993.

[Gag94] Gagné, D., Desbiens, J. & Nault, G.,"A Multi-Agent System Simulating Crew Interaction in a Military Aircraft"., In *Proceedings of the Second World Congress on Expert Systems.* Estoril, Portugal, 1994.

[Gas89] Gasser, L. & Huhns, M.N. (Eds), *Distributed Artificial Intelligence: Volume II*, Los Altos, California: Morgan Kaufmann, 1989.

[Gen87] Genesereth, M. & Nilsson, N.,*Logical Foundations of Artificial Intelligence.*, Morgan Kaufmann, 1987.

[Gia94] Giarratano, J. & Riley, G.,*Expert Systems: Principles and Programming.*, PWS Publishing Company, 1994.

[Han93] Hanks,S., Pollack, M. & Cohen, P.,"Benchmarks, Test Beds, Controlled Experimentation and the Design of Agent Architectures.", In *AI Magazine*, Vol. 14, No. 4, Winter 1993.

[Hew73] Hewit, C., Bishop, P. & Steiger, R., "A Universal Modular Actor Formalism for Artificial Intelligence.", In *Proceedings of the 3rd Joint Conference on Artificial Intelligence (IJCAI73).*, Stanford, California, 1973.

[Hew77] Hewit, C., "Viewing Control Structures as Pattern of Passing Messages.", In *Journal of Artificial Intelligence.*, Vol 8, No. 3, 1977.

[Hew85] Hewit, C., "The Challenge of Open Systems.", *BYTE Magazine*, Vol. 10, No. 4, April 1985.

[Huh87] Huhns, M.N. (Ed), *Distributed Artificial Intelligence*, Los Altos, California: Morgan Kaufmann, 1987.

[Koz93] Koza, J.,*Genetic Programming.*, The MIT Press, 1993.

[Min85] Minsky, M., *The Society of the Mind.*, Simon and Schuster, New York, 1985.

[Ten88] Tenney, R. & Sandell, JR., "Strategies for Distributed Decisionmaking." In Bond & Gasser (Eds) *Readings in Distributed Artificial Intelligence.*, Los Altos, California: Morgan Kaufmann, 1989.