

comp



Canadian Artificial Intelligence Intelligence Artificielle au Canada

Autumn 1996

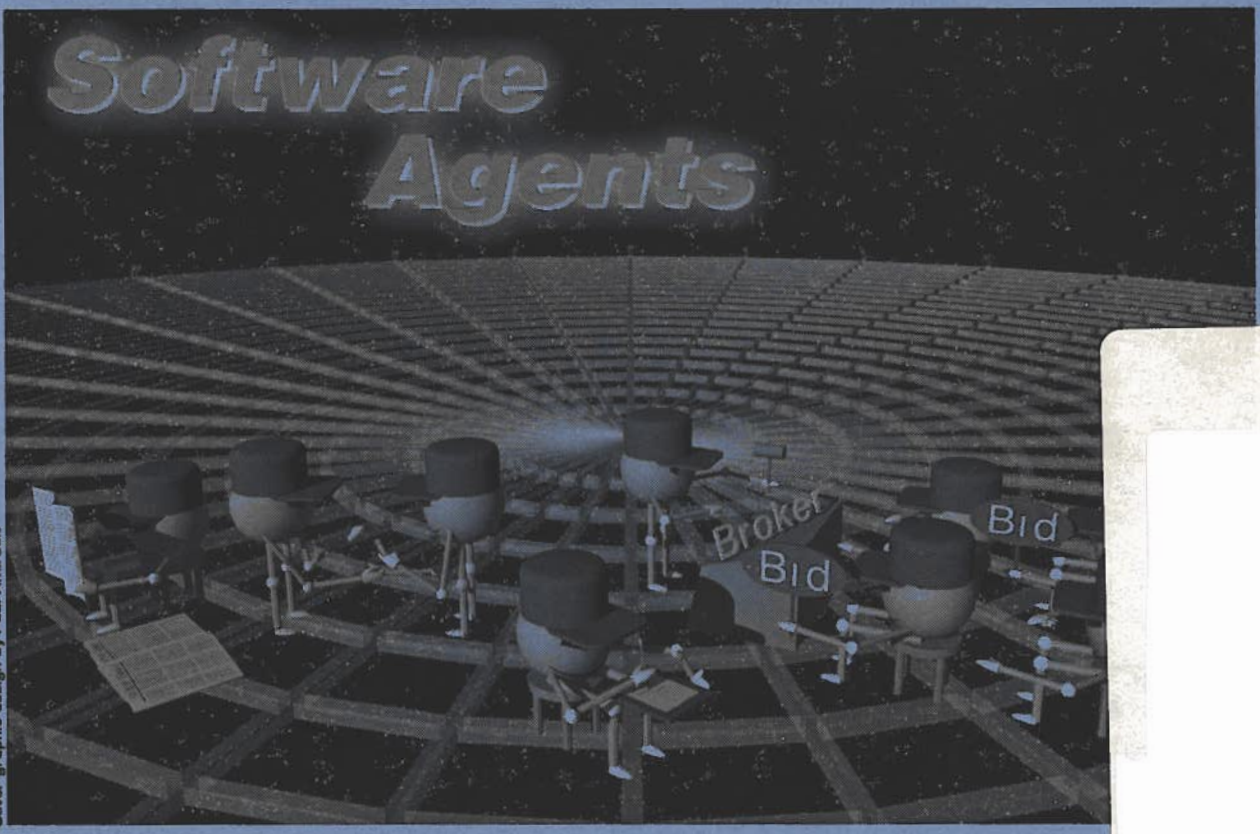
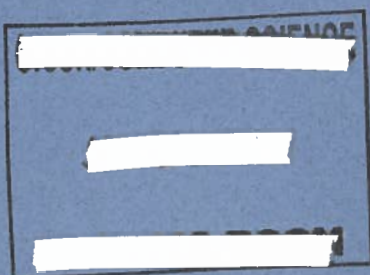
No. 40

automne 1996

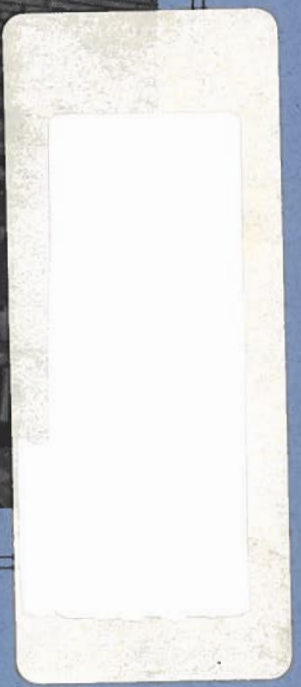
An official publication of CSCSI, the Canadian Society for Computational Studies of Intelligence
Une publication officielle de la SCEIO, la Société canadienne pour l'étude de l'intelligence par ordinateur

**Special Issue
Software Agents**

**Edition speciale
Agents Logiciels**



Cover graphic design by Paul Amireault





Canadian Artificial Intelligence

Intelligence Artificielle au Canada

Autumn 1996

No. 40

automne 1996

Canada's National AI magazine.

Editors - Rédacteurs en Chef

Peter Turney and Sue Abu-Hakima

Editor Emeritus - Rédacteur Emeritus

Roy Masrani

Managing Editor - Chef de Pupitre

Arlene Merling

Production Team - Equipe de production

Arlene Merling, Greg Klymchuk, Carol Tubman

Academia - Académiques

Vacant

Canadian AI Success Stories

Récits de Succès d'IA au Canada

Peter Turney

PRECARN Update - Nouvelles de PRECARN

Harry Rogers

Book Reviews - Critiques de livres

Graeme Hirst

Translation - Traduction

Alain Désilets, Benoit Farley, Norman Vinson

Advertising - Publicité

Arlene Merling

Canadian Artificial Intelligence is published three times a year by the Canadian Society for Computational Studies of Intelligence (CSCSI). *Intelligence Artificielle au Canada* est publiée trimestriellement par la Société canadienne pour l'étude de l'intelligence par ordinateur (SCEIO). Canadian Publications Mail Product Sales Agreement No. 507032.

ISSN 0823-9339

Copyright © 1996, Canadian Society for Computational Studies of Intelligence. All rights reserved; *Canadian Artificial Intelligence* may not be reproduced in any form without the written permission of the editors. Printed in Canada by Riley's Reproductions & Printing Ltd. *Canadian Artificial Intelligence* is published with the assistance of the Alberta Research Council. The opinions expressed herein are those of their respective authors and are not necessarily those of their employers, CSCSI, *Canadian Artificial Intelligence*, the editors, CIPS, the National Research Council or the Alberta Research Council.

Copyright © 1996, Société canadienne pour l'étude de l'intelligence par ordinateur. Tout droit réservé; *Intelligence artificielle au Canada* ne doit être reproduite par quelque moyen que ce soit sans le consentement écrit des éditeurs. Imprimée au Canada par Riley's Reproductions & Printing Ltd. *Intelligence artificielle au Canada* est publiée avec l'aide du Conseil de Recherche de l'Alberta. Les opinions exprimées dans ce magazine sont celles de leurs auteurs respectifs et non pas nécessairement celles de leurs employeurs, de la SCEIO, de *Intelligence artificielle au Canada*, des éditeurs, de l'Association canadienne en informatique, de Conseil National de Recherches du Canada, ou du Conseil de Recherche de l'Alberta.

CSCSI Executive

President - Président:

Stan Matwin, Computer Science Dept., University of Ottawa, Ottawa, ON K1N 6N5; stan@csi.uottawa.ca

Past-President - Président Précédent:

Janice Glasgow, Dept. of Computing & Information Science, Queen's U., Kingston, ON K7L 3N6; janice@qcis.queensu.ca

Vice-President - Vice-Président:

Allan Jepson, Department of Computer Science, University of Toronto, Toronto ON M5S 1A4; jepson@vis.utoronto.ca

Secretary - Secrétaire:

Fred Popowich, School of Computing Science, Simon Fraser University, Burnaby BC V5A 1S6; popowich@cs.sfu.ca

Treasurer - Trésorier:

Peter van Beek, Dept. of Computing Science, U of A, Edmonton, AB T6G 2H1; vanbeek@cs.ualberta.ca

Editors - Rédacteurs en Chef

Peter Turney, Institute for Information Technology, NRC, Ottawa, ON K1A 0R6; peter@ai.iit.nrc.ca
Sue Abu-Hakima, Institute for Information Technology, NRC, Ottawa, ON K1A 0R6; sue@ai.iit.nrc.ca

Contents

Contenu

Communications 2 Communications

Feature Articles

Nick Cercone — 1996 CSCSI/SCEIO Distinguished Service Award Winner
Gord McCalla

Agents: From Theoretical Foundations to Practical Applications
Jean-Francois Arcand and Sophie Pelletier

The Agent Building Shell: A Tool for Building Enterprise Multi-Agent Systems
Mihai Barbuceanu and Mark S. Fox

Multi-agent Systems at Laval University
B. Chaib-draa

Intelligent Agent Structures for the Internet
Leslie L. Daigle and Sima Newell

Highly Autonomous Sensor Models
Fernando Figueroa

LALO: A New Agent Oriented Programming Language and Environment
D. Gauvin, H. Marchal, C. Saldanha

The SIGMA Project: Market-Based Agents for Intelligent Information Access
Grigoris J. Karkoulas

Agent Research in the Cognitive Robotics Group
Y. Lesperance, H.J. Levesque, F. Lin, D. Marcu, R. Reiter, and R.B. Scherl

Distributed Agent Systems for Intelligent Manufacturing
Douglas H. Norrie and Brian R. Gaines

Personal Assistants for the Office Professional
Cliff Grossner and T. Radhakrishnan

How to Get (and Keep) a Research Grant
Ian H. Witten
(Revised and updated by Janice I. Glasgow)

Gros Titres

4 Nick Cercone – Récipiendaire du prix de la SCEIO pour service exceptionnel pour l'année 1996
Gord McCalla

6 Les agents: des fondements théoriques aux applications pratiques
Jean-Francois Arcand et Sophie Pelletier

9 Le "Agent Building Shell": un outil pour la construction de systèmes d'entreprise multi-agents
Mihai Barbuceanu et Mark S. Fox

12 Les systèmes multi-agents à l'Université Laval
B. Chaib-draa

16 Structures d'agents intelligents pour l'internet
Leslie L. Daigle et Sima Newell

19 Modèles de capteurs hautement autonomes
Fernando Figueroa

22 LALO: un nouveau langage de programmation orienté agent et un nouvel environnement
D. Gauvin, H. Marchal, et C. Saldanha

25 Le projet SIGMA: des agents basés sur le marché pour l'accès intelligent à l'information
Grigoris J. Karkoulas

28 Recherche sur les agents dans le Groupe de robotique cognitive
Y. Lesperance, H.J. Levesque, F. Lin, D. Marcu, R. Reiter, et R.B. Scherl

31 Des systèmes d'agents distribués pour la fabrication intelligente
Douglas H. Norrie et Brian R. Gaines

34 Des assistants personnels pour le professionnel de bureau
Cliff Grossner et T. Radhakrishnan

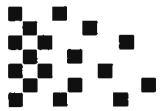
39 Comment recevoir (et conserver) une subvention de recherche.
Ian H. Witten
(Révisé par Janice I. Glasgow)

PRECARN Update 47 Nouvelles de PRECARN

Book Reviews 47 Critiques de livres



Recycled / Recyclable



Special Issue: Software agents

Innes A. Ferguson


It is evident to me from reading a number of respected AI publications and from attending various research and trade conferences recently that there is a strong and growing interest in software agents research and development in Canada. The purpose of this special issue is to present a broad view of some of the more interesting software agents work currently being undertaken in Canadian universities and government and industrial R&D organizations.

As the contributions in this issue illustrate, there is a fairly wide range of definitions and perspectives concerning software agents and software agent technology. This, in part, can be attributed to the relatively young and dynamic nature of the field. It is also, however, a testament to the highly multidisciplinary nature of agents and to the (consequent) broad range of application domains within which software agents are starting to appear.

As a more or less emerging field, there has been an uncomfortably large amount of hype associated with software agents. In the mid to late 1980's, much excitement ensued from the belief that a number of AI techniques had matured enough to warrant combining some of them inside integrated reasoning architectures, or agents. Just as this initial excitement started to wane in the early 1990's, renewed optimism grew with the emergence of the Internet and World Wide Web (WWW) as potentially fertile environments for developing and fielding agent-based applications. A number of research and commercial applications of agent technology have been witnessed to date, the most popular being in the areas of resource discovery and information filtering. While few, if any, of the commercial agent systems available today exhibit much in the way of intelligence — real or artificial — a number of them are proving popular with users who, caught in the flood of available online information, are seeking ever more advanced levels of automated computer assistance to help cope with the resulting complexity and volume of information processing.

Although the WWW features in a number of the papers in this issue of *Canadian Artificial Intelligence*, several other

interesting domains are addressed, ranging from personal office assistance (meeting and calendar management, telephone call processing) through robotic control to intelligent manufacturing. In tackling these domains, the various agent systems described herein put forward and employ a number of novel and leading edge techniques for representing knowledge, reasoning and planning, adapting to different environments, and coordinating with multiple agents. A number of the papers also present principled approaches to the architectural design, engineering, and programming of software agents. What these papers demonstrate, in fact, is that alongside the seemingly inevitable hype (object-oriented programming was in a similar position around a decade ago) there is also a considerable amount of mature, well-founded research and development work already taking place in software agents.


In addition to acknowledging the help of the magazine's editorial staff, I would like to thank all the contributors for their support and commitment (and timeliness!) in helping create this special issue. Finally, I hope the readership will take advantage of the information — and pointers to further information — included in this issue to add to and further enhance Canada's prominent position in this very current, active, and exciting field. 

Innes Ferguson is a member of the Agent Team at Mitsubishi Electric Europe, London, UK. His current work involves designing and applying intelligent agent technology toward advanced information publishing, discovery, and retrieval tools for the Internet. He had previously held research positions in artificial intelligence and software agents at BNR and NRC, both in Ottawa, Ontario. He received a Bachelor's Degree in Computer Science from Edinburgh University (Scotland) in 1985 and, while on a BNR Postgraduate Scholarship, a Doctorate in AI (intelligent agents) from Cambridge University (England) in 1992. He can be reached at innes@dlib.com.

Communication from the Ex-Co-Editor

Peter Turney

This is my last issue as co-editor of *Canadian Artificial Intelligence*. My colleague and co-editor, Suhayya Abu-Hakima, will be continuing on her own as the editor of the magazine. I have enjoyed very much the challenges of my two-year term as co-editor. Thanks to Suhayya for sharing the duties and responsibilities of editorship with me; thanks to Arlene Merling for managing the magazine and doing the layout; thanks to Janice Glasgow, Stan Matwin, Mike Halasz,

and Innes Ferguson for their help with the special issues; thanks to Greg Klymchuk for technical assistance; thanks to Benoit Farley and Alain Désilets for translation; thanks to Carol Tubman for help with proof-reading; and thanks to all our authors and readers, who are, of course, the most important part of the magazine. I leave my post confident that the magazine continues in good hands. 

Announcing the New CSCSI/SCEIO Executive

As the new president of the CSCSI, I would like to introduce myself and the new executive, and tell you some of the plans that we have for the society for the next couple of years. After having served as secretary of the society for the past two years, I am looking forward to working for the society as its president over the next two years. I will be working with our vice president Renee Elio, our secretary Guy Minot, our treasurer Howard Hamilton, and our past president Stan Matwin. Suhayya (Sue) Abu-Hakima will now be the editor of our magazine, and Arlene Merling will continue in her role of managing editor.

Over the next couple of years, we would like to increase the presence of the CSCSI in the artificial intelligence committee. Here are a few ideas that have been discussed to increase our local presence, our national and international presence, and our electronic presence.

Increased Local Presence

We would like to explore the idea of having the CSCSI sponsor local events at different locations throughout the country. Such events could include an 'evening of AI lectures' in cities that have several universities or businesses with AI interests. One person from each institution could speak on a 'controversial' topic, to be followed by a discussion session. It would be a great way to encourage the interaction of people and expose new AI students to a larger community.

Another idea that has also been discussed is having a local CSCSI contact at each institution. This person could help the society make contacts with new students, and could help coordinate local CSCSI activities.

Increased National and International Presence

Currently, the CSCSI holds an AI conference only every second year. Although this conference does bring together AI researchers from across Canada and around the world, changes should be made to make this conference more successful. Additionally, the CSCSI could make its presence known by taking a more active role in the sponsorship or organization of additional special topic international workshops at different locations around the country.

Increased Electronic Presence

In the past, the CSCSI has relied almost entirely on its magazine for communication with its members. Recently we have started to take greater advantage of on-line communication. Our magazine is now available electronically, and there is other information available at the CSCSI web site. We would like to expand our web site to be a greater service to our members, and to raise the external profile of our society. Additionally, we plan to establish more direct contact with our members through the use of the CSCSI e-mail list that is now being maintained by CIPS.

If you have any ideas concerning the activities of your society, just let us know. Our e-mail boxes are always open!

*Fred Popowich, President CSCSI
Associate Professor, School of Computing Science
Simon Fraser University
Email: popowich@cs.sfu.ca*



CSCSI/SCEIO Executive 1996 - 1998

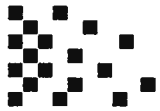
Fred Popowich, Simon Fraser University, President

Renee Elio, University of Alberta, Vice-President

Guy Mineau, Université Laval, Secretary

Howard Hamilton, University of Regina, Treasurer

Sue Abu-Hakima, National Research Council (NRC), Magazine Editor



Nick Cercone — 1996 CSCSI/SCEIO Distinguished Service Award Winner

Gord McCalla

Gord McCalla souligne les contributions de Nick Cercone pendant les quinze dernières années à la communauté de l'intelligence artificielle, qui lui ont valu le prix de la SCEIO pour service exceptionnel en 1996.

Gord McCalla highlights Nick Cercone's contributions to the Artificial Intelligence community over the last fifteen years leading to conferment of the 1996 CSCSI/SCEIO Distinguished Service Award.

October, 1983. 6:00 a.m. It is a traditional West Coast fall day. Grey, cool, and moist. And, Nick Cercone has an idea: why not create a new artificial intelligence journal, to be sponsored by the CSCSI/SCEIO? Nick and I are each on a sabbatical leave, living next door to each other in Vancouver, he the President of the CSCSI/SCEIO, and I the Vice-President. By the time I arrive for our daily mid-morning de-briefing, Nick has sketched out the idea on his ancient (even then) North Star computer and printed out these preliminary ruminations on his low quality (even then) dot matrix printer. We spend the rest of the day discussing the nature of the journal, putting together a potential editorial board, and formulating a plan for interesting both the AI community and potential publishers. Six months later, the *Computational Intelligence* journal is a reality, co-edited by Nick and me, the first explicitly international journal to be published by the National Research Council of Canada Journals, and the first new journal of any sort for NRCC in thirteen years. Now published by Blackwell, it is in Volume 13 and continues to be a leading international journal in AI. Nick and I are still co-editors.

This is typical of Nick Cercone. Not only a source of great ideas, but also plenty of follow-through to actually make things happen. Creating a new AI journal was not the only thing we did during our mutual sabbaticals in 1983-84. Under the auspices of the CSCSI/SCEIO, we also spawned the idea of a workshop entitled Theoretical Approaches to Natural Language Understanding (TANLU) that was held in Halifax in June of 1985. We produced a survey of Canadian AI research that proved invaluable to many in Canada as the AI boom (remember that?) hit full force (this report was later adapted for publication in AAAI's *AI magazine*), we put together papers on AI in Canada for the 1984 CIPS and 1984 CSCSI/SCEIO conferences, and we attempted to inform the Canadian AI community about the fledgling CIAR AI and Robotics program. It was an active time for AI in Canada, and Nick was actively involved in making things happen.

And, this was just one year in Nick's hectic life. His entire career has been one where he has made things happen. Here

is just a partial list of his service to AI and related areas over the last 15 years: President of the CSCSI/SCEIO (1982-84); President of the Canadian Society for Fifth Generation Research (1985-90); Co-Chair of the Planning Committee for the BC Advanced Systems Institute (ASI) in 1985 and Member of the ASI Scientific Advisory Board (since 1986); Member of the NSERC Interdisciplinary Grants Selection Committee (1988-91) and the NSERC Collaborative Grants Panel (since 1995); Leader in the IRIS Network of Centres of Excellence project (since 1989); Member of the CANARIE Board (since 1994); Member of the Canadian Genome Assessment of Technology Management Board (since 1993); Member of the National Research Council of Canada Advisory Committee on Artificial Intelligence (1989-91); President of the Canadian Association for Computer Science/ Association Informatique Canadienne (CACS/AIC), the Council of Canadian Computer Science Chairs (1991-94); Member of the Research Committee for the Centre for Image and Sound Research (CISR) (1991-93) and Member of the CISR Board of Directors (1989-93).

He has also served his discipline in a major way through conference and journal activities. In addition to his co-founding and continued co-editing of the *Computational Intelligence journal* (noted above), he has served (and is still serving) on five journal editorial boards. He was also Program Chair of the 4th Biennial Conference of the CSCSI/SCEIO (1982) and Program Co-Chair of the 7th Biennial CSCSI/SCEIO Conference (1988); Program Chair of the Theoretical Approaches to Natural Language Understanding Workshop (TANLU) (1985); Program Chair of the 7th International IEEE Conference on Data Engineering (1990) and General Chair of the 8th such Data Engineering conference (1991); Program Chair of the 3rd International Computational Intelligence Conference (1990); Vice-Chair of the 13th International Conference on Distributed Computer Systems (1992); as well as serving on literally dozens of programming committees over the years and refereeing scores of scientific papers. He has always been active in academic and university-related activities, most notably as Chair of the Simon Fraser University Computing Science


Department (1981-85), of the Centre for Systems Science at Simon Fraser University (1987-92), and Associate Vice-President (Research) and Dean of Graduate Studies at the University of Regina (since 1993). Through these offices, he has had great influence on the direction of computer and information science in Canada, with a special focus on encouraging technology interactions between the university and industrial sectors.

Nick is also a great traveler, giving talks and establishing interactions with researchers throughout the world, with a special emphasis on creating and maintaining interconnections with researchers in Asia. Through these travels, Nick has spread the word about his own, and other Canadian AI research. I won't go into any stories about these trips, but next time you see Nick you might ask him about his unorthodox return trip from India in 1987, or about a certain large, blue kimono!

Any academic, of course, serves his discipline best through his research and scholarly activities. Nick has had a stellar research career, publishing some 150 scientific papers in books, journals, and conferences. He has edited or co-edited five books. His research contributions span several areas, most notably natural language understanding, knowledge-based systems, knowledge discovery in databases, and computation in the humanities. He and his research collaborators have snared many millions of dollars in research funding over the years, through which several generations of researchers and graduate students have been funded. Nick's research supervision and teaching have been a source of inspiration to his students, and he continues to actively pursue these teaching activities at both the graduate and undergraduate levels even as he has taken on more and more senior administrative roles.

Lest one think that this man is some kind of superman, however, let me assure you that Nick sometimes displays all the attributes expected of the normal academic boffin. To illustrate, I would like to conclude with one more story. The scene is November 1983. Nick and I are in Chicago to attend the program committee meeting of the first International IEEE Data Engineering Conference, traveling on tickets arranged carefully by Nick some weeks in advance. The PC meeting is being held in conjunction with a general U.S. computer conference. We attend the opening night reception for the conference and are surprised to find that U.S. computer scientists apparently dress in black tie for such occasions. Scruffy as we are, we nevertheless wander through the reception, gulp down some wonderful smoked salmon, circulate for a bit exchanging a few pleasantries, before retiring fairly early. After all, we must be up bright and early the next morning for the 8:30 a.m. start of the PC meeting. The next morning dawns, cool, grey, and moist (making us Vancouver-ites feel right at home), and we go in search of the PC meeting. It is nowhere to be found. In a semi-panic we take the desperate measure of actually looking at the information we have been sent by the PC chair about the PC meeting. We are in the right city, in the right hotel,

outside the right meeting room, at exactly the right time of day. Unfortunately, it's the wrong day! Yes, Nick's famous organizational skills have managed to get us to Chicago a whole day early. And, the conference reception the night before was, in fact, a meeting of the Illinois Bar Association!

Despite his occasional feet of clay, however, it is entirely appropriate that Nick Cercone is the winner of this year's CSCSI/SCEIO Distinguished Service Award. Congratulations, Nick, on a job superbly well done and an award that has been thoroughly well earned. 

Gord McCalla is a Professor in the Department of Computer Science at the University of Saskatchewan. He has been involved in artificial intelligence (AI) research for over 25 years, with interests in natural language understanding, knowledge representation, and dynamic planning. These interests have been combined and expanded within the general context of developing artificial intelligence applications in education. In particular, he has worked on systems to support learning of programming CLISP and PL(C), has investigated basic issues in student modelling and domain knowledge representation, and has been involved in investigations into instructional planning, tutorial dialogue and alternative instructional paradigms. Numerous papers, invited presentations, and talks have come out of this research, including two co-edited books. Additionally, he has been active in the CSCSI/SCEIO, serving various executive roles, including President and Vice-President. He has served on program committees of numerous conferences (including AAAI, IJCAI, and CSCSI), and has been program chair of several conferences including AI 96 in Toronto, the 1992 International Conference on Intelligent Tutoring Systems, and the 1995 International Conference on Computers in Education held in Singapore (where he was co-chair). He is a member (and former director) of the Laboratory for Advanced Research in Intelligent Educational Systems (ARIES) at the University of Saskatchewan. He is currently involved in the Telelearning Network of Centres of Excellence as the director of sub-project 6.2.4, working on collaborative learning systems.

Ed. Note: A profile of Nick Cercone appeared in the Summer 1993 issue of Canadian Artificial Intelligence magazine authored by Connie Bryson, a free-lance technical writer based in Alberta.

Agents: From Theoretical Foundations to Practical Applications

Jean-Francois Arcand and Sophie Pelletier

This research was supported by the Centre for Information Technology Innovation (CITI). Both authors are now with Microcell Labs, 1250, boul Rene-Levesque Ouest, suite 400, Montreal (Quebec) Canada, H3B 4WB.

Résumé

Cet article porte sur la recherche sur les architectures multi-agents basées sur la psychologie cognitive et les réseaux neuronaux artificiels. Le point central de notre architecture multi-agents à base cognitive est l'adaptabilité. Les agents, confrontés à des environnements dynamiques, doivent avoir la capacité d'apprendre en généralisant les expériences passées, pour pouvoir réagir intelligemment devant de nouvelles situations. Cette expertise accumulée, de paire avec la détermination dynamique du profil de l'utilisateur, permet aux agents intelligents de rendre service efficacement à une grande variété d'utilisateurs dans une multitude de scénarios divers.

Abstract

This paper presents research on multiagent architectures based on cognitive psychology and artificial neural networks. Adaptivity represents the central point of our Cognitive-Based Multiagent Architecture. Agents faced with dynamic environments must have the ability to learn by generalization of prior experience in order to react intelligently to new situations. This cumulated expertise, together with dynamic user profiling, allows intelligent agents to serve efficiently a wide range of scenarios and users.

Introduction

During the past few years, CITI's Performance Support System (PSS) group has been exploring the tremendous benefits that can be gained from using adaptive, intelligent, and autonomous agents for intelligent user assistance. The theoretical foundation for these situational agents combines several areas of research: cognitive psychology, adaptive knowledge representation techniques, agent and multi-agent architectures, and economic market models. Central to all of the agents implemented by the PSS group is their ability to acquire long-term knowledge based on past experiences. Just as contextual data and agents' task definitions are essential in mapping agents' reactions to current situations, long-term adaptive knowledge becomes essential when all possible worlds cannot be specified *a priori* within the agent. Agents faced with dynamic environments must have the ability to learn from generalization of prior experience in order to react intelligently to new situations. This cumulated expertise, together with dynamic user profiling, allows intelligent agents to efficiently serve a wide range of scenarios and users. In this way, resource-constrained agents have

more solid grounds for making better choices when negotiating for resources. As well, agents with similar expertise can enter into trainer-trainee types of relationships.

Mapping Agents to Cognitive Psychology Concepts

An adaptive system must be designed to assist users in their problem-solving task, not to complicate it. Since users' knowledge about the system and skill level in interacting with the interface change with experience, and since users differ in their choice of problem-solving knowledge representation, it is useful to envision an adaptive qualitative model of the user's problem-solving behavior. This allows the overall task to be approached with realistic expectations of what can be achieved. Clearly, cognitive psychology issues play a major role in modeling the user. If the agent is to adapt to an individual user, it must encompass information about the user's cognitive limitations or strengths as well as the user's perceptual strengths and weaknesses.

Cognitive Model

Current research suggests that embedded user models would allow more intelligent and helpful human-computer interaction. In the course of our research (Arcand, 1995), several proposed cognitive models were analyzed and Rasmussen's model was viewed as the most appropriate one for representing a cognitive model of the user in agents (Rasmussen, 1983).

Rasmussen's model is a cognitive modeling framework focusing on real-time, multi-tasking domains, and has been successfully applied and validated in several domains. The model considers problem-solving to consist of three different levels of behavior: skill-based, rule-based, and knowledge-based. Skill-based behavior, the lowest level of behavior, occurs at an unconscious level; most of our behavior is skill-based. Actions such as moving the hand to pick something up are skill-based. Rule-based behavior is an intermediate level of behavior evident in familiar situations. At this level, we are conscious of our thoughts and our behavior is governed by stored rules. Knowledge-based behavior is the highest level of behavior and is used in unfamiliar situations, is goal-controlled and is typified by functional reasoning. While these types of behavior have fuzzy boundaries, they are very useful in describing the range of behavior that must be included in any model that

attempts to account completely for human behavior.

Rasmussen's model can also be thought of as a learning or abstraction hierarchy. A human first encountering a problem typically uses some kind of functional reasoning (knowledge-based behavior) to draw analogies with what is already known. Thus, the problem is dissected into smaller, more familiar, problems the human already knows how to solve (using skill-based behavior) by rules that have been developed earlier (rule-based behavior). If the same problem is seen frequently and is simple enough, eventually only skill-based actions may be involved. In terms of Rasmussen's model, we are concerned with learning the rules and the skills associated with particular behaviors.

Agent Modeling

On the basis of Rasmussen's model, a generic multi-agent architecture has been developed which takes advantage of the hierarchical distribution of knowledge. This architecture, called the Cognition-Based Architecture (CBA), rests on the interaction of three classes of agents: Junior Intelligent Agents (JIA), Senior Intelligent Agents (SIA), and Facilitator Agents (FA) (Arcand, 1996). The first two classes encapsulate knowledge on how to support the user in performing tasks; the last one is internal to the system and facilitates communication among other agents and optimization of the multi-agent world.

JIA's are specialized in domain specific know-how, and represent the skill-based and rule-based levels. In an intelligent user support environment, these agents have the ability to take into account contextual information about the user's current work session. SIA's possess the knowledge-based information relating to a user's tasks, and consist of expertise built up from a history of past interaction between the user and the system. JIA's and SIA's are paired in order to create a complete model of the user. This allows the system to react on a contextual basis if possible, or to generalize from its past experience if no model is known for the current context.

Dividing the user model into two classes of agents results in two major advantages: it diminishes the complexity of the knowledge base as the latter is built in a modular fashion, and it allows for the selection of the most appropriate knowledge representation technique at every level. For instance, if rule-based systems are appropriate for JIA's, SIA's can request the use of techniques with a higher capacity to generalize from known information, as well as to learn and forget with time, thus implementing true adaptiveness into the user support system. The technology of artificial neural networks is one of the few that offers these required characteristics.

Another benefit gained from splitting the user model among collaborative agents is the possibility of having users share their SIA's in order to build group knowledge and benefit from each others' usage of the system. Novice users' JIA's could team up with experts' SIA's to facilitate expertise transfer. In essence, if the expertise of the best user can be

captured easily and economically and made available to other users, significant economic benefits would accrue. Another way to accomplish this expertise transfer is to have SIA's support one another when one does not yet possess sufficient knowledge. Relations between all agents of the architecture must therefore be kept dynamic.

Such dynamism brings another dimension to the architecture. In addition to the option to choose which agents team up, agents can have the authority to punish and reward their collaborators, and thus influence other agents in their future collaboration decisions. We believe that such dynamic economy driven agent teams will become part of the solution for multi-agent worlds evolving on the World Wide Web (WWW). Agents tasked to complete searches and transactions on the Web on behalf of users will be able to team up with the best possible collaborators on the market in order to maximize both their efficiency and effectiveness in meeting their users' requests.

Adaptive Knowledge Representation Techniques

One of the key issues in designing agents is to choose the knowledge representation technique best suited to the agent's needs, particularly for truly adaptive agents as described in the context of the Cognition-Based Architecture. Different research projects with highly adaptive agents and their multi-agent worlds lead us to explore different knowledge representation techniques. However, distributed artificial neural networks lie at the kernel of the designed adaptive models. At CITI, we have successfully applied DANNs to adaptive modeling in such diverse fields as information filtering (Pelletier, 1996), human factors (Arcand, 1994) (Ramstein, 1996), collaborative work (Arcand, 1994), etc.


Distributed Artificial Neural Networks (DANN) have been developed to simulate human problem-solving resources and mechanisms and have shown strong abilities in learning and flexible knowledge processing (i.e., generalize knowledge to novel situations). DANNs are seen as super-networks (distributed meaning shared and simplified), comprised of a number of sub-networks, that can communicate with each other. Such super-networks are intended to facilitate the modeling of complex and heterogeneous realities. The designer's task is thus to divide a complex problem into a series of simpler problems, according to Rasmussen's model, thus dividing the work among the agents.

With the ability to learn from a series of training examples, artificial neural network models provide the possibility of extracting user characteristics and providing on-line help depending on current user behavior. DANNs are extremely fast; therefore adaptation could be offered on-line in real-time. DANNs can be used to extract information such as cognitive goals, strategies, command sequences, and information needs. This can then be used to provide adaptive advice to the user or to intelligently filter incoming cooperative information.

Conclusion

This article presented multiagent research using cognitive psychology principles within a human-computer system. In the framework of open multi-agent systems, free market economy concepts allow systems to perform better by eliminating agents whose credibility or utility are in doubt and by improving the performance of remaining agents. We believe that the use of cognitive psychology principles avoids restricting knowledge representation to artificial intelligence techniques by providing a framework for applying artificial neural networks.

References

- Arcand, J.F., (1994). An Artificial Neural Network for the Ergonomic Evaluation of a Human-Computer Interface, in *Proceedings of IEEE Sixth International Conference on Tools with Artificial Intelligence*, November 6-9, New-Orleans, LA. pp. 606-608.
- Arcand, J.F., Chafetz, R., (1994). Using Markov Chains, Adaptive Neural Networks and Evolutionary Programming to Create an Intelligent Agent, *Proceedings of International Symposium on Artificial Neural Networks*, December 15-17, Tainan, Taiwan, pp. 526-536.
- Arcand, J.F., Pelletier, S. (1995). A Framework for Creating Societies of Agents, pp. 235-238 in Pearson, D.W., Steele, N.C., and Albrecht (eds) *Artificial Neural Nets and Genetic Algorithms*, Springer-Verlag, Vienna, New-York.
- Arcand, J.F., Pelletier, S. (1996) Cognition Based Multiagent Architecture in Intelligent Agents: Theories, Architectures and Languages. *Lecture Notes in Artificial Intelligence*, Volume II, Springer-Verlag, pp. 267-282.
- Pelletier, S. Arcand, J.F, Velisarios, J. (1996). Stealth: A Personal Digital Assistant for Information Filtering, in *Proceedings of the 1996 Practical Applications of Intelligent Agents and Multi-Agents Conference*, London, April 22-24.
- Ramstein, C., Arcand, J.F., Deveault, M. (1996). Adaptive User Interface with Force Feedback, in *Proceedings of ACM Conference on Human Factors in Computing Systems, CHI 96*, April 14-18, Vancouver, Canada, pp. 408-409.
- Rasmussen, J. 1983. Skill, Rules, and Knowledge: Signals, Signs, and Symbols, and Other Distinction in Human Performance Models. *IEEE Transactions on Systems, Man, Cybernetics*, Vol. SMC-13, May/June, 257-266. 

Jean-Francois Arcand is a mathematician working in the field of human-computer interaction and artificial neural networks. His research interests include information filtering, adaptive interfaces, intelligent agents, and multi-agent architectures. He has over twenty publications in books and conference proceedings related to these fields. Since September 1996, he is working at Microcell Labs, a research and development organization dedicated to advancing the state of the art in personal communication services (PCS). Email: Jean-Francois.Arcand@Microcell.net

Sophie Pelletier holds a B.Sc. in Computer Sciences from the University of Sherbrooke. She is presently completing a M.Sc. in Computer Engineering at the École Polytechnique de Montréal on the subject of multi-agent architectures for information retrieval in heterogenous data sources. She has worked as a researcher for the Centre for Information Technology Innovation (CITI) and is now part of the research team at Microcell Labs in Montreal. Her research interests include the design of open distributed multi-agent architectures and of intelligent autonomous agents, as well as mobile computing. Email: Sophie-Julie.Pelletier@Microcell.net

Check out our Website!

Canadian Artificial Intelligence Magazine

Accessible to CSCSI/SCEIO members only

If you would like to try electronic access, please send an E-mail message to peter@ai.iit.nrc.ca, with the following information:

- (1) your full name
- (2) your e-mail address
- (3) a short user name (5-8 characters, for web access)
- (4) a password (5-8 characters, for web access), and
- (5) your CSCSI/SCEIO membership identification number, which can be found on the mailing label of the magazine.

The Agent Building Shell: A Tool for Building Enterprise Multi-Agent Systems

Mihai Barbuceanu and Mark S. Fox

Résumé

Le concept d'agent offre un niveau d'abstraction qui permet de construire des systèmes informatiques qui interagissent globalement à travers des réseaux, reliant les gens, les organismes et les machines sur une plate-forme virtuelle unique. On appelle "agents" les entités informatiques qui opèrent à ce niveau. Les systèmes à base d'agents sont naturellement propices à la modélisation des entreprises modernes, qu'on peut concevoir comme des réseaux mondiaux de fournisseurs, d'usines, d'entrepôts, de centres de distribution et de détaillants à travers lesquelles les matières premières sont obtenues, transformées en produits, livrées aux clients, traitées et améliorées. Du point de vue pratique, tout système de soutien à la création de systèmes à base d'agents doit permettre la réutilisation de descriptions de mécanismes de coordination, de composantes de systèmes, de services et de bases de connaissances. Sur cette base, nous développons un outil pour la création d'agents qui fournit des langages et des services réutilisables, évitant ainsi aux développeurs de construire des systèmes à base d'agents à partir de zéro, et garantissant que les services essentiels d'inter-opération, de communication et de coopération seront toujours présents pour supporter les applications.

Abstract

The agent view provides a level of abstraction at which we construe computational systems that interoperate globally across networks, linking people, organizations, and machines on a single virtual platform. We call the computational entities that can operate at this level *agents*. Agent systems can naturally model modern enterprises, which are best seen as world-wide networks of suppliers, factories, warehouses, distribution centers, and retailers through which raw materials are acquired, transformed into products, delivered to customers, serviced, and enhanced. From the practical standpoint, any support for building agent systems must provide the ability to reuse descriptions of coordination mechanisms, system components, services, and knowledge bases. Based on this recognition, we are developing an Agent Building Shell that provides reusable languages and services for agent construction, relieving developers from the effort of building agent systems from scratch, and guaranteeing that essential interoperation, communication and cooperation services will always be there to support applications.

1. Motivation

The agent view provides a level of abstraction at which we construe computational systems that interoperate globally

across networks, linking people, organizations, and machines on a single virtual platform. We call the computational entities that can operate at this level *agents*. Agent systems can naturally model modern enterprises, which are best seen as world-wide networks of suppliers, factories, warehouses, distribution centers, and retailers through which raw materials are acquired, transformed into products, delivered to customers, serviced, and enhanced. In order to operate efficiently, enterprise functions must work in a coordinated manner. But the dynamics of the enterprise and of the world market make this difficult; exchange rates unpredictably go up and down, customers change or cancel orders, materials do not arrive on time, production facilities fail, workers are ill, etc. causing deviations from plan. In many cases, these events can not be dealt with locally, i.e., within the scope of a single supply chain "agent," requiring several agents to coordinate in order to revise plans, schedules, or decisions. In the manufacturing domain, the agility with which the enterprise supply chain is managed at the tactical and operational levels in order to enable timely dissemination of information, accurate coordination of decisions, and management of actions among people and systems, is what ultimately determines the efficient achievement of enterprise goals and the viability of the enterprise on the world market.

2. The Agent Building Shell

From the practical standpoint, any support for building agent systems must provide the ability to reuse descriptions of coordination mechanisms, system components, services, and knowledge bases. Based on this recognition, we are developing an Agent Building Shell [4, 2] that provides reusable languages and services for agent construction, relieving developers from the effort of building agent systems from scratch and guaranteeing that essential interoperation, communication, and cooperation services will always be there to support applications. We have built a layered architecture of the Agent Building Shell comprising:

The *coordination language*, providing services for defining distributed agent configurations, managing communication, defining and managing structured interactions amongst agents, external software integration and in context acquisition, and debugging of coordination knowledge. The communication part is based on KQML [11], while the coordination language [2, 3, 5] is a novel contribution of this work.

The *cooperative conflict management service*, providing a general model for reasoning about retraction in a multiagent system. If an agent receives contradictory information from other agents — for example p from Agent1, q from Agent2,

the agent believing that $p \wedge q \Rightarrow \text{false}$ — it applies this model to retract beliefs supporting the contradiction and reinstall consistency both locally and with its neighbors. The model relies on the credibility of the information sources balanced against the utility of the information, as measured, for example, in terms of costs to be paid when undoing decisions as a consequence of belief retraction. The model reinstates agent-level consistency and, through negotiation with the agents affected by the retracted beliefs, extends the consistent state to surrounding agents. Also, the model stipulates the kind of cooperative behavior that an agent must exhibit towards other agents in the process of reestablishing a consistent state [7, 2].

The *cooperative information distribution services* provide permanently active information distribution services allowing agents to stay informed about significant events without having to explicitly demand other agents to provide this information each and every time they need it. Agents advertise their long-term topics of interest to the community. Agents that can supply relevant information will do so whenever the information is available and as long as the interest persists. If previously sent information is later invalidated, senders which are aware of that will behave cooperatively by notifying receivers.

The *ontology layer*, consisting of the actual conceptualizations agents maintain about their domain, environment, and self. Some conceptualizations are shared among agents to allow them to communicate in terms that are semantically unified. The environment and self representations use a shared organization ontology that captures the structure of organizations, the roles, goals, actions, empowerment, and responsibility of member agents of the organization [10].

The *knowledge management layer*, providing support for general purpose representation and inference. It is used to represent an agent's conceptualizations and beliefs about its domain, operation environment, and its own capabilities. It provides support for nonmonotonic reasoning and general purpose deductive reasoning [8].

3. Coordination

We have built a domain independent coordination language (COOL) that provides a conceptualization of the coordination task in terms of structured "conversations" amongst agents. The approach builds on the following assumptions: (i) autonomous agents have plans that explicitly include interactions with other agents; (ii) these plans are based on limited assumptions about the possible actions/reactions of the other agents; (iii) the plans may be incomplete and inaccurate and the knowledge to extend them may become available only during execution; (iv) agents are able to extend and modify their plans during execution. COOL is an agent planning and execution language that embeds these principles. The most important construct employed by COOL is that of a plan that explicitly represents interactions and alternative courses of action, that may be incomplete or

inaccurate and that is modifiable during execution. This construct is called conversation-class, to emphasize the importance of interactions in agent plans. When interacting, agents instantiate their own conversation-classes, creating actual conversations that maintain the execution status of the plans described by the conversation-classes. Conversation-classes consist of states, sets of conversation rules managing the transitions amongst states, specifications of control mechanisms for rule execution and others. Conversation rules specify patterns of received messages and agent internal conditions that, when satisfied, trigger a transition to another state, send out messages, and execute actions. Some rules are triggered automatically upon state entry or exit. Others are triggered as specified timeouts expire, allowing agents to operate in real time.

COOL provides a multiple conversation management mechanism that allows agents to maintain a dynamically constructed hierarchy of conversations and run them simultaneously. Child conversations are created by parent conversations, taking over some part of the interaction. Child conversations may propagate upwards (to parents) messages or situations they are not prepared to deal with. Agents start in an initial conversation and, as messages are received and actions done, new conversations are created and added to the hierarchy. Some conversations can be put on wait for others to reach a specified state and be resumed when this happens. This allows agents to switch focus of attention as imposed by dynamic events.

A challenging aspect when dealing with coordination knowledge in a multi-agent environment stems from the social nature of coordination knowledge coupled with the logical, spatial and temporal distribution of agents. Essentially, this limits the applicability of approaches based on designing coordination structures in advance, off-line, on the assumption of complete knowledge and totally predictable interactions. Rather, we should view coordination knowledge as dynamically emerging through the unpredictable interactions of participants who necessarily have limited knowledge and who are usually in the process of doing their own work. This means that the coordination system must leverage the ability of participants (agents) to dynamically create and alter interaction structures on the fly and to build up common interaction structures without being temporally or spatially collocated.

In our system, we satisfy this requirement by allowing (i) incremental modifications of the cooperation structures, e.g., by adding or modifying knowledge expressed in rules and conversation objects, (ii) system operation with incompletely specified coordination structures, in a manner allowing users to dynamically intervene at appropriate times and take any action they consider appropriate, and (iii) system operation in a user controlled mode in which the user can inspect the state of the interaction and take alternative actions not necessarily represented in the system. This is achieved by operating with incomplete rules. Whenever a conversation encounters an incomplete rule, a special graphical interface

is popped up and the user is given the discretion to fill in any piece of missing knowledge and make any decision about the continuation of the conversation. Using structure editors, users can create new COOL structures or modify existing ones, incrementally improving the existing coordination knowledge. As this intervention is always situated in the actual execution context, knowledge acquisition/debugging/extension is equally situated, thus more accurate and easier to perform. By being able to operate with any amount of incomplete knowledge, prototypes can be rapidly built, demonstrated, and evaluated. Refinement can then follow as above, shortening the time and cost of development.

4. Current Status and Future Directions

Currently, all layers have been implemented and experimented with. The information distribution and conflict management services are embodied in a generic Information Agent that functions as an intelligent mediator in our enterprise architecture [6]. The Information Agent makes use of description logic knowledge management [8] to intelligently solve subscriptions to information by proving the subsumption of available information by topics of interest. The conflict management model [7] is also implemented in this context and makes use of extended truth-maintenance algorithms to determine the impact of contradictory information across the beliefs held by an agent. The coordination system has been evaluated on several problems, ranging from well-known test problems like n-queens to the supply chain of our TOVE virtual enterprise [9] and to supply chain coordination projects carried out in cooperation with industry. In all situations, the coordination model and the acquisition tool enabled us to quickly prototype the system and build running versions demonstrating the required behavior. Often, an initial (incomplete) version of the system has been built in a few hours enabling us to immediately demonstrate its functionality. We have built models containing hundreds of conversation rules and tens of conversation classes in one to two weeks. Moreover, we have found the approach explainable to industrial engineers interested in modeling manufacturing processes.

Our major priority at the moment continues to be gathering empirical evidence for the adequacy of the approach to industrial applications and for that matter we are jointly working with several industries. Apart from that, we are making our COOL agents accessible through the WWW and improving the integration of all components of the Agent Building Shell. Since our approach is in an essential way managing workflow, we have started addressing organizational workflow modeling and enactment as part of our participation in the Globeman/IMS effort. Last but not least, explaining the decisions and behavior of multi-agent systems will become more and more important as we move into more complex applications. Having explicit representations of coordination mechanisms forms the basis for providing such explanations and we are studying the issue as part of another joint effort with industry.

5. References and Further Readings

1. <http://www.ie.utoronto.ca/EIL/ABS-page/ABS-intro> contains many papers and examples about our agent research.
2. M. Barbuceanu and M. S. Fox. Capturing and Modeling Coordination Knowledge for Multi-Agent Systems. To appear in the *International Journal on Cooperative and Intelligent Information Systems*, 1996.
3. M. Barbuceanu and M. S. Fox. Coordinating Multiple Agents in the Supply Chain. To appear in *Proceedings Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, IEEE Press, 1996.
4. M. Barbuceanu and M.S.Fox. The Architecture of an Agent Building Shell. In M. Wooldridge, J.P. Mueller, M. Tambe (eds) *Intelligent Agents II*, Springer Verlag Lecture Notes in AI, vol. 1037, March 1996.
5. M. Barbuceanu and M.S.Fox. COOL: A Language for Describing Coordination in Multi-Agent Systems. In V. Lesser (ed) *Proceedings of First International Conference on Multi-Agent Systems*, MIT Press/AAAI Press, San Francisco, 1995.
6. M. Barbuceanu and M.S.Fox. The Information Agent: An Infrastructure for Collaboration in the Integrated Enterprise. In S.M.Deen (editor) *Cooperative Knowledge Based Systems*, DAKE Centre, 1994.
7. M. Barbuceanu and M.S.Fox. Conflict Management with a Credibility/Deniability Model. In M. Klein and S. Lander (Co-chairs) *Models of Conflict Management for Cooperative Problem Solving*, Tech. Report WS-94-04, AAAI Press.
8. M. Barbuceanu. Models: Toward Integrated Knowledge Modeling Environments, *Knowledge Acquisition*, 5, pp. 245-304, 1993.
9. M. S. Fox. A Common-Sense Model of the Enterprise. In *Proceedings of Industrial Engineering Research Conference*, 1993.
10. M. S. Fox, M. Barbuceanu, and M. Gruninger. An Organisation Ontology for Enterprise Modeling: Preliminary Concepts for Linking Structure and Behavior. In *Proceedings of the Fourth Workshop on Enabling Technologies, Infrastructure for Collaborative Enterprises*, IEEE Computer Society Press, 1995.
11. T. Finin, et al. Specification of the KQML Agent Communication Language. The DARPA Knowledge Sharing Initiative, External Interfaces Working Group, 1992.

Mihai Barbuceanu holds a Ph.D. in computer science from the Technical University of Bucharest and is currently a Research Scientist in the Enterprise Integration Laboratory at the University of Toronto. His current research focuses on coordination and multi-agent systems. Previously, he has done work in knowledge representation, description

(continued, page 15)

Multi-agent Systems at Laval University

B. Chaib-draa

Résumé

Ceci est un bref compte rendu des recherches effectuées actuellement sur les aspects multiagents, au sein du département d'informatique de l'Université Laval. Ce compte rendu est divisé en trois axes: 1) les architectures d'agents, 2) les aspects théoriques portant sur le multiagent; 3) les applications en cours ou envisagées. Pour ce qui est des architectures d'agents, nous travaillons présentement sur des modèles hiérarchiques du type de celui de Rasmussen, c'est à dire une architecture qui fait ressortir les aspects habiletés, règles et connaissances. Pour l'aspect théorique, nous nous orientons vers la logique descriptive et l'algèbre des relations. En ce qui concerne les applications, nous portons nos efforts sur le trafic routier, la gestion des bases hétérogènes distribuées et le génie concurrent.

Abstract

In this short paper, we describe research in multiagent systems (MAS) at Laval University. The research is divided into 1) agent architectures, 2) theoretical aspects and, 3) applications. In the context of agents, we are interested by the hierarchical human model, particularly the skill, rule, knowledge (SRK) model of Rasmussen. Our vision of theoretical aspects for multiagent systems is based on descriptive logics and relational algebra. Finally, in the context of applications, we are working on urban traffic, cooperation between heterogeneous distributed databases, and concurrent engineering.

Introduction

In multiagent systems (MAS), the agents generally preexist and are autonomous and typically heterogeneous. Research in this context is concerned with coordinating intelligent behaviors among a collection of autonomous agents, that is, how these agents can coordinate their knowledge, goals, skills, and plans jointly to take action and to solve problems. More specifically, in this type of environment:

- 1) agents may be working toward a single global goal, or toward separate individual goals that interact;
- 2) agents are generally self-motivated, autonomous, and rational;
- 3) agents might share knowledge about tasks and partial works;
- 4) agents must reason about the process of coordination among them.

Thus, coordination is central to multiagent systems; without it, any benefits of interaction are not possible and the behavior of the group of agents can become anarchic. The concept of coordination between agents is central to our work at Laval University. In this short paper, we describe this work.

Agent Architectures

To produce coordinating behaviors in MAS, most research has concentrated on developing groups in which both control and data are distributed. Distributed control means that agents are autonomous (to some degree) in their actions. Evidently, this autonomy can lead to uncoordinated activities because of the uncertainty of each agent's actions. In this context, a number of coordination techniques have been deployed. However, no technique investigated the relation between uncertainty and the situation addressed by agents since it is well known that the uncertainty decreases when the degree of familiarity of the addressed situation increases.

Our work on agent architectures is a step towards remedying this problem by providing a framework for designing multiagent systems in which agents are capable of coordinating their activities in routine, familiar, and unfamiliar situations. In routine situations, generally, agents have complete knowledge of their actions and interactions and in these conditions, it would be possible to know exactly what each agent is doing at the present moment and what it is intending to do in the future. In this context, it would be possible to avoid conflicting and redundant efforts; agents could be perfectly coordinated and the effort of achieving this state would not be prohibitively high.

Such complete knowledge about actual actions and reactions is only feasible in routine situations. In real-world domains, however, there are also familiar and unfamiliar situations. In familiar situations, agents can also coordinate their behaviors since individual acts are carried out under expectations of future actions of other agents' actions and beliefs. In unfamiliar situations, however, the coordination between agents is difficult to obtain and maintain because agents need to be constantly informed of all developments in order to elaborate their decisions. In fact, a complete analysis to determine the detailed activities of each agent is impractical in unfamiliar situations, and agents should have the capability to reason about others in order to make decisions.

Three kinds of interactions can be consequently studied in multi-agent environments: interaction in routine situations, interaction in familiar situations, and interactions in unfamiliar situations. Evidently, the coordination is easier to obtain and to maintain in routine than in unfamiliar situations (familiar situations are intermediate). Communications are also more requested in unfamiliar situation than in routine situations.

The skills, rules, and knowledge-based (SRK) processing proposed by Rasmussen (J. Rasmussen, "Information Processing and Human-Machine Interaction: An approach to Cognitive Engineering," North Holland, 1986) reflects differences in consistency of response and conscious control

of human behavior. Skill-based behavior refers to fully automated activities such as tracking or guiding, rule-based behavior to stereotyped actions such as test point checking in troubleshooting electronic circuit, and knowledge-based behavior to conscious activities involving problem solving or decision-making.

In the SRK perspective, the skill-based level denotes almost routine performances. At this level, agent performance is governed by stored patterns of predefined procedures that map directly from observation (i.e., perception) to an action. The rule-based level represents more conscious behavior when handling familiar situations. The rule-based behavior is conventionally described by a set of heuristics, that is, by a set of stocked rules. The knowledge-based level accounts for unfamiliar situations for which know-how or rules are not available. Indeed, for these situations, the control of performance must move to a higher conceptual level, in which behavior is controlled by goal and utility and more generally by the reasoning about others.

These considerations have led us to adopt Rasmussen's conceptual model as a framework to develop an agent architecture that evolves in a world inhabited by other agents. This model is driven by the goal of combining the complementary advantages of reactive, planning, and decision-making systems in order to take into account different situations which arise in multiagent environments: routines, familiar, and unfamiliar situations. First, it needs to be reactive to be able to quickly respond to changes in its environment. Secondly, it should be capable to plan its activities for a recognized task or goal. Finally, the model must also allow reasoning about others since agents should be capable of making decisions that take into account their own intentions and also others' intentions.

As previously noted, the coordination between agents is generally more easy to obtain and to maintain in routine than in unfamiliar situations. Indeed, when agents have routine and familiar behaviors, these behaviors are generally known by all agents. In this context, any agent has facilities to coordinate its activities with other agents, and communication is requested only when necessary. In order to strengthen the levels relative to routines and familiar situations, we have enriched each agent with social regularities (for instance, coordinative rules, cooperative rules, etc.) and social collectivities (e.g., roles, groups, organizations, etc.) in the form of social laws. By doing this, we assume that the agents adopt these social laws and each agent obeys these laws and will be able to assume that all others will as well.

Finally, we have demonstrated, on some scenarios of urban traffic, the applicability of our approach based on SRK model (see Applications).

Theoretical Aspects

Actually different formal tools are used to analyze and synthesize our multi-agent systems. These formal tools are based on (classic and non-classic) logics, game theory,

decision theory, and market mechanisms. Such tools should be completed by the calculus of relations if we want to represent and to reason about "social structure" in which relations between agents is the relevant unit. In this respect, we are developing a formal approach based on relational algebra for the reasoning about crisp and fuzzy relations between agents. The important elements of this approach include 1) basic notions of relation algebra, 2) how to use crisp relations in multi-agent systems, 3) how to use fuzzy relations in multi-agent systems.

In this research, we have moved some steps towards a formal theory of relations that characterize a "social structure" between agents. Precisely, we have proposed a formalism based on relation algebra for relationships between components (essentially agents, tasks, and resources) of the multi-agent systems. To achieve this, we have studied in detail 1) the basic notions of this formalism, and 2) their use in the context of multi-agent.

The most obvious consequence of using such formalism is to allow each agent to reason about relations with a "natural" tool which is the relation algebra. Such a reasoning allows agents: (i) to manage inter-dependencies between their activities in order to obtain and to maintain coordination between them, (ii) to manage their meta-level information exchange to decrease the overall communication flow.

Further efforts must be carried out in order to achieve a complete example formalized with the relation algebra approach.

Another theoretical aspect on which we are working is reasoning about causal relations between agents' goals and activities in order to contribute to agents' coordination. To do this, we have investigated the issue of using cognitive maps (CMs) for this reasoning. CMs are causal maps represented as directed graphs in which nodes represent problem construct (utilities, goals, or other concepts) and edges or arrows represent dependencies or influences between constructs. The distinguishing characteristic of a CM is a focus on abstract causal relationships among concepts, specifically on the directions, or signs (+, -, 0 for instance) of causal dependency. At that time, we can explain how CMs allow agents to derive decisions about which goals to achieve or which areas of the graph (of goals) to explore in order to contribute to coordination. In this research area, we have also constructed a formal model for CMs with precise semantics based on relation algebra. By defining this semantics, we justify the classical intuitive inference mechanisms, based on reasoning from cause to effect.

Finally, we are also working on a new formalism based on descriptive logics for reasoning in multi-agent environments. This work investigates language, as used in KL-ONE-type knowledge representation systems, from an algebraic point of view. Descriptive logics are based on two primitive syntactic types, called concepts and roles, which are usually interpreted in a "model" as sets and relations respectively. We propose an algebraic rather than a model approach and we show that descriptive logics can be naturally

accommodated in algebras of sets interacting with relations. As well, we use algebra as a formal tool for reasoning about concepts interacting with roles in multi-agent environments.

Applications

Actually, three applications have been used to validate our idea and concepts: urban traffic, a heterogeneous distributed database system, and concurrent engineering.

Urban traffic

The architecture described in the Agent Architectures section depicts a general model and can be adapted to a vast number of fields. We implemented each component of the architecture (in Common Lisp) while keeping in mind this idea of generality. In order to validate this architecture in a multi-agent environment, we have implemented some scenarios from urban traffic, using our architecture.

As stated previously, there are three levels (S-R-K: skills, rules, and knowledge) of cognitive control (for each agent) in multiagent systems (MAS). These three levels can be grouped together into two general categories. K is concerned with analytical problem solving based on symbolic representation, whereas S and R are concerned with perception and action. S and R levels can be only activated in routine and familiar situations because these low levels require that agents know the perceptual features of the environments and the knowledge relative to these situations. The K level, on the other hand, is only activated in unfamiliar situations. These considerations have been taken into account in designing our agents.

With these agents, we have developed implementations and experiments in urban traffic to verify our intuitions about the distinction between the two modes of processing: perceptual processing (S and R levels) and analytical problem solving (K level). Perceptual processing is fast, effortless, and is propitious for coordinated activities between agents, whereas analytical problem solving is slow, laborious, and can lead to conflicts between agents. To this end, we have conducted a series of experimental studies on three policies of the crossroads scenario. The policy 1 refers to a routine of urban traffic. In this routine, agents' activities are coordinated by traffic lights. Policy 3 refers to an unfamiliar situation of the crossroads scenario. In this situation, agents should rely on social laws to make decisions because traffic lights are off, and there is no policeman to coordinate their activities. Finally, the policy 2 refers to a complex situation where agents' activities are coordinated by a policeman, that is by a knowledgeable agent.

We examined for the cars three performance indices when comparing the policies: communication, processing time for each mode of reasoning (skills, rules and knowledge), and task effectiveness. The effectiveness is specified by two distinct parameters: errors and waiting time. For the policeman who intervenes in the policy 2, we examined two performance indices: communications and processing time for each level of the agents' cognitive control.

As we had anticipated, our implementation and experiments successfully demonstrated that perceptual processing is fast, effortless, and is propitious for coordinated activities between agents.

Heterogeneous Distributed Database System

Heterogeneous distributed database system (HDDDBMS) involves the interoperability of data sources with different structure and different implementation across a network. One approach to realize this type of integration is to build interfaces between the different databases being integrated. This approach works, for a particular case, at a specific point in time. Another possibility that we are investigating is the integration of information agents as front-ends of the different databases. We think that such agents can be an important tool for information-gathering and query-answering for the existing databases as well the Web services. We are working towards this integration using our agent architecture (see Introduction) and the theoretical aspects about relationships between agents.

Negotiation and Cooperation in Concurrent Engineering

Actually, we are working on some examples from concurrent engineering to validate our theoretical approaches on relations between agents, tasks, and resources; causal relations between agents' goals intentions and utilities; and reasoning on concepts and roles as specified by our descriptive logic.

References

- Chaib-draa, B., Desharnais, J., and Lizotte, S. A Relation Graph Formalism for Relationships Between Agents. *IEEE Trans. on Knowledge and Data Engineering* (submitted).
- Chaib-draa, B. Formal Tools for the Multiagent Systems. *IEEE Trans. on Knowledge and Data Engineering* (submitted).
- Hassane, F. and Chaib-draa, B. Evaluation of Different Algorithms for Rule Allocation in Parallel Rule-Based Systems. *Revue d'Intelligence Artificielle* (to appear).
- Chaib-draa, B. Crisp and Fuzzy Causal Reasoning in Multiagent systems. *Decision Group & Negotiation* (submitted).
- Chaib-draa, B. and Levesque, P. Hierarchical Model and Communication By Signs, Signals and Symbols in Multiagent Environments. *Journal of Experimental and Theoretic Artificial Intelligence* (to appear).
- Chaib-draa, B. Interaction Between Agents in Routine, Familiar and Unfamiliar Situations. *International Journal of Intelligent & Cooperative Information Systems* (to appear).
- Chaib-draa, B. Industrial Applications in Distributed AI. *Communications of ACM*, 38(11), 1995.
- Zhang, L. and Chaib-draa, B. A Design Methodology For Real-Time Systems to be Implemented on Multiprocessor Target Machine. *Journal of Systems and Software* (to

appear).

- Chaib-draa, B. Plans in Natural Language Dialogues. *Knowledge-Based Systems*, 6(1), pp. 67-75, 1993.
- Chaib-draa, B. Control and Communication in Distributed Problem Solving. *Revue de Liaison de la Recherche en Informatique Cognitive des Organisations*, 4(3 et 4) pp. 6-15, 1992.
- Chaib-draa, B., Moulin, B., Mandiau, R., and Millot, P. Trends in Distributed Artificial Intelligence. *Artificial Intelligence Review*, 6(1), pp. 35-66, 1992.
- Chaib-draa, B., Libert, G., and Dergal, A., Temporal Reasoning Models for Second Generation Expert Systems. *Revue de Liaison de la Recherche en Informatique Cognitive des Organisations*, 3(4) pp. 41-46, 1991.
- Chaib-draa, B. and Millot, P. A Framework for Cooperative Work: An Approach Based on the Intentionality. *Artificial Intelligence in Engineering*, 5(4), pp. 199-205, 1990.
- Chaib-draa, B. Distributed Artificial intelligence: An Overview. *Encyclopedia of Computer Science and Technology*, Kent, A. and Williams, J. (eds), 31 (suppl. 16), 1994.
- Chaib-draa, B. and Vanderveken, D. On the Success and Satisfaction of Speech Acts for Computational Agents. *Essays in Speech Act Theory*, Vanderveken, D. and Kubo, S. (eds), Benjamins Amsterdam, Philadelphia, (to appear).
- Moulin, B. and Chaib-draa, B. A Review of Distributed AI. *Foundations of Distributed Artificial Intelligence*, O'Hare, G. and Jennings, N. (eds), Wiley Inter-Science, (to appear).

Brahim Chaib-draa is an associate professor of Computer Science at Laval University. His research and teaching focus on agents, multi-agent environments, speech acts and

computational discourse, and formal logics. He has published papers on these aspects in journals and conferences. He earned a Ph.D. in distributed intelligent systems and has held academic and industrial positions in France and in Africa. Email: chaib@iad.ift.ulaval.ca

The The Agent Building Shell: A Tool for Building Enterprise Multi-Agent Systems

Continued from page 11

logics, knowledge acquisition, generic problem-solving models, design modeling, and others, creating problem solving systems and languages and applying them to practical problems. He can be contacted at the Enterprise Integration Laboratory, University of Toronto, 4 Taddle Creek Road, Rosebrugh Building, Toronto, Ontario, M5S 3G9
Email: mihai@ie.utoronto.ca

Mark Fox received his B.Sc. in Computer Science from the University of Toronto in 1975 and his Ph.D. in Computer Science from Carnegie Mellon University in 1983. In 1991, Dr. Fox returned to the University of Toronto where he received the NSERC Research Chair in Enterprise Integration and was appointed Professor of Industrial Engineering, Computer Science, and Management Science. In 1992, he was appointed Director of the Collaborative Program in Integrated Manufacturing. Dr. Fox was elected a Fellow of American Association for Artificial Intelligence in 1991, and a Joint Fellow of the Canadian Institute for Advanced Research and PRECARN in 1992.

Contact: Tel: 416-929-4283, fax: 416-944-3276

Email: msf@novator.com, <http://www.novator.com/novator/>

Submissions to Canadian Artificial Intelligence

Canadian Artificial Intelligence welcomes submissions on any matter related to artificial intelligence.

Please send your contribution, electronic preferred, with an abstract, and a short bio to:

Sue Abu-Hakima, Institute for Information Technology
Editor, *Canadian Artificial Intelligence*
Institute for Information Technology
National Research Council Canada
M-50 Montreal Road
Ottawa, Ontario, Canada
K1A 0R6 or — suhayya@ai.iit.nrc.ca
Telephone (613) 993-8564 Fax (613) 952-7151

Book reviews and candidate books to review should be sent to:

Graeme Hirst
Canadian Artificial Intelligence
Department of Computer Science
University of Toronto
Toronto, Ontario, Canada
M5S 1A4 or — gh@ai.toronto.edu

Intelligent Agent Structures for the Internet

Leslie L. Daigle and Sima Newell

Résumé

Dans un environnement largement distribué d'informations hétérogènes et de calcul informatique tel que l'internet, un logiciel basé sur des agents peut incorporer des principes d'intelligence artificielle à deux niveaux: à l'intérieur d'agents individuels, ou par la participation d'agents de base dans un ensemble intelligent. Cet article décrit avec plus de détails les concepts de construction de systèmes d'information intelligents dans l'internet grâce à l'utilisation de la technologie URA (Uniform Resource Agent), et présente quelques travaux préliminaires pour décrire effectivement les capacités des agents.

Abstract

In a widely distributed, heterogeneous information resource and computing environment such as the Internet, agent software can embody principles of artificial intelligence at two levels – within individual agent objects, or through the participation of basic agents in an intelligent collective.

This paper describes more fully the concepts of building intelligent information systems in the Internet through the use of the URA technology, and introduces some preliminary work done on effectively describing agent capabilities.

Introduction

Work on a "Uniform Resource Agent" (URA) framework has defined a virtual object structure in which to specify an Internet information activity in terms of components for specifying input requirements, target resources, experience data, activity script, response filter script, as well as general meta-data about the URA. A URA is a formalization of an activity. These agent objects may range in level of "intelligence," depending on the scripting code used to carry out the activity, as well as the use made of experience and other environment information.

The standard URA object methods provide a mechanism for a client (human, software, or other agent) to learn about a particular URA it has never encountered before, and invoke it, without having to know anything about how the URA carries out its task. Thus, URAs become manageable "black boxes" that can participate in intelligent collectives to carry out Internet activities.

However, the ability to participate in collectives is less than useful if a constructor cannot find relevant component agent objects. That is, given a set of URAs, it is not trivial to find the appropriate one to which a given task should be delegated. As a starting point, it would seem that users with similar backgrounds and interests should be able to share their "hot lists" of Internet activities, and that once someone has sifted through irrelevant information, another user with a similar background should be spared the same task. An intelligent system must therefore incorporate a way to store information about the user, available agents, and

correspondences between user needs and agent abilities. Some preliminary work in this area is discussed below.

URAs and Intelligent Internet Behaviour

A URA, completely instantiated, embodies the necessary skills to contact one or more (Internet) information resources and combine the results in an answer to the URA client. So, for example, someone who knows where all the net's cool music sites are might write a URA to search through them and return pointers to music files. That URA then embodies a certain set of skills — finding music files on the net — and can perform that task for any client. The URA is a procedural representation of some information resource knowledge.

A more complex URA might perform some post-processing on the results returned from the information resources — rather than simply returning hits from a search (formatted one way or another), the URA could use the information to reason about things ("There are no new music files since you last checked," etc).

Since URAs execute within the client's computing resource space (i.e., on their machine, or at some site that has been provided for URAs), they are reflective of the client, not any information service. A URA's history is tied to its involvement with a particular user, and that history information can be used to allow the URA to "learn," or adapt its behaviour to the client's preferences.

In principle (not yet in the prototypical implementations that have been developed — see [4]), URAs can call on other URAs to carry out their specialized tasks. This yields a structure of agents working together to carry out a larger conceptual task than any of the agents can handle themselves. The URAs are building blocks for complex Internet information activities.

The ability to construct such collectivities of activity is increasingly important as the Internet information space evolves. This space must permit the customization of views of its content. Individual users must be able to tailor their interactions with the Internet to support their work tasks. In the same way that retrieving a file has become transparent to users who click on a WWW link anchor, searching for information must be supportable as a component part of a composite task.

This can be supported in an interaction model that has three primary components: an information need, an information processing task or activity, and information resources. That is, a client's information need can be satisfied by some processing task which will draw on specific resources. In the current Internet environment, the bulk of the processing associated with satisfying a particular need is embedded in software applications (such as WWW browsers), and these applications access information resources directly through a fixed set of communications

protocol standards. The potential danger is that this puts a barrier between clients and resources, because the language of communication is generalized to support so many possible applications that it cannot support any one deeply.

Rather than falling into this end-heavy distribution of information activities, URAs can act as standardized mechanisms for encapsulating client information processing tasks. In this way, client information processing can be delegated to a task defined outside the realm of a particular application. There are activities that are client-centered, not resource-centered, and for these it is not appropriate for it to be buried in information servers.

While the URA technology itself only provides a framework within which agents can be built, and the degree of "intelligence" associated with each agent is at the discretion of its creator, URAs can be used to support very sophisticated Internet information tasks.

Matching Agents and Users — Preliminary work

Given a set of agents that perform Internet tasks, it is a challenge to locate those that are of interest to a given user. Suppose, for example, that Keith is a musicologist specializing in the work of Vivaldi, and Sally enjoys listening to Vivaldi's music. Although the keyword descriptors for Keith and Sally's interests in this area are identical (music, classical, Vivaldi), their information needs are probably quite different. Similarly, if Sally has created a URA that seeks (music, classical, Vivaldi)-related information from the Internet, and then filters it to her liking, this URA will differ from one which Keith has built. How then, can we better characterize users and their information needs? And what data is needed to describe a URA such that users with similar characteristics can find those agents of interest?

Modeling Users

The problem in the (music, classical, Vivaldi) model of Keith and Sally's interests is that these keywords provide no information about the reasons for their interests. But these reasons, which stem from a user's experience, are different in Keith and Sally's cases. Essentially, the list of words (music, classical, Vivaldi) does not adequately model either of the two English sentences originally used to describe each user. It thus stands to reason that a simple set of keywords does not provide enough information about the user's background to find agents (and thus data) of interest.

Asking a user to provide topic-related keywords is relatively easy. Modeling a user's background is not quite as simple. In an effort to develop a useful working model, we have constructed a World Wide Web (WWW) form-based survey [1] of users. Part I of this survey is oriented towards modeling users' backgrounds and certain of their interests. Its format [2][3] is based on the standard user profile used in business: the curriculum vitae.

However, the questions are posed in a very particular way. For example, in describing their employment, users are asked to provide a career area, job function, and duration

of employment for each job held. These three items are considered to be orthogonal descriptors of the user's job. That is, the information obtained in one category (e.g., career-area = engineering) is independent of that gathered from the other two (e.g., job-function = manager and duration = five-years).

Finding orthogonal descriptors for each aspect of the user's background will ultimately facilitate the comparison of different users. Currently, the choices for each descriptor are limited to allow comparisons to be made between a relatively small number of users.

URA Meta-Information

Part II of the survey focuses on describing search topics of interest to a user. The survey questions relate to the topic name and associated keywords, whether the user has a broad interest in the topic or is only interested in very specific related areas, and how concerned a user would be about the privacy of his search in this area. (No mention is made in the survey of using an agent as the basis of the search mechanism, but this is the underlying theory.)

Unfortunately, we know that a topic and keywords are not sufficient to describe the function of an agent. This inadequacy may be addressed by associating the creator's experiences and background with the URA. This combination of topics, keywords, and user background form a set of meta-information about the agent.


Again, we have chosen orthogonal descriptors for each topic: its name, technical level, scope, and privacy level [2]. The technical level describes the depth of knowledge needed in the topic area; the scope provides a measure of breadth. Here, "technical" does not imply "scientific": Keith's technical level with respect to Vivaldi's music is certainly higher than Sally's.

A URA's creator may explicitly decide the values of the four descriptors. If not, their initial values may be inferred from the creator's profile. In addition, many topics and sub-topics could be associated with an agent. Each of these should have explicit or inferred initial values for the four descriptors.

Using Meta-Information and the User Profile to locate URAs

Given a set of users with certain profile characteristics, and a set of agents with appropriate meta-information descriptors, it becomes possible to create an intelligent system to return a set of URAs that are of interest to any particular user. With the addition of relevance feedback, the system could progressively fine-tune a user's profile, and thus learn to provide URAs of greater interest. The key is that a finite set of orthogonal descriptors is used as URA meta-information, and that a similar set is associated with each topic of interest to a user. If these descriptors are carefully chosen, the problem of finding information of interest in the Internet may be more effectively resolved.

References

- [1] S. Newell, L. Daigle, "Internet Activity Survey Form," <<http://www.bunyip.com/urasurvey.html>>, Montreal, 1996.
- [2] L. Daigle, S. Newell, "Intelligent Agents and the Internet Information Infrastructure," *INET'96*, Montreal, 1996.
- [3] E. Rich, "Stereotypes and User Modeling," in *User Models in Dialog Systems*, Springer-Verlag, New York, 1989.
- [4] The Silk Project homepage, <<http://www.bunyip.com/products/silk/>>. 

Sima Newell holds a B.Eng. in Electrical Engineering from McGill University (Montreal, Canada) and is currently pursuing her Master's degree in Expert Systems in the same department. She has been working at Bunyip Information Systems Inc. since Spring 1995, first in Internet Operational Services, and then in her present capacity as a researcher. Her work involves exploring Internet information systems; she is currently designing a system architecture that uses user models and intelligent agents for electronic information retrieval.

Email: sima@bunyip.com

Leslie L. Daigle holds a B.Sc. in Mathematics and Computer Science from McGill University, and an M.Sc. in Computing and Information Science from the University of Guelph. She has worked with Bunyip Information Systems, Inc., since 1993. As the Project Manager for Bunyip's Desktop Internet Resource Discovery Client, Silk, Leslie was the principal researcher developing the Uniform Resource Agent (URA) technology. She is active in the Internet Engineering Task Force, particularly in the areas of UR and Directory Services development. Now Vice President, Research at Bunyip, Leslie is ever hopeful of pursuing her Ph.D. at McGill in the area of multimedia retrieval systems, and never misses an opportunity to argue the short-comings of text-only retrieval technology.*

Email: leslie@bunyip.com

BIG NEWS COMES IN SMALL PACKAGES

*Join CSCSI/SCEIO and receive
Canadian Artificial Intelligence magazine*

See page 38 for membership information



**Canadian
Artificial
Intelligence**

**Intelligence
Artificielle
au Canada**

Highly Autonomous Sensor Models

Fernando Figueroa

Résumé

Une théorie formelle pour le développement d'un modèle générique de capteur autonome est proposée et implantée. On peut considérer ce modèle comme recouvrant un champ assez spécifique des modèles d'agents autonomes. L'aspect le plus intrigant de ce travail est qu'il est centré sur le capteur lui-même, alors que les agents autonomes considèrent que le système d'acquisition sensoriel est complet et aussi précis qu'il est nécessaire. La construction de systèmes intelligents en ingénierie implique l'enchâssement des capacités de l'opérateur dans le modèle. Ces capacités ne consistent pas seulement de règles et de faits, mais aussi de méthodologies de raisonnement et de prise de décision. Nous présentons donc la notion selon laquelle un capteur peut-être vu comme un agent autonome et nous le désignons hautement autonome, puisqu'aucun système ne peut-être tout à fait autonome. Non seulement un capteur hautement autonome interprète-t-il les données acquises conformément à une base de connaissances intégrée sous forme de système expert, mais il est aussi capable d'apprendre et de ce fait, d'améliorer ses performances avec le temps. L'idée derrière le modèle est de combiner les capacités du capteur physique et un opérateur expert contrôlant le capteur en temps réel. Le modèle est générique et peut-être utilisé pour modéliser n'importe quel capteur comme un capteur hautement autonome. Le but de notre travail est de créer des capteurs fiables, précis, qui peuvent installer eux-mêmes, se calibrer eux-mêmes, et qui fournissent une information complète. Dans les industries de procédés, dans les navires, dans les usines de production d'électricité, et dans bien d'autres, où l'on trouve des systèmes automatiques qui font un usage intensif de capteurs, une large proportion des budgets est dépensée pour l'installation, la calibration et l'entretien des capteurs. Avec le modèle de capteur hautement autonome, on veut diminuer ces dépenses sans compromettre l'intégrité de l'information fournie par les capteurs.

Abstract

A formal theory for the development of a generic model of an autonomous sensor is proposed and implemented. This model can be viewed as addressing a somewhat specific area of autonomous agent models. Perhaps the most intriguing aspect of this work is that it concentrates on the sensor itself whereas autonomous agents assume that the sensory input is complete and accurate to the degree that it needs to be. Building intelligent systems in engineering implies embedding the capabilities of the system operator in the model. These capabilities are not simply rules and facts, but also reasoning and decision making methodologies. So we present the notion of considering a sensor as an autonomous agent itself and call it highly autonomous, since

no system can be completely autonomous. A highly autonomous sensor (HAS) not only interprets the acquired data in accordance with an embedded expert system knowledge base, but is also capable of learning and thereby improving its performance over time. The inspiration for the model is to combine the capabilities of the physical sensor and an expert operator monitoring the sensor in real-time. The model is generic and can be used to model any sensor as a HAS. The motivation for this work is to create sensors that are self-installing, self-calibrating, reliable, accurate, and that provide complete information. Many sensor intensive automated systems such as process industries, ships, power plants, and others spend a sizable portion of their budgets in installation, calibration, and maintenance of sensors. The HAS model is intended to decrease these expenses without compromising the integrity of the information the sensors provide.

Introduction

Development of autonomous agent models has generally been linked with autonomous vehicles that must navigate unknown terrains avoiding static and moving obstacles as they try to reach a desired destination. The literature related to autonomous agents is quite extensive, resides primarily in computer science circles, and generally deals with two issues: the agent and the simulated environment where the agent navigates. In engineering, a large number of systems having "intelligence" are being developed, e.g., intelligent controllers, automatic systems modelers and designers, etc. Whereas autonomous agent models are more comprehensive in their attempt to imitate humans' autonomous operation, engineering "intelligent" systems are normally associated with a defined area of knowledge (are not comprehensive) and can be implemented with less hardware and software resources. Since autonomous agents operate in an all encompassing world, their knowledge domain can not be very detailed, limited by lack of resources (hardware and software). However, more recent autonomous agent models include tools that allow the agent to operate in more focused knowledge domains with increased detail. Given these developments, methods and architectures developed to model autonomous agents are now suitable to model engineering "intelligent" systems. Often, however, the use of autonomous agent models in engineering may require some modifications, further extensions, and/or new interpretations.

The HAS model was developed in collaboration with my colleague Dr. Ajay Mahajan. Detailed descriptions of the model can be found in refereed publications [Mahajan (1994), Mahajan and Figueroa (1995), Figueroa and Mahajan (1994)]. We have developed a generic structure and

methodology to model any sensor as highly autonomous. A HAS will be described as an autonomous agent that operates in the environment where it senses the value of a parameter of interest. For example, a thermistor operates in a room and it senses the value of the temperature. Although it would be proper to include a literature survey on "intelligent" sensors, given the space limitations, I refer the reader to our recent publications where adequate references are provided.

Description of the Model

The HAS model operates in both deliberative and reactive modes [Ferguson (1995), Brooks (1986, 1991)]. In the reactive mode, it cleans in real-time the raw data from spurious noise such as large spikes that are easily identified and eliminated. In the deliberative mode, it uses the data from the Sensing Element to (1) extract qualitative behaviors of the sensor and the measurand, (2) determine trends, (3) reason about the trends determined to provide a basis for maintenance, modification and evolution of the knowledge data bases (learning), and to trigger a hard calibration when needed.

The HAS has the following components: a Sensing Element that provides the raw signal; a Sensor Knowledge Base that contains quantitative data such as the sensor operating specifications (sensitivity, conversion constant, sensor time constant, spike behaviors due to foreseeable perturbations, frequency response, etc.); numeric data such as tables for drift, sensitivity and response plots, etc.; and qualitative data such as that found in the manuals indicating conditions and procedures one must observe to insure proper operation of the sensor. This component also includes statistical and analytical methods to perform the following functions:

- (1) immediate filtering of the raw data and transfer of the data to the control system for utilization,
- (2) extraction of qualitative sensor and measurand behaviors from the raw data by a combination of statistical and heuristic techniques,
- (3) quantitative interpretation of the data in the form of a representative value.

The Sensor Knowledge Base ultimately provides the accurate, reliable, and complete value describing the parameter that the sensor is measuring. A Measurand Knowledge Base contains specifications of the measurand such as its time response constant, and a plausible set of qualitative behaviors. These behaviors might be, for example: constant with noise, monotonic increase, step change at the appropriate speed, harmonic at defined frequencies, etc. The sensor and measurand qualitative behaviors are treated as Envisioned Behaviors and guide the analysis of the qualitative version of the measured data. Once the measured data has been identified with one of the Envisioned Behaviors, then a quantitative analysis can be performed to diagnose and calibrate the system. For example, if the measurand behavior is matched with an envisioned step change behavior, one could use the step change to measure

the measurand time-response and update the Measurand Knowledge Base. In another example, if a monotonic decrease behavior has been identified, one could then try to determine if this is actually a sensor drift behavior. One way to determine drift may be by comparing the monotonic decrease with drift tables in the Sensor Knowledge Base, and examine if the conditions that cause drift (e.g., aging, temperature changes) have indeed occurred. Another component of the HAS model is the Learning module. This module updates the knowledge bases in the system when new behaviors have been discovered and/or existing behaviors have been modified. The current technique to discover new behaviors is to notice if behaviors not yet in the knowledge bases occur a number of times. If this is the case, then these become candidates to be included in the HAS model. It is possible to run the sensor in a self-installation mode by letting the Learning module discover all the knowledge in the system given only a so called "normal behavior" that it uses as reference. The "normal behavior" is "constant with noise." A Maintenance module updates the knowledge in the system such as changing the sensor or measurand parameters as a result of analysis of the data (e.g., step-change, drift, etc.) Finally, a Hard Calibration module calibrates the sensor numeric specifications by connecting a known input. It is usually difficult to physically have this module as part of the sensor, and Hard Calibration usually implies the need for intervention by the operator.

The HAS as an Autonomous Agent

Given the prior description, it is important to relate the HAS model to accepted autonomous agent models. The HAS exhibits behaviors much like Ferguson's TouringMachine [Ferguson (1995)]. It displays deliberative and non-deliberative behavior. It is capable of autonomous operation. That is, it has its own goals (interpret data accurately, reliably, and completely), and it has a repertoire of behaviors (knowledge bases) to carry out its goals and to respond to environmental conditions. In addition, it has a goal that apparently is not present in any autonomous agent. The HAS must also evolve with experience in terms of modeling its own behavior in addition to the environment's behavior. This goal would be equivalent to having a TouringMachine improve its own model so that it may self-predict its own responses to stimuli.

Since the HAS model is generic, it is appropriate to relate it to Gensim, a simulation environment developed by Anderson and Evans (1995). This is a simulation environment with a collection of facilities so that the user may build complete environments to design and test autonomous agents. The high level structure of Gensim has the same components of the HAS. Actions in Gensim are carried out by the agent and its effects are manifested by the simulator. In the HAS, actions are executed to modify knowledge bases of the sensor and the environment, as well as to modify the incoming data so as to extract an accurate and complete interpretation of the environmental parameter being measured. This is

done by substituting faulty data values by extrapolated values predicted based on the current trend. The focus function in Gensim allows the agent to focus on certain aspects of the environment. (The agent informs the simulator of its interests in some objects in the environment.) This function is performed by the HAS as it focuses on envisioned behaviors of the measurand. Moreover, the HAS also focuses on envisioned sensor behaviors since it must also interpret the condition of the sensor. The perceptual information in the HAS is acquired by the sensor element. And the domain knowledge in the HAS encompasses sensor, operator, and measurand knowledge domains with some shared knowledge among them.

At the operation level, Gensim provides tools for reasoning with the time element. In Gensim, the agent can be interrupted every time interval P is to be presented with sensory information. This is similar to the sampling time in the HAS model where data from the sensors is read at every sample time. The sampling interval is defined as the smallest time increment to time-stamp events, and other time-related behaviors are defined as a function of the sampling interval. Two time-related behaviors in Gensim include the rate at which an agent commits to actions (A), and the rate at which changes are made to the environment (E). In the HAS models these elements are very important, and they correspond to the sensor's response time constant (similar to A) and the environment's response time constant (similar to E).


The HAS model includes a learning module which builds upon existing sensor and measurand behaviors to discover new behaviors or modify/fine-tune existing behaviors. This activity results in continuous self-evolution of the sensor and measurand models. The Gensim environment also appears to provide the tools necessary to implement learning about the environment. Anderson and Evans indicate that, "Perception must exist to confirm the agent's expectations of the world, but also must provide the agent with new information (not necessarily what it expects to see or is directed toward) in order to form the expectations of the world that guide its sensory abilities." Thus the agent is given incentive to explore objects in the world.

Future Work

Another aspect of intelligent behavior addressed in Gensim is that of fusing information from various sensors to reinforce perception. The HAS does not yet have communications capabilities built in, but future work will focus on sensor fusion and integration associated with a network of HASs. Currently, communications protocol standards for sensors networks are being developed (Fieldbus), but modifications will be needed to take full advantage of the more complete information HASs provide.

References

Anderson, J., and Evans, M., "A Generic Simulation System for Intelligent Agent Designs," *Applied Artificial*

- Intelligence*, V. 9, N. 5, October, 1995, pp. 527-562.
- Brooks, R. A., "Intelligence without representation," *Artificial Intelligence*, Vol. 47, 1991, pp. 139-159.
- Brooks, R. A., "A Robust Layered Control System for a mobile robot," *IEEE Journal of Robotics and Automation*, Vol. 2, 1986, pp.14-23.
- Ferguson, I., "Integrating Models and Behaviors in Autonomous Agents: Some Lessons on Action Control," *Proc. AAAI Spring Symposium on Lessons Learned from Implemented Software Architectures for Physical Agents*, Palo Alto, CA, March, 1995.
- Figueroa, F. and Mahajan, A., "Generic Model of an Autonomous Sensor," *Mechatronics*, Vol. 4, No. 3, 1994, pp. 295-315.
- Mahajan, A., "Dynamic Across Time Autonomous - Sensing, Interpretation, Model Learning, and Maintenance Theory," Ph.D. Thesis, Mechanical Engineering Dept., Tulane University, New Orleans, LA, USA, May 1994.
- Mahajan, A, and Figueroa, F, "Dynamic Across Time Autonomous - Sensing, Interpretation, Model Learning, and Maintenance Theory," *Mechatronics*, Vol. 5, No. 6, 1995, pp. 665-693. 

Fernando Figueroa was born in Peru. He received a Bachelor's Degree in Mechanical Engineering from the University of Hartford, Hartford, Connecticut, USA (1983), a M.S. Degree (1984) and a Ph.D. Degree (1988) in Mechanical Engineering from Pennsylvania State University. Since 1988, he has been with the Faculty of Mechanical Engineering at Tulane University. Since 1995, he has held the NSERC/Monenco AGRA Associate Chair of Instrumentation and Control, at the University of New Brunswick, Mechanical Engineering Department. He worked at NASA Johnson Space Center in Houston as a Summer Faculty Fellow (1993, 1994). His areas of interest include intelligent/autonomous sensors/instrumentation and systems, robotics, mobile robots, ultrasonic sensing, and automatic control. Contact Information: Tel: (506)447-3097, Fax: (506)453-5025, Email: figueroa@unb.ca

Distributed Agent Systems for Intelligent Manufacturing

Continued from page 33

Systems and, in particular, in Multi-Agent Applications in Manufacturing. Contact: <http://www.ucalgary.ca/~norrie/>

Dr. Brian R. Gaines is Killam Memorial Research Professor and Director of the Knowledge Science Institute at the University of Calgary. He received his B.A., M.A. and Ph.D. from Trinity College, Cambridge, and is a Chartered Engineer, Chartered Psychologist, and a Fellow of the Institution of Electrical Engineers, the British Computer Society and the British Psychological Society. Contact: gaines@cpsc.ucalgary.ca 403-220-5901 Fax: 403-284-4707 <http://ksi.cpsc.ucalgary.ca/KSI>

LALO: a new Agent Oriented Programming Language and Environment

Daniel Gauvin, Hervé Marchal, Carlos Saldanha

Résumé

Dernièrement, la technologie des agents logiciels a été couverte de façon importante par la presse scientifique et même par la presse populaire. On retrouve les agents dans des champs divers tels que le génie logiciel, la gestion de réseau, les systèmes simultanés, la robotique, les interfaces humain-ordinateur et l'intelligence artificielle. Malgré ses utilisations diverses, la notion de ce qu'est un agent varie d'un champ à un autre et même d'un système à un autre.

Dans cet article, nous adhérons à la définition commune donnée par la communauté scientifique de l'intelligence artificielle distribuée (IAD). Un agent est considéré comme une entité autonome capable de décisions et d'actions rationnelles. Récemment, Yoav Shoham [Sho 93] a proposé le paradigme de programmation orientée agent (POA) comme une spécialisation du paradigme de programmation orientée objet, et a développé une implantation sous la forme du langage Agent-0. Depuis, plusieurs équipes de recherche ont proposé leurs propres langage POA (par exemple, voir Agent-k [Dav 94], Congolog [Les 95], Cool [Kol 95]). Dans cet article, nous décrivons LALO, acronyme français pour Langage d'Agent Logiciel Objet, construit au CRIM (Centre de recherche informatique de Montréal), qui diffère de façon significative des langages mentionnés plus haut.

1. Introduction

Lately, software agent technology has received extensive coverage by the scientific and even the popular press. Agent concepts are mentioned in diverse fields such as software engineering, network management, concurrent systems, robotics, human/machine interfaces, and artificial intelligence. In spite of its diverse uses, the notion of what an agent is varies from area-to-area and even from system-to-system.

In this paper we adhere to the common definition provided by the scientific community in Distributed Artificial Intelligence (DAI). An agent is seen as an autonomous entity capable of rational decisions and actions. Recently, Yoav Shoham [Sho 93] proposed the agent-oriented programming (AOP) paradigm as a specialization of the object-oriented programming paradigm and developed an implementation in the form of the Agent-0 language. Since then, many research teams proposed their own AOP language (see, for example, Agent-k [Dav 94], Congolog [Les 95], Cool [Kol 95]). In this paper, we describe LALO, a French acronym for Object Software Agent Language being constructed at CRIM (Centre de recherche informatique de Montréal) which differs significantly from the above mentioned languages.

The major differences are related to the way agent behavior is specified and to the software architecture and implementation. LALO does not impose any programming restrictions on the specification of agent behavior. Depending on the application, the developer chooses the most appropriate computational method. For example, behavior can be specified as a set of rules, a decision table, a neural network, or even as an algorithmic procedure. The programmer is given the freedom to specialize application agents by associating behavior methods with new agent classes. As a result, the new agents are better suited to the area of interest than the generic agent classes available in LALO. Another important characteristic of LALO is that it is a compiled rather than an interpreted language. This feature was designed to overcome the inadequacy of interpreters in handling certain types of applications such as those requiring real-time processing.

Section 2 of the paper presents a brief review of the AOP paradigm. It is followed by a section describing the LALO language and the corresponding programming framework. Finally, a small example is given in Section 4.

2. Agent Oriented Programming

Agent Oriented Programming proposes a collaborative view of computing where agents exchange information, query other agents, offer services, accept or reject tasks, and compete or cooperate in accomplishing tasks. In this context, an agent is defined as an entity having internal states categorized as beliefs, capabilities, decisions, and commitments. Collectively, their status describes an agent's mental state, which changes over time. As such, time is an essential agent parameter used in monitoring and controlling mental activity.

Beliefs represent an agent's perception of the world in which it interacts. Capabilities relate to the actions an agent can perform. An agent will take the commitment or decide to execute an action in relation to its beliefs and its capabilities. The relation between an agent's mental state and its capabilities determines the agent behavior in response to changes in the environment (including time). Agent behavior is essentially the sequence of actions it performs. However, prior to performing the action, there must be a commitment or decision to act. The mental state influences both the commitments made and the substance of those commitments (i.e., how actions are realized).

3. The LALO Environment

This section presents the main components of the LALO

agent-oriented programming environment. We first describe the main components of the environment followed by a presentation of the LALO AOP language.

3.1. Main components

The LALO programming environment consists of several C++ libraries, a compiler, and some utility programs. The architecture is based on a hierarchy of agent classes. A LALO agent is one instance of one particular class in the hierarchy. The class of a particular agent can be specified in the LALO program and its choice depends on the programming technique used to specify its behavior.

At the top of the hierarchy is the class `BasicAgent` which implements the communication mechanisms. The main components of a `BasicAgent` are: an object of the class `InRouter` to handle incoming messages, an object of the class `OutRouter` to handle outgoing messages and an object of the class `InputBuffer` containing received messages. When a `BasicAgent` starts its execution loop, it creates an `InRouter` and an `OutRouter`. In UNIX, each starts a child process, `AFW_in_process` and `AFW_out_process` respectively, in order to implement true asynchronous communication. After the successful creation of the two child processes, the agent enters a wait state for new messages. When a new message is received, the agent places it in the input buffer. The messages in the input buffer are handled in a first in first out order. When the input buffer is empty, the agent returns to his wait state.

The AOP paradigm is implemented by the `LaloAgent` class which specializes `BasicAgent` by adding specific objects in order to represent its beliefs, capabilities, commitments, and decisions. The execution loop is overloaded in order to take into account the execution of commitments and decisions.

The `RBasedAgent` class further specializes `LaloAgent` by adding three different rule bases and inference engines. One rule base handles the incoming messages, the second the execution of the commitments and decisions, and finally the third is a general reasoning rule base. The execution loop inherited from `LaloAgent` is overloaded in order to incorporate a rule based reasoning mechanism in the behavior of the agent. This class implements a particular way to define the behavior, based on a rule based paradigm and illustrates how the basic class hierarchy can be extended in order to implement new reasoning mechanisms. The agents can communicate by using messages in the KQML ([Fin 94]), and HTTP protocols. The default behavior implemented in the classes described above can handle most of the messages defined in those two protocols. Furthermore, the architecture allows the user to implement other protocols if necessary. The compiler translates a LALO program into a C++ source file that can be compiled with any C++ compiler. It is also used by the agents themselves (those that inherit from `LaloAgent`) to compile dynamically the contents of KQML messages that are in the LALO language.

3.2. The LALO AOP language

LALO is a compiled language based on the concepts of agent beliefs, capabilities, decisions and commitments defined in the previous sections. The basic elements of the language are beliefs and tasks, two concepts which depend on time. The time associated with each belief or task is assigned by preceding the action or fact with a temporal operator. For example, the temporal operator `BEGIN_AT` can be used to indicate when a task should start. Similarly, the temporal operator `AT` can be used to indicate at which time a belief holds.

Agent communication actions are based on KQML which is a language employing a flexible protocol for exchanging messages between agents. A KQML message consists of a performative and any number of arguments. The performative conveys the meaning of the message; however, the content of the message can be in any format. KQML defines a number of reserved performatives along with their corresponding arguments. In LALO, there is a predefined communication action for each KQML performative. When a communication action is executed by an agent, an appropriate KQML message is formed and sent to the receiver.

A LALO agent program is divided into five parts; namely, the identification, the task declarations, the initial decisions, the initial beliefs, and the behavior. In identification, the author of the program (both name and email address) and the agent class and name is given. The task declarations component describes the capabilities of the agent. Initial decisions correspond to the tasks an agent will execute when it is created. As such, a decision is a special kind of commitment where the parameter specifying the agent committed to is ignored (the agent is committed to itself). The next part of a LALO program describes an agent's initial beliefs.

Finally, the last part of the LALO program defines the behavior of the agent. This must be done in accordance with the reasoning mechanism used by the agent class specified by the developer. The programming environment is designed to provide a library of agent behavior programming methods. For example, a class of rule-based agents will employ a rule syntax for defining behavior, and an inference mechanism capable of interpreting the rules. In the contrast, a plan-based agent class will require agent behavior to be specified as a series of plans, while a plans-hierarchy reasoning mechanism will activate those plans at run-time.

4. Example

In order to illustrate our language, the following presents a small LALO program:

```
AUTHOR: Hervé Marchal;  
EMAIL: hmarchal@crim.ca;  
AGENT_CLASS: RBasedAgent;  
AGENT_NAME: Daniel;
```


TASKS:

PUBLIC COMPOSITE promote-yourself(to);

DECISIONS:

BEGIN_AT #begin: promote-yourself(to:Carlos);

BELIEFS:

AT #begin: my-friend(agent: Carlos);

BEHAVIORS:

// (R1) If an agent asks me to tell a third agent that a fact is true

// and I believe that it is true, I commit to the requesting agent

IF

RECEIVED:

achieve(sender: ?ag1, content:

BEGIN_AT ?t1: tell(receiver: ?ag2, content: AT ?t2: ?f);

BELIEF: AT ?t2: ?f;

THEN

COMMITMENT_TO ?ag1:

BEGIN_AT ?t1: tell(receiver: ?ag2, content: AT ?t2: ?f);

// (R2) If I believe that it is false, I do not commit

IF

RECEIVED:

achieve(sender: ?ag1, content:

BEGIN_AT ?t1: tell(receiver: ?ag2, content: AT ?t2: ?f));

BELIEF: AT ?t2 : NOT ?f;

THEN

COMMITMENT_TO #myself:

BEGIN_AT #now:

sorry(receiver: ?ag1, in-reply-to: ?current_message,

comment: "I don't believe in that");

// (R3) If I don't know whether the requesting agent is my friend,

// I will believe that the fact is true and take the commitment

IF

RECEIVED:

achieve(sender: ?ag1, content:

BEGIN_AT ?t1: tell(receiver: ?ag2, content: AT ?t2: ?f));

UNBELIEF: AT ?t2 : ?f;

BELIEF : AT_NOW : my-friend(agent : ?ag1);

THEN

BELIEF: AT ?t2 : ?f;

COMMITMENT_TO ?ag1:

BEGIN_AT ?t1: tell(receiver: ?ag2, content: AT ?t2: ?f);

5. Conclusion

An agent-oriented programming environment has been described in this paper. The main characteristics include a compiled language and a flexible mechanism for defining the behavior of individual agents in problem-solving. We

are currently testing the environment. Some extensions and modifications that we are planning include a more elaborate handling of time in order to allow one to specify repetitive tasks and to adjust the time grain. More information on LALO is available from our web site at <http://www.crim.ca/gbsc/lalo>.

References

- [Dav 94] Davies, W. H. E., Edwards, P. Agent-K: An Integration of AOP and KQML. *Proceedings of the CIKM '94 Workshop on Intelligent Information Agents*, Y. Labron & T. Finin (eds), National Institute of Standards and Technology, Gaithersburg, Maryland, USA, 1994.
- [Fin 94] Finin, T., Fritzson, R., McKay, D., McEntire, R. KQML as an Agent Communication Language. *The Proceedings of the Third International Conference on Information and Knowledge Management (CIKM '94)*, ACM Press, November, 1994.
- [Kol 95] Kolb, M. A cooperation language. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS 95)*, San Francisco, California, USA, June 12-14 1995. AAAI Press.
- [Les 95] Lesperance, Y., Levesque, H. J., Lin, F., Marcu, D., Reiter, R., Scherl, R. B. Fondements d'une approche logique à la programmation d'agents. In *Proceedings of the Troisième Journées Francophones sur le Artificielle Distribuée & les Systèmes Multi-Agents (IADSMA 95)*, St Baldoph, Savoie, France, 15-17 Mars 1995.
- [Sho 93] Shoham, Y. Agent-oriented programming. *Artificial Intelligence*, 60(1), pages 51-92, 1993.
- Daniel Gauvin is research advisor at the Centre de Recherche Informatique de Montréal (CRIM). He joined the CRIM in 1990 after he completed a master degree in computer science at the University of Montréal. Since then he has participated in several knowledge-based systems projects and is now lead investigator in CRIM's multi-agent systems development environment called LALO. E-mail: gauvin@crim.ca.*

Hervé Marchal is senior research agent at the CRIM. After completing a DESS S.C.H.M. at Orsay University (France) in 1989, he joined CRIM in 1994 and works in conjunction with D.Gauvin on LALO.

Carlos Saldanha is currently the director of the Knowledge-based Systems group at CRIM and continues to play the role of lead researcher in the group, a position he has held since 1992.



The SIGMA Project: Market-Based Agents for Intelligent Information Access

Grigoris J. Karakoulas

This project is the result of a joint effort with the Interactive Information Group of the National Research Council of Canada. We especially thank Innes Ferguson, Martin Brooks, and Dale Schuurmans.

Introduction

Comme l'espace de l'information sauvegardée électroniquement continue de s'étendre à travers les réseaux d'ordinateurs, le besoin d'un accès intelligent à l'information à l'intérieur d'espaces multidimensionnels, partiellement structurés et bruyants, est impératif. Comme exemples de tels espaces, on peut mentionner le Usenet, les sources de nouvelles, le WWW et les dépôts d'information d'entreprise. Le problème de concevoir un système de filtrage d'information (FI) représente un défi, à cause de la volatilité de ces espaces et de la capacité limitée des usagers à spécifier des intérêts globaux dans un champ fondamentalement incertain de tels espaces. De plus, la distribution et la dynamique inhérentes des réseaux d'information actuels tendraient naturellement à suggérer une approche robuste, décentralisée et multi-agents à la conception d'un système de FI.

Le but de notre travail dans SIGMA (acronyme anglais pour System of Information Gathering Market-Based Agents – Système d'agents basés sur le marché pour la collecte d'information) est de développer une structure d'agents logiciels hétérogènes et asynchrones pour gérer la recherche d'information dans un environnement de réseau largement distribué; d'effectuer un indexage automatisé d'éléments d'information semi-structurés; et d'interagir avec l'utilisateur pour créer des filtres d'information personnalisés à partir des réactions de l'utilisateur, par adaptation et apprentissage. Dans cet article, nous présentons brièvement le modèle de marché de SIGMA et ses applications à une classe particulière de problèmes de FI (Karakoulas & Ferguson 1995; 1996). Le modèle définit une économie computationnelle que nous avons utilisée comme base pour construire un système de FI multi-usagers

1. Project Goals and Objectives

As the space of electronically stored information continues to expand across computer networks, the need for intelligent access to information within multi-dimensional, partially structured, and noisy spaces becomes imperative. Examples of such spaces are the Usenet, newswire feeds, the World Wide Web, and enterprise-wide information repositories. The problem of designing an information filtering (IF) system is challenging because of the volatility of these spaces and the limited ability of users to specify global interests over an inherently uncertain area of such space. In addition, the inherent distribution and dynamics of today's information networks would naturally appear to suggest some form of

robust, decentralized, multi-agent approach to the design of an IF system.

The goal of our work in SIGMA (System of Information Gathering Market-Based Agents) is to develop a framework of heterogeneous, asynchronous, software agents that manage information retrieval across a widely distributed network environment; perform automated indexing of semi-structured information items; and interact with the user to create personalized information filters from user's feedback through adaptation and learning. Various adaptation models have been proposed for dynamic multi-agent coordination, including ones based on metaphors from artificial life (Menczer et al. 1995) and from computational markets (Wellman 1994), which do not use pre-fabricated coordination strategies. As such, these adaptive, distributed algorithms appear more likely to scale with problem size and so be better suited to handle the increasing complexity of today's heterogeneous information networks. In our own work, we address the issues of scalability and robustness by proposing an adaptive model for multi-agent coordination based on the metaphor of economic markets and by integrating different machine learning techniques into the model.

In this paper we briefly present the market model of SIGMA and its application to a particular class of IF problems (Karakoulas & Ferguson 1995; 1996). The model defines a computational economy which we have used as the basis for constructing a multi-user IF system.

2. The Market-based Framework of Agents

Markets have been devised within economics for allocating limited or scarce resources among competing agents. They provide the machinery for decentralized decision-making where each agent processes only asymmetric and local information in order to evaluate decisions regarding goods and services. In addition, because of their decentralized and local nature, markets can spontaneously be developed for the exchange of goods according to local needs; they can also adjust to unforeseeable changes. Similarly to economic markets, the computational economy of SIGMA can be defined in terms of goods and agents that produce and consume those goods. The goods traded in SIGMA are information items (e.g., news articles) in different representation forms depending on the stage of processing. The agents are of three general categories: (i) the consumer agents, (ii) the producer agents and (iii) the broker agents.

A consumer represents a user's goal and preferences for a task over a time period. Producers transform goods from an input form into an output form according to their learning techniques. In response to a consumer's demand for goods they enter the local market and compete with each other to serve as efficiently as possible the demands for goods from other agents — consumers or other producers — within the market. A producer determines the price that maximizes its profit by learning how to produce goods that a consumer is expected to buy and estimating their demand. A broker is responsible for implementing the bidding policy for a particular consumer's demand for goods, namely setting up the auction each time a demand is posed in the market and deciding which producers win the bidding. The broker also maintains the history of the performance of the producers in serving the particular consumer's demand. The producers that succeed in the bidding sell goods to the consumer at their respective prices. The goods are of different quality. The consumer is endowed with a budget. At any time, the consumer is allowed to "shop around" by probabilistically assigning its preferences among the producers and allocating its budget for buying one or more goods. Given its choices for goods, these stochastic preferences are reinforced by the feedback that the consumer receives from the environment.

The feedback for each good is also propagated by the consumer to the producer from which the product was bought.

Different learning techniques have been developed and incorporated into the SIGMA computational market for dealing with the following two issues: (i) how a producer, denoted by PG (Product Generator) in the figure, can learn to bid by using its past performance in the market; and (ii) how a consumer, denoted by PS (Product Selector) in the figure, can learn to choose goods from a mixture of producers so that its cumulative reinforcement is maximized. These learning techniques can be considered independent of the particular IF task. In contrast, learning how to produce by a PG depends on the particular domain; in that case, a PG is learning a mapping from the initial information space to a compact representation of part of that space that is of potential interest to the user.

The SIGMA market model described above has been implemented as a collection of specialized CALVIN agents (Ferguson and Davlourous 1995). The CALVIN agent framework is an open architecture for facilitating the development of highly concurrent, embedded agent-oriented applications. As such, the framework provides application developers with a powerful set of agent programming tools

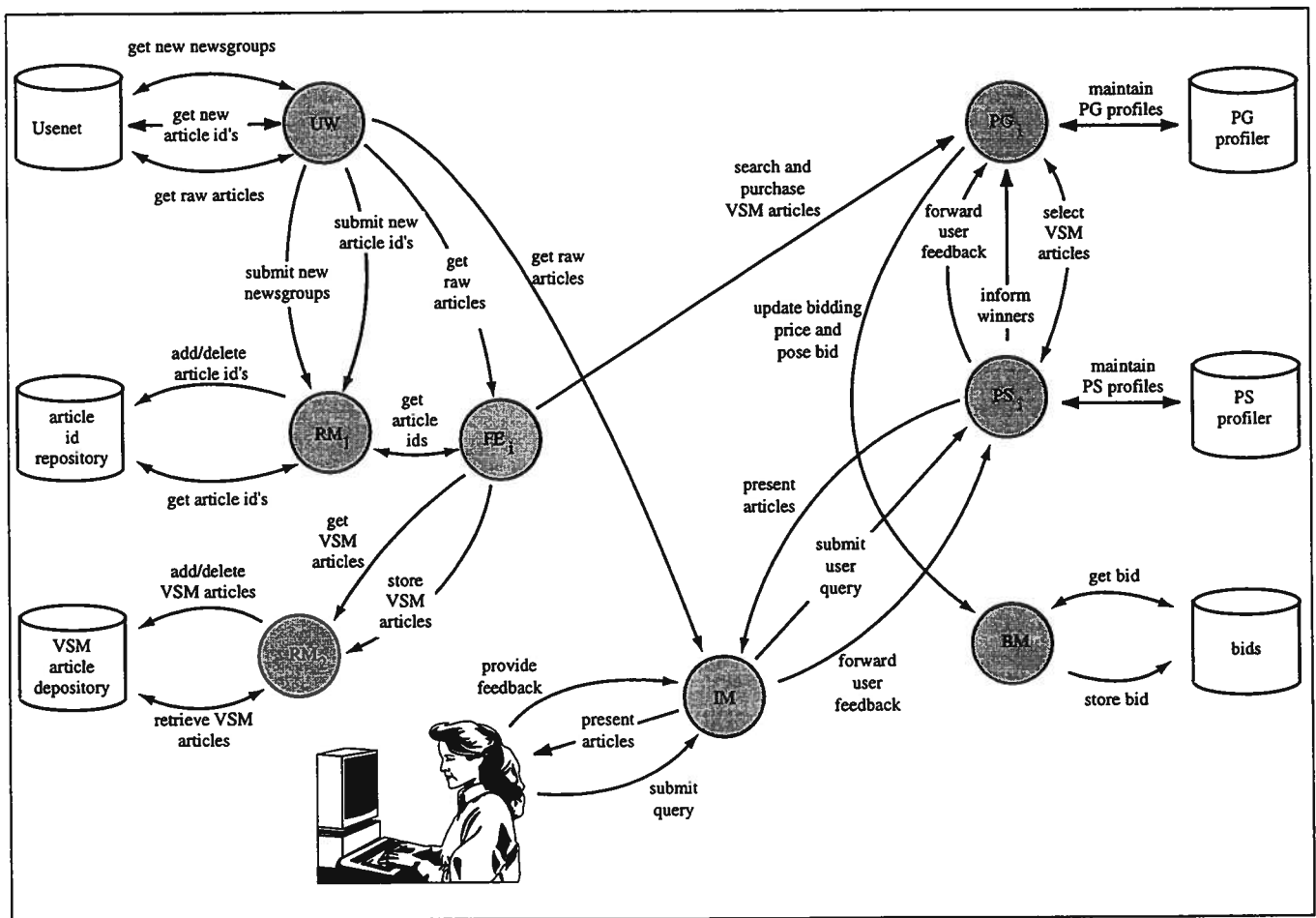


Figure 1. SIGMA agent level system description.

including libraries of intra- and inter-agent protocols (in the latter category, we are currently using KAPI (Kuokka and Harada 1995)), sensory and effectory apparatus, internal agent behavior APIs, persistent storage management, and support for pre-emptive (anytime) scheduling of behaviors, among others.

3. Applications for Intelligent Information Access

In the Usenet IF task the type of good which is exchanged most within SIGMA is the Vector Space Model (VSM) representation of an article (Salton & McGill 1983). Like any other type of document, an article contains structured information in its header part (author, subject, newsgroup, etc.) as well as unstructured information in its text part. A query submitted by a user to SIGMA consists of fields that correspond to the structured part of a news article – namely author(s), subject(s), newsgroup(s), and organization(s) – and a field that corresponds to the unstructured part – namely text keyword(s).

The SIGMA computational market for netnews IF (see Figure 1) consists of a consumer type of agent, called Profile Selector (PS), and two main types of producers: (i) the Feature Extractors (FE) which transform each news article (a generic input good) into a document indexing representation by using VSM; and (ii) the Profile Generators (PG) which are mid-producers since each of them takes as input a subset of the output of the FE producers and transforms it into a profile (a compact representation of documents) which the PG agent expects will satisfy the respective consumer's interests (Karakoulas & Ferguson 1996). Results from current experiments on this IF task suggest that heterogeneity and adaptivity in the agents architecture and behavior have a significant impact on increasing the signal-to-noise ratio in accessing interesting articles in Usenet.

While our initial interest in computational markets has largely been focused on the design and application of adaptive information filtering techniques to WWW-based news, it is our belief that as the Internet and WWW become increasingly commercialized, the need for effective profit-based agents which can act on behalf of their owners and seek payment for services rendered will increase dramatically. In addition, the adoption of intranets by large organizations for integrating corporate-wide distributed heterogeneous information repositories opens up the possibilities for interesting applications of SIGMA such as intelligent dissemination and access to changes of information in corporate as well as external networks.

References

Ferguson, I.A. and Davlouros, J.D. 1995. On establishing multi-sensory multi-channel communications among networked users. In *Proceedings IJCAI Workshop on AI in Distributed Information Networks*, pp. 103-109.

Karakoulas, G.J. and Ferguson, I.A. 1995. A computational market for information filtering in multi-dimensional spaces. In *Proceedings AAAI Fall Symposium on AI Applications in Knowledge Navigation and Retrieval*, pp. 78-83.

Karakoulas, G. and Ferguson, I. 1996. SIGMA: Integrating learning techniques in computational markets for information filtering. In *Proceedings of AAAI 96 Spring Symposium on Machine Learning and Information Access*, AAAI Press.

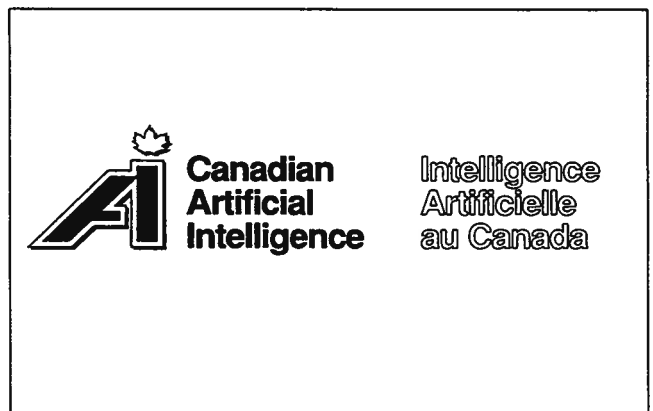
Kuokka, D. and Harada, L. 1995. Matchmaking for information agents. In *Proceedings International Joint Conference on Artificial Intelligence*, pp. 672-678.

Menczer, F., Belew, R.K., and Willuhn, W. 1995. Artificial life applied to adaptive information agents. In *Proceedings AAAI Spring Symposium on Information Gathering from Heterogeneous, Distributed Environments*.

Salton, G. and McGill, M. 1983. *Introduction to modern information retrieval*. McGraw-Hill.

Wellman, M.P. 1994. A computational market model for distributed configuration design. In *Proceedings Conference of the American Association for Artificial Intelligence*, Seattle, WA, 401-407.

Grigoris Karakoulas is at the Department of Global Analytics, Canadian Imperial Bank of Commerce. He is also a Visiting Research Scientist at the Department of Computer Science, University of Toronto. He obtained his M.Sc. in Computer Science in 1987 from University of Bradford in UK, and his Ph.D. in Computer Science in 1993 from Athens University in Greece. During 1993-1995, he was Visiting Research Fellow at the Institute for Information Technology, National Research Council of Canada. His current research interests include multi-agent learning in computational markets, reinforcement learning, machine learning techniques for financial problems, and software agents for intelligent information access. His address is: Global Analytics, Canadian Imperial Bank of Commerce, 161 Bay St., BCE-8, P.O. Box 500, Toronto, Ontario M5J 2S8; e-mail: karakoul@cibc.ca



Agent Research in the Cognitive Robotics Group

Yves Lesperance, Hector J. Levesque, Fangzhen Lin, Daniel Marcu, Ray Reiter, and Richard B. Scherl

Résumé

Pour construire un agent qui agit intelligemment et à qui on peut donner des instructions de haut niveau sur ce qu'il y a à faire, il faut lui fournir une base de connaissances contenant de l'information sur son environnement et sur ses capacités, et des méthodes pour raisonner sur les conséquences de ses actions, pour réviser ses connaissances lorsqu'il fait des observations, et pour prédire les actions d'autres agents. Ceci exige que nous nous penchions sur les questions suivantes sur la représentation des connaissances.

Abstract

To build an agent that acts intelligently and can be given high-level instructions about what to do, one needs to give it a knowledge base containing information about its environment and capabilities, and methods for reasoning about the effects of its actions, revising its knowledge when it makes observations, and predicting the actions of other agents. This requires addressing the following knowledge representation issues.

Capturing the Prerequisites and Effects of Actions

A method is needed to represent succinctly the conditions under which actions can be performed (their prerequisites) and the conditions that they are expected to change (their effects). The "frame problem" concerns the effects of actions; how can they be specified without requiring one to stipulate explicitly the numerous conditions not affected by the action.

Perception and Other Knowledge-Producing Actions

For embodied agents, most actions affect physical properties of the agent and its environment, for example, moving down a corridor. But other actions, such as reading a message one has received, need to be characterized differently, in that they affect the agent's state of knowledge, and the knowledge they yield may be required for subsequent actions or decisions. To design an intelligent agent, it is necessary to integrate perception and action with the rest of the problem-solving or decision-making architecture. For example, a robot should be able to reason about perception if only to decide when it is appropriate to try to look for something, rather than try to infer where it should be on the basis of its knowledge of the world.

Natural Events and Actions by Other Agents

Unless the agent is the only source of change in the domain, it must be able to reason about natural events and actions by other agents in order to adapt its behavior to their occurrence.

Goals, Intentions, Ability, and Rational Choice

When an agent must interact with other agents and does not have complete knowledge of their internal state and the programs they follow, it becomes useful to model them as entities that make rational choices of actions based on their goals, commitments, knowledge, and abilities. This is essential if the agents are to cooperate to perform some tasks.

A Theory of Agency

In the past few years, the Cognitive Robotics Group has been working on a theory of rational agency based on the situation calculus that addresses these questions. Reiter [Reiter91] laid the foundation for our theory by showing how the prerequisites and effects of primitive actions can be expressed in a way that solves the frame problem for a large class of cases. Scherl and Levesque [SL93] extended this solution to handle knowledge-producing actions. As well, we developed a way of representing complex actions – actions constructed out of the primitive ones using standard control structures such as sequencing, conditionals, iteration, and nondeterminism – such that these complex actions inherit the solution to the frame problem from that for primitive actions.

The GOLOG and CONGOLOG Programming Languages

The set of complex action expressions handled by the theory forms the basis of a new logic programming language that we call GOLOG (alGOI in LOGic) [LRLLS96, LLLMRS94]. Here is an example of a GOLOG program that an agent organizing a meeting might be running:

```
proc scheduleMeeting(organizer,Participant,period)
for p: Participant(p) do
request(scheduleMgr(p),
addToSchedule(p,period,meeting,organizer));
queryWhether(scheduleMgr(p),
agreedToMeet(p,self,period,organizer))
endFor;
while ~KnowWhether(self, forall p [Participant(p) ->
agreedToMeet(p,self,period,organizer)]) do
senseMsg;
if ~Empty(MsgQ(self)) then removeMsg endIf
endWhile;
/* then, inform organizer of result, etc. */
endProc
```

The example comes from [LLLMRS96]. The abstract communication actions `request` and `queryWhether` are defined in terms of the `sendMsg` primitive action.

Perhaps the most distinctive feature of GOLOG is that its interpreter automatically maintains an explicit representation of the dynamic world being modeled, on the basis of user supplied axioms about the preconditions and effects of primitive actions and the initial state of the world. This allows GOLOG programs to reason about the state of the world and consider the effects of various possible courses of action before committing to a particular behavior. The net effect is that programs may be written at a much higher level of abstraction than is usually possible. Note that our approach focuses on high-level programming rather than plan synthesis at run-time; but nondeterminism can be used to specify sketchy plans, leaving it up to the interpreter to search for an executable sequence of primitive actions that will achieve the desired effects. When an implementation of the primitive actions is provided, a GOLOG program can be executed in a real environment; in the absence of this, the interpreter can nevertheless produce a simulated execution. A prototype GOLOG interpreter has been implemented in PROLOG.

For applications involving multiple interacting agents, as well as cases where reactive or event-driven behavior is required, it is useful to extend the set of complex actions handled in GOLOG to include concurrent processes, priorities, interrupts, and multiple agents [LLLMRS96]. We call the extended language CONGOLOG (CONcurrent GOLOG). A prototype interpreter for it has also been implemented in PROLOG.

Applications

To evaluate our theoretical framework and the GOLOG/CONGOLOG interpreters as implementation tools, we have been experimenting with various types of applications. One project under way involves robotics applications. GOLOG makes it relatively easy to specify intelligent strategies for a robot to accomplish its tasks. So it seems that a GOLOG-based reasoning module should be very useful as a component of a hierarchical control system for a robot; this should yield more powerful and adaptable robotics systems. Our first application in this area has been mail delivery in an office environment [LLLMRS94]. A high-level controller for the robot was programmed in GOLOG and interfaced to a robotics software package that supports path planning and local navigation. So far, we have only used the system in simulation mode; but we are acquiring a RWI B-21 robot and will soon be running experiments in using the GOLOG controller to have it perform tasks. A group at the University of Bonn has also been experimenting with GOLOG to control its RWI B-21. In collaboration with Michael Jenkin's team at York University, we are studying how the GOLOG reasoning module is best integrated with the other components of a hierarchical control system for robots in order to get maximum performance.

In contexts such as robotics, sensor noise and "control error" are immediate problems. We have made some headway in modeling this within our theory [BHL95]. A new account of planning has also been developed to deal

with that fact that robot plans typically involve sensing [Levesque96]. Earlier work on agent-centered representations [LL95] has also been incorporated in our framework [SLL95].


In another project, we are developing tools based on CONGOLOG for modeling business and organizational processes [YML96]. In contrast to the operational view of conventional modeling tools, CONGOLOG takes a logical view of processes. This should prove advantageous when it comes to modeling system behavior under incompletely known conditions and proving properties about the system. CONGOLOG can produce a simulation directly from the system specification. To test our framework, we are using it to model processes from real organizations. One such experiment involves modeling the procedures followed by nuclear plant operators. Related to this is the problem of how one can model continuous processes in a state-based framework like the situation calculus. [Reiter 96] and [Pinto 94] develop solutions to this which provide the foundations for the design and implementation of simulators for physical systems [Kelley96].

Work is also under way on more classical intelligent software agent applications. Ruhman [Ruhman96] implemented a multi-agent "home banking assistant" tool that can be used to monitor account balances and select the best transaction to perform when a balance moves outside a target range. The system involves a number of GOLOG agents that communicate using TCP/IP. A CONGOLOG treatment of the popular office meeting scheduling problem has also been developed [LLLMRS96]. An implementation of the design is under way.

As mentioned earlier, when agents start interacting with others without having complete knowledge of the situation, it is advantageous for them to view other agents as having goals, intentions, and abilities, and as making rational choices. This allows them to anticipate and influence the behavior of other agents, and cooperate with them. It also supports an abstract view of communication acts as action that affect other agents' mental states as opposed to mere message passing. We have started extending our framework to deal with abilities, goals, intentions, and rational choice [SLL95,LLLS96].

References

- [BHL95] F. Bacchus, J.Y. Halpern, and H.J. Levesque. Reasoning about Noisy Sensors in the Situation Calculus. In *Proceedings of IJCAI-95*, pp. 1933-1940, Montreal, August, 1995.
- [Kelley96] T.G. Kelley. Reasoning about physical systems with the situation calculus, in *Proc. Common Sense 96: Third Symposium on Logical Formalizations of Commonsense Reasoning*, Stanford, CA, January, 1996.
- [LL95] Y. Lesperance and H.J. Levesque. Indexical Knowledge and Robot Action – A Logical Account. *Artificial Intelligence*, 73, pp. 69-115, 1995.

- [LLLMRS94] Y. Lesperance, H.J. Levesque, F. Lin, D. Marcu, R. Reiter, and R.B. Scherl. A Logical Approach to High-Level Robot Programming - A Progress Report. In Benjamin Kuipers, editor, *Control of the Physical World by Intelligent Systems, Papers from the 1994 AAAI Fall Symposium*, pp. 79-85, New Orleans, LA, November, 1994.
- [LLLMRS96] Y. Lesperance, H.J. Levesque, F. Lin, D. Marcu, R. Reiter, and R.B. Scherl. Foundations of a Logical Approach to Agent Programming, in M. Wooldridge, J.P. Mueller, and M. Tambe, editors, *Intelligent Agents Volume II - Proceedings of the 1995 Workshop on Agent Theories, Architectures, and Languages (ATAL-95)*, pp. 331-346, Springer-Verlag, Lecture Notes in Artificial Intelligence, 1996. A French version appeared in *Actes des Troisièmes Journées Francophones sur l'Intelligence Artificielle Distribuée et les Systèmes Multi-Agents*, Chambéry-St-Badolph, France, March, 1995.
- [LLLS96] Y. Lesperance, H.J. Levesque, F. Lin, and R.B. Scherl. Ability and Knowing How in the Situation Calculus. In preparation. 1996.
- [LRLS96] H.J. Levesque, R. Reiter, Y. Lesperance, F. Lin, and R.B. Scherl. GOLOG: A Logic Programming Language for Dynamic Domains, to appear in *Journal of Logic Programming*, special issue on Reasoning about Action and Change, 1996.
- [Levesque95] H.J. Levesque. What is planning in the presence of sensing? To appear in *The Proceedings of the Thirteenth National Conference on Artificial Intelligence, AAAI-96*, Portland, Oregon, August 1996.
- [Pinto94] J. Pinto. Temporal Reasoning in the Situation Calculus. Ph.D. Thesis, Dept. of Computer Science, Univ. of Toronto, January 1994.
- [Reiter96] R. Reiter. Natural actions, concurrency and continuous time in the situation calculus. *Proc. Common Sense 96: Third Symposium on Logical Formalizations of Commonsense Reasoning*, Stanford, CA, January 1996.
- [Reiter91] R. Reiter. The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy*, Vladimir Lifschitz (ed.), pp.359-380, Academic Press, San Diego, CA, 1991.
- [Ruhman96] S. Ruhman. GOLOG as an agent-programming language: Experiments in developing banking applications. M.Sc. Thesis, Dept. of Computer Science, University of Toronto, 1996.
- [SLL95] R.B. Scherl, H.J. Levesque, and Y. Lesperance. The Situation Calculus with Sensing and Indexical Knowledge, in Moshe Koppel and Eli Shamir, editors, *Proceedings of BISFAI'95: The Fourth Bar-Ilan Symposium on Foundations of Artificial Intelligence*, pp. 86-95, Ramat Gan and Jerusalem, Israel, June, 1995.
- [SL93] R.B. Scherl and H.J. Levesque. The Frame Problem and Knowledge Producing Actions. In *Proceedings of AAAI-93*, pp. 689-695, Washington, DC, July, 1993.
- [SLL95] S. Shapiro, Y. Lesperance, and H.J. Levesque. Goals and Rational Action in the Situation Calculus - A Preliminary Report, in *Working Notes of the AAAI Fall Symposium on Rational Agency: Concepts, Theories, Models, and Applications*, Cambridge, MA, November, 1995.
- [YML96] E. Yu, J. Mylopoulos, and Y. Lesperance, Modelling the Organization: New Concepts and Tools for Re-Engineering, to appear in *IEEE Expert*, special issue on Artificial Intelligence Applications in Enterprise Modelling, June 1996.
- Copies of these and other relevant sources of information may be found at: <http://www.cs.toronto.edu/~cogrobo> 
-
- Yves Lesperance is an Assistant Professor in Computer Science at York University. He received his Ph.D. in Computer Science from the University of Toronto in 1991. His current work focuses on the development of logic-based tools for modeling and designing intelligent agents. lesperan@yorku.ca <http://www.cs.yorku.ca/People/lesperan>*
- Fangzhen Lin received his doctorate from Stanford University in 1991. He pursued research at the University of Toronto from 1992 to 1996. He is currently working at the Hong Kong University of Science & Technology.*
- Richard Scherl is currently an Assistant Professor in the Department of Computer and Information Science at the New Jersey Institute of Technology. He received his Ph.D. from the University of Illinois in 1992.*
- Hector Levesque is a Professor of Computer Science at the University of Toronto. He received his Ph.D. in 1981 from the University of Toronto. His current work focuses on how to keep reasoning computationally tractable and on the knowledge representation problems faced by an autonomous agent interacting with a dynamic and incompletely known world.*
- Ray Reiter holds a B.A. and M.A. in mathematics from the University of Toronto, and a Ph.D. in Computer Science from the University of Michigan. He is currently Professor of Computer Science at the University of Toronto. His research interests are in artificial intelligence, with emphasis on foundations for knowledge representation. He is a Fellow of the American Association for Artificial Intelligence, of the ACM, and of the Canadian Institute for Advanced Research. In 1993 he received the Research Excellence Award, granted biennially by the International Joint Conference in Artificial Intelligence.*
- Daniel Marcu is a Ph.D. student in the Department of Computer Science, University of Toronto. He works in the area of computational linguistics, knowledge representation and reasoning for natural language systems, and robotics. His current work focuses on building theories and systems for agents that communicate in plain English.*

Distributed Agent Systems for Intelligent Manufacturing

Douglas H. Norrie and Brian R. Gaines

Introduction

Les architectures pour la fabrication intelligente formant des sociétés d'agents coordinateurs sont devenues un sujet de recherche important au cours de la dernière décennie. Pendant cette même période, la recherche sur les systèmes de fabrication intelligente a connu une collaboration internationale plus grande, ainsi qu'une coordination accrue des sociétés de chercheurs à travers le monde. Les modèles et les technologies de coordination distribuée d'agents en cours de développement pour gérer la fabrication, s'appliquent aussi bien à la gestion de la recherche en collaboration distribuée. Le programme de recherche international des Systèmes de fabrication intelligente tente de systématiser et de rendre opérationnel le savoir mondial sur les systèmes avancés de fabrication en vue de générer de nouveaux paradigmes [3]. Cet article met l'accent sur la recherche sur les systèmes de fabrication intelligente à base d'agents effectuée à l'Institut de la science de la connaissance et à la Division d'ingénierie de la fabrication de l'Université de Calgary pour supporter le cycle de vie de fabrication dans une entreprise distribuée. On y décrit des applications à la coordination de recherche en fabrication et à des activités et procédés de fabrication spécifiques.

Introduction

Architectures for intelligent manufacturing systems as societies of coordinating agents have become a major research theme in the last decade. At the same time, research activities on intelligent manufacturing systems have increasingly involved international collaboration and the coordination of societies of human research agents worldwide. The distributed agent coordination models and technologies being developed to manage manufacturing are also applicable to the management of distributed collaborative research. The international Intelligent Manufacturing Systems (IMS) research program is an attempt to systematize and make operational world-wide knowledge of advanced manufacturing systems as a basis for new paradigms [3].

This article focuses on agent-based IMS research undertaken in the Knowledge Science Institute and the Division of Manufacturing Engineering at the University of Calgary to support the manufacturing life cycle in a distributed enterprise. Applications to the coordination of manufacturing research and to specific manufacturing activities and processes are described.

The following describes two multi-agent systems embodying new approaches and for which working prototypes have been developed. The first of these is a high-level Mediator agent system which supports the coordination in concurrent fashion, of geographically distributed application processes via Internet communication [4]. The second multi-agent system is a concurrent engineering application which supports coordinated simultaneous design, process planning, routing, and scheduling activities [7].

Mediator

Mediator is an agent-based, open-architecture information and knowledge management system designed to provide a flexible technology to support the management of complex manufacturing environments. A heterogeneous environment is assumed in which the sub-systems are geographically dispersed and involve different application packages, not necessarily designed to work together, multiple platforms,

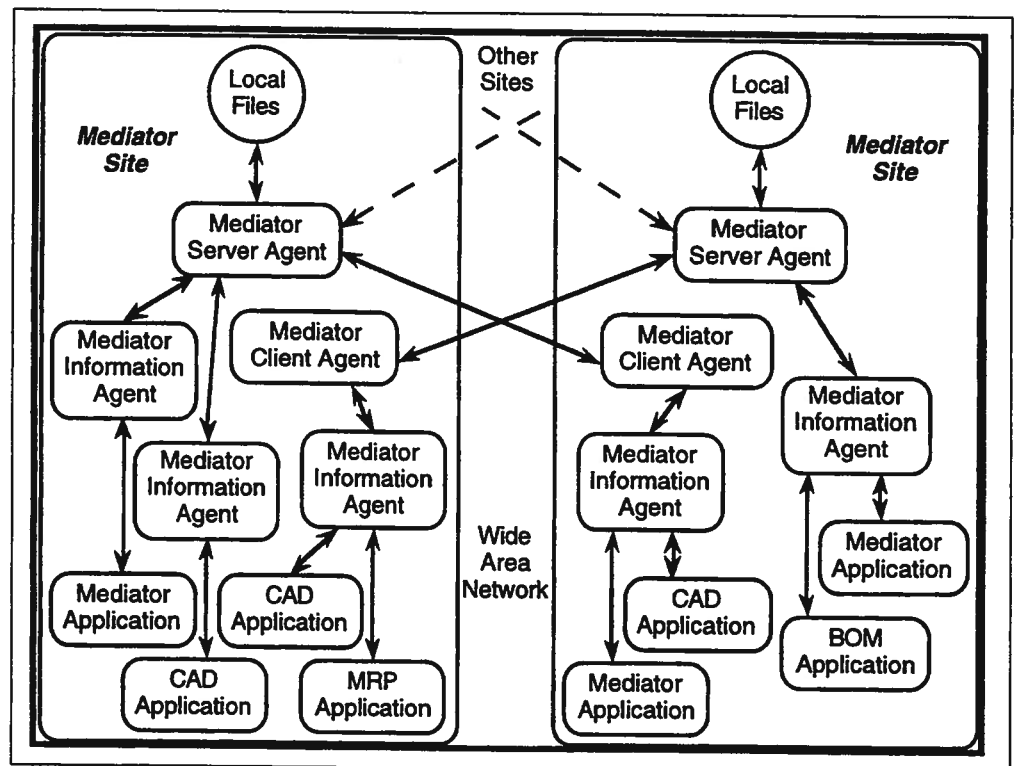


Figure 1. Mediator Operation Over a Network

protocols and forms of user interface. The function of Mediator is to provide a knowledge support system for the managers and system operators involved in running a virtual factory. It is designed to facilitate communication, compliance with constraints including physical restrictions and legal obligations, and to generally represent knowledge about any activity or sub-system relevant to the manufacturing process.

Figure 1 shows the way in which Mediator operates over a network. A server agent at a site manages a knowledge base consisting of a set of files from different applications. Concept maps are used to represent the files and relations between them. Files may be opened from the maps in the appropriate applications. Since the maps and hypermedia documents of Mediator are also files, the system can be used to support large-scale linked knowledge structures. Client agents at remote sites connect to server agents across the network and allow files to be accessed remotely in the same way as they are locally.

Concurrent Engineering MAS

The architecture of this concurrent engineering application is based on a heterogeneous agent paradigm in which every physical or other entity in the system, such as design features, parts, and manufacturing shop floor machines, is associated with a reasoning agent. These design, resource, part, and coordination agents are integrated into an open ended system in which dynamic virtual clusters of agents interact to carry out the necessary concurrent engineering activities [8]. Novel

features are: dynamic system organization through virtual clustering (the clusters are created, coordinated, and destroyed as required for task accomplishment); agent reasoning through an asynchronous message-triggered inferencing process quite different from conventional inferencing techniques; specially developed mechanisms to provide each agent with its own independent thread of execution control and to dynamically coordinate these. The system has been tested when distributed across three workstations connected by a local area network.

Figure 2 shows a simplified view of the architecture. The components include a CAD sub-systems, a Shop-Floor Manager, and Resource agent communities. These multi-agent composite components have reasoning, control, and communication capabilities for both their internal activities and their collaborative actions with other agents. This synergy achieves flexibility, expandability, fault tolerance, and reconfigurability on a real-time basis.

The system has been implemented on a network of HP 9000/715 machines running HPUX 9.01. The CAD sub-systems have been implemented using AutoCAD R12, Advanced Modeling Extension V2, a light-weight process library developed for this purpose, and C++. The Shop-Floor Manager and resource agents have been implemented using Smalltalk VisualWorks 2.0. Every agent in the system has an independent thread of execution control. The HPUX communication capabilities have been used to provide an asynchronous messaging mechanism for communication among the sub-systems. The constituent individual agents use facilities provided by their respective local environment, namely, the light weight process library and Smalltalk.

The CAD sub-system provides facilities for feature-based design and comprises Part Agents, Feature Agents, a Design Agent, a Geometric Interface Agent, and an Environment Manager. A part is designed by repeated instantiations of features on a blank geometry. The Part Agent is the repository for both product data and knowledge, and dynamically updates itself as the design progresses. Every feature type present in the design system and every feature instantiated in the design process is represented by a Feature Agent. The design system has more than twenty-five non-degenerate feature types for prismatic components.

The Design Agent here is the human expert. The Geometric Interface Agent acts as an intermediary for translating information between the graphical display, the constructive solid geometry manipulation front end, and other agents. The Environment Manager coordinates the activities of the agents in the design system by acting as a message redirector. It also interfaces local agents with manufacturing resource agents through shop-floor manager.

The Shop-Floor manager plays a vital role in shop floor control by establishing and maintaining virtual agent communities and by redirecting messages. It imposes an adaptive hierarchy within a resource community as required. The Shop-Floor Manager is comprised of multiple

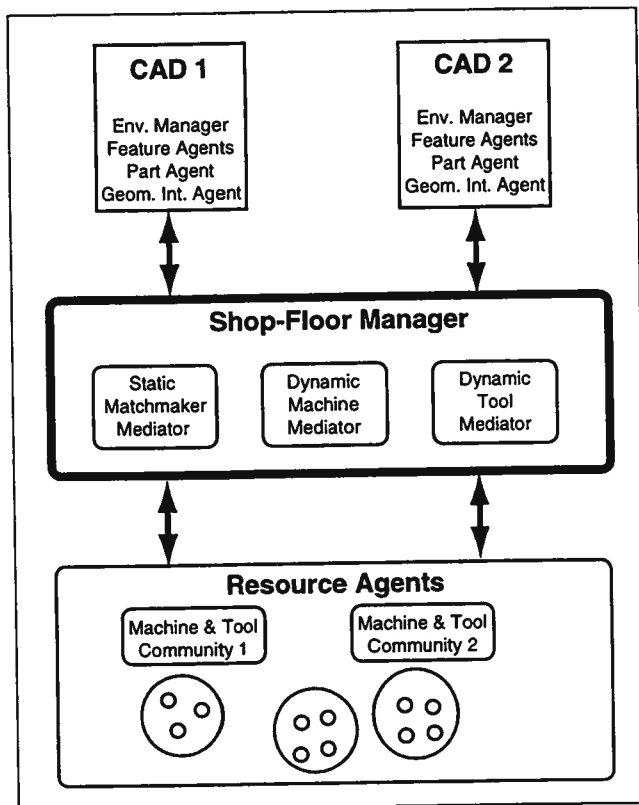


Figure 2. System Architecture

components, namely, a Static Matchmaker mediator, and a number of dynamic machine and tool mediators. The mediators have a generic coordination architecture [9] and provide facilities for matchmaking, cloning, clustering, dynamic coordination, and arbitration.

Each machine on the shop floor is represented by an autonomous machine agent which has knowledge of its own machine's physical and process capabilities, potential or assigned tooling, and production schedule. Each tool in the shop floor is represented by an autonomous tool agent which has knowledge about the tool's shape, and its tolerance capabilities in combination with a particular machine and work material under standard operating parameters. It also maintains a record of the schedule for the tool's use.

Other Agent Research at the University of Calgary

Other agent projects related to manufacturing include a development system for agent software [6], planning and control of autonomous guided vehicles (AGVs) [5], and intelligent control [1].

The applications of agent-based systems to manufacturing are part of a long-term program of research into the fundamentals of intelligent, adaptive agents, including communities of agents in human society and their interactions with technology.

The collective stance model [2] views the human species as a single agent recursively partitioned in space and time into sub-agents that are similar to the whole. These parts include societies, organizations, groups, individuals, roles, and neurological functions. Notions of expertise arise because the species adapts as a whole through adaptation of its interacting parts. The phenomena of expertise correspond to those leading to distribution of tasks and functional differentiation of the parts. The mechanism is one of positive feedback from parts of the agent allocating resources for action to other parts on the basis of those latter parts past performance of similar activities. Distribution and differentiation follow if performance is rewarded and low performers of tasks, being excluded by the feedback mechanism from opportunities for performance of those tasks, seek out alternative tasks where there is less competition. The knowledge-level phenomena of expertise, such as meaning and its representation in language and overt knowledge, arise as byproducts of the communication, coordination, and modeling processes associated with the basic exchange-theoretic behavioral model.

The collective stance model has been applied to many areas of human-technology interaction, including educational systems and the use of Mediator to support a learning web [10].

References

- [1] Balasubramanian, S. and Norrie, D.H. "Intelligent Manufacturing Control," *Proceedings of 1996 Canadian Conference on Electrical and Computer*

Engineering, 26-29 May 1996, Calgary, IEEE Publication, 4 pp., 1996 (in press).

- [2] Gaines, B.R. "The Collective Stance in Modeling Expertise in Individuals and Organizations," *International Journal of Expert Systems*, Vol. 7, No. 1, pp. 21-51, 1994.
- [3] Gaines, B.R. and Norrie, D.H. "Knowledge Systematization in the International IMS Research Program," in *Proceedings of 1995 IEEE International Conference on Systems, Man and Cybernetics*, IEEE, New York, pp. 958-963, 1995.
- [4] Gaines, B.R. and Norrie, D.H. and Lapsley, A.Z. "Mediator: an Intelligent Information System Supporting the Virtual Manufacturing Enterprise," in *Proceedings of 1995 IEEE International Conference on Systems, Man and Cybernetics*, IEEE, New York, pp. 964-969, 1995.
- [5] Kwok, A. and Norrie, D.H. "Intelligent Agent Systems for Manufacturing Applications," *Journal of Intelligent Manufacturing*, Vol. 4, pp. 285-293, 1993.
- [6] Kwok, A. and Norrie, D.H. "A Development System for Intelligent Agent Manufacturing Software," *International Journal of Integrated Manufacturing Systems*, Vol. 5, No. 4/5, pp. 64-76, 1994.
- [7] Maturana, F.P. and Norrie, D.H. "A Generic Mediator for Multi-Agent Coordination in a Distributed Manufacturing System," in *Proceedings of 1995 IEEE International Conference on Systems, Man and Cybernetics*, IEEE, New York, pp. 952-957, 1995.
- [8] Maturana, F., Balasubramanian, S. and Norrie, D.H. "A Multi-Agent Approach to Integrated Planning and Scheduling for Concurrent Engineering," *Proceedings of Third International Conference on Concurrent Engineering*, 26-28 August, 1996, Toronto, Canada, 8 pp., 1996 (in press).
- [9] Maturana, F.P. and Norrie, D.H. "Multi-Agent Coordination Using Dynamic Virtual Clustering in a Distributed Manufacturing System," *Proceedings of Fifth Industrial Engineering Research Conference (IERC5)*, May 18-20, 1996, Minneapolis, 6 pp., 1996 (in press).
- [10] Norrie, D.H. and Gaines, B.R. "The Learning Web: A System View and an Agent-Oriented Model," *International Journal of Educational Telecommunications*, Vol. 1, No. 1, pp. 23-41, 1995.

Copies of these papers and other relevant articles may be found at: <http://ksi.cpsc.ucalgary.ca/articles/> and <http://www.ucalgary.ca/~norrie/>

Douglas H. Norrie currently holds the Nortel Chair in Intelligent Manufacturing at The University of Calgary, Alberta, Canada. He is also Professor of Mechanical Engineering and Adjunct Professor of Computer Science at the same institution. His research interests are in Intelligent
(Continued, page 21)

Personal Assistants for the Office Professional

Cliff Grossner and T. Radhakrishnan

Résumé

Des chercheurs étudient actuellement la conception d'agents logiciels qui agiront en tant qu'assistants personnels au professionnel de bureau: par exemple, des agents qui filtrent le courrier, aident au traitement des appels téléphoniques, et aident à la gestion de l'agenda quotidien du professionnel. Ces agents devront interagir avec le professionnel d'une façon naturelle pour lui, autrement on ne réussira pas à les introduire dans les bureaux de demain. De plus, lorsqu'ils rempliront leurs fonctions, ces agents devront interagir les uns avec les autres de façon à partager l'information et à résoudre des conflits; par exemple, lorsque l'agent qui est responsable des appels téléphoniques voudra localiser son propriétaire, il devra consulter l'agent responsable de l'agenda. Les architectures d'agents utilisées pour la conception d'agents assistants au professionnel de bureau devront posséder plusieurs caractéristiques: une représentation explicite du problème que l'agent doit résoudre, la possibilité d'acquérir un ensemble initial de connaissances, la possibilité pour un agent d'acquérir de nouvelles connaissances, la modélisation des interactions entre un agent et les autres agents dans son environnement, et la possibilité pour les concepteurs de l'agent de vérifier la validité de la base de connaissances de l'agent. Dans cet article, nous décrivons nos modèles d'agents logiciels, qui peuvent soutenir le développement d'architectures d'agents.

Abstract

Researchers are now studying the design of software agents that will act as personal assistants to the office professional, such as agents that filter email, support the processing of telephone calls, and support the management of the professional's daily calendar. These agents will have to interact with the office professional in a manner that is natural for the professional, or they will not be successfully integrated into the office of tomorrow. In addition, when performing their functions, these agents will have to interact with each other in order to share information and to resolve conflicts; for example, when the agent that is responsible for processing telephone calls wants to locate its owner, it will have to consult the agent responsible for calendar management. Agent architectures used for the design of agents to aid the office professional will have to support several features: an explicit representation of the problem that the agent is solving, support the acquisition of an initial body of knowledge, support the ability for the agent to acquire new knowledge, support modeling of interactions that occur between an agent and the other agents in its environment, and permit the agent's designers to validate the knowledge base of the agent. In this paper, we describe our models for software agents, which can support the development of agent architectures.

Introduction

In many of the business offices in Canada, the technology used to support a professional has not advanced much since the introduction of the electric typewriter, except for the introduction of the word processor. There are many factors that can be responsible for this, and one factor that is certainly a problem is the gap between how office professionals currently carry out their daily functions and the human interface that today's computers can provide. In fact, when presented with the option of learning to use a new software package, many professionals choose to continue with the method they are currently using because the interface provided by this software package is too clumsy and requires too much time for the professional to learn.

We believe that with the development of *intelligent software agents* that assist a professional by acting as a personal assistant, it will be possible to further enhance the productivity and the convenience of working because these agents will be able to provide interfaces that do not require large effort on the part of the professional. The personal agent of an office professional will "know" about the professional's needs, working habits, and constraints, and this knowledge will be represented in a suitable form [1]. This agent would also be a learning agent, in which case its knowledge will be improved incrementally, and any knowledge added to the agents' knowledge base must be validated [2]. Learning could come through several means, including the agent observing the office professional performing routine tasks [3].

An office is considered to be an open system by several researchers [4,5]. Such a system has to operate in an environment whose boundaries are not clearly specified. There are several different tasks that an office professional performs on a daily basis, such as reading email, scheduling and attending meetings, and processing voice mail messages. Let us suppose that each of these tasks is assisted by an agent. Thus, there will be several agents assisting a single office professional. An agent, as we understand, will have the following three attributes:

- It is autonomous; it will operate as an independent process and interact synchronously with other agents on a peer-to-peer level.
- It encapsulates functionality; it serves as a container of "human knowledge" that is needed by the agent to assist a user in a specific task. This knowledge refers both to computational methods and the associated data. An agent can be accessed only through a well defined agent interface. In this sense, it resembles an object as described by an Object Oriented paradigm.
- It uses a predefined agent communication language; multiple agents communicate by passing messages whose

syntax and semantics are formally defined by a standard agent communication language.

It is possible that an agent providing a service to the office professional will actually consist of multiple agents interconnected through a network, and each agent may play a different role. For example, there could be specialized interface agents for interacting with the office professional, or agents that are able to connect to open environments such as the Internet. All these agents acting together will form an *agent architecture* that is suitable to provide the required service to the office professional.

Agent-to-agent communication requires the sending agent to create some model of the relevant aspects of the receiving agent(s). This is particularly true when problem-solving involves alternating phases of *planning* and *execution*. In the planning phase, the agents involved in planning will collect information or create models of the other agents, so that they may create a "good plan" that will be fruitfully executed in the following execution phase. Thus, in certain cases agent modeling becomes an important aspect in supporting agent communication.

In this report, we will consider how to model software agents. Our models are designed to capture the actions that are taken by agents that interact to solve a problem, and to capture the data items that must be shared among the agents as they are problem-solving. First, we will consider how to model the problem that is to be solved by the agents. Using this model, we will then show how to model the agents themselves, given that these agents are implemented as rule-based systems. Then, we describe how to model the interaction between the agents in terms of the actions that are taken by each agent to solve the problem, and in terms of the data items that must be shared between the agents.

Models for Software Agents

Our model for representing problems is based upon the state space model that is used by many researchers for generic problem-solving [6]. In the state space model, problem-solving is seen as a traversal of a set of states, and at each state a number of different *actions* may be taken leading the problem-solver into a new state. After traversing many states, the problem-solver will solve the problem, reaching a final state in the state space. We propose the state space notion to include the existence of states that are referred to as *goal* states [7]. Goal states are states that are identified by the problem-solver as being important to reach in solving the problem. Reaching a goal state implies that a "meaningful" step has been taken in solving the problem.

In our model for representing problems, we identify two types of goals: *concrete* goals and *abstract* goals. A concrete goal is a goal that is an atomic unit in terms of which abstract goals are defined. Concrete goals are mapped to states in the state space model, and the transitions in the state space model will correspond to the steps required to achieve one concrete goal starting from another concrete goal. Abstract goals are goals that are achieved when the

problem-solver achieves several concrete goals, and it is common that there will be a number of different combinations of concrete goals that lead to the achievement of the same abstract goal. In effect, these different combinations for achieving an abstract goal represent alternative methods for achieving that abstract goal. The appropriate method for achieving an abstract goal each time that goal is to be achieved will depend on the current state of the problem being solved and the preferences of the problem-solver.

There exist many different relationships between goals, and we make use of these relationships to determine the combinations of concrete goals that are appropriate for achieving an abstract goal. In our model, we focus on the use of the subgoal relationship [8], where we include a logical function as a component of the subgoal relationship. This representation, shown in Figure 1, permits us to capture the different actions that a problem-solver may take when problem-solving as a network of goals interconnected by the subgoal relationship. We refer to this network of goals as a Designer's Goal Graph (DGG). If we consider the goal network shown in Figure 1, we can see that there are abstract goals and concrete goals that are interconnected by XOR (X), OR (O), and AND (A) subgoals relationships. The XOR subgoal relationship is used to express that an abstract goal is achieved by one of a number of different alternatives, and only one of the alternatives would be chosen to be achieved each time the abstract goal is to be achieved. The OR subgoal relationship also indicates that an abstract goal is achieved by one of a number of different alternatives, but that it may be advantageous to pursue alternative approaches for solving the same goal simultaneously. If several alternatives for achieving a goal are pursued simultaneously and one of the alternatives succeeds, all ongoing attempts to achieve other competing alternatives would be abandoned. In the case of the AND subgoal relationship, an abstract goal is achieved when the set of subgoals on the AND list are achieved.

Let us now consider how we can model an agent, assuming that the problem to be solved by the agent has been modeled using a designer's goal graph. The key to modeling agents that are implemented as rule-based systems is to define a notion of a *rule-based execution path* that is analogous to an execution path for conventional software. For this purpose, we have developed the path model [9]. The path model permits an agent to be modeled as a set of paths, where each path achieves a concrete goal. Given the concrete goals that are defined in the designer's goal graph for the problem that the agent is designed to solve, the path model defines sequences of interdependent rules that will achieve a concrete goal when all the rules in the path fire; thus, for each concrete goal, the path model will define one or more sequences of rules that can achieve that goal. Analysis of the rules in each path indicates the data items that are required by all the rules in a path to fire; thus the path model also captures the data items that are required to achieve each concrete goal.

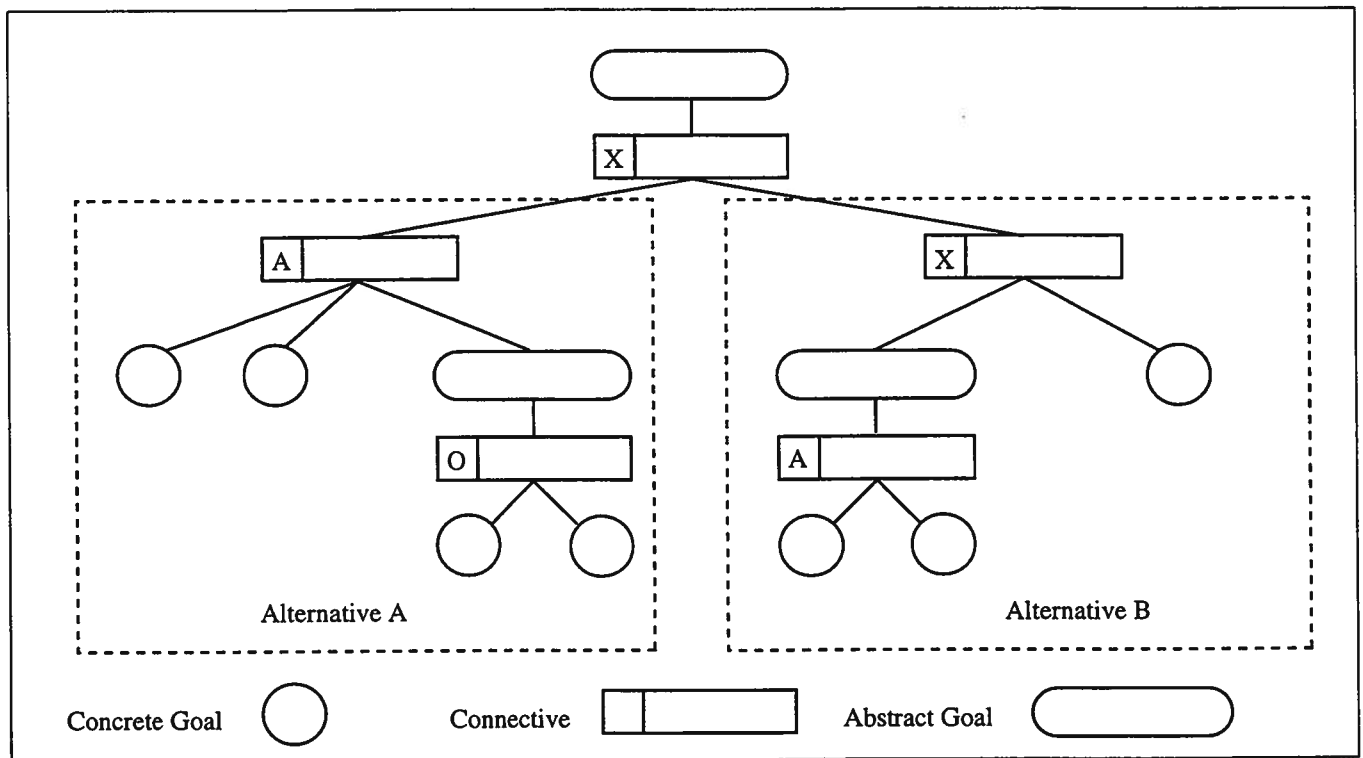


Figure 1. Problem Representation (Designer's Goal Graph)

Given the designer's goal graph, the paths used to achieve a concrete goal, and the data items required to achieve that goal, we can now consider the different sequences of actions that can be used by an agent to achieve an abstract goal. Again, we will make use of the goal graph to define the notion of a *line of reasoning*. A line of reasoning is a partially ordered sequence of tuples, and each tuple consists of a concrete goal and a path that achieves that goal. The concrete goals in a line of reasoning satisfy the constraints for achieving an abstract goal that is contained in the designer's goal graph. The tuples belonging to a line of reasoning are partially ordered by the precedence constraints between the concrete goals in each tuple. Each line of reasoning represents the sequence of actions that are to be taken to achieve an abstract goal as well as the data items that are required by the agent to achieve that abstract goal.

The interactions between software agents can be modeled in terms of the actions that are taken by each agent to solve the problem, and in terms of the data items that must be shared between the agents. At any time in the problem-solving process, each agent will be attempting to achieve one or more abstract goals. Each agent will determine a line of reasoning that is appropriate to achieve the abstract goal it is mandated to achieve. Making use of space-time diagrams, we can determine the data items that must be shared between the agents. We will also be able to determine at which stage in the problem-solving process an agent producing a data item should make the data item available, and determine which agents will require access to that data item.

As an example, consider one possible interaction between

three software agents, as modeled in Figure 2. Here, IM is an agent that monitors news sources for specific articles, MF is an agent that filters multimedia messages, and MS is an agent that schedules meetings. Each of these agents is implemented as a rule-based system, the problem domain that each agent has been designed to solve is modeled using a designer's goal graph, and the rule base of each agent has been analyzed to discover all of the execution paths that can be used to achieve a concrete goal. In Figure 2, we show the concrete goals that are achieved and the messages passed by these three agents when agent IM detects that a news article has been posted about which the office professional would want to be notified. IM determines that it does not currently know the location of the office professional, and decides to send a request to agent MF, shown by (A) in Figure 2. Agent MF decides to see if Agent MS can locate the office professional (B). Eventually, both Agent MF and MS respond indicating that they do not know where to locate the office professional (D). Agent MS and IM both take on the goal of asking other agents if they know the location of the office professional. Agent MS is able to determine the location of the office professional with the help of other agents (not shown in Figure 2), and notifies Agent IM (E). Agent IM can then deliver a message to the office professional.

In Figure 2, we show only one of many different sequences of actions that might have been pursued by Agents IM, MF, and MS when IM determines that it must contact the office professional. The different sequences of actions that can occur will be modeled using the designer's goal graph and the different paths that are present in the rule base of each

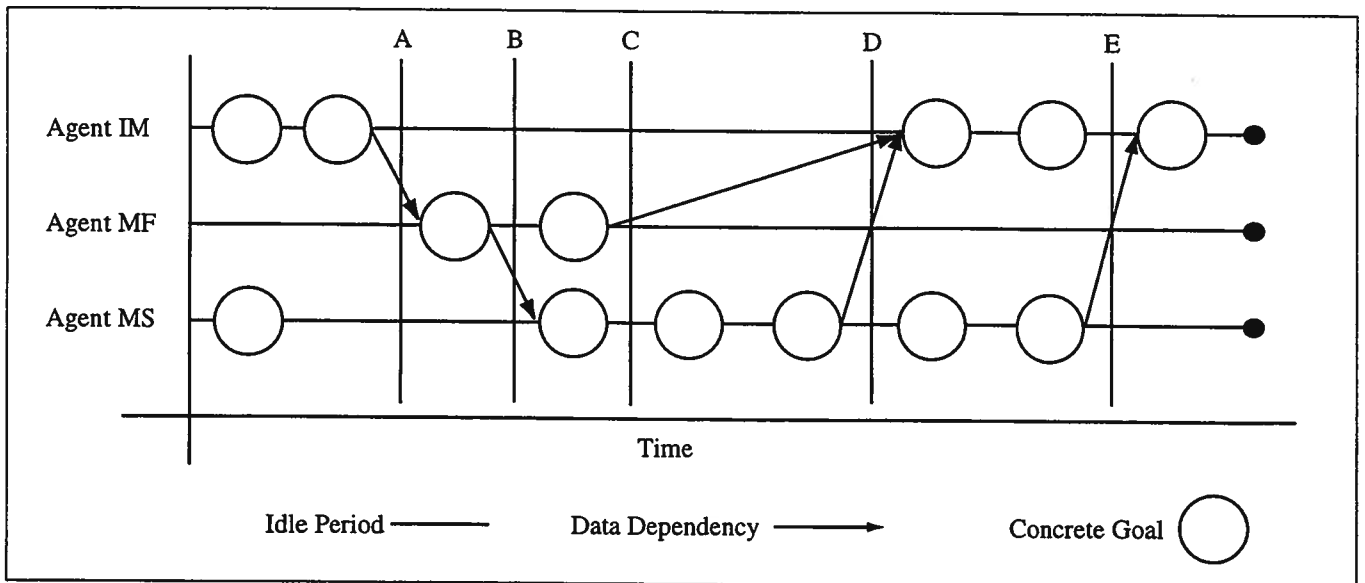


Figure 2. Modeling Agent Interaction

agent. Of course, the data messages that are passed between the agents will vary with the lines of reasoning each agent uses to achieve their goals.

When the interaction between agents is modeled as we describe, the data dependencies between the agents and the idle times that result due to the data dependencies are known.

Selected Research Issues

In an area of application for software agents, such as office information systems, there are several technical and research issues to be studied. In this section, we will describe four such issues that are relevant to our research interests: (1) learning in the context of a rule-based agent, (2) validation and verification of rule-based systems, (3) application level protocols for agent to agent, and (4) user models for the support of effective end user to agent interactions.

In our research, the knowledge base of an agent is represented in the form of a rule base. An agent starts with a set of knowledge and this knowledge is added to incrementally as the agent learns. In order to facilitate learning, an agent is designed to observe "situation-action" pairs. These actions are as defined in the designer's goal graph for the problem domain; an example action for the meeting scheduling problem would be *requesting for the postponement of a meeting*. Situations are characterized in terms of the data items required to achieve an action as provided by the path model, and external input from end users. Which situations an agent should observe is an open ended question. As new knowledge is added to the knowledge base of the agent, its validation and verification are important to ensure that the rules added do not "contradict" the existing rules. We have been working on both static and dynamic validation of rule-based systems [10].

For software agents, agent-to-agent communication is a key issue. The language used for agent communication, the

protocol adopted to suit the underlying application domain, and the acceptability of such protocols and languages among researchers are essential in the development of agent-based technology. We have developed the Consensus protocol which is designed to be used by a set of cooperating agents [8]. An interface agent is a software system that interacts with both the end user and agents designed to solve a specific problem. For an interface agent to communicate effectively with the end user, we need suitable user models in which the interface agent can develop appropriate responses or requests. This is also one of our areas of research.

Summary

We have been working on the above research topics for the last several years. Each of our research initiatives has been taken up as an independent piece of work. Currently, we are working on integrating these individual research initiatives within the context of software agents designed to aid the office professional.

References

- [1] Lashkari, Metral, M. and Maes, P. Collaborative interface agents. In *Proceedings of the 12th National Conference on Artificial Intelligence (AAAI-94)*, pages 444-450, July 1994.
- [2] Maes, P. Agents that reduce work and information overload. *Communications of the ACM*, 37(7):30-40, July 1994.
- [3] Malone, T.W. How do people organize their ideas? Implications for the design of Office Information Systems. *ACM Trans. on OIS*, 1(1): 99-112, Jan. 1989.
- [4] Hewitt, C. Offices are open systems. *ACM Trans. on OIS*, 4(2):271-287, July 1968.
- [5] Ho, C.S., Hang, Y. and Kuo, T.S. A society model for office information systems. *ACM Trans. on OIS*,

4(2):104-131, April 1986.

- [6] Grossner, C., Preece, A., Gokulchander, P., Radhakrishnan, T. and Suen, C.Y. Exploring the structure of rule-based systems. In *Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI-93)*, 1993, pp. 704-709.
- [7] Grossner, C., Preece, A., Radhakrishnan, T. and Newborn, M. Sharing data in a multi-agent system. Submitted to *International Journal of Cooperative Information Systems*, Dec 1995.
- [8] Clark, Grossner, C. and Radhakrishnan, T. Consensus and Compromise: planning in cooperating agents. To appear in *International Journal of Cooperative Information Systems*, 1996.
- [9] Grossner, C., Chander, P.G., Preece, A. and Radhakrishnan, T. Revealing the structure of rule-based systems. To appear in *Journal of Expert Systems Research and Applications*, 1996.
- [10] Preece, A., Grossner, C. and Radhakrishnan, T. Validating dynamic properties of rule-based systems. To appear in *International Journal of Human Computer Studies*, 1996.

Clifford Grossner is now working as a Member of Scientific Staff for Nortel Technology in Montreal. In addition, Cliff is an Assistant Adjunct Professor at Concordia University. Cliff obtained his Bachelor's and Master's degrees in Computer Science from Concordia University in 1980 and 1982, and he obtained a Ph.D. from McGill University in 1995, also in Computer Science. Current interests include Artificial Intelligence, Agent Architectures, Distributed Systems, User Interface Design, and Distributed AI.

Dr. T. Radhakrishnan received his computer education from the Indian Institute of Technology, Kanpur, India. Since 1975, he has been teaching and conducting research at Concordia University in Montreal, Canada where he is now a professor. He has supervised more than thirty graduate students and his interests are in user interface agents, cooperating agents, multimedia applications in distributed processing, and the social aspects of computing. His interests are primarily in applied research. He is co-author of a book on Computer Organization, published by Prentice Hall, that has gone through four editions in the last twenty years.

CSCSI/SCEIO Membership

I wish to join CSCSI/SCEIO and receive Canadian Artificial Intelligence

- Web Access Only (\$30.00* Cdn./yr.)
- Printed Copy and Web Access (\$40.00 *Cdn./yr.)

I am a student

- Web Access Only (\$15.00* Cdn./yr.)
- Printed Copy and Web Access (\$15.00* Cdn./yr.)

I am a member of CIPS

- Web Access Only (\$25.00* Cdn./yr.)
- Printed Copy and Web Access (\$30.00* Cdn./yr.)

Name _____

Mailing
Address _____

E- mail
Address _____

Please mail your membership to:

CIPS
430 King Street West, Suite 106
Toronto, Ontario
M5V 1L5

Phone: (416) 593 - 4040

Fax: (416) 593 - 5184

For more information contact CIPS or a member
of the executive.

*Includes Applicable G.S.T.

How to Get (and keep) a Research Grant

by Ian H. Witten

Revised and updated by Janice I. Glasgow

"How to get a research grant" initially appeared as a feature article in Canadian Artificial Intelligence, No. 24, July 1990. The updated version was prepared in March 1996.

Abstract

Ian Witten a écrit l'article intitulé "Comment recevoir une subvention de recherche" juste après avoir siégé trois ans sur le comité de sélection des subventions du Conseil de recherche en sciences naturelles et en génie (CRSNG) du Canada. Cette version révisée de cet article a été préparée par Janice Glasgow après un passage similaire sur le comité du CRSNG. La mise à jour du document était motivée par le climat en continuel bouleversement entourant la subvention à la recherche au Canada. À cause des coupures budgétaires fédérales, recevoir (et conserver) une subvention du CRSNG devient de plus en plus difficile. Ainsi, la qualité de la proposition pour obtenir une subvention est une composante encore plus cruciale du processus de subvention qu'elle ne l'était au moment de la rédaction de l'article original.

1. Introduction

Ian Witten wrote the article "How to get a research grant" after having just spent three years on the Natural Sciences and Engineering Research Council of Canada (NSERC) grant selection committee for Computer and Information Science. This revised version of the paper was prepared by Janice Glasgow after a similar term on the NSERC committee. The updating of the document was motivated by the ever-changing climate for research funding in Canada. Because of federal budget cutbacks, getting (and keeping) an NSERC grant is becoming increasingly more difficult. Thus, the quality of a grant proposal is an even more crucial component of the funding process than it was at the time of the original article. The job of an NSERC grant selection committee member is arduous, but worthwhile and interesting. It provides an opportunity to see most of the computer science research going on in Canadian universities, and although committee members suffer from terrible information overload they do gain an appreciation for the breadth and excellence of the work being carried out. The most painful part of the job is the extraordinarily inadequate amount of money that granting agencies have to work with, and the need to reduce, or even cancel, funding for many worthwhile projects because of the extremely competitive nature of the process and the dire shortage of funds. For most Canadian researchers, an NSERC research grant (formerly referred to as operating grant) provides the core funding for their research; thus, losing this funding could significantly affect their research career. The second most painful part of the job, which prompted Ian Witten to write

the initial article, is seeing how many capable researchers remain unfunded because they are not aware of how to write good research proposals; interesting projects go by the board because they are inadequately presented. In the hotly competitive environment in which the grant selection committee operates, it is inevitable that inadequate or poorly-prepared research proposals receive little benefit of the doubt. The onus lies squarely on the applicant to provide clear evidence on which the committee can base a decision. This article summarizes what the authors have learned about how to write research proposals, through having had to evaluate many such proposals — good and bad — over the years. Provided certain mistakes are avoided, the excellence of a proposal hinges on the originality and impact of the research, and this article will not help you with that! But there are some simple guidelines that should be followed to generate a well-presented proposal. Several factors are taken into account when evaluating a research grant application:

- the quality of the proposed research;
- the quality of the researcher;
- the training of highly qualified personnel; and,
- the need for funds.

The quality of a research proposal stems from a well-planned, long-range program; this is addressed in Section 2. The quality and impact of the work must be reflected in a well-written document, as addressed in Section 3. A researcher's reputation, which is built over time, strongly influences how his or her proposal is seen, and Section 4 gives some advice on how to present yourself in the best light. It also discusses the importance of training students and researchers. Section 5 sketches how a grant selection committee actually works. Section 6 gives some information about refereeing research grant applications, an activity that — though often seen as a chore — is absolutely essential for the health of the discipline. This article is targeted at proposals for NSERC computer science research grants, which are intended to provide basic support for an individual researcher's work. However, many of the ideas presented apply to any research proposal. The NSERC research grants program stresses long-term funding for individual researchers' programs, rather than funding for a particular project; other granting programs may have different priorities. It should be emphasized that the views expressed here do not necessarily reflect the official policy of NSERC or any other body.

2. Research ideas

To do research, you must formulate a question (or hypothesis) that your work will strive to answer (or achieve). This should not just be an isolated question, but one related to a long-term research theme that evolves over a substantial part of your career — certainly much longer than the four-year term of the average research grant. Moreover, you may begin with not just a single question, but a few (although not too many) that differ in risk, and hence potential value. You must be able to evaluate these research questions yourself, so that you can pick good ones and present them clearly. Also, they should fit together into a coherent program with definable long- and short-term objectives.

2.1 Generating research questions

In computer science, it should not be hard to come up with good research questions. The field is young and there is much to do. Technology changes constantly, radically altering the boundaries of what is feasible, and new possibilities for research are continually opening up. There are fertile opportunities in replicating previous work more systematically and in greater depth, i.e., rational reconstruction of programs, experimental evaluation and comparison, tightening up existing conceptual frameworks, and so on. There are plenty of avenues for research in computer science! Nevertheless, it may still be difficult to generate specific research questions. Just trying to think them up can easily lead to mental blocks. Good ideas often come from reading, discussing, explaining, and, (best of all) teaching what someone else is doing. Group discussions can be fertile breeding grounds for new ideas. Read current research papers in areas that interest you, force yourself to present and explain them to others, and ideas will strike you. In our experience, it is not the authors' suggestions for future research that spawn the best questions; those suggestions are ones the authors themselves have not been able (or bothered) to pursue successfully. People who write research papers generally know far more about what they are doing than the reader, and problems that they identify but leave unsolved may well be really tough! It may be better to capitalize on your more detached position to escape from the author's mind-set and think more laterally about what he or she is working on.

2.2 Relating ideas to a theme

Strive to give your research some breadth of scope and long-term continuity, without appearing to spread yourself too thin. This is not easy to achieve, but merits serious effort. As months stretch into years and years into decades, your results should build up and strengthen each other so that real progress can be perceived towards answering significant and difficult questions. An alternative research strategy is more opportunistic; identify problems that others have formulated but failed to solve properly, and jump in with a new technique of which they are unaware and show how it can be applied. This kind of predatory strategy is often adopted by those

who have special knowledge of — or an obsession with! — a particular viewpoint or tool. One danger is that to a person with a hammer, everything looks like a nail; you may be blind to the inappropriateness of your pet methodology for many of the applications you investigate. Another is that while good and plentiful results may be obtained quite quickly, over the long term the research program as a whole may take on a scrappy, uncoordinated, character. Thus, it may be better to focus your long-term efforts on particular kinds of problems rather than on solving a string of small, weakly related puzzles.

2.3 Safe versus risky research

By its very nature, it is hard to plan research, and any avenue — no matter how good it seems — may turn out to be sterile, unfeasible, or simply incorrect. On the other hand, beware of promising to work on too many things, for your proposal will be criticized as being “unfocused.” Reviews of proposals sometimes state explicitly that the evaluation would have been higher if fewer ideas had been included. You can spoil a good proposal by adding more to it. Propose a mix of questions to work on — some short-term and obviously answerable, others long-term, more risky, but potentially more valuable. It is important to take chances in research, and equally important to be aware of the risks being taken. Kuhn (1970) defines “normal science” as research firmly based upon one or more past scientific achievements, achievements that are acknowledged by the scientific community to supply the foundation for further practice. He contrasts this with “scientific revolutions” that question and restructure established practice: “non-cumulative developmental episodes in which an older paradigm is replaced in whole or in part by an incompatible new one.” Kuhn's distinction, which is designed for a grand scale (like Copernicus's or Einstein's revolutions in physics), also applies in miniature at the level of the individual researcher: safe versus risky research. Be aware of this distinction and propose work on different levels.

2.4 Evaluating research ideas

You have to evaluate your own ideas, assess their strengths and weaknesses, sharpen them, and present them in the most favourable light. When you specify a goal, how will you know if you reach it? Of course, you may not expect to attain your goals, but if by chance you achieve complete success you ought to be able to tell that you have done so! Many research proposals specify goals that are so vague they could never be reached (or already have been — sometimes it is difficult to tell). It is essential to formulate goals sufficiently precisely so that it will be possible to determine when they have been reached, and (if it is not completely obvious) you must explain how you will know. Goals that are stated in a way that makes it difficult to decide if they have already been achieved, or ones that are clearly completely out of reach, will destroy the credibility of any proposal. Are your goals worthwhile, and why? The

onus is on you to convince your reviewers that, if you are successful, you will have accomplished something worth doing. Of course, you might fail. But if you do succeed, it is reasonable to ask what contributions will have been made to scientific knowledge (i.e., results that others can build on) or to practice (i.e., general techniques that others can apply). If you intend to prove a theorem that no one cares about, or tackle a particular application in a way that does not shed light on others, then research funding will be difficult to obtain. Have you identified a rational approach to tackling your chosen problem? Of course, it is tough to plan research; strategies may change depending on results you obtain along the way. But it is essential to have some idea of what methods you will apply to attempt to solve your proposed problem. You must plan something more concrete than just "waiting for inspiration" or even "reading about the problem (and waiting for inspiration)"! Since research is evidently unpredictable and difficult to plan, alternative lines of attack are sometimes useful.

2.5 Cross-discipline research

By its nature, computer science research often crosses disciplines; historically, computing has often been coupled with mathematics, electrical engineering, and psychology. More recently, we have seen proposals that relate computer science research with other disciplines, such as biology, mechanical engineering, linguistics, and education. Unfortunately, research that crosses disciplines often suffers in the NSERC research grant process; grant selection committees generally favour research that promotes their own discipline and even with evaluations from other committees, it is often difficult to assess the impact and significance of work that involves multiple areas of expertise. It is extra tough to write a proposal that is cross-disciplinary. First, you may not know *a priori* what committee the proposal will end up in (this is determined by NSERC in consultation with the committee chairs). Thus, you do not know what audience you are addressing, e.g., a proposal written for a computing committee might have a different focus than one written for a math committee! For this reason, it is often best to focus your proposal in one discipline, and use the cross-disciplinary aspect to help demonstrate the significance and impact of the research. For example, you might propose to do research in knowledge representation, but state that the work can be motivated and evaluated in terms of cognitive psychology criteria. It is important, however, to stress the impact and significance it has on the focus discipline since it will be experts from that community who will be judging your proposal (with possibly some external advice from the secondary discipline). NSERC does have a special committee for judging interdisciplinary proposals. However, your research must span at least three disciplines to be eligible for consideration by this committee. This document does not address such research. Despite the problems discussed above, the computing committee does recognize the importance of research that crosses borders. It certainly strengthens your

own research if it can be demonstrated that it has significance in other areas. Just be careful that you are not just applying known technology to a new problem domain; your proposal will be judged primarily on how you are contributing to the advancement of computer science.

3. The research proposal

Carefully read and follow all instructions provided with the grant application. Your application may be disqualified if you do not follow specifications, such as font size and format. As well, committee members do not appreciate reading material presented in a non-standard way — and the last thing you want to do is make an overextended reviewer unhappy! Given that you have the ideas, how do you describe them and make them sound worth funding? First, remember that you are describing your ideas to a colleague, not a business promoter; be positive and optimistic about your work, but avoid making a sales pitch. Your basic problem, as pointed out by Bundy (1988), is threefold. It is to convince the selection committee that:

- you have identified a well-formulated goal;
- attaining this goal would be a significant contribution to computer science;
- you have a good chance of attaining the goal with the resources available.

One of the complexities of writing a research proposal is that you have to address two audiences: 1) the internal and external reviewers, who are likely to be experts in your field of interest, and 2) the remainder of the committee, who are computer scientists but may have limited knowledge of the area in which you are working. Your proposal must have something for both audiences; there should be enough depth and detail to please the expert, but you must also convince the non-expert of the importance and impact of your proposed research. In particular, *the abstract should be written for the general computer science audience.*

3.1 Describing your ideas

Your proposal will be evaluated by experienced, and probably sympathetic, researchers. They have been through it themselves and understand the difficulty of proposal writing and conducting research. They realize that research is difficult to plan and do not expect to be able to glean every last detail about what you want to do just by reading the proposal. But they can tell a lot about you, and the way you think, from your writing. They expect you to have thought pretty hard about your ideas, and to have worked conscientiously to explain and present them as clearly and straightforwardly as possible. They want to give you a chance, but they must justify it to themselves (and to others). It is up to you to provide the evidence for a positive decision. Do not make your research description a sales brochure. The kind of people who evaluate it will probably react negatively to salesmanship. On the other hand, you must make it clear that what you propose to do is worthwhile and has a good chance of success. Acknowledge difficulties honestly; do

not try to pull the wool over the readers' eyes. If there are snags or potential problems, say so; reviewers will be impressed by your candor. If the difficulties are ones that they have not thought of, they may be impressed by your intelligence, too. It is only reasonable to assume that you have thought through your proposal more thoroughly than the reviewers have; consequently, if they see problems that you do not seem to have noticed, then they will be less than impressed with your efforts. It would reflect badly on your proposal if you were to describe obstacles that seem completely insurmountable, but you presumably will not be proposing work that you judge to be unfeasible. You cannot really lose by being honest about the problems you expect to encounter.

3.2 The researcher

As well as having good ideas, you must explain why you are fully — and perhaps uniquely — qualified to carry them out. Of course, since they are your ideas, you automatically have a head start over others. You must know the background for the work, i.e., the relevant literature in the field and how it relates to your research. Your proposal should contain a brief (one page) section that reviews this prior work.¹ Space will not permit a comprehensive literature survey, and you will be unable to include many references. That makes it all the more important to select judiciously, thereby demonstrating that you have solid knowledge of the field, and the ability and good taste to make the very best use of limited space. Do not be overly introverted; mention other work besides your own. It gives a bad impression to have all (or even most) references to yourself or to a closed circle of collaborators. Avoid being involved in a small clique of researchers who publish in the same places and whose results are referred to only by one another. Make sure your literature review is up-to-date, including recent publications in the area if they exist, and if not, consider explaining why (lest your proposal be seen as belonging to a bygone era). For a senior researcher, the “track record” of work (especially recent) in the area will obviously play an important role in the evaluation of the proposal. Do not waste space by listing your own papers twice, once in the reference list and again in the personal data form (PDF) or resume. Invent a way to cross-reference from the proposal to the PDF (e.g., by numbering entries in your publication list and using letters to identify other references in the proposal). If you are a relatively junior researcher who does not have an extensive track record, do not fret — your proposal will be judged relative to others at similar stages in their career. New applicants are evaluated primarily on their potential. In these cases where there is generally little track record, the quality of the proposal is even more crucial. Those who are being considered for renewal for the first time are judged on whether they have begun to develop an independent research program. Publications that are based only on research done during a Ph.D. or postdoc may not convince the committee of this. When considering subsequent renewals, evidence of

an independent long-term research program is essential. This does not mean that team research is not deemed important, quite the contrary; recent communications from NSERC have emphasized that collaborative and concerted activities should be actively encouraged and that grant selection committees should give credit to effective research interactions. This was not always perceived to be the case in the past, when it was considered that more emphasis was placed on sole authored papers and there was sometimes a lack of appreciation for strong multi-authored papers. The current sentiment is that creativity and innovation are at the heart of all research advances, whether made individually or as part of a group effort. In an NSERC research grant proposal, it is useful for the applicant to clearly define his/her personal contributions to joint research (there is a place for this in the PDF). There is not much you can do to boost your track record, other than presenting your accomplishments fairly and accurately (see Section 4). However, an important source of information concerning whether you are the right person for the job is the understanding and insight you display when presenting and discussing the research in the proposal. The payoff for explaining your ideas clearly, eloquently, insightfully, and candidly cannot be stressed too strongly.

3.3 Structure of a proposal

Any proposal should review the context of the research, articulate the goals that will be pursued, summarize relevant prior work, describe a research plan, and give some indication of why the research is useful. A progress report on completed research is also required for renewals. The background should be brief and set the context for the proposal in terms of an overall research theme. The goals should project a fabric of interwoven ideas, augmenting and contributing to each other, with a mix of short- and long-term, safe and risky, research. One useful technique is to break down an overall goal into several interacting sub-goals or objectives — but beware of proposing too much. Also ensure that sub-goals fit together into a cohesive research program, rather than what might appear to be a collection of unrelated projects. The majority of the proposal should be devoted to a careful description of the research objectives and the methodology whereby these objectives will be achieved. For the research plan, you should at least know how you are going to start out and have some ideas for future options. Do not schedule research too firmly or too far into the future; that is unrealistic. Be prepared to describe alternative scenarios for the later stages, which hinge on how the early research turns out. It might be useful to look at the problem from different points of view (theory, simulation, experimental implementations, human behaviour, etc.) provided you have the background and resources to carry this out. Be mindful of the need to evaluate your ideas, not just develop and implement them. If successful, what will be the effect of the research; how will others be able to build on the results? Will they contribute to the advancement of science, or merely develop a wonderful

“look Ma, no hands” system that leaves others no better off? Sometimes such systems leave others worse off if they cannot replicate or follow up on your results.

3.4 The progress report

If you have previously been funded, you must summarize the progress made under the previous grant: What specific contributions have been made, where have they been published, and who has taken them up, applied them, or developed them further? If you cannot demonstrate that you have made good use of a previous grant, the chances of your grant getting renewed will clearly be diminished. Publication delays may mean that your recent work has not yet appeared in print. Some papers based on work related to research funded from other sources may also have appeared; while this is a good sign, it should not be confused with research progress stemming from the NSERC grant itself. Fortunately, your proposal will be evaluated by experienced researchers, who understand the publication business (particularly the fact that ambitious research projects may require multiple sources of funding) and that delays in publication often occur. Be careful of your balance between progress and proposed research. In the past, a senior researcher might have succeeded with a proposal that mostly describes past work and then says something to the effect of “I plan to continue doing more or less the same thing.” In the current competitive environment, this no longer works. Even those with an excellent track record must include a detailed and well-written research proposal for the work they plan to do. Note that some of the items that indicate progress have a natural place in the PDF. Use the space allotted for describing your proposal wisely.

3.5 Budget

One of the criteria for a research grant is need. The unfortunate circumstance is that almost all researchers “need” more money than the committee can possibly provide. Despite this, budgets should be carefully prepared and justified. Most importantly, be realistic; some applicants have lost credibility by proposing unrealistic budget items. For example, if you are a new researcher, then it might be difficult (but not impossible) to justify funding for a postdoc. Similarly, if you are a senior researcher who has not supervised graduate students recently, then a budget that includes support of several students might be questioned. Make certain that your budget items are allowable by NSERC. Since the guidelines change year to year, check these for each new application. A budget should reflect the actual cost of carrying out your proposed research — even if such funding is not available from the granting agency. Note that the committee cannot give you more money than you ask for, so do not be too modest in what you request! Researchers are often funded from multiple sources, suggesting that there may be some question of need. In general, additional sources of funding are seen as a positive reflection on the researcher and his/her work. However,

there are cases where the amount of time that the applicant has for the new proposal might be a consideration. It is important that you make clear how work being proposed is unique and distinct from research funded from other sources.² One major budget item affecting the need for funds is the support of graduate students. For those who are not supporting students, either by choice or by circumstance (e.g., there is no graduate program at their institution), the perceived need for funds is typically less than for those supporting students.

3.6 Preparing the proposal

It is important to take great care to present your ideas clearly; the people who evaluate your proposal are busy, even overloaded, volunteers. Thus, they will probably react negatively to any signs of sloppiness in either thinking (fuzzy goals, inadequate background, unacknowledged problems, etc.) or presentation (poor proofreading, spelling errors, infelicitous formatting, incomplete references, etc.). If you are not sufficiently motivated or excited by your ideas to spend time honing the content and presentation of your proposal, you cannot expect a sympathetic hearing from whoever is obliged to evaluate it. Reviewers do not generally look favourably on superficial or “popularized” proposals. Make sure there is plenty of technical content for them to pick up on. If the proposed research is highly technical, do not shy away from reflecting the technicalities in the proposal. There is nothing wrong with including a few equations if necessary, even diagrams (though be careful, especially with the latter, to ensure good use of space). Have others read your proposal before submitting it. Encourage them to be critical, to emulate a tough reviewer, to pick out holes and ambiguities, to misunderstand where at all possible — in short, to look for ways to dislike the proposal. Probably the actual reviewers will be more sympathetic, but you should prepare the proposal to withstand a critical onslaught. Also, if possible, read other proposals — particularly those that have been successful in the past — to pick up clues of how your presentation could be improved. Some departments and faculties have research committees or officers that provide assistance in preparing proposals; take advantage of any such resources that are available to you. It will probably be four years before you have another chance to apply for a grant, so it takes a long time to recover from funding cuts resulting from a poorly-prepared proposal. This is particularly true for first-time applicants, where the implication of a good versus a bad proposal is funding versus non-funding. Proposals are restricted to a certain number of pages. You do not have to cover them all, but a clear exposition of complex ideas takes a certain amount of writing and most successful proposals occupy the majority of the allotted space. Do not buck the system by using a tiny typeface (chances are, your proposal will become ineligible for funding if it does not conform to the proposed standards). Prepare the proposal in a straightforward way that will not upset the reader. It is better to get the bulk of your message

across properly than to try to communicate the whole thing in detail and fail completely! Do not try to cheat by sending in more than the maximum number of pages as the proposal will be truncated before it even reaches the reviewers and important parts may be lost. Think of it as an exercise; part of the test is seeing how effectively you can work within specified constraints.

4. The personal data form (PDF)

Along with the research proposal, you will have to submit a PDF giving information about your qualifications, the positions you have held (list them in reverse chronological order), the number of students you have supervised, your publication list, and other information. Make sure you document industrial and consulting work, along with any "technology transfer" activity. Consider showing thesis titles and publications by students under your supervision, listing your undergraduate and graduate students and postdocs by name (along with their career progress), summarizing your refereeing activity, your published reviews, and so on. What you decide to include reflects your priorities and general professionalism; it will be used by the reviewer to build a picture of you and your work.

4.1 The publication list

This is perhaps the most important part of the PDF and you should take great care in preparing it. Gather together under separate headings papers in refereed journals (clearly indicating their "accepted" or "published" status), papers in refereed conference proceedings, other refereed items like book chapters, books, non-refereed articles, and so on. Make sure you provide correct and complete references to your papers. It is essential to be scrupulously honest when preparing the publication list. Reviewers react negatively to any suspicion of cheating. Make sure you know for certain which of your publications are refereed. Only list publications for the specified time period. Avoid duplication in your publication list. If a conference paper was subsequently published as a book chapter, for example, choose one section in which to include it and note with that entry that it also appeared elsewhere. In general, if it is a reprint or a revision of an earlier paper, say so, and only list it once (you do gain credit from the fact that someone evidently thought it was worth reprinting!). Avoid writing different papers with the same (or very similar) titles. Submitted papers should be collected together and clearly identified as such. People disagree on whether you should specify the journals to which your papers have been submitted. The argument in favour is that it gives readers a chance to judge whether you are submitting your work to appropriate places. On the other hand, it might be interpreted as an attempt to glorify yourself by association. Never succumb to the temptation to mislead reviewers on the status of submitted papers — it is quite possible that someone will check with the editor of the journal and discover deception (it happens). If a paper has survived one round of refereeing and been re-submitted for

a second, say so. If it has been accepted subject to minor corrections and approval by the editor, say so, giving the date of acceptance. If in doubt, spell it out. These remarks are intended as guidelines rather than rules, and in practice, there is some latitude in interpreting them. Some people prefer to list both conference papers and journal papers that are straightforward extensions of them separately, which is permissible so long as they are clearly cross-referenced. The refereed or non-refereed status of papers is sometimes not clear-cut, particularly in the case of invited papers — and ultimately of course, it is the quality of the material that counts, not where it appears. The most important thing is to be open and honest about the status of your work. If you are suspected of misrepresentation, your application will suffer and so will your reputation. Justify the journals and conferences where you are publishing, particularly if they are not ones that are easily recognizable by members of the committee. For example, interdisciplinary work might be best presented in forums that are accessible by both disciplines.

4.2 Explanation of research contributions

Make clear what your personal contribution is to joint publications. If a publication is primarily the work of a graduate student under your supervision — say so. If the order of authors reflects their respective contribution — say so. If anything is not clear, then a reviewer might assume the worst case scenario.

4.3 Contributions to the training of highly qualified personnel

The training of students and researchers is an important criterion in the assessment of a research grant proposal. As with publications, the focus is not just on quantity, but also on quality; one graduate student who goes on to be a first class researcher is obviously preferable to three unemployed graduates. Do not just list numbers; list thesis titles, say where students went after graduating, mention awards or publications they may have received under your supervision, list co-supervisors, etc. Doing so indicates that the training of students and researchers is a high priority to you. The merit of training is judged in conjunction with the quality of the researcher. Someone who has a poor track record, or a badly presented proposal, may not be considered the best person to train new researchers. In such cases, having supervised a large number of students might be thought of as a negative, rather than a positive, contribution.

4.4 Other evidence of impact

This section of the PDF allows you to present further evidence to the significance of your research. In particular, it is useful to list awards and honours related to your work. In some cases, you might wish to describe the significance of the honour if it is not a well known one, e.g., a listing of a best paper award could include the number of papers submitted and accepted to the conference. Prestigious invited

talks also support the significance of your research. Positions on program committees and journal boards suggest that your peers value your contributions. One obvious evidence of impact in computer science is the dissemination of software resulting from research. If you have written programs that are being used by other researchers or industry, this could be an important contribution. Consulting activities are also worth mentioning, if they relate to your research.

4.5 Delays in research

If there have been any significant reasons for delay in research, then it is may be useful to specify these in the PDF. The committee is sympathetic to the fact that there may be situations that may result in decreased research productivity, and this can be taken into consideration when evaluating your research record. Situations such as maternity leave or illness are valid reasons for a temporary slow down in activity. Administrative responsibilities may also have impaired your progress. It is important, however, that these delays be perceived as temporary (e.g., if your research has slowed down because you have had a term as an associate dean and have subsequently been appointed to a term as dean, then there is no reason to think that the situation will improve). Also, be careful not to cite delays that are perceived as a normal part of a researcher's responsibility (e.g., graduate student advising or program committee membership).

4.6 Additional material

You may have the opportunity to submit additional material, such as preprints or reprints, to support your application. Unfortunately, reviewers are often forced to guess the quality of a paper from the journal or conference in which it appears, but if you can, submit actual papers. This provides a welcome

opportunity to evaluate the research contributions more closely. Be sure to select reasonably recent work, and make it your best work! Do not include papers just because they have been published in prestigious journals. It may be better to choose good papers that have appeared in obscure places, or have not yet been published, as the reviewer will otherwise be quite unable to evaluate this work.

5. How a grant selection committee works

It helps to know a little about how a grant selection committee works. Following is a description of how the current (1995/96) Computing and Information Science grant selection committee operates; other NSERC committees tend to have similar procedures, but we cannot guarantee this. Unlike most other disciplines, computing has only one committee that handles all applications (e.g., Math is split into two subcommittees). The proposals reviewed by the computing grant selection committee come mainly from computer science departments of Canadian universities; applications may also come from other departments, such as business, library science, math, and electrical engineering. Certain research institutes (e.g., CRIM) and colleges have employees who are eligible for funding. As well, researchers in industry who are adjunct professors at a university may be eligible to apply for a grant.³ The computing committee currently has fourteen members. Each application is read carefully by nine members of the committee. Two or three members, who are especially knowledgeable about the relevant research area, are specifically assigned to each proposal as "internal reviewers" to evaluate the proposal thoroughly, read the additional material, and prepare and present a recommendation to the committee. The natural tendency is for internal reviewers to champion their applications where

- It is not clear what question or hypothesis is being addressed by the proposal.
- It is not clear what the outcome of the research might be, or what would constitute success or failure.
- The question being addressed is woolly or ill-formed.
- It is not clear why the question is worth addressing.
- The proposal is just a routine application of known techniques.
- Industry ought to be doing it instead.
- There is no evidence that the proposer has new ideas that make it possible to succeed where others have failed.
- A new idea is claimed but insufficient methodological details are given to judge whether it looks feasible.
- The proposer seems unaware of related research.
- The proposed research has already been done (or appears to have been done).
- The proposer seems to be attempting too much for the funding requested and the time-scale envisaged.
- The proposal is too expensive for the probable gain.

Table 1. Some reasons for rejecting a research proposal (adapted from Bundy, 1988)

merited, and the other committee members serve as a critical sounding-board for the representation. The meeting proceeds quickly — on average, there is less than ten minutes available per application (though in cases where there is disagreement, the discussion could last longer). Internal reviewers summarize the application, highlighting the applicants' credentials, what they propose to do, their evaluation, and finally their recommendation. If they judge the proposal to be good, then they will act as a proponent, trying to persuade the other committee members of the virtues of the case. You should strive to make it easy for them! Re-read your application and imagine someone having to defend it on your behalf in the space of a few minutes. Obviously you must highlight salient points in the summary: goals, prior achievements, objectives, research plan, evaluation methodology. Proposals are considered in categories; all new applicants are handled first, then applications from researchers whose current grants fall into the same funding range (\$0 - \$15,000; \$15,000 - \$30,000; >\$30,000) are considered consecutively. The recommendation of an internal reviewer includes a proposed amount for funding. Each committee member must balance a budget, thus a recommendation to increase the funding of one applicant might imply the decrease or non-funding of another grant application. As your representatives present your case, the other members of the committee are reviewing the application and external referee reports trying to assess the case and decide whether they can agree with the recommendations or not. They have studied it before, of course, but there may be hundreds of other applications and memories will need refreshing. Table 1, adapted from Bundy (1988), summarizes common reasons why proposals are rejected — bear these in mind as you prepare your proposal. There might be disagreement between the internal reviewers or with another committee member — maybe even an argument (it has happened)! As the discussion proceeds, the rest of the committee is silently scanning your application, listening, and thinking about it. Just imagine the impact of a poorly-prepared, scrappy proposal, and contrast it with the effect of a beautiful, tastefully-arranged document. The final recommendation for funding is done by a “Dutch auction” method. An initial funding amount is proposed; this is generally the highest of the amounts suggested by the internal reviewers, but a higher amount can be recommended by another member of the committee if they feel it is warranted. Voting then proceeds where only those who are designated readers (recall that there are nine readers and five non-readers for each proposal) have the right to vote. The amount incrementally decreases (usually by \$1,000 decrements) until a majority of eligible voters have their hands up. Anyone who is in conflict with a proposal (e.g., at the same university or a co-author) is a non-reader and must leave the room during the discussion and voting process. The final amount decided upon is only a recommendation to NSERC; given that the overall budget may be altered after competition week, proposed amounts may vary from

those eventually awarded.

6. Refereeing grant applications

Selection committees depend heavily on timely and carefully prepared reviews by outside members of the research community. Each application is sent to several external referees for evaluation. Some are suggested by the applicant, others by the committee. The responses are made available to all members and referred to frequently in the committee's deliberations. Refereeing other people's applications is widely perceived as a time-consuming chore, although it can be interesting. Ultimately, it is in our discipline's interests to have the fairest possible funding decisions, and conscientious reviews play a crucial role in this. For example, NSERC evaluates the functioning of the Computer and Information Science committee, and the computer science community at large, by the response rate to review requests; this is the kind of thing that helps whenever the committee makes requests for a larger slice of the cake (such as in the reallocation exercise that occurs every four years). If you care about the quality of computer science research and future funding for the discipline, you should feel obliged to contribute your share to the refereeing process.

It is important to prepare reviews thoughtfully and to the best of your ability. Unqualified praise gives the impression that you are trying to do the applicant a favour; unqualified criticism suggests that you have a biased view. In any case, it is helpful for you to summarize your previous knowledge of the applicant's work and your personal acquaintance of him or her, if any. One-line reviews give the impression that you have not taken time to reflect upon the proposal or evaluate it properly. On the other hand, no one wants to read a review that is longer than the proposal itself (yes, it does happen!). The best reviewers evaluate proposals carefully and summarize the contributions fairly, mentioning both positive and negative aspects and weighing the evidence for and against funding. Writing good reviews is just another aspect of your professionalism; it will be noticed and will enhance your reputation.

7. Conclusion

No amount of care and effort in preparing a research grant proposal will compensate for a weak research program. However, a poorly prepared proposal can prevent a strong research proposal from being funded at the level it deserves. The authors of this document hope that a better understanding of the NSERC review process can assist researchers in having a clearer idea of how to best present their ideas and contributions for future research grant competitions.

Notes


1. Some applicants prefer to combine the literature review with the description of proposed work. This is allowable and sometimes more effective than putting it in a separate section.

2. This should be done on the supplementary budget page, rather than in the research proposal itself.
3. Eligibility requirements vary from year to year, so check with NSERC if there is any uncertainty about the current policies.

Acknowledgments

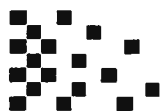
Ian Witten is grateful to Rick Bunt, Brian Gaines, Saul Greenberg, Carl Hamacher, and Helmut Jurgensen for making valuable comments on a draft of the original article. Janice Glasgow thanks Uri Ascher for his insightful comments on a draft of the revised version of the document, and Nancy Barker for her assistance in recreating the initial document.

References

- Bundy, A. (1988) "How to get a SERC grant," *AISB Quarterly* 65: 7-9.
- Kuhn, T.S. (1970) *The structure of scientific revolutions*. University of Chicago Press, second edition. 

Ian Witten is a Professor of Computer Science at the University of Waikato, New Zealand and the University of Calgary.

Janice Glasgow is a Professor of Computing and Information Science at Queen's University.



PRECARN UPDATE NOUVELLES DE PRECARN

Harry Rogers

PRECARN is a consortium of 35 industry partners that conducts pre-competitive research in the areas of Robotics and Intelligent Systems. PRECARN also administers the Institute for Robotics and Intelligent Systems (IRIS), one of the federal Networks of Centres of Excellence. The IRIS network engages 23 universities and 350 investigators, graduate students and other professionals. Combined, the two organizations will invest approximately \$75 million in collaborative R&D projects between now and the year 2000.

"Advanced technology has a key role to play in promoting sustainable growth in Canada's new knowledge-based economy. Advanced technology provides us with the opportunity and ability to build national prosperity using our most important resource: our people."


These were the words that Industry Minister John Manley addressed to an audience of university and industry representatives at a recent conference on developing the advanced technology economy. This bodes well for researchers in the fields of knowledge related technologies, including intelligent systems.

Phase 2 of IRIS, which ends in early 1998, is just passing its halfway mark but already planning has begun for a Phase 3. A national consultation process on the Phase 3 mission, the framework for the network, and its projects' content will

be commencing shortly. Persons interested in contributing to the consultation process are invited to contact PRECARN for a copy of the position paper as well as to learn of the location of the consultation meetings scheduled for early November across the country.

As well, the Technology-Gap program has attracted considerable interest from IRIS Principal Investigators. The T-Gap program has enabled Principal Investigators to obtain funding to develop prototypes using technologies that have been developed in their work. Demonstrations of these prototypes in such areas as Minimally Invasive Surgical tools, Stereoscopic Vision, and very high speed positioning devices are expected by the spring of 1997.

And finally, the PRECARN consortium will be opening another competition for new collaborative research projects in October. PRECARN contributes up to 40% of the eligible costs of projects. For further information on these matters, contact our offices or visit us at our web-site.

PRECARN Associates Inc., Tel: 613-727-9576, Fax:613-727-5672, Web: www.precarn.ca, E-mail: info@precarn.ca
Address: Suite 300, 30 Colonnade Road, Nepean, Ontario K2E 7J6 




BOOK REVIEWS CRITIQUES DE LIVRES

Edited by Graeme Hirst

Please note that no book reviews were received by press time. Readers who wish to review books for *Canadian Artificial Intelligence* should write, outlining their qualifications, to the book review editor, Graeme Hirst, Department of Computer Science, University of Toronto, Toronto, Canada M5S 1A4, or send electronic mail to gh@cs.toronto.edu or gh@cs.utoronto.ca. Obviously, we

cannot promise the availability of books in anyone's exact area of interest.

Authors and publishers who wish their books to be considered for review in *Canadian Artificial Intelligence* should send a copy to the book review editor at the address above. All books received will be listed, but not all can be reviewed. 

CALL FOR PAPERS

IASTED International Conference Artificial Intelligence and Soft Computing

July 27-August 1, 1997

Banff, Canada

SPONSOR

**International Association of Science and Technology
for Development (IASTED)**

In cooperation with:

**American Association for Artificial Intelligence
(AAAI)
Canadian Society for Computational Studies of
Intelligence (CSCSI)**

SCOPE

Artificial intelligence
Expert systems
Neural networks
Knowledge acquisition
Knowledge representation
Knowledge engineering
Distributed artificial intelligence
Image understanding
Logic programming
Advisory systems
User interface
Theory of neural networks
Modelling
Simulation
Architecture
Learning algorithms

Object-oriented techniques
Expert systems design procedures and
tools
Natural languages
Computational linguistics
Architectures for natural languages
Machine learning
Fuzzy expert systems
Approximate reasoning
Automated reasoning
Reasoning methods
Planning and scheduling
Applications, all areas including:
Engineering, Science, Business,
Economics, Power systems, Transportation, Decision making,
Management, Environmental systems, Banking, Government,
Education, Others

Algorithms
Intelligent databases
Intelligent control
Intelligent agents
Temporal and spatial reasoning
Fuzzy logic
Fuzzy systems
Verification, validation
Testing

INTERNATIONAL PROGRAM COMMITTEE

H. Adeli, Ohio State University, USA
E. Arce-Medina, National Polytech. Inst., Mexico
R. De Mori, McGill University, Canada
T. Feuring, WWU Muenster, Germany
N. Ishii, Nagoya Inst. of Technology, Japan
M. Jamshidi, University of New Mexico, USA
W.W.L. Keerthipala, Nanyang Tech. Univ., Singapore
S.-G. Lee, Hannam University, South Korea
Z.-N. Li, Simon Fraser University, Canada
D. Lukose, University of New England, Australia
N. Mani, Monash University, Australia

D. Martland, Brunel University, UK
A.R. Mikler, Ames Laboratory, USA
A.R. Mirzai, Iran Univ. of Science & Tech., Iran
M. Morando, CNR, Italy
D. Mulvaney, Loughborough, UK
N. Serbedzija, GMD FIRST, Germany
A. Sung, New Mexico Tech., USA
S. Valenti, University of Ancona, Italy
H. VanLandingham, Virginia Tech., USA
J. Wang, Chinese Univ. of Hong Kong, Hong Kong
S. Zhao, Governors State University, USA

SUBMISSION OF ABSTRACTS

Initial paper selection will be based upon abstracts. An abstract should present a clear and concise view of the motivation of the subject, give an outline of the paper, and provide a list of references. The abstract should not exceed 600 words. The International Program Committee will make the decision concerning the acceptance of the papers. Three copies of the abstracts should be received at the IASTED Secretariat ASC'97 (address below) by **January 6, 1997**. Authors should provide a maximum of five keywords describing their work and must include a statement confirming that if their paper is accepted, one of the authors will attend the conference to present it. Please include the full name, affiliation, full address, telephone number, fax number, and email address of the corresponding author. Final papers and registration must be received by **May 2, 1997**. Papers received after that date will not be included in the proceedings. The International Program Committee reserves the right to reject any final manuscripts if quality is poor.

IMPORTANT DEADLINES

Abstracts due January 6, 1997
Acceptance notification March 3, 1997
Camera-Ready Manuscripts
and Registrations May 2, 1997

For more information or to be placed on the mailing list,
please contact:

IASTED Secretariat ASC'97

#80, 4500-16 Avenue NW, Calgary, AB Canada T3B 0M6

Tel: 403-288-1195 Fax: 403-247-6851

Email: iasted@cadvision.com, Web site: <http://www.iasted.com>