Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

Document
D-91-10

MAAMAW'91

Pre-Proceedings of the 3rd European Workshop on "Modeling Autonomous Agents and Multi-Agent Worlds"

Donald D. Steiner, Jürgen Müller (Eds.)

August 1991

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH

Postfach 20 80
D-6750 Kaiserslautern
Tel.: (+49 631) 205-3211/13
Fax: (+49 631) 205-3210

Stuhlsatzenhausweg 3
D-6600 Saarbrücken 11
Tel.: (+49 681) 302-5252
Fax: (+49 681) 302-5341

Deutsches Forschungszentrum für Künstliche Intelligenz

The German Research Center for Artificial Intelligence (Deutsches Forschungszentrum für Künstliche Intelligenz, DFKI) with sites in Kaiserslautern und Saarbrücken is a non-profit organization which was founded in 1988 by the shareholder companies ADV/Orga, AEG, IBM, Insiders, Fraunhofer Gesellschaft, GMD, Krupp-Atlas, Mannesmann-Kienzle, Philips, Siemens and Siemens-Nixdorf. Research projects conducted at the DFKI are funded by the German Ministry for Research and Technology, by the shareholder companies, or by other industrial contracts.

The DFKI conducts application-oriented basic research in the field of artificial intelligence and other related subfields of computer science. The overall goal is to construct *systems with technical knowledge and common sense* which - by using AI methods - implement a problem solution for a selected application area. Currently, there are the following research areas at the DFKI:

- Intelligent Engineering Systems
- Intelligent User Interfaces
- Intelligent Communication Networks
- Intelligent Cooperative Systems.

The DFKI strives at making its research results available to the scientific community. There exist many contacts to domestic and foreign research institutions, both in academy and industry. The DFKI hosts technology transfer workshops for shareholders and other interested groups in order to inform about the current state of research.

From its beginning, the DFKI has provided an attractive working environment for AI researchers from Germany and from all over the world. The goal is to have a staff of about 100 researchers at the end of the building-up phase.

Prof. Dr. Gerhard Barth
Director

MAAMAW'91 Pre-Proceedings of the 3rd European Workshop on "Modeling Autonomous Agents and Multi-Agent Worlds"

Donald D. Steiner, Jürgen Müller (Eds.)

DFKI-D-91-10

MAAMAW '91

Third European Workshop on
Modeling Autonomous Agents and Multi Agent Worlds

5 - 7 August 1991
Dorint Hotel, Kaiserslautern, Germany

The purpose of this workshop is to stimulate exchange and discussion of research in the field of multi-agent systems. A multi-agent system consists of at least two agents that are engaged in some tasks that may require coordination, cooperation and/or competition. An autonomous agent has its own goals, capabilities and knowledge. The actions of an agent occur in the context of other agents that may have structures and strategies different from the agent's own. Multi-agent problems arise when several autonomous agents share a common environment. These problems may result from limited resources, shared or competing goals, etc. While classical Distributed Artificial Intelligence (DAI) was mainly concerned with distributed problem solving leading to a common global goal, DAI has recently been moving closer to a multi-agent perspective, allowing agents to have unrelated goals. However, we emphasize multi-agent systems of all sorts from simple to complex agents and agent organizations.

PROGRAM CHAIRMEN

Yves Demazeau (F - LIFIA/IMAG/CNRS)
Eric Werner (D - University of Hamburg)

PROGRAM COMMITTEE

John Campbell (UK - University College of London)
David Connah (UK - Philips Research Laboratory)
Rosaria Conte (I - CNR Rome)
Mauro Di Manzo (I - University of Ancona)
Jean Erceau (F - ONERA Chatillon Bagneux)
Jacques Ferber (F - LAFORIA/University of Paris 6-7)
Julia Galliers (UK - University of Cambridge)
Heikki Hammainen (SF- Helsinki University of Technology)
Walter Van De Velde (B - AI-Lab / VUB University of Brussels)
Frank Martial (D- GMD Sankt Augustin)
Jean-Pierre Muller (CH - IMI University of Neuchatel)
Jürgen Müller (D - DFKI Kaiserslautern)
Martin Nilsson (S - Swedish Institute for CS, Stockholm)
John Perram (DK - University of Odense)
Nigel Seel (UK - STC Technology Ltd. Harlow)
Genevieve Teil (F - CSI / Ecole des Mines de Paris)

LOCAL ORGANIZERS

Donald Steiner (Siemens AG / DFKI)
Jürgen Müller (DFKI)

SPONSORS

Deutsche Forschungsgesellschaft (DFG)
Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI)
Daimler-Benz AG
Siemens AG

PROGRAM

Third European Workshop on
Modeling Autonomous Agents and Multi Agent Worlds

5-7 August 1991
Dorint Hotel, Kaiserslautern, Germany

Sunday, 4 August 1991

17:30 Registration

19:00 Reception

Monday, 5 August 1991

8:00 Registration

8:30 Introduction and Welcome

9.00 THE ROLE OF REPRESENTATION IN INTERACTION: DISCOVERING FOCAL POINTS AMONG ALTERNATIVE SOLUTIONS
Jeffrey Rosenschein (invited speaker) & Sarit Kraus
Hebrew University, Jerusalem (IS)

10:00 SANP : A COMMUNICATION LEVEL PROTOCOL FOR NEGOTIATIONS
Man Kit Chang & Carson C. Woo
University of British Columbia, Vancouver (CDN)

10:45 - Break -

11:00 SOCIAL PLANS : A PRELIMINARY REPORT
Anand S. Rao, Michael P. Georgeff & Elizabeth A. Sonenberg
Australian AI Institute (AUS) & University of Melbourne (AUS)

11:45 COLLABORATIVE PLAN CONSTRUCTION FOR MULTIAGENT MUTUAL PLANNING
Ei-Ichi Osawa & Mario Tokoro
Sony Computer Science Laboratory Inc., Tokyo (J)

12:30 - Lunch -

15:00 COOPERATIVE PROBLEM-SOLVING GUIDED BY INTENTIONS AND PERCEPTION
Birgit Burmeister & Kurt Sundermeyer
Daimler-Benz AG Research Institute, Berlin (D)

15:45 A MULTI-AGENT ANALOGICAL REPRESENTATION FOR PHYSICAL OBJECTS
Lucia Maria Gambardella & Marc Haex
IDSIA, Lugano (CH)

16:30 Discussion

16:45 - Break -

17:30 Departure from hotel to Edenkoben

18:30 Winetasting and Dinner

21:30 Return to hotel

Tuesday, 6 August 1991

9:00 VARIABLE COUPLING OF AGENTS TO THEIR ENVIRONMENT: COMBINING SITUATED AND SYMBOLIC AUTOMATA
George Kiss (invited speaker)
The Open University, Milton Keynes (UK)

10:00 TOWARD AN ARCHITECTURE FOR ADAPTIVE, RATIONAL, MOBILE AGENTS
Innes A. Ferguson
University of Cambridge (UK)

10:45 - Break -

11:00 ECO-PROBLEM-SOLVING MODEL: RESULTS OF THE N-PUZZLE
Alexis Drogoul & Christophe Dubreuil
LAFORIA - Universite Paris VI, Paris (F) & CERT-ONERA, Toulouse (F)

11:45 EXPLOITING EMERGENT BEHAVIOUR IN MULTI-AGENT SYSTEMS
Peter Wavish
Philips Research Laboratories, Redhill (UK)

12:30 - Lunch -

15:00 HOW TO MOVE (PHYSICALLY SPEAKING) IN A MULTI-AGENT WORLD
Jean-Claude Latombe (invited speaker)
Stanford University (USA)

16:00 A DISTRIBUTED ARTIFICIAL INTELLIGENCE VIEW ON GENERAL PURPOSE VISION SYSTEMS
Olivier Boissier & Yves Demazeau
LIFIA, Grenoble (F)

16:45 - Break -

17:00 REAL-TIME PERFORMANCE OF INTELLIGENT AUTONOMOUS AGENTS
Anne Collinot & Barbara Hayes-Roth
LAFORIA/IBP Universite Paris VI (F) & Stanford University (USA)

17:45 Discussion

18:30 Departure from hotel to Ritterkeller, Frankenstein

19:00 Banquet

22:00 Return to hotel

Wednesday, 7 August 1991

9:00 PANEL
THE DYNAMICS OF KNOWLEDGE AND ORGANIZATION IN MULTI-AGENT SYSTEMS

- ORGANIZATIONAL ACTIVITY THROUGH CONVERSATION IN OPEN SYSTEMS
  Chisato Numaoka
  Sony Computer Science Laboratory Inc. (J)

- A SYSTEM FOR BELIEF REVISION IN A MULTI-AGENT CONTEXT
  Aldo Franco Dragoni
  University of Ancona (I)

- ON BEING RESPONSIBLE
  Nick Jennings
  University of London (UK)

10:30 - Break -

10:45 PLURALITY: EXPLAINING WHY DAI SYSTEMS WORK AND WHY THEY DON'T.
Les Gasser (invited speaker)
Universite de Paris VI and CSI/Ecole des Mines (F)

11:45 TOWARDS A SEMANTICS OF DESIRES
Georges Kiss & Han Reichgelt
The Open University, Milton Keynes (UK) & University of Nottingham (UK

12:30 - Lunch -

15:00 DEPENDENCE RELATIONS AMONG AUTONOMOUS AGENTS
Christiano Castelfranchi & Maria Miceli & Amedeo Cesta
CNR Institute of Psychology, Rome (I) & University of Rome (I)

15:45 A GAME THEORETIC APPROACH TO DISTRIBUTED ARTIFICIAL INTELLIGENCE AND THE PURSUIT PROBLEM
Ran Levy & J.S. Rosenschein
Hebrew University, Jerusalem (IS)

16:30 Discussion & Conclusion

17:00 End

The Role of Representation in Interaction: Discovering Focal Points among Alternative Solutions

Sarit Kraus*
Jeffrey S. Rosenschein
Computer Science Department
Hebrew University
Givat Ram, Jerusalem, Israel
sarit@cs.huji.ac.il, jeff@cs.huji.ac.il

June 27, 1991

Abstract

Representation can be critical in enabling agents to interact effectively. Alternative methods for representing and reasoning about the world can radically affect the ability of agents to reach cooperative solutions. Focal points are examined as a particularly compelling example of the importance of representation; we consider the algorithms that might be used by resource-constrained agents in discovering prominent solutions of their interaction.

*Kraus is also affiliated with the Graduate School for Library Studies.

1 Introduction

There are various kinds of interactions that sophisticated human agents can easily handle, and yet whose formal representation is problematic. The inability to formally represent crucial features causes difficulties for conventional interaction techniques. For example, humans are often able to recognize a particular structure to a problem that helps them coordinate their actions to mutual benefit. There is sometimes a "special" attribute of a coordinated act, and both agents, in recognizing the specialness, can choose their actions suitably. Schelling [13] called these prominent coordinated actions "focal points." Intuitively, a focal point is a conspicuous point of agreement to which interacting agents gravitate.

In this paper, we discuss the concept of focal points, emphasizing how they could be discovered by an automated agent and used by several to coordinate their actions. A number of other researchers in Distributed Artificial Intelligence (DAI) have also recognized representation issues as critical for effective interaction, including Durfee [4], Werner [15], Singh [14], and Cohen and Levesque [1].

In Section 2 we present the notion of focal points, including illustrative examples, and in Section 2.2 we discuss their properties at greater length. Standard representations (such as those of game theory) are unable to exploit focal points in finding coordinated actions, and we examine this problem in Section 3. Central to the idea of focal points is the ease with which they are found; this suggests that it is important to take into consideration the time that passes as agents carry out their reasoning. In order to capture the passage of time, we make use of Step Logic [6], discussed in Section 4.

In Section 5 we present our focal point algorithm, along with the specific rules by which focal points are discovered. Because the algorithm will not necessarily result in a single candidate focal point, there is a need to reduce the candidate set as much as possible. The resolution of multiple solutions is discussed in Section 5.4. Finally, in Section 6, we discuss how the presence or absence of knowledge among the agents affects their focal point computation.

2 Focal Points

2.1 Examples of Focal Points

Schelling proposes no formal definition of focal points. We follow his lead by demonstrating the concept via examples.

Imagine two players on a TV game show. The emcee explains to the players the simple rules of the game: each is to go to a separate, private room, where they will be handed a pile of 100 $1 bills. They are each, in isolation, to divide the single pile into two piles, A and B, with any distribution of bills between the piles. Then their distributions will be announced, and if they are identical (i.e., the players' A piles are the same size, and their B piles are the same size), they will each win a Mercedes. If their distributions are not identical, they will

receive the consolation prize (a home version of the game).

Readers may, at this point, wish to consider for a moment what choice they themselves would make if they were playing this particular game.

Schelling [13] discussed his experiments with a game of this type, and found that the overwhelming majority of players chose to divide the 100 $1 bills into two equal piles, 50 bills in each. Informal runs of this game, done by this paper's authors, seem to point to the same conclusion: most people are drawn to the 50–50 split, even though there are another 100 possible choices (if we allow the empty pile). The reasoning goes something like this: "Since success in the game requires us only to anticipate each other's choice, and since at one level of analysis all the choices are equivalent, I must look for any uniqueness that will distinguish a particular option in both of our minds, and rely on the other agent's doing likewise." In this case, equivalence between piles is a property that is true only of the 50–50 split, making it, in Schelling's vocabulary, a *focal point.* Intuitively, as was mentioned above, a focal point is a prominent point of agreement to which interacting agents gravitate.

A similar example has two contestants asked to write down, secretly, "some positive integer," with a prize to be awarded if they both write down the same positive integer. Although there are infinitely many winning solutions to this interaction, Schelling found that most people tended (unsurprisingly) to write down the number 1. Given the range of choices, this number has the unique property that it is the only one without a predecessor in the set. Thus, it is a focal point.

Focal points can also arise in non-numeric domains, as in the following example. You have parachuted into the countryside represented by Figure 1 (the bent lines represent roads, the small boxes are houses, and there is a river, spanned by a bridge, horizontally bisecting the middle of the picture). Another person whom you want to join up with has parachuted into the area also, but you are (unexpectedly) out of communication with her. Where do you go to meet up with one another?

Most people, when presented with this case, are reported by Schelling to choose the bridge as a meeting place. There is no guarantee that your partner will go through the same line of reasoning, but the bridge is a prominent solution, a focal point, and one towards which participants gravitate.

2.2 Competing Focal Points

There are a number of intuitive properties that seem to qualify a given agreement as a focal point. Among these properties are uniqueness, symmetry, and extremeness. Our formal solution below will encode these intuitions into a logic that could be used by an agent.

Even when we consider these special properties, more must be done to identify focal points. *There are bound to be competing potential focal points, since there is* something *unique about any solution.* Another fairly strong contender for a solution in the original game presented above is the choice of 0 bills in A, and 100 bills in B (or vice versa). Of course, it is precisely the "vice versa" aspect of this solution that makes it appear less appealing in comparison with the 50–50 split.

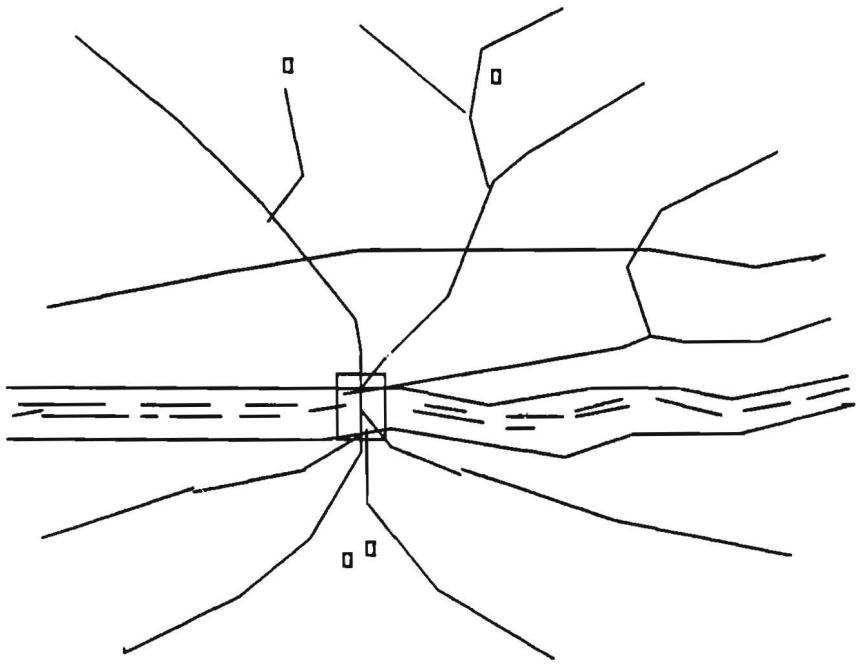


Figure 1: The Parachute Problem

Any solution, though, will have something to recommend it—but the less obvious that something is, the less attractive the alternative becomes, precisely because it becomes less obvious that the other agent will duplicate our line of reasoning. For example, the choice of 10–90 recommends itself, since it is the only choice where the number of tens in both piles is a perfect square (1 squared and 3 squared), and where at the same time the first pile is smaller than the second. And of course, we might choose 16–84 as our split, reasoning that our partner will realize, as we did, that these are the only years in the 20th century (whose last two digits add up to 100) that have seen the election of United States presidents with the same number of letters in their last names (Wilson in 1916 and Reagan in 1984).

This is a farfetched example, but the point should be clear: a focal point is produced not only because it satisfies one of the intuitive principles mentioned above, but because it seems computationally more accessible—it seems more likely that the other agent will also recognize the point than that he will recognize competing points.

2.3 The Role of Communication

One way of altering the prominence of a focal point, or creating a new one, is through communication. Schelling presents two communication scenarios of his TV game show. In the first, one contestant shouts out "60–40!" as he is being led to his isolation booth. The emcee decides not to stop the game, but warns the players against any further communication.

The second contestant is now faced with a new prominence. While it is true that 50–50 is an attractive alternative, it is given real competition by the 60–40 possibility. Especially if the other player knows that the content of the shout could be understood, they both have reason to ascribe prominence to this solution point.

Another of Schelling's scenarios has the emcee doing the communication. Imagine that no shouting has occurred on stage, but after you are in your isolation booth you are visited by the emcee. He says, "I have visited the other player, and I am giving you the same advice that I gave him: choose 37–63." With that, he departs. Again, a new piece of information has been introduced, and with it the possibility that the other agent will settle on this suddenly prominent option.

What has happened in both these communication scenarios is that the very act of communication, even though it is extremely limited, has given new prominence to particular solutions. After all, in the first scenario, the 60–40 split became the only solution with the property of having been broadcast by the other player. In the second scenario, the 37–63 split became the only solution with the property of having been advised by the emcee. This is similar to the prominence attached to the 50–50 split in our first example, the only solution with the property that the piles are equivalent.

In this paper, we will not be discussing communication in focal point interactions; this is a subject for future research.

2.4 Automated Agents and Focal Points

Although the concept of focal points was originally introduced with regard to human interactions, they have relevance for automated interactions as well. From a machine's point of view, a focal point is an instantiation of a variable in a statement or action. When automated agents are designed to operate in realistic domains, they will need to analyze interactions in a sophisticated way. Groups of agents might find focal points useful because it can help coordinate actions when communication is difficult or impossible.

For example, consider the case of automated agents that are working together on Mars, but have lost communication with one another (e.g., their radio frequency has developed interference). They would like to meet again so as to reestablish their line of communication, but need to independently decide where the meeting will take place. The agents could not establish an *a priori* protocol for how to get back together, because they did not have sufficient information about what the terrain would be like. The search for a focal point meeting place would be a natural mechanism for solving this problem.

In addition, if an automated agent needs to interact with humans it will be helpful to act in a "natural" way that the human can also anticipate and coordinate with. For example, one of the Mars workers above might be a human, and the automated search for a focal point meeting place mirrors his own thought processes. Another case might be that of a robot cleaning up an auditorium, coming across a left article, and having to reason about where to put it so that it will be found by the owner the following day.

More fundamentally, focal points provide a test case for representation and reasoning in

multi-agent domains. They are a hard problem that conventional techniques cannot address. By studying focal points, we gain insights into representing and reasoning about multi-agent encounters.

3 The Failure of Standard Representations

Focal points are an interesting interaction problem to study precisely because they provide clear examples of the failure of simple interaction representations.

Consider the original problem given above, with two contestants on a TV game show. The problem of dividing up a stack of 100 $1 bills can be represented very easily using a payoff matrix, with $40,000 marked in all the boxes down the diagonal and zero everywhere else. The failing of this representation is that it does not allow the agents to reason about anything other than the relationships among the payoffs, and these relationships are wholly uninteresting. There are 101 payoffs of $40,000, but there is no other way of reasoning about why one action is better than any other.

Reasoning about the matrix layout, which might help in this case, is wholly outside of Game Theory's use of the payoff matrix [10]. In fact, one agent (in Game Theory) cannot be sure that the other agent sees the matrix in exactly the same way that he does. That is, there is some mapping between the other agent's choices and the matrix, but we don't know what that mapping is—the $40,000 payoffs may be scattered throughout his personal representation of the matrix (as long as there is only one per row and one per column).

The same issue arises in the problem where the agents are to choose a positive integer: one representation would have high payoffs down the diagonal and zeros everywhere else, but other functionally equivalent representations would have the payoffs scattered around the matrix, and the agents can't (in Game Theory) use a rule like "choose the upper left-hand corner." There is no common view of the matrix.

The solution to the problem cannot simply be to introduce an ordering on the matrix, because in real-life encounters the matrix really isn't commonly perceived by the agents: it is the interaction that is commonly perceived, and the matrix is only an internal representation of an external reality. If, for example, two agents (A and B) meet, and each agent has two potential actions, A_{lift}, A_{break}, B_{push}, B_{clear}, then who can say that A's internal representation will match B's, and how safely could they hope to employ any simple rule that relies on the particular ordering they have attached to these moves?

Standard logic also fails to provide the solution to focal points. Computational complexity seems central to identifying focal points. Not only must a solution to a given problem satisfy a property like uniqueness in order to qualify as a focal point, it must also be easier to find than other solutions with similar properties. It is therefore necessary to model the computational process itself in the reasoning procedure as we search for focal points. Classical first order logic does not model the computational process. We turn, instead, to a modification of first order logic, called *step logic*, that deals explicitly with the passage of time as an agent reasons.

There is some related work in artificial intelligence that addresses the issues of the passage of time during the reasoning process. In [8], [7], and [12], decision-theoretic approaches are used to optimize the value of computation under uncertain and varying resource limitations. In these works, deadlines and the passage of time while reasoning are taken into consideration in computing the expected computational utility. Dean and Boddy [2] formulated an algorithmic approach to solution of time-dependent planning problems by introducing "anytime algorithms" that capture the notion that utility is a monotonic function of deliberation time.

Although we have chosen not to use these tools in examining focal points, it appears that they might someday be modified appropriately for the task. Using decision-theoretic techniques to find focal points might be especially suitable in cases where alternative outcomes have natural associated utilities. This is left for future research.

4 Discussion of Step Logic

Our current project employs the formalism of "step-logics," introduced by Elgot-Drapkin, Miller, and Perlis ([3, 6, 5]) where inferences are parameterized by the time taken for their inference, and in which these time parameters themselves can play a role in the specification of the inference rules and axioms.[1] Step-logics offer a natural representation of the evolving process of reasoning itself. A *step* is a fundamental unit roughly characterized by the time it takes the agent to draw a single inference.

Observations, which are inputs from the external world, may arise at the beginning of a discrete time-step. When an observation appears, it is considered a belief in the same time-step. Apart from his observations at the beginning of step i, the only information available to the agent is a snapshot of his deduction process completed up to and including step $i-1$. During step i the agent applies all available inference rules in parallel, but only to beliefs at step $i-1$ (denoted by $\mathcal{F}acts_{i-1}$); new beliefs thus generated through applications of inference rules are not available for use in further inference until step $i+1$. For example, consider the following reasoning (shown is an application of modus ponens) from step i to step $i+1$.

i: White(c271); House(c271); White(e99); House(e99); House(e31);
Now(i); White(x) $\wedge$ House(x) $\rightarrow$ Big(x) ...

$i+1$: Big(c271); Big(e99);
White(c271); House(c271); White(e99); House(e99); House(e31);
Now($i+1$); White(x) $\wedge$ House(x) $\rightarrow$ Big(x) ...

In effect, step-logics are first-order logics suitably modified to include a $Now(i)$ predicate, where the value of i changes at the end of a time-step.

[1] Step logics have also been used for planning in deadline situations [9].

5 The Focal Point Algorithm

The intuition behind our focal point algorithm is that the agent, at each step i, will look for candidates in the domain that have certain properties (like uniqueness). If something in the domain has the property, it is a focal point at step i. As time goes on, new beliefs are derived (e.g., through modus ponens), and the domain over which the search is being conducted also expands (through observations or consideration of new conjunctive properties). Then the search for candidate focal points is repeated—and an old focal point may, given the new information, no longer be one. The search for focal points is cut-off at some depth of computation, depending on time constraints, at which point the agent resolves competing focal points to the best of his ability.

Let us now consider the details of the above process. We first consider the way in which the agent models the (changing) domain, then the rules that qualify a candidate as a focal point. Finally, we consider the ways in which an agent resolves competing focal points.

5.1 Domain of Consideration

Before the process starts the agent is given two finite sets enumerating the domain constants (one, *Pred*, is a set of predicates, and the second, *Term*, is a set of term constants) over which the focal point computation is going to be done initially. Both lists can grow as the computation progresses.

TV Show Example: The vectors that sum to 100, with no element less than 0, can be given as an initial finite domain over which properties will be discovered.

It should be noted that these finite sets represent the *explicit* knowledge of the agent, not its implicit knowledge. For example, an agent may implicitly be aware of the infinite set of positive integers, but for the moment only be considering the finite set of integers from 1 to 500. As time goes on, numbers above 500 may come under the explicit scope of consideration.

5.1.1 Addition of Term Constants

There are two mechanisms for adding new explicit terms. The first is observation, where new term constants are observed over time (e.g., a new bridge is observed). The second mechanism is the use of inductive rules, such as a successor rule that generates new integers or a rule that generates new primes.

Example 1: At step i, the domain includes Bridge(C125). At step $i+1$ we have Observe{Bridge(C237)}. At step $i+2$ we then have C237 in *Term*.

Example 2: If $Int(x) \rightarrow Int(x+1)$ is a rule at step i, and $Int(5)$ is known at step i, then at step $i+1$ the agent will know $Int(5+1)$. Assuming that the agent has the requisite procedure attached to the symbol $+$, he will (in step $i+2$) add the term 6 to *Term*.

5.1.2 Addition of Predicate Constants

Consider an agent searching for focal points. When he starts, he considers attributes that might be held by only a single object in his domain. For example, there might be only one object that is Red. However, if such a unique object does not exist, then he may consider conjunctions of attributes. For example, there might be only one House that is Red. We want to capture this intuition in our algorithm.

When the process starts, *Pred* is equal to the finite set of predicates provided to the agent. At the second step, the agent considers binary conjunctions of predicates from the original list. At step three, he considers ternary conjunctions of predicates from the original list, and so on. The following lines describe the evolution of Pred through successive steps.

$$\text{step 1: } Pred_1 = \{\text{domain constant predicates and their negations}\} = \{P_1, \neg P_1, P_2, \neg P_2 \ldots\}$$

$$\text{step 2 : } Pred_2 = \{\text{binary combinations of predicates of } Pred_1\} = \{P_1 \wedge P_2, P_1 \wedge P_3, P_2 \wedge P_3, \ldots\} \cup Pred_1$$

$$\text{step 3 : } Pred_3 = \{\text{ternary combinations of predicates of } Pred_1\} = \{P_1 \wedge P_2 \wedge P_3, P_2 \wedge P_3 \wedge P_4, \ldots\} \cup Pred_2$$

5.1.3 Explicit and Easily Computed Knowledge

We want agents, in their search for focal points, to consider both explicit knowledge and "obvious" knowledge that is easily computed from their databases. For example, if "less than" is a predicate that the agent is considering, and both 5 and 6 are terms of which he is aware, then we want the agent to use the knowledge that 5 is less than 6, even though this fact is not explicitly represented in his database.

We therefore use a special notation to signify that a fact is "known" at the previous level. We write $\in^*$ to mean that the fact is either explicitly listed in *Facts* at level i, or that it can be simply computed over the constant terms *Term* known at level i.

The question of what can be simply computed is domain dependent, as well as agent dependent. There is an analogy here with the idea of "operational" in the Explanation Based Learning literature [11]. Checking "less than" might be operational in some machines; in other machines, deciding in a game of chess whether a given board position is reachable from the current state might be operational because of specialized hardware.

5.2 Focal Point Rules

In this section we present the actual rules by which an agent identifies candidates for focal points. We make no claims for completeness here. These rules provide good coverage of the Focal Point examples in [13], but additional rules may be appropriate in other cases.

Identification of focal points is a two stage process. First the agent identifies candidates by looking for meta-characteristics of objects, such as uniqueness. Second, the agent resolves competing candidates to the best of his ability (using other rules) and decides on one or more focal points.

5.2.1 Uniqueness

An object may be a focal point if it is the only object with a given property. Formally, if in $i-1$ we have $P \in \mathcal{P}red_{i-1}$, and there exists an $x \in \mathcal{T}erm_{i-1}$ such that

$$P(x) \in^* \mathcal{F}acts_{i-1} \forall y \in \mathcal{T}erm, y \neq x[P(y) \notin^* \mathcal{F}acts_{i-1}],$$

then in step i we will have

$$\text{Unique}(x, P, i).$$

Note that Unique is a "meta-predicate" that does not itself appear in the *Pred* set. Note also that the term x is considered unique with respect to the predicate P; this will be important later when competing focal points must be resolved.

Example: This rule would be applicable in the case where we know about only one Bridge, namely C125.

Both x and y can be vectors, in which case they will be denoted by $[x]$ and $[y]$. Another example of uniqueness (using equality on elements of a vector) is the following: $P([x,y]) \equiv x = y$ where the domain is defined to be vectors such that $Sum([x,y]) \equiv x + y = 100$. This causes us to choose the vector [50,50] over all others whose elements sum to 100.

5.2.2 Uniqueness Complement

Lack of information can also cause a solution to be prominent.

An object may be a focal point if it is the only object without a given property. Formally, if in $i-1$ we have $P \in \mathcal{P}red_{i-1}$, and there exists an $x \in \mathcal{T}erm_{i-1}$ such that

$$P(x) \notin^* \mathcal{F}acts_{i-1} \forall y \in \mathcal{T}erm, y \neq x[P(y) \in^* \mathcal{F}acts_{i-1}],$$

then in step i we will have

$$\text{Unique-Comp}(x, P, i).$$

Example: This rule would be applicable in the case where we know that everybody in the domain is a member of the Democratic Party, except that we have no information one way or the other about John. Although we don't know whether or not John is also a member, this lack of knowledge causes him to be prominent.

5.2.3 Centrality

Another meta-predicate is the concept of Centrality, the intuitive property of a central point around which a domain (or sub-domain) is symmetric.

An object may be a focal point if it is a central object within a given domain. Formally, if in $i-1$ we have $P \in \mathcal{P}red_{i-1}$, and there exists an $x \in \mathcal{T}erm_{i-1}$ such that

$$P(x) \in^* \mathcal{F}acts_{i-1}$$

$$\forall y \in \mathcal{T}erm, y \neq x \wedge P(y) \in^* \mathcal{F}acts_{i-1},$$

$$\exists z \in \mathcal{T}erm, z \neq y \wedge P(z) \in^* \mathcal{F}acts_{i-1},$$

$$\text{such that } y - x = x - z$$

then in step i we will have

$$\text{Central}(x, P, i).$$

Example: In the range between 0 and 100, the number 50 is Central (where P is the predicate Integer).

5.2.4 Extreme

An object can sometimes be prominent because it is the highest object, or the tallest, or the smallest, among the elements of the domain.

An object may be a focal point if it is an extreme object in a totally-ordered domain. Formally, if in $i-1$ we have $P \in \mathcal{P}red_{i-1}$, and there exists an $x \in \mathcal{T}erm_{i-1}$ such that

$$\forall y \in \mathcal{T}erm_{i-1}, y \neq x, (P(x,y) \in^* \mathcal{F}acts_{i-1} \wedge P(y,x) \notin^* \mathcal{F}acts_{i-1}),$$

then in step i we will have

$$\text{Extreme}(x, P, i).$$

Example: In the range between 1 and 10000, the number 1 is Extreme-Total (with the predicate P being "less than").

5.3 Dealing with Functions

All of the rules above can be generalized by using, instead of x and y, $f(x)$ and $f(y)$, functions that return values given the x and y terms or vectors. First, we must assume that the agent has been given a third finite list (in addition to *Pred* and *Term*) that enumerates the domain of functions: *Func*. Then, as an example, we could write the Extreme property as follows: if in $i-1$ we have $P \in \mathcal{P}red_{i-1}$ and function $f \in \mathcal{F}unc_{i-1}$, and there exists an $x \in \mathcal{T}erm_{i-1}$ such that

$$\forall y \in \mathcal{T}erm_{i-1}, y \neq x, (P(f(x), f(y)) \in^* \mathcal{F}acts_{i-1} \wedge P(f(y), f(x)) \notin^* \mathcal{F}acts_{i-1}),$$

then in step i we will have

$$\text{Extreme}(x, P, i).$$

The original rule above is then the case when the function f is the identity function.

Consider the following example:

$$Famous(Smith), Famous(Jones), Famous(Brown), Famous(\text{Father-of}(Smith))$$

Using our original focal point rules, neither Smith, Jones, nor Brown would be a prominent solution. However, extending the technique using functions, Smith becomes a prominent solution (since he is the only one with a famous father).

The set of functions *Func* will grow over time both through observation, and through composition. When the process starts, *Func* is equal to the finite set of functions provided to the agent. At the second step, the agent considers binary compositions of functions from the original list. At step three, he considers ternary compositions of functions from the original list, and so on.

$$\text{step 1: } Func_1 = \{\text{domain constant functions}\} = \{F_1, F_2, \ldots\}$$

$$\text{step 2 : } Func_2 = \{\text{binary combinations of functions of } Func_1\} =$$

$$\{F_1 \circ F_2, F_1 \circ F_3, F_2 \circ F_3, \ldots\} \cup Func_1$$

$$\text{step 3 : } Func_3 = \{\text{ternary combinations of functions of } Func_1\} =$$

$$\{F_1 \circ F_2 \circ F_3, F_2 \circ F_3 \circ F_4, \ldots\} \cup Func_2$$

5.4 Computing Focal Points—The Resolution Rules

The rules above specify when an object is unique, or extreme, etc.; they do not relate directly to the question of when the object is actually a focal point. We thus need a rule to use in tying together these attributes with the notion of focal point.

The most straightforward approach is to relate each of the meta-predicates above with the focal point attribute:

$$\frac{i: \quad \text{Unique}(x, P, i)}{i+1: \quad \text{FocalPoint}(x, i)}$$

$$\frac{i: \quad \text{Unique-Comp}(x, P, i)}{i+1: \quad \text{FocalPoint}(x, i)}$$

$$\frac{i: \quad \text{Central}(x, P, i)}{i+1: \quad \text{FocalPoint}(x, i)}$$

$$\frac{i: \ \text{Extreme}(x, P, i)}{i+1: \ \text{FocalPoint}(x, i)}$$

These rules of course may not supply us with a unique focal point, since there could be a term that satisfies Unique, another that satisfies Unique-Comp, etc. There could even be two separate terms that are Unique with respect to different predicates. There is still utility for the agent in discovering the set of focal points, since even if the choice is made among them probabilistically, there is an increased chance for coordination among the agents.

We will not attempt here to provide additional rules that guarantee a single focal point. Instead, we illustrate that one could introduce additional rules so as to reduce the size of the focal point set.

It is critical to resolve among focal points so that ones that are discovered more easily have higher priority. Step logic provides us with a natural tool for dealing with this. Using step logic, there are several mechanisms for relating priority to complexity; we here present one.

A focal point might be generated (given the above rules) at a given level, then not be a focal point at a subsequent level. The agents look for focal points only up to a certain level k. At this level, there might be several competing focal points that are still valid (e.g., arising from different rules, or from different predicates). As an initial winnowing mechanism, the focal points that were generated earliest are kept and the others discarded.

Example: In the range between 1 and 10000, the number 1 is Extreme-Total (with the predicate P being "less than"), and 10000 is Extreme-Total (with the predicate P being "greater than"), after the first step.

If the domain of considered integers grows at each step, 1 will still be extreme while 10000 will no longer be extreme. Thus, at the end of the process, 1 will be chosen since it has been "extreme" for the longest period. This disambiguates between the two extreme ends of a finite domain that is growing in only one direction.

The algorithm only considers "term-property" pairs; if a term was a focal point because of some property at level i, then was no longer a focal point because of that property at level i+1, then again became a focal point because of a different property at level i+2 (and remains a focal point until the end), then it is considered to have been generated at level i+2.[2]

The intuition is that, since the other agent may not go as deep in the deduction as we have in looking for a focal point, we are more likely to match the other agent by taking the earliest focal point. It is the solution that we still believe in most likely to have been reached by the other agent.[3]

[2]The idea behind looking at term-property pairs in order to establish the first appearance of a focal point is that once a focal point has disappeared because of other terms with the same property, its prominence because of that original property is completely negated.

[3]Other approaches present themselves, such as considering the *coverage* of a focal point e.g., if a term is a focal point for much of the deduction, though it is not at the final step, we would still consider it a likely

We may also choose to introduce rules that assign a priority to the meta-predicates (like Unique) so that, for example, a unique object gets priority as a focal point over an extreme object.

5.4.1 The Relation of Actions to Focal Points

There is an important relationship between the actions that are to be coordinated by interacting agents, and the focal points in a domain. This relationship can help agents resolve competing focal points.

Consider the following example (due to Schelling). Five candidates, Smith, Jones, Brown, Robinson, and White, are running for public office. In the first round of voting, the following results occurred:
$f(Smith) = 19$
$f(Jones) = 28$
$f(Brown) = 15$
$f(Robinson) = 29$
$f(White) = 9$

You are now to choose whom to vote for, given that you will win money if your choice wins the election, and that you have no other interest in the outcome.

Considering our focal point rules (with the function modification), there are two Extreme candidates for focal points: Extreme($White, <, i$) and Extreme($Robinson, >, i$). It seems reasonable that in choosing whom to vote for, the "greater-than" relation is of greater relevance than the "less-than" relation (Schelling's own experiments confirm this). We would like to capture this intuition in our Focal Point algorithm.

Other examples might assign "tallest" as the predicate most relevant to choosing a prominent basketball player, and "shortest" as the predicate most relevant to choosing a prominent jockey. There might also be relevant predicates used for the uniqueness property. For example, we might have a group of people, one of whom will be chosen to help us move furniture. We must choose one of the group, matching someone else's choice. It turns out that only one person is a swimmer, and only one (other) person is blond. Although the uniqueness criterion applies to both, being a swimmer is more relevant than hair color to the assignment of someone for a physical task.

Obviously, the representation of relevance is intimately connected with our representation of action. The details of this connection are beyond the scope of the current paper, and are left for future work.

---

solution. We could also then probabilistically weight the steps of the deduction, so that (for example) earlier steps receive more weight than later steps. These methods are left for future work.

6 Interesting Knowledge Categories

In the full paper, we will consider several different cases involving varying degrees of knowledge among agents, and the consequences that these different knowledge conditions impose on the search for focal points. It will sometimes be necessary (or simply appropriate) to modify the algorithm presented above so as to exploit the particular knowledge situation that exists.

Among the knowledge conditions that we will examine are the following:

1. There is common knowledge on everything, including the original axioms, run-time observations, the domain of predicates, terms, and functions, and the agents' computational "power" (i.e., how deep the search for focal points will go). We consider cases where the agents search to the same depth, and where they do not search to the same depth. In the case where search depth is identical, we show that if there is a set of focal points, the set will be generated identically by both, i.e., the procedure above is deterministic. With minor modification, the algorithm generates the same set for both even when the search depth is not identical, but is known.

2. There is common knowledge about everything other than the power of computation (i.e., how deep the focal point search will go). We then consider whether the set of focal points is monotonic, and whether the agents will reach the same focal point under certain conditions.

3. The agents have implicit common knowledge: the explicit expression (syntax) of the knowledge is not identical, but the closure under deduction is the same. We will examine whether the agents, under certain conditions, reach the same focal point.

7 Conclusions

We have presented the concept of focal point solutions to interaction problems, and discussed why conventional representation techniques are insufficient for focal point discovery. An algorithm was developed that allows for the uncovering of focal points through the use of step-logic, special inference rules, and sets of predicates, functions, and terms that change over time. The technique is particularly well-suited for modeling the time-dependent nature of focal point search. Further work is needed to characterize the knowledge situations when these techniques will converge, and for integrating theories of utility into the focal point calculation.

8 Acknowledgments

This research has been partially supported by the Israel National Council for Research and Development (Grant 032-8284), and by the Center for Science Absorption, Office of Aliya

Absorption, the State of Israel.

References

[1] Philip R. Cohen and Hector J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42(3), 1990.

[2] Thomas Dean and Mark Boddy. An analysis of time-dependent planning. In *Proceedings, AAAI88*, pages 49–54, 1988.

[3] J. Drapkin, M. Miller, and Donald Perlis. Life on a desert island. In *Proc. Workshop on The Frame Problem in Artificial Intelligence*, pages 349–357. American Association for Artificial Intelligence, 1987.

[4] Edmund H. Durfee. *Coordination of Distributed Problem Solvers.* Kluwer Academic Publishers, Boston, 1988.

[5] J. Elgot-Drapkin. *Step-Logic: Reasoning situated in time.* PhD thesis, Univ. of Maryland, 1988.

[6] J. Elgot-Drapkin and D. Perlis. Reasoning situated in time: basic co ıcepts. *Journal of Experimental and Theoretical Artificial Intelligence*, 1990.

[7] E. Horvitz, G. Cooper, and D. Heckerma. Reflection and action under scare resources: Theoretical principles and empirical study. In *Proceedings of IJCAI-89*, pages 1121–1127, Detroit, Michigan, 1989.

[8] E. J. Horvitz. Reasoning under varying and uncertain resource constraints. In *Proceeding, AAAI88*, pages 111–116, 1988.

[9] S. Kraus, M. Nirkhe, and D. Perlis. Toward fully deadline-coupled planning. In *Proc. 1990 DARPA Workshop on Innovative Approaches to Planning, Scheduling, and Control*, 1990.

[10] R. Duncan Luce and Howard Raiffa. *Games and Decisions, Introduction and Critical Survey.* John Wiley and Sons, New York, 1957.

[11] T. M. Mitchell, R. Keller, and S. Kedar-Cabelli. Explanation-based generalization: A unifying view. *Machine Learning*, 1(1):47–80, January 1986.

[12] S. Russell and E. Wefald. Principles of metareasoning. In *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning*, pages 400–411. Morgan-Kaufman, 1989.

[13] Thomas C. Schelling. *The Strategy of Conflict.* Oxford University Press, New York, 1963.

[14] Munindar Singh. Group intentions. In *Proceedings of the Tenth International Workshop on Distributed Artificial Intelligence*, Bandera, Texas, October 1990.

[15] Eric Werner. Toward a theory of communication and cooperation for multiagent planning. In *Proceedings of the Second Conference on Theoretical Aspects of Reasoning about Knowledge*, pages 129–143, Pacific Grove, California, March 1988.

SANP: A Communication Level Protocol for Negotiations

Man Kit Chang and *Carson C. Woo*
University of British Columbia*

Abstract

Organizations are distributed open systems where agents (workers, department, etc.) cooperate with one another to achieve organizational goals. However, conflict is also an inherent component of the process. In designing computer systems which aim to automate organizational activities, the conflicts among agents and how to resolve them need to be considered. Distributed Artificial Intelligent (DAI) researchers have long been interested in the question of how to resolve conflicts among cooperative agents in distributed problem solving environments. Negotiation has been suggested by many researchers as an important technique to resolve conflicts. However, most of the existing negotiation protocols used in DAI systems are inflexible. Actual protocols rarely take into account the result from negotiation research. In this paper, we propose a negotiation protocol, SANP (Speech Act based Negotiation Protocol), that is based on Ballmer and Brennenstuhl's speech act theory, and on the negotiation analysis literature from other disciplines. SANP is a flexible protocol that supports multi-level negotiations between two parties. In addition, SANP also allows third party arbitration if the parties involved in the negotiation cannot resolve the conflict themselves.

1 Introduction

Organizations are open systems that are composed of many interdependent and interconnected subsystems [17,18]. To accomplish organizational goals, work has to be distributed among agents (i.e., employees, departments, etc.) and their effort has to be coordinated in the direction of reaching the goals. However, each agent may have his own goals that may be inconsistent with goals of others. In an open system, agents receive incomplete and inconsistent information from outside of the system or from different parts within the system, and they develop their own viewpoints and beliefs that may be incompatible with the beliefs of other agents. Agents may also have different interpretations of the same situation caused by different personal knowledge and beliefs [15]. Hence, conflict is inevitable. Different kinds of conflict may arise: (1) conflict of interest when two agents compete for scarce resources; (2) conflict of value; (3) cognitive conflict when two agents differ in their thought processes or perceptions; and (4) goal conflict when desired outcomes of two agents differ [24]. This paper investigates the computer communication support needed for resolving conflicts by negotiation.

Distributed Artificial Intelligence (DAI) researchers have long been interested in how to resolve conflict among cooperative agents in distributed problem solving environments. However, Galliers

*Authors' address: Faculty of Commerce and Business Administration, The University of British Columbia, 2053 Main Mall, Vancouver, B.C., Canada V6T 1Z2. INTERNET: Carson_Woo@mtsg.ubc.ca for Carson Woo.

[13] points out that most existing DAI research projects avoid dealing with the problem of conflict among agents by assuming that cooperative agents are benevolent, agree with each other, and ready to adpot each other's goals. On the other hand, most of the projects that consider conflict assume the existence of a centralized authority for making decisions. In open systems, however, we cannot make this assumption.

Many methods have been proposed to reconciliate disparities among agents [5]. We are interested in the application of negotiation which has been suggested by many researchers as an important method to resolve multiagent conflicts in open systems [5,9,15,17,18,29]. Negotiation is a process of communication between agents in which conflicting goals are reframed, conflicting issues are identified and narrowed, alternative solutions are proposed, attacked, and defended, and agreements are reached and confirmed [23]. All these functions should be provided if we want to support negotiation between agents to resolve conflicts in an organizational information system. It is important that agents can present arguments that support their beliefs and goals, so that the other party in the negotiation process can understand the rationale of the beliefs or claims. The articulation of the arguments may help to identify issues on which the parties disagree and subsequently to solve the problem.

However, most existing negotiation protocols do not support all these functions of negotiation. Most of them are inflexible and are designed only to coordinate the execution of tasks. The primary aim of our research is to develop a flexible negotiation protocol that supports organizational work. SANP (Speech Act based Negotiation Protocol) is based on Ballmer and Brennenstuhl's speech act classification. SANP incorporates several strategies that the parties can employ in negotiation, such as "delay" and "appeal". In the next section, we shall review the related research projects that use negotiation to coordinate distributed problem solvers. Section three describes Ballmer and Brennenstuhl's speech act classification. Our proposed negotiation protocol is presented in section four. An example application will be given in section five. Section six outlines the usage of our protocol and future research work.

2 Related Work

The need for a communication protocol was recognized by researchers in the DAI field. Campbell and D'Inverno [6], for example, suggest that communication protocols can be used to control excessive communication between agents that uses a great deal of computer resource. Several researchers have proposed negotiation protocols and frameworks to support the cooperation of distributed problem solvers or agents. Some of these are Contract Net Protocol [9,19,29], Multistage Negotiation Protocol [8], Partial Global Planning [11,12], and PERSUADER [30].

Most of these protocols (except PERSUADER) view negotiation as a process of exchanging contracts or plans that result from planning or reasoning processes. In this case, each agent will have a better idea of what other agents plan to do and adjust their plans accordingly. These protocols suffer the problem of inflexibility in that they only allow agents to negotiate at a fixed level. That is, agents cannot negotiate the selection of a particular action, assumptions, and criteria of decision making. In addition, the design of these protocols is not based on any theory or model of negotiation, nor does it result from an analysis of negotiation processes. They are only designed to solve the problem at hand, and they are not general enough to be used in other problem domains. In particular, it is difficult to apply these protocols to the organizational environment to support the negotiation of distributed workers. For example, while establishing policies or constructing budgets, it may not be easy to gain consensus by only exchanging plans.

PERSUADER [30] does allow negotiation to be done at different levels (i.e., slightly more flexible). It uses a mediated negotiation model based on an integration of case-based reasoning and multiattribute utility theory to generate proposal and counterproposal based on the feedback from the other party. The most important feature is that it can generate persuasive argumentation to change the other party's mind. Although the system provides a good support for negotiation, it is not exactly a negotiation protocol, it functions more like a negotiation support system. Furthermore, it does not support many strategies that can be used in negotiation, such as delaying the discussion.

3 Speech Act Theory

The central idea of the speech act theory [2,25,26] is that someone uttering a sentence is not just saying things (i.e., describing a state of affairs), but he is actually doing somethings [20]. For example, when somebody says "I promise to come", he is not only uttering the statement "I promise to come", but he actually commits himself an action to be performed in the future.

3.1 Searle's Speech Act Classification

Searle's speech act classification is the most fully developed and widely used classification. Some researchers have applied it to the computer domain [7,34,35,36,38]. Others use Searle's taxonomy to analyze natural discourse (e.g., van Eemeren and Grootendorst [32] use it to analyze argumentative discussion). Searle [26] suggests that there are five basic kinds of actions that can be performed in speaking. He bases his classification on the goal that a speaker wants to achieve by his utterance and comes up with five categories:

Assertive:	which commits the speaker to the truth of the expressed proposition.
Directive:	which are attempts by the speaker to get the hearer to do something.
Commissive:	which commits the speaker to some future course of action.
Expressive:	which expresses the psychological state about the state of affairs.
Declarative:	which brings about change by virtue of the content of the utterance.

However, this classification is being criticized as "lack a principle basis, contrary to Searle's claims, it is not even build in any systematic way on felicity conditions" [20]. Another shortcoming of Searle's work is that he only analyzes the speech act from the speaker's point of view, there is no analysis of interaction between the speaker and the hearer (i.e., the analysis is basically unidirectional). It is difficult to use this taxonomy to analyze natural discourses because most of these are interactive in nature. Another difficulty in using Searle's classification is that no matter what the function of a speech act is, we have to put it in one of the five categories. It overloads the semantic of the categories and makes it difficult to comprehend what the speaker really wants. Therefore, another classification is needed if this theory is to be useful to constructing a communication protocol.

3.2 Ballmer and Brennenstuhl's Speech Act Classification

Ballmer and Brennenstuhl [3] were interested in classifying all German speech act verbs into Searle's speech act categories. They experienced enormous difficulties, however, because the classification required constant revision with the addition of more verbs. This caused a re-analysis of all previously classified speech act verbs and, hence, this process seemed endless.

To overcome this problem, Ballmer and Brennenstuhl employed a bottom-up approach, instead of the top-down approach suggested by Searle, to classify speech act verbs. This approach considers all speech act verbs in a German verbs dictionary, and groups the verbs that are similar in meaning into semantic categories. For example, the category "Dissent" has speech act verb like "have words with someone", "quarrel", etc. Semantic categories are grouped into models according to semantic similarity. For instance, the Struggle Model consists of the categories "Attack" and "Dissent". The categories in a model and the models themselves form a natural order based on temporal relation and degree of effectiveness. For example, "Defense" is temporally ordered after "Attack", and "Threatening" is more effective than "Warning".

The models are related to each other as well. They can be grouped into four linguistic functions: (1) Expression, (2) Appeal, (3) Interaction, and (4) Discourse. The Expression linguistic function contains speech act verbs for expressing emotional states. Appeal is a linguistic function directs towards to a hearer and try to affect his course of action. Interaction is the linguistic function that two parties engage in some form of negotiations. Discourse is the linguistic function which involves a more organized verbal interaction between two parties.

For our purpose of constructing a negotiation protocol, we are most interested in the Interaction linguistic function. We will base largely on the Struggle Model and partly on Institutional Model (mainly the appeal to authority). The Struggle Model contains speech act verbs that are used in various stages of negotiation. If the parties involved in the negotiation cannot reach agreement by themselves, they can require arbitration from a higher authority which is handled in the Institutional Model.

3.3 The Struggle Model

The struggle model covers the semantic area of verbal struggle. The struggle starts by the attacker making a claim and by attacking an addressee if he disagrees with the claim. Then both parties engage in a competitive verbal fight, argue with each other which may result in the victory of one and the defeat of the other or in a compromise [3]. In the course of struggle, both parties can employ tactics such as postponing the discussion or requesting arbitration.

Figure 1 exhibits the categories in the Struggle Model and Institutional Model that we use to construct our protocol. Notice that we do not include all the categories in the models. We leave out categories that describe:

1. physical action (e.g., crash into),
2. emotional reaction after loosing a negotiation (e.g., retaliate, refusing to admit defeat) ,
3. coalition attempts with third parties,
4. nonserious behavior (e.g., play a trick on), and
5. emotional tactics (e.g., insult).

Speech act verbs related to physical actions and emotional reaction after loosing a fight are not applicable to machine-machine communication in an organizational setting. The last two items are difficult to support and their intended outcome can be modelled using other speech act categories. Coalition with a third party is left for future work.

We also made minor changes to the remaining categories. First, we combine some categories because the low level distinction is not necessary for our protocol. For example, it is not necessary to distinguish different links of claim (make justifiable claims, make negative viewed claim). Second, we rename some categories to reflect the functionality of the categories. For example, the category

CODE	CATEGORY NAME	EXAMPLE SPEECH ACT VERBS
KA_{-3a}	unacquainted status	be strange, be unknown
KA_{-2a}	make claim	ask, assert, claim opinion
KA_{-1a}	agree	agree, share the same opinion
KA_{-1c}	dissent	break with someone, have words with someone
KA_{-1d}	withdraw	abandon, abstain from, give up
KA_{0j}	argumentative attack	affirm, claim, confront
KA_{1bba}	postpone	adjourn, delay, hold up, postpone
KA_{1bbb}	admit defeat	admit, agree to, give in, give way
KA_{1bcd}	argumentative defense	bring counterarguments, contradict, disprove
KA_{1cab}	pinning down	fix, pin down, tie down
KA_{1cca}	repeated attack	claim, iterate, repeat
KA_{1ccb}	insisting	persist in, press, pursue
KA_{3b}	retreat	$\cong KA_{-1d}$
KA_{4ab}	force concession	force, overrule
KA_{4cbd}	offer compromise	make a contract, make an agreement
KA_{4cba}	one-sided compromise	admit, consent to, agree
KA_{4cbc}	counter offer	accept in part, make stipulation
KA_{4cda}	accept compromise	accept, agree with, approve of
NO_{60}	appeal	appeal to someone, bring forward
NO_{6c}	examining	hear, question
NO_{6da}	testify	give testimony, show proof
NO_{7a}	make decision	arbitrate, decide, settle

Figure 1: Example speech act verbs in selected categories.

name of KA_{1bba} is "evasive manoeuvre without loss for defender". However, the functionality of the speech act verbs in the category (e.g., adjourn, delay, hold up, postpone, put off) can be better described as "postponing".

We want to mention here that the label attached to each category represents the meaning of the speech act verbs. KA is the model name derived from the German word "Kampfmodell" which means struggle model. The subscript is used to express sequencing and subcategory information. For example, the category KA_1 is sequenced before KA_2, and KA_{1a} is a subcategory of KA_1.

In the following section, we will describe how we use these categories and result from negotiation research to construct our protocol.

4 SANP: The Negotiation Protocol

4.1 Principles behind SANP

The basis of SANP (Speech Act based Negotiation Protocol), is the Struggle Model. We use the temporal relations between the categories and the alternative choice provided by the subcategories to construct the protocol. For example, "Dissent" must appear after "Make Claim", and there are two alternatives after the sentence "Make Claim": "Agree" and "Dissent". The sequencing infor-

mation is provided by the subscript of the category labels. Note that the sequencing information ensures that if category Y has to be used after category X, then Y has a subscript number which is greater than or equal to that of X.

In constructing the protocol, we use the following rules to guide a conversation:

1. After speaking a sentence, the party must wait for the other party to reply before it is allowed to speak its next sentence.
2. The next sentence spoken is restricted by the choices given at that state in the protocol diagrams in the appendix.

Results from the negotiation literature are used to decide what should be included in the protocol and to confirm the validity of the Struggle Model. The processual model of negotiation of Gulliver [16], strategic choice model of Pruitt [22], and the analysis of argument structure of Toulmin [31] are used to support our design decisions.

Gulliver [16] proposes an 8 phase processual model of negotiation which depicts the developmental progress of the negotiation process from the initial recognition of the dispute to some kind of outcome. His model only cover two parties negotiation without the intervention of third party. The 8 phases are described as follow:

1. Search for arena: Parties agree on the location where negotiation may occur.
2. Agenda Setting: Parties agree on the issués to be negotiated.
3. Exploring the field: parties try to establish maximal limits to the issues in dispute.
4. Narrowing the difference: the parties begin to look for the possibilities of approaching actual outcomes and try to narrow their differences.
5. Preliminaries to final bargaining: parties search for a viable bargaining range, refine persisting difference, test trading possibilities, and construct a bargaining formula.
6. Final bargaining: parties exchange specific and substantive proposals and counter proposals.
7. Ritual affirmation
8. Execution of the agreement

Our protocol is not intended to support the entire process of negotiation described by Gulliver. We make a number of assumptions: the negotiation platform is computers, there are specific issues to be negotiated, and when a compromise is made it is automatically affirmed. Therefore, our protocol will only support the negotiation phases 4 to 6. More specifically, we would like to provide facilities for exchanging information among agents to increase the understanding of each other, for narrowing down differences, and for settling the dispute.

Pruitt [22] proposes a Strategic Choice model of negotiation. The model states that parties involved in a negotiation have to make strategic choice at every point in time. The choices include:

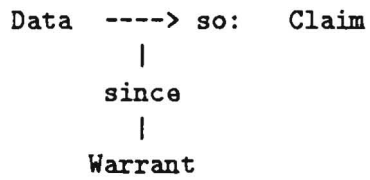
1. concede unilaterally
2. stand firm and employ pressure tactics (e.g., persuasive arguments, threat)
3. collaborate with other parties in search of a mutually acceptable solution.

Maynard [21] in his research on the structure of discourse in misdemeanor plea bargaining finds that bargaining results in three outcomes: reaching compromise, postponing discussion, and appealing to higher authority. Requests for postponement are used to collect more information or to put pressure on the other party.

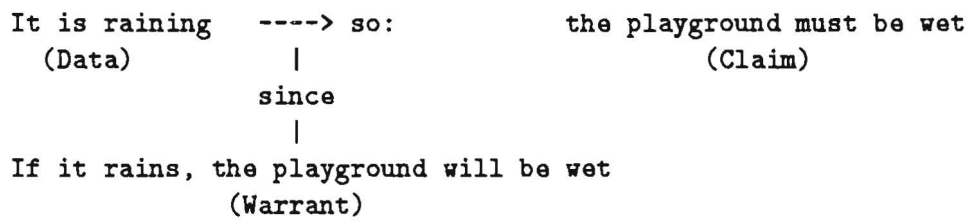
In addition to reaching a compromise, SANP is designed to support both "Delay" and "Appeal" functions. We also support all the strategies stated in the Strategic Choice model. We allow agents

to make concessions by offering one-sided and two-sided compromise, and to persuade other agents by providing arguments.

Toulmin [33] proposed a simple model of argument structure which is depicted by the following diagram:



In Toulmin's analysis, the first step in an argument is that one party expresses an opinion, which Toulmin calls a "Claim". If the claim is challenged, it has to be defended by adducing of "Data" which supports the claim. If the challenger is not satisfied with the accuracy of the data, the objection has to be removed by "Preparatory Argument". However, even if the accuracy of the data has not been questioned, the challenger can require further support for the claim. In this case, a "Warrant" which states the relation between the "Data" and "the Claim" has to be given [33]. An Example of the structure is:



From this simple structure, we conclude that arguments are recursive. We need to justify the accuracy of the "Data" by another argument. We need to do the same thing for the "Warrant", we can question its validity and require a justification.

SANP is designed to handle the recursive properties of argumentation. It supports multi-level negotiations so that the participants can identify and narrow down their differences more easily.

4.2 A Communication Level Protocol for Negotiation

In this section, we shall discuss a communication level negotiation protocol, SANP, that incorporates the requirements discussed in the previous section.

The format of a sentence in SANP consists of the two components:

<function><content>

where <function> is a speech act category name and <content> is the representation of domain knowledge (an example domain knowledge representation is given in section 5). Figures in the appendix are the protocol diagrams. The state of a conversation is represented by a node. The "-" and "+" signs labelled with the arc in the diagram are used to indicate the sentences spoken by the attacker and the defender, respectively. The "?" indicates the sentences spoken by the arbitrator. For example, "$-KA_{-2a}$" means the attacker speaks "Make Claims". The arcs originating from a node is the allowable response in that stage of negotiation. For example, in "Dissent" state, the attacker has two choices: "Retreat ($-KA_{3b}$)" or "Argumentative Attack ($-KA_{0j}$)". The rectangular box indicates the name of the subdiagram to be used at that point. "End" is used

to terminate the conversation. A detailed description of the protocol is provided in the appendix. The following discussion will highlight some important points in the protocol.

Basically, there are 6 main stages in the protocol:

1. *Starting situation:* The agents establish a common understanding of the topic and decide whether they have the necessary knowledge to enter into a discussion.
2. *Making a claim and receiving dissent:* The attacker starts by presenting his claim. It can be a plan, a request for action, or just an assertion. The defender then reasons whether he agrees with it or not.
3. *Attacking:* If the defender disagrees with the claim, the attacker will start attacking the defender by providing argument.
4. *Tactical phase:* The defender begins by defending his position, the attacker attacks the defender's argument. This process continues until they identify their differences and try to narrow them down. They can employ different tactics: request to delay, offer compromise, and insist on their own arguments. This phase corresponds to phase 4 and 5 of Gulliver's model.
5. *Entering into a settlement:* Both parties begin to offer compromise and counter on other's proposal. This phase correspond to phase 6 of Gulliver's model.
6. *End result:* The result of the negotiation can be an one sided compromise, mutual compromise, delay, or appeal to higher authority.

Initially, both parties are in the "Idle" state, the attacker needs to specify or declare the topic of discussion. If the defender does not have the necessary knowledge, the converation ends. Otherwise, both parties should have all the necessary background to understand each other (e.g., the same variable will mean the same thing to both of them). The term "ack" (used in the protocol diagram in the appendix) means the defender agrees to discuss the topic.

In the tactical phase, the protocol supports multi-level arguments by providing a looping between the "Being attacked" and "Defending" states. Therefore, both parties can request and provide deeper and deeper level of support to their arguments. This looping also aims at identifying and narrowing the differences between the views of the parties. For implementation consideration, we only allow a maximum of 100 times of looping to avoid infinite looping. We believe this number is sufficient in an ordinary argument. Moreover, if the number of iterations on the same point at the same level exceeds 5, the looping is also terminated. We handle the looping between "Offer a compromise" and "Modify the compromise" in a similar way.

Our protocol also supports the delay tactic suggested by Maynard [21]. An agent can request to delay a discussion if he does not have enough information to continue the discussion. It must be emphasized that the date of postponement is also settled by negotiation. The other party can even deny the request if he is in a higher position of authority.

Appeal to a higher authority is also supported. If the parties find that they cannot settle the dispute by themselves, they can request an arbitration. The arbitration process is simple, each party presents his own arguments and the arbitrator will make a decision based on this information. We assume that the arbitrator is either a human being or a system that has the necessary knowledge to make the decision.

In designing this protocol, we always put flexibility in high priority so that it can be applied to wide variety of situations. In the following section, we will give an example use of our protocol.

5 An Example Application

In this section, we shall describe how the proposed communication protocol is used to support the preparation of budgets. Consider two organizational workers where the attacker is responsible for preparing the departmental budget and the defender is an employee who prepares part of the departmental budget. The attacker feels that the "labor cost" budgeted by the defender is too high.

Before presenting the dialogue, we need a representation of the domain knowledge:

1. knowledge and rules are represented as equations. For example
   labor cost = total hours × salary × number of worker
2. facts are represented using variables and their corresponding values.
3. A "?" in an equation means the agent does no know the value of a variable.

We intentionally make the representation simple so that we can demonstrate our ideas easily.

In Figure 2, we present three scenarios for negotiation. The attacker begins the conversation by declaring the topic as "Next year's budget". The defender knows the topic, so he agrees to discuss it. The attacker claims that the "labor cost" should be $91,520 and the defender disagrees with the attacker. The defender provides arguments why he disagrees in line 4. In line 5, the attacker disagrees with the value of "hourly wage" and "total yearly hours" and provides his argument. There are three possible scenarios after this.

In the first scenario, the defender offers a compromise in line 6. He agrees on "hourly wage" but requires to keep the number of employee to be 5. The attacker consider this compromise to be acceptable, so he accepts $114,400 as the labor cost.

In the second scenario, the defender does not know the value for "%increase", he requests to postpone the discussion to 6/30/91. However, in line 10, the attacker wants the discussion to be resumed earlier, so he counter offers an earlier date and the defender accepts this new date.

In the third scenario, the defender agrees with the percentage of wage increase (%increase). However, he disagrees with the value of "#employee", so he presents his own calculation. In line 14, the attacker in turn disagrees with the value of "workload". However, the defender insists that the value of "workload" is 10,000 hours. Since they cannot resolve the conflict among themselves, the attacker requests an arbitration in line 16. In line 17 to 20, the attacker and the defender present their arguments to their superior. Based on his knowledge, the superior makes a decision in line 21 that is in favor of the defender.

These example scenarios show that in SANP agents can present their supporting evidence in the negotiation process and they can discover the reasons for their conflict. For instance, in scenario three, the agents finally identify the cause of disagreement on the "labor cost" to be their different assessment on the value of "workload". In addition, agents can utilize strategies such as postponement and arbitration in SANP.

6 Conclusion

We have proposed a communication level negotiation protocol, SANP, based on the Struggle Model and partly on the Institutional Model in Ballmer and Brennenstuhl's speech act classification. In designing the protocol, we also draw on the results of negotiation research. Our aim is to provide a flexible protocol which can be applied to various areas in DAI and to automate organizational work. The protocol provides the ability to multiple level negotiations and supports many tactics

1	A: Declare topic	"Next year's budget"
2	D: Ack	
3	A: Make claim (KA_{-2a})	labor cost = \$91,520
4	D: Dissent (KA_{-1c})	labor cost = \$124,800
		(Reason:)
		labor cost = hourly wage × total yearly hours
		hourly wage = \$12
		total yearly hours = 10400 hrs
5	A: Argumentative attack (KA_{0j})	hourly wage = \$11
		total yearly hours = 8320 hrs
		(Reason:)
		hourly wage = last year wage × (1+%increase)
		last year wage = \$10
		%increase = 10%
		total yearly hours =#employee × 2080
		#employee = 4

After here, there are three scenarios:

Scenario 1: Compromise

6	D: Offer compromise (KA_{4cbd})	labor cost = \$114,400
		%increase = 10%
		#employees = 5
7	A: Accept compromise (KA_{4cda})	labor cost = \$114,400
8	D: End	

Scenario 2: Postpone

10	D: Postpone (KA_{1bba})	Date = "6/30/91"
		(Reason:) %increase = ?
11	A: Counter offer (KA_{4cbc})	Date = "6/27/91"
12	D: Accept compromise (KA_{4cda})	Date = "6/27/91"
13	A: End	

Legend: "A" is the attacker, "D" is the defender.

Figure 2: An example dialogue for the budgeting example (part 1 of 2)

Scenario 3: Arbitration

14	D: Argumentative defense (KA_{1bcd})	Disagree: #employee = 5
		(Reason:)
		#employee = round(workload/2080)
		workload = 10,000 hrs

15	A: Repeated attack (KA_{1cca})	Disagree: workload = 8,000 hrs
16	D: Insisting (KA_{1ccb})	workload = 10,000 hrs
17	A: Appeal (NO_{60})	
18	S: Examining attacker (NO_{6c})	

19	A: Testify (NO_{6da})	labor cost = \$91,520
		(Reason:)
		labor cost = hourly wage × total yearly hours
		hourly wage = 11
		total yearly hours = #employee × 2080
		#employee =
		round(workload/yearly hours per employee)
		workload = 8,000 hrs

20	S: Examining defender (NO_{6c})	
21	D: Testify (NO_{6da})	Labor cost = \$114,400 workload = 10,000 hrs
22	S: make decision (NO_{7a})	workload = 10,000 hrs labor cost = \$114,400
23	S: End	

Legend: "A" is the attacker, "D" is the defender, and "S" is the arbitrator/superior.

Figure 2: An example dialogue for the budgeting example (part 2 of 2)

which are well recognized in negotiation literatures. At the communication level, all the reviewed negotiation protocols in section 2 can be captured in our protocol.

It is necessary to mention that our protocol is only a communication level protocol, users need to provide their own representation and reasoning mechanism for the domain knowledge. For example, the output of a negotiation support system is a good input to our protocol. Some argument comprehension system (such as OpEd [1]) can also be attached to our protocol.

It is also possible to automate some semi-structured negotiation in an organization by providing a set of tools to the agents. See [37] for a detailed discussion of this direction of application.

We plan to implement a prototype of SANP as a generic platform for negotiation using the Strudel package [28]. Strudel provides a set of generic tools that enable conversation and action management. In Strudel, messages contain conversational moves, such as "Make claim" in our protocol, and we can restrict the possible next moves in the message. Small programs can be written in Winterp, an objected-oriented Lisp, to process the message automatically. Other researchers can then attach their own representations or models, rules of negotiation, and problem solving algorithms of their interested problem domains to this platform to fulfill their specific needs of

negotiation.

Future work will include application of SANP to specific problem domains (e.g., budgeting) to investigate the usefulness and limitation of the protocol. We hope to gain insight from these experiences to improve the protocol.

Our protocol can be extended in several ways. First, our treatment of the appeal function is very simple. In fact, the appeal function can be viewed as another negotiation with a mediator. The Institutional model provides a good base for developing such a protocol. Second, the detailed process of arguing, such as requesting more information and criticizing opponent's argument, has not been incorporated into the protocol. The "Arguing Devices" in Ballmer and Brennensthul's classification can be used for this purpose. Third, the protocol can be extended to involve a third party in the negotiation. This party is not the arbitrator, but an agent that collaborates with one of the parties in the conflict.

SANP is not meant to be complete. Many improvements have yet to be made. The validity of the protocol is derived in part by the match between the negotiation literature and the speech act classification. The final assessment of the protocol should be based on its usefulness when it is applied to differnt problem domains.

Acknowledgement

The authors are members of the Institute for Robotics and Intelligent System (IRIS) and wish to acknowledge the support of the Networks of Centres of Excellence Program of the Government of Canada, the Natural Science and Engineering Research Council, and the participation of PRECARN Associates Inc.

The authors also wish to thank Marius A. Janson for his constructive criticisms and useful suggestions for the presentation and the content of this paper.

References

[1] Alvarado, S. J., Dyer, M. G. and Flower, M., "Editorial Comprehension in OpEd Through Argument Units", *Proceedings AAAI-86*, (Philadelphia, PA, August 11-15, 1980), 205-256.

[2] Austin, J. L., *How to Do Things with Words*, Oxford University Press, 1962.

[3] Ballmer, T. and Brennenstuhl, W., *Speech Act Classification: A Study in the Lexical Analysis of English Speech Activity Verbs*, Springer-Verlag, Berlin, Heidelberg, 1981.

[4] Bond, A. H. and Gasser, L., eds., *Reading in Distributed Artificial Intelligence*, Morgan Kaufmann Publishers, Inc., San Mateo, California, 1988.

[5] Bond, A. H. and Gasser, L., "An Analysis of Problems and Research in DAI", in *Reading in Distributed Artificial Intelligence*, A. H. Bond and L. Gasser, eds., Morgan Kaufmann Publishers, Inc., San Mateo, California, 1988, 3-35.

[6] Campbell, J. A. and D'Inverno, M. P., "Knowledge Interchange Protocol", in *Decentralized A.I.*, Y. Demazeau and J. P. Muller, eds., Elsevier Science Publishers B. V., Amsterdam, Netherlands, 1990, 63-80.

[7] De Cindio, F., Simone, C., Vassallo, A., and Zanaboni, A., "CHAOS: A Knowledge-Based System for Conversing within Offices", in *Office Knowledge: Representation, Management, and Utilization*, W. Lamersdorf ed., Elsevier Science Publisher B. V., Holland, 1988, 257-276.

[8] Conry, S. E., Meyer, R. A., and Lesser, V. R., "Multistage Negotiation in Distributed Planning", in *Reading in Distributed Artificial Intelligence,* A. H. Bond and L. Gasser, eds., Morgan Kaufmann Publishers, Inc., San Mateo, California, 1988, 367-384.

[9] Davis, R. and Smith, R. G., "Negotiation as a Metaphor for Distributed Problem Solving", *Artificial Intelligence,* Vol. 20, No. 1, 1983, 63-109.

[10] Demazeau, Y. and Muller, J. P., eds., *Decentralized A.I.,* Elsevier Science Publishers B. V., Amsterdam, Netherlands, 1990.

[11] Durfee, E. H. and Lesser, V. R., "Using Partial Global Plans to Coordinate Distributed Problem Solvers", in *Reading in Distributed Artificial Intelligence,* A. H. Bond and L. Gasser, eds., Morgan Kaufmann Publishers, Inc., San Mateo, California, 1988, 285-293.

[12] Durfee, E. H. and Lesser, V. R., "Negotiation Task Decomposition and Allocation Using Partial Global Planning", in *Distributed Artificial Intelligence*, Volume II, L. Gasser and M. N. Huhns, eds., Pitman, London, 1989, 229-243.

[13] Galliers, J. R., "The Positive Role of Conflict in Cooperative Multiagent Systems", in *Decentralized A. I.,* Y. Demazeau and J. P. Muller, eds., Elsevier Science Publishers B. V., Amsterdam, Netherlands, 1990, 33-46.

[14] Gasser, L. and Huhns, M. H., eds., *Distributed Artificial Intelligence*, Volume II, Pitman, London, 1989.

[15] Gerson, E. M. and Star, S. L., "Analyzing Due Process in the Workplace", *ACM Transactions on Office Information Systems,* Vol. 4, No. 3, July 1986, 257-270.

[16] Gulliver, P. H., *Disputes and Negotiations: A Cross-Cultural Perspective,* Academic Press, New York, 1979.

[17] Hewitt, C., "Offices Are Open Systems", *ACM Transactions on Office Information Systems,* Vol. 4, No. 3, July 1986, 271-287.

[18] Hewitt, C., "Open Information Systems Semantics for Distributed Artificial Intelligence", *Artificial Intelligence,* Vol. 47, No. 1, 1991, 79-106.

[19] Koo, Charles C., "A Commitment-based Communication Model for Distributed Office Environments," *Proceeding of Conference on Office Information System,* (Palo Alto, California, March 23-25, 1988), 291-298.

[20] Levinson, S. C., *Pragmatics,* Cambridge University Press, New York, 1983.

[21] Maynard, D. W., "The Structure of Discourse in Misdemeanor Plea Bargaining", *Law & Society Review,* Vol. 18, No. 1, 1984, 75-104.

[22] Pruitt, D. G., *Negotiation Behavior,* Academic Press, New York, 1981.

[23] Putman, L. L. and Poole, M. S., "Conflict and Negotiation", in *Handbook of Organizational Communication: An Interdisciplinary Perspective,* F. M. Jablin, et al., eds., Sage, Newbury Park, CA, 1987, 549-599.

[24] Rahim, M. A., *Managing Conflict in Organization,* Praeger, New York, 1986.

[25] Searle, J. R., *Speech Acts: An Essay in the Philosophy of Language,* Cambridge University Press, New York, 1969.

[26] Searle, J. R. "A Taxonomy of Illocutionary Acts", In *Language, Mind and Knowledge,* Minnesota Studies in the Philosophy of Science, Vol. 7, K. Gunderson, ed., University of Minnesota Press, Minneapolis, 1975, 344-369.

[27] Searle, J. R. and Vanderveken, D., *Foundation of Illocutionary Logic,* Cambridge University Press, London, 1985.

[28] Shepherd, A., Mayer, N., and Kuchinsky, A., "Strudel - An Extensible Electronic Conversation Toolkit", *CSCW 90 Proceedings,* (Los Angeles, California, October 7-10, 1990), 93-104.

[29] Smith, R. G., "The Contract Net Protocol: High Level Communication and Control in a Distributed Problem Solver", *IEEE Transactions on Computing,* Vol. C-29, No. 12, December 1980, 1104-1113.

[30] Sycara, K., "Multiagent Compromise via Negotiation," in *Distributed Artificial Intelligence,* Volume II, L. Gasser and M. Nuhns, eds., Pitman Publishing, London, 1989, 119-137.

[31] Toulmin, S. E., *The Uses of Argument,* Cambridge University Press, Cambridge, 1969.

[32] van Eemeren, F. H. and Grootendorst, R., *Speech Acts in Argumentative Discussions,* Foris Publication, Dordrecht, 1983.

[33] van Eemeren, F. H, Grootendorst, R., and Kruiger, T., *The Study of Argumentation,* Irvington Publishers Inc., New York, 1984.

[34] Winograd, T. and Flores, F., *Understanding Computers and Cognition,* Addison-Wesley, NJ, 1986.

[35] Woo, C. C., "SACT: A Tool for Automating Semi-Structured Organizational Communication", *Processing of Office Information System,* (Cambridge, Massachusetts, April 25-27, 1990), 89-98.

[36] Woo, Carson. C. and Zeng, Tao, "An Application Layer Communication Protocol for Supporting Organizational Work", Working Paper 90-MIS-019, University of British Columbia, Vancouver, Canada, Auguest 1990.

[37] Woo, Carson C., "Communication Tools for Facilitating the Automation of Semi-Structure and Recurring Negotiations in Organizations", Working Paper 91-MIS-006, University of British Columbia, Vancouver, Canada, January 1991.

[38] Zeng, T., *An Organizational Communication Protocol Based on Speech Acts: Design, Verification and Formal Specification,* M.Sc. thesis, Department of Computer Science, University of British Columbia, Vancouver, Canada, 1990.

Appendix: Descriptions of the SANP Protocol

The detail of the protocol is described in this appendix. Symbols and notations used in the protocol diagram is given in section 4.2. The title of the sections and subsections (in *italic*) corresponds to the subdiagram names and nodes, repectively.

Opening Stage

Idle

At the beginning of the conversation, parties involved should establish common understanding. They need to agree on the topic of discussion and make sure they both have the knowledge for that topic. Attacker begins the conversation by saying "Declare topic" with the topic as content.

Opening

If the defender has the knowledge of the topic, speaks "ack", which means it agrees to discuss the topic.

If the defender does not have any knowledge of the topic, speaks "end" to end the conversation.

Make Claim

Ready to make claim

The attacker speaks "make claim" with the claim as content. The claim can be a statement or a request for action, as well as a plan. In an organization most conversations will be requests for action [34] and solicitation of commitments. For every request for action, the final result should be either having a commitment, may be in the form of a contract, or rejecting the request.

Claim made

If the defender agrees with the claim of attacker, speaks "agreement".

Otherwise, speaks "dissent" with the reasoning steps used to disprove the attacker's claim as the content.

Dissent

If the attacker agent 1) agrees with defender, or 2) does not have necessary information to continue the discussion, or 3) does not have any supporting argument, speaks "withdraw" to withdraw the claim.

If the attacker disagrees with the argument of the defender, speaks "argumentative attack" with information of what he disagrees together with argument to support them as content.

Agreement

The attacker speaks "offer compromise" with the claim as its content. The content may be in the form of a contract so that both parties can keep track of the commitment made (as in [19]).

Attacker offers compromise

The defender speaks "accept compromise" and enter the contract into its knowledge base.

Tactical Phase

Being attacked

If the defender agrees with the attacker's argument, he should speak "admitted defeat", and enters into an agreement with the attacker. The process will be described in the "**Defender Admitted Defeat**" section.

If the defender does not have necessary information or knowledge, speaks "postpone" to require a postponement of the discussion.

If the defender does not agree with the attacker's argument, speaks "argumentative defense" with what he disagrees and supporting argument as the content.

If 1) the defender runs out of argument or 2) the number of iterations between *Being attacked* and *Defending* on the same point exceeds five, or 3) the total iterations between *Being attacked* and *Defending* on the whole argument exceeds 100, the defender can either speak 1) "insisting" with his argument as content or 2) "offer compromise" with the compromise specified by the user as content.

Defending

If the attacker agrees with the argument of the defender, it should speak "withdraw" with empty content.

If 1) the attacker runs out of argument or 2) the number of iterations between *Defending* and *Being attack* on the same point exceeds five, the attacker can either speak 1) "insisting" with his argument as content or 2) "offer compromise" with the compromise as content.

Defender Requests Postponement

Defender requests postponement

If the attacker agrees with the date of postponement, speaks "accept compromise".

If the attacker agrees to postpone but with different date, speaks "counter offer" with the new date as content.

If the attacker is in higher authority, it can speak "forcing concession" with the postpone date as the content.

If both parties cannot reach an agreement on the postpone date, attacker can speak "appeal" to require an arbitration from higher authority.

If the attacker is in higher authority, it can speak "pinning down" to force a compromise from the defender.

However, if the number of iterations between *Defender requests postponement* and *Attacker modify date* exceeds five, attacker is forced to choose other responses.

Attacker modify date

If the defender agrees with the date, it should speak "accept compromise" with the date as content.

If the defender want a different date, it should speak "counter offer" with the new date as content.

If the defender is in higher authority, it can speak "forcing concession" with the postpone date as the content.

If both parties cannot reach an agreement on the postpone date, defender can speak "appeal" to require an arbitration from higher authority.

Pinning down

The defender has to search for a compromise in its knowledge base. It should speaks "offer compromise" with the compromise in the knowledge base, or if there isn't any, with the attacker's argument as content.

Defender Admitted Defeat

Defender admitted defeat

The attacker speaks "offer compromise" with the claim as its content. The content may be in the form of a contract so that both parties can keep track of the commitment made (as in [19]).

Attacker offers compromise

The defender speaks "accept compromise" and enter the contract into its knowledge base.

Defender Offers Compromise

Defender offers a compromise

If the attacker agrees with the compromise, speaks "accept compromise" with the compromise as content.

If the attacker agrees only partly with the compromise, it can speak "counter offer" with the modified compromise as content.

If the attacker is in higher authority, it can speak "forcing concession" with its proposed compromise as content.

The attacker can also speak "appeal" to require an arbitration from higher authority.

However, if the number of iterations between *Defender offers compromise* and *attacker modify compromise* exceeds five, the attacker is forced to choose other response.

Attacker modify the compromise

If the defender agrees with the compromise, speaks "accept compromise" with the compromise as content.

If the defender agrees only partly with the compromise, it can speak "provisos in compromise" with the modified compromise as content.

If the defender is in higher authority, it can speak "forcing concession" with its proposed compromise as content.

The defender can also speak "appeal" to require an arbitration from higher authority.

Attacker Offers Compromise

Similar to **Defender Offers Compromise.**

Defender Insists On Argument

Defender insists on argument

If there is alternative in the knowledge base, speaks "offer compromise" with the alternative as content or if there isn't any, uses the defender argument as the content.

The attacker can speak "appeal" to require an arbitration from a higher authority.

If the attacker is in higher authority, it can speaks "forcing concession" with its own argument as content.

Attacker Insists On Argument

Similar to **Defender Insists On Argument.**

Request Arbitration

Arbitration requested

The arbitrator speaks "examining attacker" to ask the attacker to testify.

Attacker being questioned

The attacker speaks "testifying" with its argument as content.

Attacker testified

The arbitrator speaks "examining defender" to ask the defender to testify.

Defender being question

The defender speaks "testifying" with its argument as content.

Defender testified

The arbitrator speaks "make decision" with its decision as the content.

Defender Forcing a Concession

Defender forcing a concession

If the defender has not changed its initial position, speaks "withdraw" to withdraw the claim.

If the defender has changed its initial position, speaks "one-sided compromise" with the defender's argument or proposal as content.

Attacker Forcing a Concession

Attacker forcing a concession

The defender speaks "one-sided compromise" with the proposal of the attacker as content.

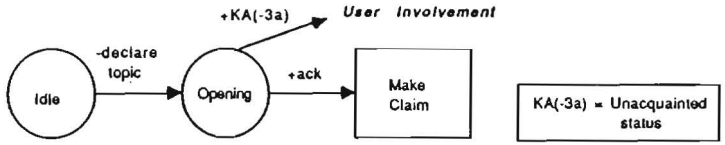
Retreat

Withdraw claim

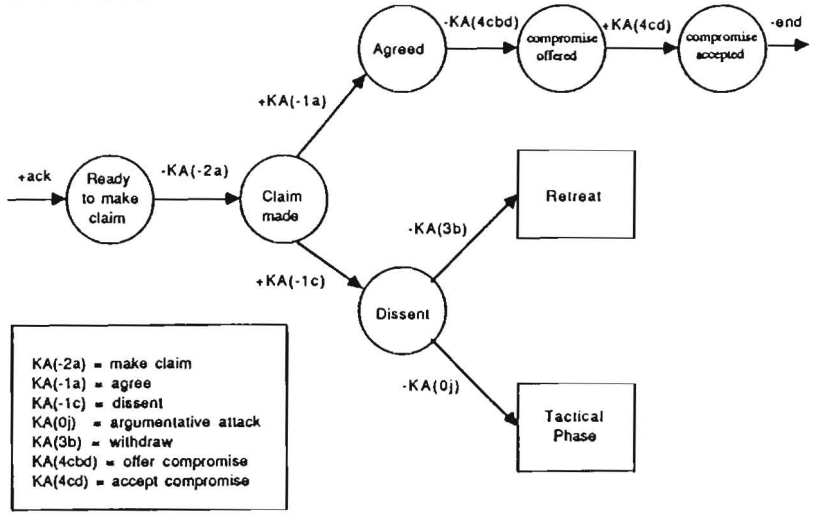
The defender speaks "end" to end the conversation.

Protocol Diagrams for SANP

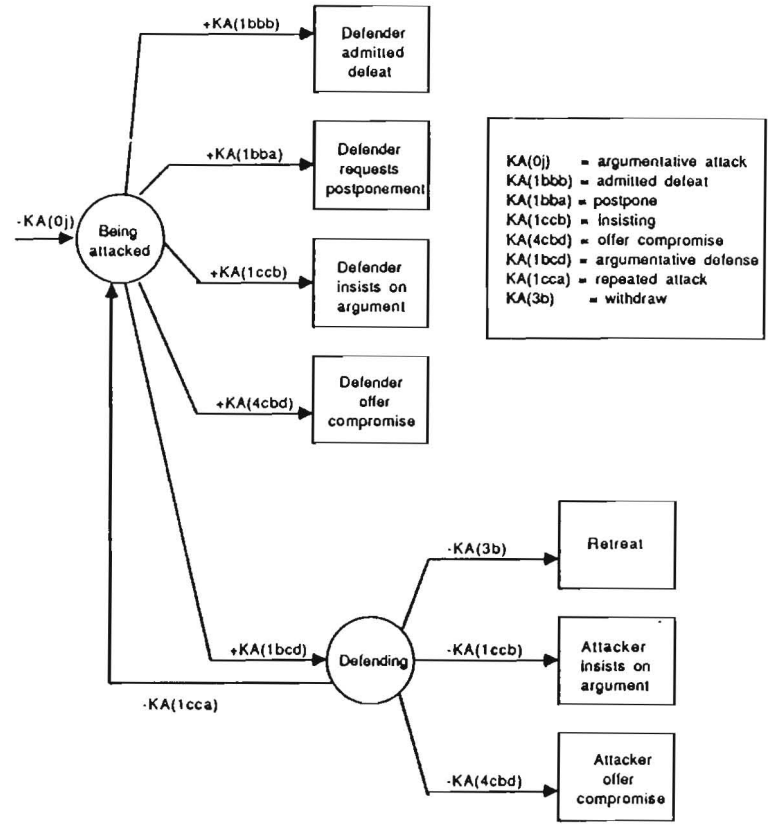
OPENING STAGE



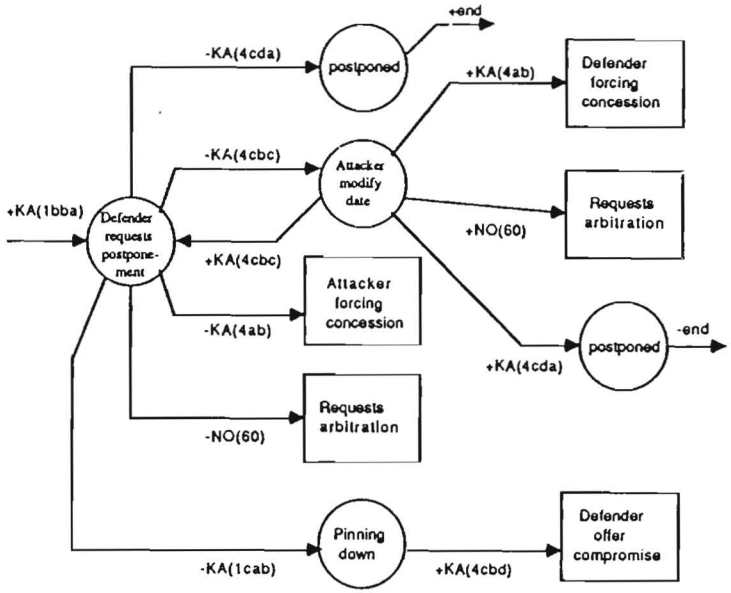
MAKE CLAIM



TACTICAL PHASE



DEFENDER REQUEST POSTPONEMENT

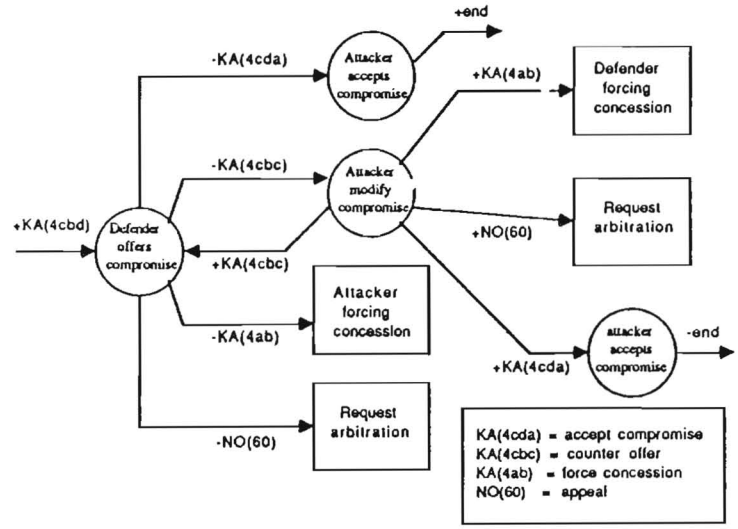


KA(4cda) = accept compromise
KA(4cbc) = counter offer
KA(K4ab) = force concession
KA(1cab) = pinning down
KA(4cbd) = offer compromise
NO(60) = request arbitration

DEFENDER ADMITTED DEFEAT

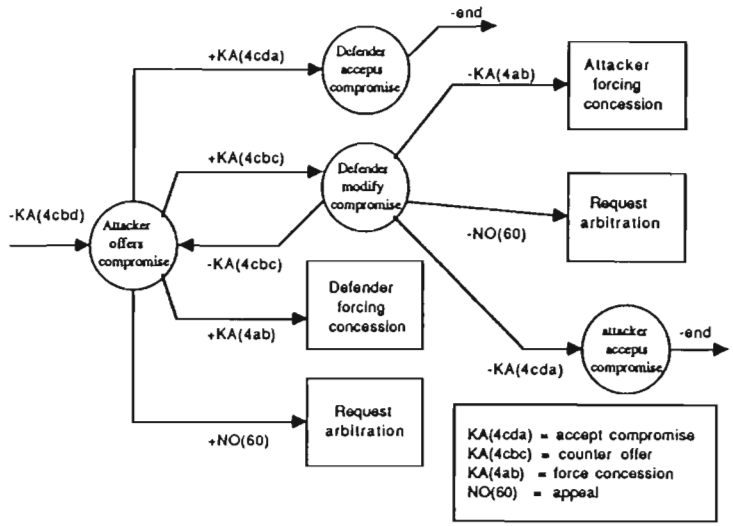
KA(4cbd) = offer compromise
KA(4cd) = accept compromise

DEFENDER OFFER COMPROMISE

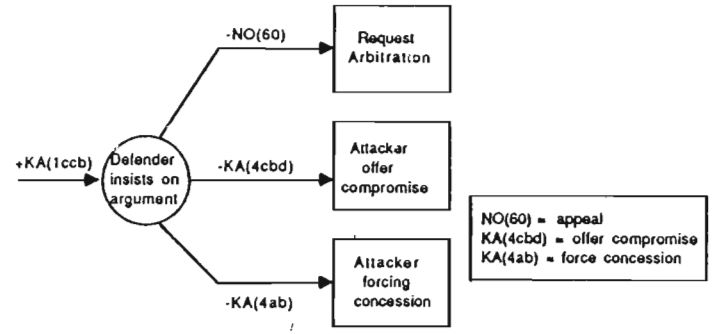


KA(4cda) = accept compromise
KA(4cbc) = counter offer
KA(4ab) = force concession
NO(60) = appeal

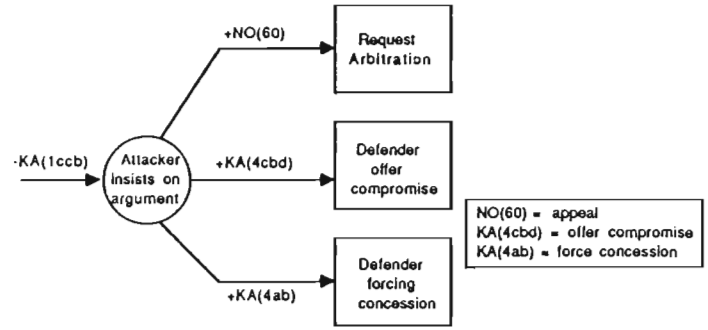
ATTACKER OFFER COMPROMISE



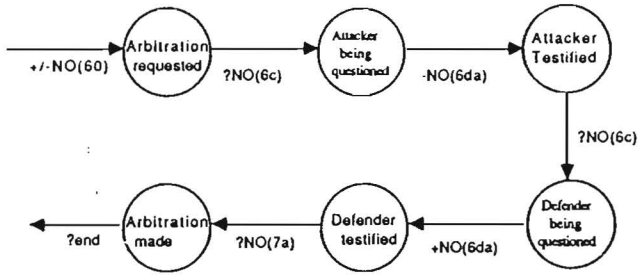
DEFENDER INSISTS ON ARGUMENT



ATTACKER INSISTS ON ARGUMENT

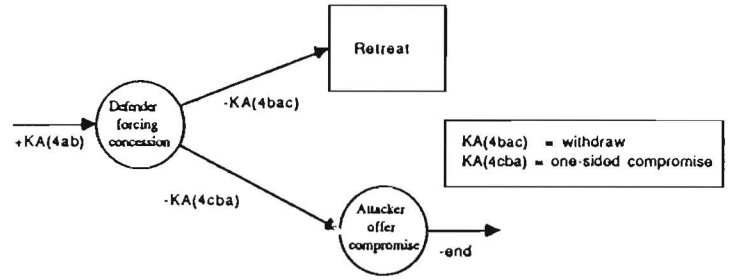


REQUEST ARBITRATION



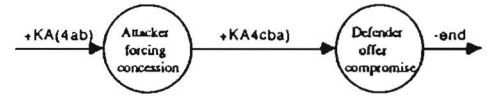
NO(6c) = Examining
NO(6da) = Testify
NO(7a) = Make decision

DEFENDER FORCING CONCESSION



KA(4bac) = withdraw
KA(4cba) = one-sided compromise

ATTACKER FORCING CONCESSION



KA(4cba) = one-sided compromise

RETREAT

Social Plans: A Preliminary Report*

Anand S. Rao
Australian Artificial Intelligence Institute
Carlton, Victoria 3053, Australia
Email: anand@aaii.oz.au

Michael P. Georgeff
Australian Artificial Intelligence Institute
Carlton, Victoria 3053, Australia
Email: georgeff@aaii.oz.au

Elizabeth A. Sonenberg
The University of Melbourne
Parkville, Victoria 3052, Australia
Email: eas@cs.mu.oz

Abstract

The formalization of multi-agent autonomous systems requires a rich ontology for capturing a variety of collective behaviours and a powerful semantics for distinguishing between collective agents having, executing, and jointly intending a social plan. In addition, success and failure executions of plans should be distinguished. In this paper, we introduce the notion of *social agents* and *social plans*, formalize some of the above issues, and briefly discuss how social agents can perform hierarchical planning.

1 Introduction

Situated agents are systems embedded in dynamic environments; they continuously sense their environment and effect changes to it by performing actions. These agents have to balance the time they devote to thinking against the time they take acting. Also, they need to balance the need to react to new situations against the need to continue pursuing long-term purposes and goals.

One of the critical considerations in the design of situated agents is that these agents are resource bounded; that is, they must reason and act under possibly stringent constraints on time and information. According to Bratman [1], the intentions of the agent play a crucial role in such cases. Viewed as a commitment to present and future plans, intentions constrain the deliberation and planning process and hence reduce the time spent reasoning. Systems (formal or implemented) that give primary importance to the notion of intention are called Belief-Desire-Intention (BDI) architectures [2].

Although individual situated agents can be adequately modeled within the BDI framework, modeling of a group of such agents involved in a collaborative activity requires a number of additional notions. In particular, we need an understanding of mutual beliefs, joint goals, joint intentions, social plan structures, social roles, negotiation, communication, and organizational structures. While the formalization of a comprehensive multi-agent BDI-architecture is still an open problem, in this paper we attempt to set some of the foundations of such a theory.

*This research was in part supported by a *Generic Industry Research and Development Grant* from the Department of Industry, Technology and Commerce, Australia and in part by the Australian Civil Aviation Authority.

2 Overview

Joint actions among a group of agents often involves a commitment from all members of the group that each one do their respective parts. Such commitments are often formalized as joint intentions among a group of agents. For example, if two agents want to jointly lift a table, each needs to individually intend to lift one end of the table and believe that the other person will lift the other end. Also, the group needs to mutually believe in the above [17]. However, even before the two agents can form a joint intention to lift the table, they need to share an abstract specification of how to lift a table jointly. In other words, even before forming joint intentions, the agents require recipes or abstract structures that specify how and in what order joint actions should be carried out.

The distinction between plans as abstract structures and plans as a mental attitude that the agent is committed to bringing about is very important [11, 10, 5]. The former, which we shall call *social plan structures*, corresponds to the english usage "I have a plan to ...". The latter, called *intended plans* or *intentions*, corresponds to the usage "I plan to ...".

It is also important to distinguish between *successful* and *failure* executions of plan structures. This distinction is especially important for situated agents because of the stringent synchronization conditions required for joint actions and the possibly serious side-effects arising from failed attempts to perform a given action [4]. For example, when two agents are jointly lifting a glass table and one of them loses his grip on the table, he may not have the opportunity to reattempt the task because either the table may have been smashed or the other agent may not be able to continue holding on to his end. Thus, agents need to track the success or failure of their planned actions and inform their partners if something goes wrong. Levesque *et. al.* [9] discuss how agents should give up their joint commitment when one of the partners has succeeded in the joint action or finds it impossible to accomplish it.

A great variety of human joint actions involve hierarchical social organizations. Thus, a group of agents may consist not only of individual agents but also of other groups. This hierarchical organization of agents needs to be reflected in social plan structures. We do this by introducing the notion of a social agent which is an abstract entity denoting a collection of other individual or social agents. For example, a long table may require three teams of two agents each to lift it, a team for each of the ends and one at the center. One needs to represent the synchronization of actions between the three different teams and the synchronization within each team.

The outline of the paper is as follows. First, we describe the ontology of social plans and illustrate its expressive power by providing examples of social plans and we informally describe their execution. Then, we present an expressive branching-time logic based on CTL* [3]. With the help of these two logics we introduce a semantics for successful and failure executions of social plans. We then extend the standard possible-worlds model to provide the semantics of mutual beliefs, joint goals, and joint intentions. Finally, we briefly describe how such a formal theory of social plans can be used within a multi-agent BDI-architecture. We conclude with a brief description of related work in existing multi-agent and autonomous systems.

3 Syntax

3.1 Social Plans

Social plan structures are syntactic entities that are invoked by a group of agents in particular situations to satisfy certain ends. These ends are achieved by different agents synchronizing their actions as specified by the social plan structure. We adopt standard first-order logic and modal temporal logic to describe situations and a variation of dynamic logic [7] to describe social plan structures.

Having a plan—as distinct from intending or executing a plan—involves not only specifying how to carry out the plan, but also knowing under what conditions such a plan can be [usefully] executed. This collection of information is called a social plan structure. The method of carrying out the plan is called the *body* of the plan structure, and the circumstances under which the plan can be executed is called the *precondition* of the plan structure. In addition, it is necessary to know which agent has the plan, and what the plan accomplishes. Syntactically, we write this as *has-plan*(p x), where p denotes a social plan structure and x denotes both what the plan accomplishes and who has the plan.

The precondition of a social plan structure is specified by a well-formed state formula, defined in the next subsection. The body is specified by a social plan expression. Social plan expressions are similar to formulas in dynamic logic, except that we explicitly specify the agent who is to carry out the plan and introduce some additional plan operators. We also distinguish between *plan types* and the events or actions that occur in the real world. Intuitively, a plan type is an abstract structure that, when executed by an agent, results in the occurrence of an action in the real world.

More specifically, a *primitive plan expression* is a pair consisting of a primitive plan type and an agent. As in dynamic logic, we introduce more complex plan expressions by means of plan operators. These include operators for sequencing (;), parallelism (||), and non-deterministic choice (|). We also allow the operators ? and !, which operate on well-formed state formulas to convert them into plan types. Intuitively, $?\alpha$ is a plan type that tests if the condition α is true and $!\alpha$ is a plan type that achieves α.

We consider two types of agents – individual agents and social agents. Social agents are abstract entities that denote more than one agent. Examples of social agents include *team*, *organization*, *family*, and *friends*. Social agents refer to a set of other constituent social or individual agents. Thus we have a recursive notion of social agents which is more expressive than the notion of a group of agents.[1]

More formally, well-formed social plan expressions can be defined as follows:

Definition 1

- If p is a (primitive or non-primitive) plan type and y is an individual or social agent, then $(p\ y)$ is a well-formed social plan expression;
- If α is a well-formed formula and y is an individual or social agent, then $(!\alpha\ y)$ and $(?\alpha\ y)$ are well-formed social plan expressions; and
- If x_1 and x_2 are well-formed social plan expressions, then $(x_1;x_2)$, $(x_1||x_2)$, and $(x_1|x_2)$ are well-formed social plan expressions.

Note that the above definition is general enough to cover individual plan expressions, i.e., plan expressions involving only individual agents.

We also need to be able to describe *executions* of plans by agents. As the execution of a plan by an agent results in the occurrence of an action, we call such descriptions *action formulas*. These action formulas describe whether the execution was a success or a failure, and whether it occurred in the past or will occur in the future.

Definition 2 If x is a well-formed social plan expression, then $<x>$, $<x>_s$, $<x>_f$, $[x]$, $[x]_s$, and $[x]_f$ are well-formed action formulas.

[1] A group or set of individual agents has been used to illustrate concepts such as common knowledge, everyone in the group knows, someone in the group knows and so on [6].

The first three action formulas denote immediate future executions and the last three indicate immediate past executions. The subscripts "s" and "f" denote success and failure executions, respectively. Without the subscript, the execution can either be a success or a failure.

3.2 Temporal and Modal Operators

An agent who has social plan structures as described above must be capable of executing these social plans in the real world. Hence, the temporal model of the real-world must be expressive enough to capture the different operations on plan types.

We use a formalism similar to Computation Tree Logic, CTL* [3], to describe the temporal structures. The temporal structure in CTL* is a tree with branching futures and a single past. A distinction is made between *state formulas* and *path formulas*: the former are evaluated at a specified time point in a time tree and the latter over a specified path in a time tree. We introduce two modal operators, optional and inevitable, which operate on path formulas. A path formula ψ is said to be optional if, at a particular time point in a time tree, ψ is true of at least one path emanating from that point; it is inevitable if ψ is true of all paths emanating from that point.[2] The standard temporal operators $\bigcirc$ (next), $\Diamond$ (eventually), and $\Box$ (always) operate over state and path formulas.

Unlike CTL*, we introduce two types of arcs between time points: success arcs and failure arcs. An arc connecting two time points is labeled by a primitive plan type. If the arc is a success arc, the primitive plan type is said to be a successful; if it is a failure arc, the primitive plan type is considered to have failed.

The modal operators BEL, GOAL, and INTEND are used to denote individual beliefs, goals, and intentions. The corresponding joint attitudes— namely, mutual beliefs, joint goals, and joint intentions— are denoted by MBEL, JGOAL, and JINTEND, respectively. We also use the operators EBEL, EGOAL, and EINTEND to denote the beliefs, goals, and intentions of all the members of a social agent. All joint propositional attitudes are defined in terms of the individual propositional attitudes. The detailed definitions are given later.

Now we can formally define the notion of well-formed state and path formulas. The former are defined as follows:

- any first-order formula is a state formula;
- if ϕ_1 and ϕ_2 are state formulas and x is an individual or plan variable, then $\neg\phi_1$, $\phi_1 \vee \phi_2$, and $\exists x\ \phi_1(x)$ are state formulas;
- if ϕ is a well-formed action formula then ϕ is also a state formula;
- if ϕ is a state formula and y is an individual agent then BEL(y ϕ), GOAL(y ϕ), and INTEND(y ϕ) are state formulas; and
- if ϕ is a state formula and y is a social agent then MBEL(y ϕ), JGOAL(y ϕ), JINTEND(y ϕ), EBEL(y ϕ), EGOAL(y ϕ), and EINTEND(y ϕ) are state formulas; and
- if ψ is a path formula, then optional(ψ) and inevitable(ψ) are a state formulas.

A path formula can be defined as follows:

- any state formula is also a path formula; and
- if ψ_1 and ψ_2 are path formulas, then $\neg\psi_1$, $\psi_1 \vee \psi_2$, $\Diamond\psi_1$, $\bigcirc\psi_1$ are path formulas.

[2] In CTL*, E and A are used to denote these operators.

4 Examples

In this section, we provide different examples to illustrate the expressive power of the formalism.

Cooperative and competitive sequences of actions among multiple agents: In this example, we assume multiple agents perform sequences of actions in parallel, in which only the start of the sequences need be synchronized; primitive plans within the sequence need not be synchronized.

A plan for such cases can be given as $((e_1\ a_1)\ ;\ (e_2\ a_1)\ ;\ (e_3\ a_1))\ ||\ ((e_1\ a_2)\ ;\ (e_2\ a_2)\ ;\ (e_3\ a_2))$.

A concrete example of this plan type would be a triathlon-race in which two agents have to cycle, swim and run, in that sequence. Although the agents have to synchronize at the start of the race, the subsequent primitive plans need not be synchronized. In other words, we can distinguish between a triathlon-contest and a sequence consisting of a cycle race followed by a swimming race and finally a running race.

Cooperative and competitive activities among teams: In this case we have multiple social agents involved in a cooperative or competitive activity. Further, each social agent refers to a set of individual agents, who are also involved in some cooperative or competitive activity.

For example, if s_1 and s_2 are two social agents who want to achieve α in parallel, the top-level plan structure would be $(!\alpha\ s_1)\ ||\ (!\alpha\ s_2)$. A possible body for the social plan structure for $(!\alpha\ s_1)$ is the plan expression of the form $(e_1\ a_1)\ ;\ (e_1\ a_2)\ ;\ (e_1\ a_3)$. A similar social plan structure can be defined for s_2 to achieve α.

A concrete example of this type is a relay race with two teams. A team running is equivalent to three members of the team running one after the other. Note that we have used the same primitive plan e_1 for all individuals only to match with the concrete example—in general they can be different primitive plan types.

Partial planning: Using the achievement plan expression and the notion of social plan structures, we can illustrate how an agent can decompose a higher-level goal into lower-level sub-goals, which again can be decomposed into further lower-level sub-goals, until one finally reaches primitive plan types. Thus the agent can execute a social plan with his future goals at different levels of abstraction.

Consider a plan of the form $(!\alpha\ a)$. A social plan structure whose purpose is $(!\alpha\ a)$ and body is $(!\alpha_1\ a);(!\alpha_2\ a)$ allows decomposition of the top-level goal into two sub-goals. Each one of these sub-goals can be further decomposed.

A concrete example of this is the means-end reasoning involved in getting to an airport: in order to get to the airport the agent has to get out of the building and hire a cab; in order to get out of the building the agent has a plan which might require him to find the route and follow it; and so on.

Tracking the success or failure of one's own actions: For certain critical tasks, autonomous agents need to test and verify the success or failure of their executions and take appropriate measures based on these tests.

Consider a plan expression of the form $(e_1\ a_1)\ ;\ (\ (?[e_1\ a_1]_s\ a_1)\ ;\ (e_2\ a_1))\ |\ (?[e_1\ a_1]_f\ a_1)\ ;\ (e_1\ a_1)))$. This plan expression states that the agent a_1 does e_1 and then tests if it was successful or not. If it is successful,he proceeds with e_2: if it fails, he repeats the primitive plan e_1.

For example, an autonomous robot trying to put out a fire tests if it has been successful or not. If it has been successful, it goes ahead with some other task; otherwise, it repeats the act.

Tracking the success or failure of other's actions: This is in principle very similar to the previous example. When multiple agents have joint goals they need to continuously track the success or failure of other agent's actions that have a direct influence on their own actions.

Consider a plan expression of the form $(e_1\ a_1)\ ;\ (\ (?[e_1\ a_1]_s\ a_2)\ ;\ ((e_2\ a_1)\ ||\ e_3\ a_2))\ |\ (?[e_1\ a_1]_f\ a_2)\ ;\ ((e_4\ a_1)\ ||\ (e_5\ a_2)))$. This plan expression states that the agent a_1 does e_1 and then agent a_2 tests if

a_1 was successful or not. If it was successful, a_1 does e_2 in parallel with a_2 doing e_3; otherwise a_1 does e_4 in parallel with a_2 doing e_5.

A concrete example of this is the case in which a_1 is a student and a_2 a teacher. The teacher tests the student on a particular lesson. If the student is successful, the teacher goes on to teach the next lesson while the student listens; if unsuccessful, the teacher repeats the previous lesson with the student listening.

Accepting another agent's beliefs: This illustrates how an agent can test if some other agent believes in a certain formula and, if successful, accept the other agent's beliefs as one's own.

A formula that illustrates this is $[(?\ \mathsf{BEL}(a_1\ \phi)\ a_2)]_s \supset \mathsf{BEL}(a_2\ \phi)$. Note that this statement is much stronger than agent a_2 believing that a_1 believes in ϕ and thereby changing his beliefs. This involves an active act of verification by agent a_2 which is absent in the case of beliefs about beliefs.

Mutually verified common goal being sufficient for forming a joint goal: This provides a weaker condition for forming joint goals. It states that if each agent can verify that the other agent has the goal to achieve a state ϕ then they can together adopt it as a joint goal.

This can be represented by the formula $[(?\ \mathsf{GOAL}(a_1 <!\phi\ a_1>_s)\ a_2]_s \wedge [(?\ \mathsf{GOAL}(a_2 <!\phi\ a_2>_s)\ a_1]_s \supset \mathsf{JGOAL}(a <!\phi\ a>_s)$, where a_1 and a_2 are members of the social agent a.

Helping other agents by informing them of the futility of their actions: This illustrates the use of temporal operators and their interaction with social plans. If an agent believes that it is inevitable that sometime in the future the other agent is going to fail in his action, an helpful agent can form a plan to convince the other agent to believe about the futility of such an action.

The formula corresponding to this is $\mathsf{BEL}(a_2\ \mathsf{inevitable}\Diamond[e_1\ a_1]_f) \supset <!\ \mathsf{BEL}(a_1\ \mathsf{inevitable}\Diamond[e_1\ a_1]_f)\ a_2>_s$.

5 Semantics

5.1 Logical Preliminaries

We first define an interpretation that is an extension of a standard Kripke interpretation of possible worlds. The extension involves each possible world being a temporal structure.

Definition 3 : An interpretation M is defined to be a tuple, M = ⟨W, IA, SA, PP, P, $\mathcal{PLANS}$, $\mathcal{MEMBERS}$, T, $\prec$, U, SPS, $\mathcal{PSA}$, $\mathcal{B}$, $\mathcal{G}$, $\mathcal{I}$, Φ⟩. W is a set of worlds, IA is a set of individual agents, SA is a set of social agents, PP is a set of primitive plan types, P is a set of plan types, T is a set of time points, $\prec$ a binary relation on time points,[3] U is the universe of discourse, and SPS is the set of all social plan structures. $\mathcal{PSA}$ is a plan structure assignment function that maps a plan type to a social plan structure. $\mathcal{PLANS}$ is a function from individual or social agents to a set of plan types. Intuitively, this function provides the plan library of the agent. $\mathcal{MEMBERS}$ is a relation between social agents and other social and individual agents. More formally, $\mathcal{MEMBERS} \subseteq$ SA × {SA ∪ IA}. The accessibility relations, $\mathcal{B}$, $\mathcal{G}$, and $\mathcal{I}$ map an individual agent's current situation to his belief-, goal-, and intention-accessible worlds, respectively. More formally, $\mathcal{B} \subseteq$ IA × W × T × W and similarly for $\mathcal{G}$ and $\mathcal{I}$. Φ is a mapping of first-order entities to elements in U for any given world and time point.

Definition 4 : A social plan structure is a tuple $\langle \phi_{pre}\ p_{body} \rangle$, where ϕ_{pre} is any well-formed formula and p_{body} is any well-formed plan expression. We also have the functions *pre* and *body* which, given a plan type, returns the appropriate argument of the above tuple.

[3] We require that the binary relation be total, transitive and backward-linear to enforce a single past and branching future.

Definition 5 : Each world w of W, called a *time tree*, is a tuple $<T_w, \prec_w, \mathcal{S}_w, \mathcal{F}_w>$, where $T_w \subseteq$ T is a set of time points in the world w and $\prec_w$ is the same as $\prec$, restricted to time points in T_w. A *fullpath* in a world w is an infinite sequence of time points $(t_0, t_1,...)$ such that $\forall i\ (t_i, t_{i+1}) \in \mathcal{A}_w$. We use the notation $(w_{t_0}, w_{t_1},...)$ to make the world of a particular fullpath explicit. The arc functions $\mathcal{S}_w$ and $\mathcal{F}_w$ map time points to a primitive plan type. More formally, $\mathcal{S}_w$: $T_w \times T_w \mapsto 2^{PP}$ and similarly for $\mathcal{F}_w$. The domains of $\mathcal{S}_w$ and $\mathcal{F}_w$ are disjoint. Intuitively, for any two time points for which the arc function $\mathcal{S}_w$ is defined, its value represents the primitive plan that successfully occurred (or was performed by agent(s)) between those time points. Similarly, the value of the arc function $\mathcal{F}_w$ represents the failure of a primitive plan occurring between those time points.

5.2 Semantics of Temporal Modalities

The semantics of temporal modalities is straightforward. Both $\bigcirc\psi$ and $\Diamond\psi$ are path formulas and are evaluated over a particular path. The formula optional(ψ) is a state formula and is true if there is at least one path where ψ is true. More formally, we have:

$M, v, (w_{t_0}, w_{t_1}, ...) \models \bigcirc\psi$ iff $M, v, (w_{t_1}, ...) \models \psi$.
$M, v, (w_{t_0}, w_{t_1}, ...) \models \Diamond\psi$ iff $\exists k, k \geq 0$ such that $M, v, (w_{t_k}, ...) \models \psi$.
$M, v, w_{t_0} \models$ optional(ψ) iff there exists a fullpath $(w_{t_0}, w_{t_1},...)$ such that $M, v, (w_{t_0}, w_{t_1},...) \models \psi$.

The formula inevitable(ψ) is defined as $\neg$optional($\neg\psi$) and $\Box\psi$ is defined as $\neg\Diamond\neg\psi$.

5.3 Semantics of Social Plan Executions

A social or individual agent has a library of social plans. All plans serve a purpose, which is either to achieve a certain condition (as in $!\alpha$) or to test for a certain condition (as in $?\alpha$).

We say that an agent y has a plan type p to achieve the condition α if whenever the plan has been successfully executed, the condition α holds. We also require that the plan be in the agent's plan library.

$M, v, w_{t_0} \models$ *has-plan*$(p\ (!\alpha\ y))$ iff
(a) $p \in \mathcal{PLANS}(y)$ and
(b) $M, v, w_{t_0} \models$ inevitable$\Box([(p\ y)]_s \supset \alpha)$.

Having a plan to test for a certain condition is very similar. We say that an agent y has a plan type p to test for condition α if, prior to the successful execution of the plan, the condition α holds. As before we require that the plan be in the agent's plan library.

$M, v, w_{t_0} \models$ *has-plan*$(p\ (?\alpha\ y))$ iff
(a) $p \in \mathcal{PLANS}(y)$ and
(b) $M, v, w_{t_0} \models$ inevitable$\Box(<(p\ y)>_s \supset \alpha)$.

Next we consider what it means for an agent to execute a plan type. We say that an agent y successfully executes a plan type p if the precondition of the plan is satisfied and the body is executed successfully.

$M, v, (w_{t_0}, w_{t_1},...) \models <(p\ y)>_s$ iff
$M, v, (w_{t_0}, w_{t_1},...) \models pre(p) \wedge <body(p)>_s$.

The past execution of plans is somewhat more complicated to specify. We say that a plan type p has been successfully executed by agent y if the body of the plan has been executed successfully, the precondition held at some time in the past when the execution of the body started, and there was no other successful execution of the body in between.

$M, v, w_{t_n} \models [(p\ y)]_s$ iff
(a) $\exists t_0,\ t_0 \prec t_n$ such that $M, v, (w_{t_0},...) \models pre(p) \wedge <(p\ y)>_s$;
(b) $M, v, w_{t_n} \models [body(p)]_s$; and
(c) $\nexists t_i,\ t_0 \prec t_i \prec t_n$, such that $M, v, w_{t_i} \models [body(p)]_s$.

The body of a plan could contain an expression to achieve or test for a certain condition. An agent y is said to achieve successfully the condition α if there is a plan type p whose purpose is to achieve α, and the plan is executed successfully. Similarly for testing a condition. More formally, if x stands for $(!\alpha\ y)$ or $(?\alpha\ y)$, we have:

$M, v, (w_{t_0}, w_{t_1},...) \models <x>_s$ iff there exists an plan type p such that
(a) $M, v, w_{t_0} \models has\text{-}plan(p\ x)$ and
(b) $M, v, (w_{t_0}, w_{t_1}, ...) \models <(p\ y)>_s$.

We say that a sequence of two primitive plans is successfully executed if each one of them is executed successfully one after the other. Two parallel primitive plans are successfully executed if both of them are successfully executed at the same time, i.e., both label the same arc. Two non-deterministic primitive plans are successfully executed if either one of them is successfully executed. More formally, the successful future executions can be stated as follows:

$M, v, (w_{t_0}, w_{t_1},...) \models <(e_1\ a_1)\ ;\ (e_2\ a_2)>_s$ iff
$M, v, (w_{t_0}, w_{t_1},...) \models <(e_1\ a_1)>_s$ and $M, v, (w_{t_1},...) \models <(e_2\ a_2)>_s$.
$M, v, (w_{t_0}, w_{t_1},...) \models <(e_1\ a_1) \parallel (e_2\ a_2)>_s$ iff
$M, v, (w_{t_0}, w_{t_1},...) \models <(e_1\ a_1)>_s$ and $M, v, (w_{t_0}, w_{t_1},...) \models <(e_2\ a_2)>_s$.
$M, v, (w_{t_0}, w_{t_1},...) \models <(e_1\ a_1) \mid (e_2\ a_2)>_s$ iff
$M, v, (w_{t_0}, w_{t_1},...) \models <(e_1\ a_1)>_s$ or $M, v, (w_{t_0}, w_{t_1},...) \models <(e_2\ a_2)>_s$.

The failure executions and past executions of a sequence of primitive plans, parallel primitive plans, and non-deterministic primitive plans can be stated in a similar manner.

Finally, we consider the success or failure executions of primitive plan types. This is straightforward: a primitive plan is successfully executed if it labels a success arc and fails if it labels a failure arc. If e is a primitive plan type then we have the following semantics:

$M, v, (w_{t_0}, w_{t_1}, ...) \models <e>_s$ iff for some t_1, $e \in \mathcal{S}_w(t_0\ t_1)$
$M, v, w_{t_1} \models [e]_s$ iff for some t_0, $e \in \mathcal{S}_w(t_0\ t_1)$
$M, v, (w_{t_0}, w_{t_1}, ...) \models <e>_f$ iff for some t_1, $e \in \mathcal{F}_w(t_0\ t_1)$
$M, v, w_{t_1} \models [e]_f$ iff for some t_0, $e \in \mathcal{F}_w(t_0\ t_1)$.

We define attempting an execution as either a successful execution or a failure execution, i.e., $<e> \equiv <e>_s \vee <e>_f$ and similarly for past executions.

Using these definitions we can distinguish between having and executing a social plan. More formally, the following formulas are satisfiable in our logic:

- having a plan and not executing the body of the plan:
  $has\text{-}plan(p\ x) \wedge \neg <body(p)>$; and

- executing the body of the plan but not executing it successfully:
  $<body(p)> \wedge \neg <body(p)>_s$.

In the above case x can be a plan expression to achieve or test a certain condition. The last property is the same as execution failure of the body of p.

5.4 Semantics of Mutual Beliefs and Joint Goals

Belief is modeled in the conventional way. That is, instead of one world we have a set of different possible worlds. A particular time point in a particular world is called a *situation.* For each situation we associate a set of *belief-accessible*, *goal-accessible*, and *intention-accessible* worlds; intuitively, those worlds that the agent *believes* to be possible, *desires* to bring about, or *commits* to achieving, respectively. Unlike most conventional models of belief, however, each belief-, goal-, and intention-accessible world is a time tree. Multiple possible worlds result from the agent's lack of knowledge about the state of the world. But within each of these possible worlds, the branching future represents the *choice* of actions available to the agent. Moving from belief to goal to intention worlds amounts to successively pruning the paths of the time tree; intuitively, to making increasingly selective choices about one's future actions.

The belief relation maps a possible world at a time point for a particular agent to a set of belief-accessible worlds. We say that an agent a has a belief ϕ, denoted $\mathsf{BEL}(a\ \phi)$, at time point t if and only if ϕ is true in all the belief-accessible worlds of the agent at time t. We use $B_t^w(a)$ to denote the set of belief-accessible worlds of agent a from world w and time t, i.e., $B_t^w(a) = \{\ w' \mid \mathcal{B}(a\ w\ t\ w')\}$.

The semantics for beliefs can be defined formally as follows:

$M, v, w_t \models \mathsf{BEL}(a\ \phi)$ iff $\forall w' \in \mathcal{B}_t^w(a)\ M, v, w'_t \models \phi$.

The semantics of goals and intentions are defined analogously by using the relations $\mathcal{G}$ and $\mathcal{I}$ [14].

The main semantic constraint imposed on the belief, goal, and intention relation is that for each belief-accessible world there exists a sub-world which is goal-accessible and, in turn, for each goal-accessible world there exists a sub-world which is intention-accessible. This semantic constraint is called strong realism and is formalized elsewhere [14]. Defining O-formulas to be well-formed formulas that contain no positive occurrences of inevitable (or negative occurrences of optional) outside the scope of belief, goal, or modal operators, we have the following axiom of strong realism.

Strong Realism Axiom: $\mathsf{INTEND}(a\ \psi) \supset \mathsf{GOAL}(a\ \psi) \supset \mathsf{BEL}(a\ \psi)$, where ψ is any O-formula.

This axiom states that, if the agent intends optionally to do an action, he should have a goal that optionally he is going to do the action and also believe that he will optionally do it. Weaker forms of this axiom and their corresponding semantic conditions are discussed by us elsewhere [12].

Now we can show that having an *intention* towards the body of a plan is different from *having* a plan and also different from *executing* the body of the plan. In other words, having a plan does not entail intention to execute the body of the plan structure and executing the body of the plan structure does not entail an intention to do so. More formally, the following formulas are satisfiable in our logic:

- having a plan and not intending to execute the body of the plan:
  $\mathit{has\text{-}plan}(p\ !(\alpha\ a)) \wedge \neg\mathsf{INTEND}(a\ <body(p)>)$; and

- executing the body of the plan and not intending to execute the body of the plan:
  $<body(p)> \wedge \neg\mathsf{INTEND}(a\ <body(p)>)$.

Next we examine the semantics of all members of a social agent believing a formula. The formula $\mathsf{EBEL}(y\ \phi)$ is satisfiable iff all members of the social agent y believe in ϕ. If the member is an individual

agent, he believes in it; if the member is another social agent, all members of that social agent believe in it. Thus, unlike previous work [6], the definition of "everyone believes" is recursive.

$$\mathsf{EBEL}(y\ \phi) \equiv \bigwedge_{\{a \mid members(y\ a)\ and\ a \in IA\}} \mathsf{BEL}(a\ \phi) \wedge \bigwedge_{\{z \mid members(y\ z)\ and\ z \in SA\}} \mathsf{EBEL}(z\ \phi).$$

The satisfaction of EGOAL and EINTEND are defined likewise.

Now we can define the mutual belief ϕ of a social agent as being all members of the social agent believing ϕ and all of them believing that ϕ is mutually believed. Joint goal ϕ of a social agent is defined to be all members of the social agent having the goal ϕ and mutually believing that ϕ is held as a joint goal. Joint intentions are defined in the same way as joint goals.

$$\mathsf{MBEL}(y\ \phi) \equiv \mathsf{EBEL}(y\ \phi) \wedge \mathsf{EBEL}(y\ \mathsf{MBEL}(y\ \phi))$$
$$\mathsf{JGOAL}(y\ \phi) \equiv \mathsf{EGOAL}(y\ \phi) \wedge \mathsf{MBEL}(y\ \mathsf{JGOAL}(y\ \phi))$$
$$\mathsf{JINTEND}(y\ \phi) \equiv \mathsf{EINTEND}(y\ \phi) \wedge \mathsf{MBEL}(y\ \mathsf{JINTEND}(y\ \phi)).$$

Note the asymmetry between the definitions of MBEL and JGOAL; while MBEL allows arbitrary nestings of BEL operators, JGOAL allows arbitrary nestings of BEL operators with the innermost operator being a GOAL operator. However, there is a symmetry between the definitions of JGOAL and JINTEND; both allow arbitrary nestings of BEL operators with the innermost operator being GOAL and INTEND, respectively.

The above definitions together with the strong realism axiom yields the following important theorem.

Theorem 1 : $\models \mathsf{JINTEND}(y\ \psi) \supset \mathsf{JGOAL}(y\ \psi) \supset \mathsf{MBEL}(y\ \psi)$, where ψ is any O-formula.

This theorem states that, if a social agent jointly intends an O-formula, the social agent also has it as a joint goal and also mutually believes it. Note that this multi-agent version of strong realism is a consequence of our definitions of joint propositional attitudes and the strong realism axiom for individual agents; it need not be defined as an axiom.

Consider the interesting case where ψ is $<(e\ a);(f\ b)>$. If a social agent y, consisting of members a and b, jointly intends this formula, we have the following formulas being true:

- individual beliefs, goals, and intentions:
  1. $\mathsf{INTEND}(a <(e\ a);(f\ b)>)$;
  2. $\mathsf{INTEND}(b <(e\ a);(f\ b)>)$;
  3. $\mathsf{GOAL}(a <(e\ a);(f\ b)>)$;
  4. $\mathsf{GOAL}(b <(e\ a);(f\ b)>)$;
  5. $\mathsf{BEL}(a <(e\ a);(f\ b)>)$; and
  6. $\mathsf{BEL}(b <(e\ a);(f\ b)>)$;
- beliefs about individual beliefs, goals, and intentions:
  1. $\mathsf{BEL}(a\ \mathsf{INTEND}(b <(e\ a);(f\ b)>))$;
  2. $\mathsf{BEL}(b\ \mathsf{INTEND}(a <(e\ a);(f\ b)>))$;
  3. $\mathsf{BEL}(a\ \mathsf{GOAL}(b <(e\ a);(f\ b)>))$;

4. BEL(b GOAL(a <(e a);(f b)>));
5. BEL(a BEL(b <(e a);(f b)>)); and
6. BEL(b BEL(a <(e a);(f b)>)).

This nesting of beliefs can repeat itself up to an arbitrary depth. Thus, by adopting the strong realism axiom for individual propositional attitudes and defining social propositional attitudes in the above manner, we are able to derive all the important conditions for joint action.

As with individual intentions, we can show that having a *joint intention* towards the body of a social plan is different from *having* a social plan and also different from *executing* the body of the social plan. More formally, the following formulas are satisfiable in our logic:

- having a plan and not jointly intending to execute the body of the plan: *has-plan*(p !(α y)) $\wedge$ $\neg$JINTEND(y <*body*(p)>); and
- executing the body of the plan and not jointly intending to execute the body of the plan: <*body*(p)> $\wedge$ $\neg$JINTEND(y <*body*(p)>).

6 Multi-Agent BDI-Architecture

In this section, we briefly discuss how one can make use of the above formalism in designing a multi-agent BDI-architecture. First, we consider the single agent scenario.

We would like to model the means-end reasoning of a single agent within a BDI-architecture. We can do this in a number of different ways. We first present the minimal version that all rational agents need to satisfy and then a strong version satisfied by strongly-committed rational agents.

We can say that, whenever an agent intends the body of a plan structure, then he must have a goal towards the purpose of the plan and the preconditions must be believed.

$\models$ *has-plan*(p (!α a)) $\wedge$ INTEND(a <*body*(p)>) $\supset$ GOAL(a <(!α a)>$_s$) $\wedge$ BEL(a *pre*(p))

However, this requirement alone is not sufficient for the agent to form intentions and act based on them. We need additional constraints that would force the agent to form intentions.

The stronger version of the means-end reasoning axiom can be stated as follows: If an individual agent has a plan p and has acquired the goal towards the purpose of this plan, and believes in the precondition of the plan, he will intend to execute the body of the plan. The body of the plan may contain other achievement plan expressions. An agent intending such an achievement plan expression would then be forced to have a goal to achieve it (by the strong realism axiom). This goal may result in further intentions to execute the body of other social plan structures. This hierarchical planning proceeds until the agent has executed the body of his top-level plan structure. Thus, for an individual agent a we have the following axiom for means-end reasoning:

$\models$ *has-plan*(p (!α a)) $\wedge$ GOAL(a <(!α a)>$_s$) $\wedge$ BEL(a *pre*(p)) $\supset$ INTEND(a <*body*(p)>)

Note that, whenever the premise of the axiom is true, the agent is going to intend the body of the plan structure. However, the agent may not act on all such intentions; an agent acts only if his immediate intention is towards a non-deterministic action [14]. In other words, if the agent has multiple present-directed intentions, he needs to deliberate and choose the best possible action before acting. The agent is allowed to have multiple future-directed intentions as he can keep postponing deliberation until he is forced to act.

The scenario for multiple agents is very similar — one considers joint attitudes rather than individual attitudes. Thus, if a social agent has a social plan p and has acquired the joint goal towards the purpose of this plan, and mutually believe in the precondition of the plan then the agent will jointly intend to execute the body of the plan. This joint intention would trigger the social agent to acquire other joint and individual goals which might trigger further joint intentions, and so on. As before, if the social agent has successfully executed the body of the social plan structure, we can say that the social agent mutually believes the postcondition. Thus for a social agent y we have the following axiom for hierarchical planning:

$$\models \textit{has-plan}(p\ (!\alpha\ y)) \wedge \mathsf{JGOAL}(y\ <(!\alpha\ y)>_s) \wedge \mathsf{MBEL}(y\ \textit{pre}(p)) \supset \mathsf{JINTEND}(y\ <\textit{body}(p)>)$$

The above axioms also hold when the agent has a plan to test for a certain condition.

In this section we have illustrated a simple design of rational agents that can perform hierarchical planning by having social plans and adopting joint goals and joint intentions. This, however, should be viewed only as a preliminary step; one needs to formalize how the deliberation and negotiation of agents lead to the formation of joint intentions (similar to the single-agent case discussed elsewhere [13]); when and how agents reconsider their joint and individual intentions; how the social roles and commitments affect the joint goals and joint intentions of agents.

7 Related Work and Conclusion

In this section, we briefly describe related work on the formalization of joint intentions. One key question to be considered is: *What is needed to characterise collective intention in addition to the conjunction of individual intentions?* There is no general agreement as to the answer to this question.

In the work of Tuomela and Miller [17] and Grosz and Sidner [5], joint intentions are reduced to intentions-in-action of individual agents and mutual beliefs about such intentions. Others, such as Searle [15] and Hobbs [8], argue that joint intentions are not reducible in this way. A central example used to discuss the reducibility question is the MBA example introduced by Searle [15]. All MBA graduates are (successfully) taught that each can serve humanity by pursuing his own selfish interests. If each agent intends to serve humanity by pursuing his own interests, each agent believes that every other MBA graduate would do the same, and there is a mutual belief to this effect, then under the definitions given by Tuomela and Miller, and Grosz and Sidner, the MBA graduates would be said to have a joint intention. However, Searle argues that, in this scenario, there is no joint intention and, further, that the ideology of the particular business school, accepted by all graduates, is that there should not be a joint intention. In our formalization, a social agent (all MBA graduates) who jointly intends to serve humanity must also have a joint goal to serve humanity. This joint goal could have been obtained by some form of prior communication (like all the MBA graduates meeting together on graduation day and jointly adopting the goal to serve humanity). If there is no such joint goal, there is no joint intention.

In his analysis, Hobbs recasts Searle's position in the language of beliefs, goals, and plans. He argues for a notion of a *collective agent* and the need for joint intentions to be related to the attitude of joint goals and appropriate mutual beliefs. Hobbs also reiterates the importance of concepts such as *commitment*, which creates mutual belief in a collective plan, and *responsibility*, which holds each agent to his part, to a more complete understanding of the concept of joint intention.

Commitment is the core of the formalization adopted by Levesque, Cohen, and Nunes [9] in their approach to joint intention. In particular, they capture the commitment of an individual agent to communicate his private belief about the success or failure of a joint goal to other members involved in

the joint activity. This formalization does not suffer from some of the drawbacks discussed by Searle and Hobbs. However, the logic is not expressive enough to capture some important types of collective behaviour – mainly because of the lack of a *collective agent* and the notion of an *plan types.*

Singh [16] adopts a different approach to defining joint intentions, one which does not explicitly invoke other propositional attitudes. He presents a theory of the intentions of a group of agents in terms of the actions done by the members of the group of agents and their social structure, as it emerges from their interactions. To address the *side-effect problem*, Singh recognizes the desirability of distinguishing between a strategy and the purpose of that strategy. In Singh's theory, intentions are ascribed from observation of agent interactions. The formalism does not allow explicit reasoning about the attitudes of other agents.

Werner [18] provides a comprehensive theory of social structures, social groups, and social roles. Intentions and joint intentions are an integral part of his theory. In Werner's approach an intentional state is a class of strategies that guide the actions of a given agent. A strategy is a mapping from (partial) histories or information states to alternative (partial) histories. These strategies include the individual, collective and communicative actions of agents. Werner provides a theory of communication which enables one to formalize how messages affect the intentions of an agent and also addresses important issues such as the (possibly changing) roles of agents within a group setting.

In conclusion, the primary contributions of this paper are: (a) the introduction of collective agents, called *social agents*; (b) an ontology for social plan structures; (c) provision of a semantics of successful and failure executions of such social plans; (d) semantics of collective attitudes such as mutual belief, joint goal, and joint intention; and (e) the foundations for a theory of multi-agent BDI-architecture that can reason with the above entities. As a result of this, we can distinguish a social agent *having* a social plan from a social agent *executing* a social plan from a social agent *jointly intending* to execute a social plan.

References

[1] M. E. Bratman. *Intentions, Plans, and Practical Reason.* Harvard University Press, Massachusetts, 1987.

[2] M. E. Bratman, D. Israel, and M. E. Pollack. Plans and resource-bounded practical reasoning. *Computational Intelligence*, 4:349–355, 1988.

[3] E. A. Emerson and J. Srinivasan. Branching time temporal logic. In J. W. de Bakker, W.-P. de Roever, and G. Rozenberg, editors, *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, pages 123–172. Springer-Verlag, Berlin, 1989.

[4] M. P. Georgeff and A. L. Lansky. Procedural knowledge. In *Proceedings of the IEEE Special Issue on Knowledge Representation*, volume 74, pages 1383–1398, 1986.

[5] B. J. Grosz and C. L. Sidner. Plans for discourse. In P. R. Cohen, J. Morgan, and M. E. Pollack, editors, *Intentions in Communication.* MIT Press, Cambridge, Ma., 1990.

[6] J. Y. Halpern and Y. Moses. Knowledge and common knowledge in a distributed environment. *Journal of the Association for Computing Machinery*, 37:549–587, 1990.

[7] D. Harel. Dynamic logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic, Vol II*, pages 497–604. D. Reidel Publishing Co., New York, New York, 1984.

[8] J. R. Hobbs. Artificial intelligence and collective intentionality: Comments on Searle and on Grosz and Sidner. In P. R. Cohen, J. Morgan, and M. E. Pollack, editors, *Intentions in Communication.* MIT Press, Cambridge, Ma., 1990.

[9] H. J. Levesque, P. R. Cohen, and J. H. T. Nunes. On acting together. In *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-90)*, pages 94–99, 1990.

[10] D. J. Litman and J. Allen. Discourse processing and commonsense plans. In P. R. Cohen, J. Morgan, and M. E. Pollack, editors, *Intentions in Communication.* MIT Press, Cambridge, Ma., 1990.

[11] M. E. Pollack. Plans as complex mental attitudes. In P. R. Cohen, J. Morgan, and M. E. Pollack, editors, *Intentions in Communication.* MIT Press, Cambridge, Ma., 1990.

[12] A. S. Rao and M. P. Georgeff. Asymmetry thesis and side-effect problems in linear time and branching time intention logics. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91)*, 1991.

[13] A. S. Rao and M. P. Georgeff. Deliberation and its role in the formation of intentions. In *To appear in the Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence (UAI-91)*, 1991.

[14] A. S. Rao and M. P. Georgeff. Modeling rational agents within a BDI- rchitecture. In J. Allen, R. Fikes, and E. Sandewall, editors, *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning.* Morgan Kaufmann Publishers, San Mateo, 1991.

[15] J. R. Searle. Collective intentions and actions. In P. R. Cohen, J. Morgan, and M. E. Pollack, editors, *Intentions in Communication.* MIT Press, Cambridge, Ma., 1990.

[16] M. P. Singh. Group Intentions. In *Proceedings of the 10th International Workshop on Distributed Artificial Intelligence (DAI-10).* MCC Technical Report ACT-AI-355-90, 1990.

[17] R. Tuomela and K. Miller. We-intentions. *Philosophical Studies*, 53:367–389, 1988.

[18] E. Werner. Cooperating agents: A unified theory of communication and social structure. In L. Gasser and M. N. Huhns, editors, *Distributed Artificial Intelligence: Volume II.* Morgan Kaufmann Publishers, 1990.

Collaborative Plan Construction for Multiagent Mutual Planning

Ei-Ichi Osawa Mario Tokoro[1]
e-mail: osawa@csl.sony.co.jp, mario@csl.sony.co.jp

Sony Computer Science Laboratory Inc.
Takanawa *Muse* Building,
3-14-13 Higashi-gotanda, Shinagawa-ku,
Tokyo, 141 JAPAN

Abstract

In multiagent planning an agent sometimes needs to collaborate with others to construct complex plans, or to accomplish large organizational tasks which he cannot do alone. Since each agent in a group may have incorrect beliefs about the world, and because agent's abilities differ, construction of a coordinated plan can be confounded. In this paper we propose a scheme for constructing plans for collaborating agents from their, possibly incorrect, beliefs and partial knowledge of the world. In the proposed scheme, when agents want to accomplish a goal together, each agent first proposes a possibly incomplete individual plan based on his own beliefs and skills. Then the agents use their individual plans to mutually construct a collaborative plan to accomplish the goal. A collaborative activity can be a composite action involving parts to be done by one agent, parts to be done concurrently by agents, and parts to be done jointly. The proposed method makes it possible for each agent to decide, without excessive communication, what actions he should take in collaboration.

1 Introduction

In multiagent systems, intelligent relatively small systems called agents interact to solve problems in a cooperative way. In such a system, agents pool their skills to achieve complex goals by dynamically forming an organization or a group. Recent developments in *open distributed environments* [Tokoro 90] show that using a multiagent approach to construct systems in an open distributed environments is promising from several points of view [Gasser and Huhns 89].

Some issues in multiagent systems are not yet well understood. One of these is mutual plan construction through multiagent cooperative planning. In multiagent cooperative plan construction, several agents mutually generate collaborative plans by means of inference based on their own, possibly incorrect, beliefs and partial knowledge about the world. Mutual planning is confounded by disparities among goals and intentions, as well as inconsistencies in world knowledge.

In a multiagent system an agent may have a goal or task which he cannot do alone. *Contract-net* protocol [Davis and Smith 83] provides a way for an agent who needs help (this agent is called a manager) to dynamically decompose the task into subtasks, and to allocate the subtasks to other agents (contractors) through negotiation. The contract-net protocol also provides dynamic and opportunistic control.

In open distributed environments, services, processing capacity, and the connection topology of computing elements are continuously changing. At the same time, the granularity of agents and plans are changing dynamically. Also agents are heterogeneous. Although the contract-net type organization schemes are usually preferable in open distributed environments because of its dynamic nature, a multiagent system embodies additional complexity which makes application of the contract-net difficult.

[1] also with Keio University, 3-14-1 Hiyoshi, Kohoku-ku, Yokohama, 223 JAPAN

Two such problems in the contract-net occur in decomposition and task allocation. When the manager first decomposes the task, his fixed decomposition of the task may not suit the open distributed environment. Not only may he not know what agents are currently available, but he also may not know the changing skills of potential contractors. The manager then selects one agent per subtask through negotiation, and allocates the subtask to that agent. No single agent may have a plan to achieve the subtask by himself. Even though subcontracting is possible, this fixed task allocation strategy which assigns a subtask to only one agent may results an ineffective hierarchy of subcontracts.

If we apply the contract-net protocol to hierarchical multiagent planning, the problems become more serious. The manager wants some agent to accomplish a goal, but if he does not have sufficient knowledge to decompose a complex goal properly in an open distributed environment, he cannot ask any single agent to achieve the goal. His task allocation strategy fails. Therefore we need a more flexible strategy for selecting contractors.

Suppose the manager can somehow select several agents as collaborative contractors. Here some questions arise. What information should he provide those contractors? In other words, what information is necessary for the contractors to mutually construct collaborative plans? Also, how should the mutual plan construction be coordinated and organized?

In this paper we first describe our agent model, and then characterize our dynamic organization scheme. We are developing an experimental environment for a multiagent planning system. In this system we call the model of an agent, *SocioAgent* [Osawa and Tokoro 90]. If an agent needs help, he organizes a cooperative group using the dynamic organization scheme. In this process, he initially plays the role of manager, selecting agents and allocating goals to them, but he can also be a member of the collaborating group. We then propose a strategy and algorithm for selecting collaborative contractors. In selecting contractors, information for collaboration, which we call *suggestions for collaboration*, is generated by the manager. The contractors are said to be given a *collaborative award*, when they receive these suggestions for collaboration. Then we characterize mutual plan construction by contractor agents in terms of *elaboration of individual plans*. Elaboration of a individual plan involves inference based on each agent's partial view of the world and the suggestions for collaboration given by the manager. A contractor agent infers plans of other collaborating agents.

The organization of this paper is as follows. In Section 2 we give a concrete example of multiagent planning which illustrates the problems discussed above. Section 3 gives our basic agent model. In Section 4 we propose a strategy and algorithm for selecting collaborative agents. Then we characterize the mutual plan construction, and describe an algorithm for collaborative plan construction. Section 5 gives the relationship between our scheme and other work in this area, and Section 6 contains our conclusions.

2 Problems of Cooperative Planning in Multiagent System

Activities by two or more agents can be viewed as a composite action involving: (1) actions to be done concurrently by two or more agents; (2) actions to be done by either agent sequentially; and (3) actions to be done by both or many agents together. The first and second cases, in which agents act in a synchronous manner to cooperate and avoid conflicts, are well studied in [Georgeff 83] as coordinating performed plans. In coordinating performed plans agents are not necessarily contributing to the same goal. In collaborative activities each agent helpfully (or positively) contribute its own skills to the successful achievement of organizational tasks. In this paper we will focus on *collaborative activities* which subsumes above mentioned three cases.

An agent's possibly incorrect beliefs and partial knowledge of the world can cause various difficulties when several agents collaborate to accomplish a goal. To examine the difficulties in detail, we give the following example. The goal in the example is for agent g_3 to have block b in room r_3 (figure 1).

In this example we assume that each agent, g_1, g_2, g_3, has the action rules and beliefs about the world given in the Appendix. We add to a first-order language with equality the operator B to model

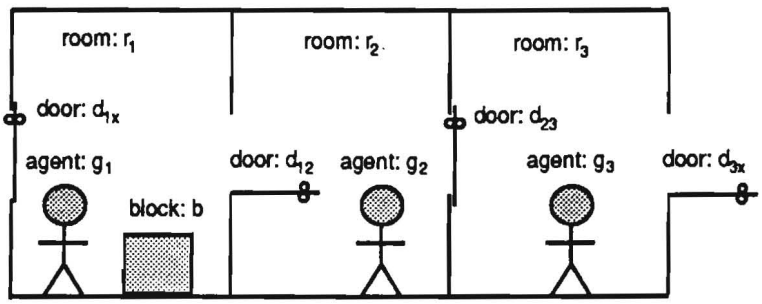


Figure 1: Moving a block along adjacent rooms

a agent's beliefs. $B(g,p,t)$ says that agent g has a belief p at a time point t. For the belief operator B we assume axioms inspired by [Halpern and Moses 85] and a frame assertion axiom (details are given in Section 3). From now on, in logical formulae, arguments which begin with upper case letters and lower case letters represent variables and ground instances, respectively.) Note that in this section, constructors such as ";" ,"||" are used for convenience to compose a sequence of actions, concurrent actions, respectively. In the following section these constructs are excluded by introducing constraints on the temporal ordering of actions.

Agent g_3 wants to have block b in room r_3. He knows that by performing $\mathsf{trans}(Agent, g_3, b)$, he can hold block b. However, since some parts of the precondition of the action, i.e. $(holding(Agent,b), in(Agent,r_3))$, don't hold at this moment, he needs to ask other agents to achieve this goal, conditioned by the fact that block b is not in room r_3 at this moment.

The contract-net provides for allocation of the goal to other agents by communication. In this case, agent g_3 plays the role of manager and announces the goal status as a task. If the two agents g_1 and g_2 individually respond to the announcement, and each can successfully generate a plan, then both give their plans to g_3 . This is called a bid. g_3 collects bids from potential contractors, evaluates them, and *awards* the task to one of the contractors.

In this example, because g_1 and g_2 have only partial knowledge of the situation, neither of them is able to generate a complete plan to achieve the goal. Therefore, if the manager requires that a bid be complete, neither returns a bid for the contract. If the manager does allow incomplete bid plans[2], agents g_1 and g_2 may return the incomplete individual plans, P_{g_1} and P_{g_2}, respectively.[3] Individual plan construction is described in the next section. These individual plans are constructed by a DCOMP-type backward production [Nilsson 80].

$$
\begin{aligned}
P_{g_1} = \{&\mathsf{pickup}(g_1,b)\\
&\mathsf{move}(g_1,b,r_1,r_2);\\
&\mathsf{move}(g_1,b,r_2,r_3)\\
&\qquad \textit{precond: } boundary(Door,r_2,r_3),\ open(Door)\}.
\end{aligned}
$$

$$
\begin{aligned}
P_{g_2} = \{\{&\mathsf{open}(g_2,d_{23});\\
&\mathsf{trans}(Agent,g_2,b)\\
&\qquad \textit{precond: } holding(Agent,b), in(Agent,r_2)\};\\
&\mathsf{move}(g_2,b,r_2,r_3)\}.
\end{aligned}
$$

Both P_{g_1} and P_{g_2} are incomplete plans, and the operators $\mathsf{move}(g_1,b,r_2,r_3)$ in P_{g_1} and $\mathsf{pickup}(g_2,b)$

[2] An incomplete plan is a plan which includes operators with unknown preconditions. These operators are called incomplete operators.

[3] Each of these incomplete individual plans is one of many incomplete alternatives of agents g_1 and g_2.

in P_{g_2} are incomplete operators.

In the contract-net, after manager g_3 receives these bids, he would select one contractor, and allocate the task to that contractor assuming that incomplete operators will be later resolved by subcontracts. However, in this case, both bids are incomplete, so it is not possible for the manager to determine which bid is better, since he also does not have complete knowledge of the situation.

The contract-net provides a general protocol for decomposing a task and awarding the subtasks to contractors. This decomposition is from the manager's point of view. Since managers in the contract-net award each task to one contractor, the allocation may not be optimal in an open distributed environment. Even though subcontracting is possible, since the granularity of agents and plans constantly changes, the fixed task allocation strategy may not be sufficient.

In the given example, if the manager selects g_2, the incomplete operator $\mathsf{trans}(Agent, g_2, b)$ in P_{g_2} can be resolved with subcontract between g_2 and g_1. In this case, the complete plan will be in the form;

$$\{\{\mathsf{open}(g_2, d_{23}) \parallel \{\mathsf{pickup}(g_1, b);\ \mathsf{move}(g_1, b, r_1, r_2)\};\ \mathsf{trans}(g_1, g_2, b)\};\ \mathsf{move}(g_2, b, r_2, r_3)\}.$$

However, if we have a whole view of the situation, we see that the plan is less efficient than the following plan, $CP_{g_1 g_2}$.

$$CP_{g_1 g_2} = \{\{\mathsf{open}(g_2, d_{23}) \parallel \{\mathsf{pickup}(g_1, b);\ \mathsf{move}(g_1, b, r_1, r_2)\}\};\ \mathsf{move}(g_1, b, r_2, r_3)\}.$$

If g_1 is selected, he still needs a subcontractor to open door d_{23}, since he does not have the skill. Since contracting is computationally expensive in a multiagent system, the above mentioned collaboration, $CP_{g_1 g_2}$, is cheaper than subcontracting.

In the following sections, we describe construction of incomplete individual plans based on partial knowledge of the world. We then propose a protocol for forming dynamic organizations and a method for mutual collaborative planning.

3 SocioAgent Model

We have developed an experimental environment for multiagent planning based on an agent model called SocioAgent. In this section we describe communication primitives, communicative actions, and the planning mechanism of SocioAgent. SocioAgent, hereafter simply called agent, has two plan construction phases. The first is *individual plan construction.* In this phase, an agent generates a *individual plan* to achieve a given goal. This is an action sequence based on his beliefs and partial knowledge of the world. Therefore, a *individual plan* can be incomplete. The second phase is *collaborative plan construction.* This process occurs when agents are to achieve an organizational goal. In this process each collaborating agent elaborates on his own initial individual plan. We characterize *collaborative plan construction* in the next section as a part of the organizational scheme we propose.

3.1 Communication Primitives

Agents have two communication primitives: **send** and **receive**. If agent *Sender* executes

$$\mathsf{send}(\{Sender\}, Recipient, Message),$$

the message $(Sender, Recipient, Message)$ is sent from agent *Sender* to agent *Recipient*. If *Sender* is omitted, it is, by default, filled in with name of the agent executing the message sending primitive.

The message receiving primitive **receive** reads a message from the incoming message queue of the receiving agent. If an agent executes

$$\mathsf{receive}(\{Sender\}),$$

the first message sent by agent *Sender* is retrieved in the form $(Sender, Message)$ from the incoming message queue. If no *Sender* is specified in a **receive** primitive, the first message in the queue is retrieved and removed. This selective message receive primitive is necessary because while an agent is collaborating with other agents, he may want to receive messages only from other group members. Also, while he is waiting for a message from a specific agent about an urgent matter, he may not want to be disturbed by messages from other agents.

3.2 Communicative Actions

By sending the message

$$\textsf{request}(Sender, Recipient, ToDo),$$

Sender indicates that he requires *Recipient* to perform the action *ToDo*.

In the following message, *Agent* asks *Recipient* to tell him whether or not he knows the truth value of P. If *Agent* is not specified, then by default, *Agent* is the sender of this message.

$$\textsf{request}(\{Agent\}, Recipient, \textsf{inform}(Recipient, \{Agent\}, P))$$

3.3 Belief Model

We assume that every agent has a set of beliefs about the world, which may include beliefs about other agents' beliefs. We add to a first-order language with equality the operator B. $B(g,p,t)$ says that agent g has a belief p at a time point t[4]. The B operator is assumed to satisfy the following axioms, where P and Q are scheme variables ranging over propositions, G ranges over agents, and T ranges over time points.

1. $B(G,P,T) \wedge B(G,P \supset Q,T) \supset B(G,Q,T)$
2. $B(G,P,T) \supset B(B(G,P,T))$
3. $\neg B(G,P \wedge \neg P,T)$
4. $\forall T(> T')\ \nexists T''(T \geq T'' > T')B(G,P,T') \wedge B(G,\neg P,T'') \supset B(G,P,T)$

3.3.1 Belief Revision

Each agent maintains a database as its belief space. An agent may revise its beliefs after executing actions, and in the course of interaction with other agents. When an agent executes an action, he adds effects of the action to his belief database. A belief $B(G',P,T)$ obtained in the course of interaction with an agent G' is also added to belief database as its own belief $B(G,P,T)$, if P satisfies the following condition without utilizing the frame assertion.

$$\nexists T'(> T)B(G,\neg P,T)$$

3.4 Individual Plan Construction

The action rules of SocioAgent are as follows.

This plan scheme is similar to that of STRIPS [Fikes and Nilsson 71] except that, in SocioAgent, each operator is associated with a temporal variable and an execution time cost which is the sum

[4] A time point is obtained from the virtual clock of the agent. Precisely speaking, a virtual clocks do not indicate the global time, but indicates the local times of the agent. However we assume that these virtual clocks are periodically adjusted, so that they maintain times which are sufficiently precise for the inference and planning process of agents.

$$\begin{aligned}&\textsf{operatorHead}(Agent, ParametersList, T, \tau)\\ &\quad \textit{precond: } B(Agent, P_1, T), \cdots, B(Agent, P_n, T)\\ &\quad \textit{body: } ActionSequence\\ &\quad \textit{effect: } B(Agent, Q_1, T), \cdots, B(Agent, Q_m, T)\end{aligned}$$

of the predicted costs of the actions in the body of the operator. The cost of a primitive action is predicted from an agent's working environment.

Individual plan construction is done by a DCOMP-type backward production [Nilsson 80]. In the backward plan construction, if all preconditions become trivial, i.e. all preconditions are compatible with the agent's beliefs, the sequence of actions obtained in the production process is regarded as a plan [Nilsson 80]. In SocioAgent, this plan is called complete. Certain incomplete plans can also form a individual plan of an agent.

In conventional plan synthesis, if an agent discovers an incomplete operator, he discards it. However in SocioAgent, an agent continues to construct a plan even though it may contain incomplete operators. Plans P_{g_1} and P_{g_2} in the previous section are examples of incomplete plans in SocioAgent. A individual plan of SocioAgent is generally given in the following form.

$$\{O_1(Agent, Objects_1, T_1, \tau_1), \cdots, O_n(Agent, Objects_n, T_n, \tau_n)\}\ C(T_1, \cdots, T_n, \tau_1, \cdots, \tau_n)$$

where $C(T_1, \cdots, T_n, \tau_1, \cdots, \tau_n)$ denotes a set of constraints on temporal ordering of actions.

Plans can be viewed as directed acyclic graphs, whose nodes are labeled with operator headers and propositions; when a node is labeled with a proposition P, it denotes any action that would achieve P. A temporally ordered partial sequence of actions in a plan is called *a partial plan.*

Using this form, the plan P_{g_1} given in the previous example can be expressed as follows.

$$\begin{aligned}P_{g_1} = \{&\textsf{pickup}(g_1, b, T_j, \tau^{g_1}_{pickup}),\\ &\textsf{move}(g_1, b, sfr_1r_2, T_{j+1}, \tau^{g_1}_{move}),\\ &\textsf{move}(g_1, b, r_2, r_3, T_{j+2}, \tau^{g_1}_{move})\\ &\qquad \textit{precond: } B(g_1, boundary(Door, r_2, r_3), T_{j+2}),\ B(g_1, open(Door), T_{j+2})\}.\\ &T_j + \tau^{g_1}_{pickup} < T_{j+1},\ T_{j+1} + \tau^{g_1}_{move} < T_{j+2}\end{aligned}$$

If we allow this sort of planning, an agent can generate many incomplete plans in his plan search process. Some incomplete plans are useful, and can be successfully completed through collaboration, and some are irrelevant and, thus useless. How can an agent choose from the incomplete plans found in its search process? We propose two criteria for selecting the preferable incomplete plan: the specificity of the plan and the cost. In order to characterize these two aspects of a plan, we address the rationality of agents.

3.5 Rationality

Rationality and its role in reasoning is discussed in [Doyle 90]. In a multiagent system, the notion of rational choice is of great significance. Choices among alternative plans and among potential contract agents occur frequently. In a society of SocioAgents, we require every agent to be individually and organizationally rational. This rationality not only makes it possible for every agent to make good

choices among alternative plans in individual plan construction, but it also enables agents to expect certain decisions and behavior from other agents in a collaborative activity. We discuss organizational rationality in the next section.

An agent rationally chooses a plan partly by comparing costs of alternative partial plans or operators. Cost is a numerical utility function which ranks alternatives according to degree of desirability. The other criteria is the specificity of the plan. If an agent has two incomplete plans as alternatives in a plan construction process, he chooses the one which is more specifically synthesized. For instance agent g_2 can construct the following individual plan P'_{g_2} in the example in Section 2.

$$
\begin{aligned}
P'_{g_2} = \{&\mathsf{open}(g_2, d_{23}, T_i, \tau^{g_2}_{open}),\\
&\mathsf{move}(g_2, nil, r_2, r_1, T_{i+1}, \tau^{g_2}_{move}),\\
&\mathsf{pickup}(g_2, b, T_{i+2}, \tau^{g_2}_{pickup})\\
&\qquad \textit{precond: } B(g_2, onfloor(b, r_1), T_{i+2}),\\
&\mathsf{move}(g_2, b, r_1, r_2, T_{i+3}, \tau^{g_2}_{move}),\\
&\mathsf{move}(g_2, b, r_2, r_3, T_{i+4}, \tau^{g_2}_{move})\}\\
&T_i + \tau^{g_2}_{open} < T_{i+4},\ T_{i+1} + \tau^{g_2}_{move} < T_{i+2},\\
&T_{i+2} + \tau^{g_2}_{pickup} < T_{i+3},\ T_{i+3} + \tau^{g_2}_{move} < T_{i+4}.
\end{aligned}
$$

This individual plan is also incomplete. However, it is much more specific than P_{g_2} given in Section 2, since P'_{g_2} contains only one unknown condition, while P_{g_2} has two. In general we cannot say that a more specific plan is better than a less specific plan, since choices are based on both cost and specificity of alternative plans.

4 Organizational Scheme and Collaborative Plan Construction

SocioAgent dynamically forms an organization according to the protocol described below, using the communication primitives and communicative actions given in the previous section. The outline of the dynamic organizational scheme we propose is similar to the contract-net up to the point at which a manager collects bids (in SocioAgent an announced task is called a request for proposal or RFP). Evaluation by the manager, awarding, and task allocation in our scheme, however, are different because we include the possibility of collaboration among contractor agents.

4.1 Dynamic Organization Scheme

In a society of SocioAgents there are special agents called *bulletin boards*. Bulletin board agents receive RFPs from ordinary agents and save them until an expiration time. Bulletin boards also send saved RFPs to requesting agents. There can be many bulletin boards in a society and all do not necessarily contain the same information. Any given agent is designed to know at least one bulletin board.

An agent that needs to request the help of other agents to achieve a goal first sends an RFP to a bulletin board. The RFP specifies the goal, an expiration time, and the manager's name. The bulletin board agents do not execute parts of RFPs, they only save them, and reply to send requests from ordinary agents. A free agent, i.e. one without a current task, can request the bulletin board to send a saved RFP to him.

When the agent receives the RFP, if he can construct a individual plan, even if it is incomplete, he sends that plan directly to the manager of the RFP. This is a bid as in the contract-net.

Upon expiration of the RFP, if the manager receives bids from more than one agent, he investigates the possibility of collaboration of all or some of these agents according to the algorithm described in the next subsection. (If he receives only one bid, he selects that bidding agent as contractor.)

If collaboration seems possible, he computes *suggestions for collaboration*, and gives bidding agents *collaborative awards* with his suggestions. In this way he requests the contractors to mutually construct a collaborative plan as elaboration of the initial individual plans, and to execute the collaborative plan to achieve the goal. If the manager finds that collaboration is not possible, he selects one agent based on his evaluation of the bids.

4.2 Collaborative Award and Suggestion for Collaboration

In this section we specify an algorithm for selecting collaborative awards, in case of multiple bid submission, and for computing suggestions for collaboration. The Figure 2 describes for two bidding agents for simplicity. Agents G_x and G_y propose individual plans $P_x(= \{O_{x(1)}; \cdots; O_{x(n)}\}C_x)$ and $P_y(= \{O_{y(1)}; \cdots; O_{y(m)}\}C_y)$, respectively. Before manager G_m computes the algorithm, he needs to eliminate those bids whose cost exceeds his requirements for plan execution time. (In the contract-net this evaluation is implemented as an eligibility test.)

1. Copy P_y to P, and empty suggestion lists S_x and S_y.
2. Let $i = 1$. While ($(1 \leq i \leq n)$ and P is not empty), do;
   If $O_{x(i)}$ matches (see the note described below) $O_{y(j)}(1 \leq j \leq m)$, then
      Add $O_{x(i)}$ and $O_{y(j)}$, with associated preconditions in case of an incomplete operator, to
         S_y and S_x, respectively.
      Remove $O_{y(j)}$ from P.
   Increment i.
3. If S_x and S_y are not empty, then
      Delete all matching operators from S_x and S_y which are inconsistent
         with respect to the temporal ordering of actions obtained from C_x and C_x.
4. If S_x and S_y are not empty, then
      Send the collaborative awards (see below), CA_x and CA_y, to agents G_x and G_y, respectively.
   Else
      Compare cost of P_x and P_y, and allocate the task to the agent whose individual plan is cheaper.

Note: 1. In matching of two operators, difference of agents and costs are ignored.

Figure 2: **An algorithm for computing suggestions for collaboration and selecting collaborative awards**

Collaborative awards CA_x and CA_y in the algorithm are given in the following forms.

$$CA_x = \mathsf{request}(G_m,\ G_x,\ \mathsf{collaborate}(G_x,\ \mathsf{suggestion}(S_x),\ P_x)).$$
$$CA_y = \mathsf{request}(G_m,\ G_y,\ \mathsf{collaborate}(G_y,\ \mathsf{suggestion}(S_y),\ P_y)).$$

CA_x, for instance, states that agent G_m requests G_x to collaboratively perform P_x with the suggestion S_x. A suggestion for collaboration given to a contractor agent includes: (1) Explicit obstacles of the other agent which collaboration may be able to resolve; (2) Actions which both collaborating agents can perform. If agents g_1 and g_2 return individual plans P_{g_1} and P'_{g_2} as given in Section 2 and 3, suggestions for collaboration S_{g_1} and S_{g_2} are as follows:

4.3 Mutually Collaborative Plan Construction

An agent who is given a collaborative award tries to construct a cooperative plan for collaboration. He does this through inference based on his initial individual plan, and the suggestions for collaboration given in the award.

$$
\begin{aligned}
S_{g_1} = \{&\mathsf{pickup}(g_2, b, T_{i+2}, \tau^{g_2}_{pickup}) \\
&\quad \textit{precond: } B(g_2, onfloor(b, r_1), T_{i+2}), \\
&\mathsf{move}(g_2, b, r_1, r_2, T_{i+3}, \tau^{g_2}_{move}), \\
&\mathsf{move}(g_2, b, r_2, r_3, T_{i+4}, \tau^{g_2}_{move})\} \\
&T_{i+2} + \tau^{g_2}_{pickup} < T_{i+3},\ T_{i+3} + \tau^{g_2}_{move} < T_{i+4}.
\end{aligned}
$$

$$
\begin{aligned}
S_{g_2} = \{&\mathsf{pickup}(g_1, b, T_j, \tau^{g_1}_{pickup}), \\
&\mathsf{move}(g_1, b, r_1, r_2, T_{j+1}, \tau^{g_1}_{move}), \\
&\mathsf{move}(g_1, b, r_2, r_3, T_{j+2}, \tau^{g_1}_{move}) \\
&\quad \textit{precond: } B(g_1, boundary(Door, r_2, r_3), T_{j+2}),\ B(g_1, open(Door), T_{j+2})\} \\
&T_j + \tau^{g_1}_{pickup} < T_{j+1},\ T_{j+1} + \tau^{g_1}_{move} < T_{j+2}.
\end{aligned}
$$

In such collaborative plan construction, each agent needs to elaborate his initial individual plans to produce efficient collaborative plans for the given goal.

Activities by two agents can be a composite action involving: (1) actions to be done concurrently by two agents, s.t. $\{\mathsf{a}(G_x, T_x), \mathsf{a}(G_y, T_y)\}$; (2) actions to be done by either agent sequentially, s.t. $\{\mathsf{a}(G_x, T_x), \mathsf{a}(G_y, T_y)\}(T_x < T_y)$; and (3) actions to be done by both agents together, s.t. $\mathsf{a}(G_x, G_y, T)$. The first and second cases, in which two agents act in a synchronous manner to cooperate and avoid conflicts, are well studied in [Georgeff 83]. In this paper we will focus on *collaborative activities* by two agents which subsumes these cases.

We raise three significant questions involved in collaboration. First, if the collaborating agents know that both of them can do the same parts of a collaborative activity, how does each agent decide which actions to perform? Second, even though some operators of the initial individual plan of one agent, e.g. G_x, are incomplete, the other agent, G_y, may be able to perform actions which render that individual plan complete. If that is the case, is it possible for G_y to infer that he should do these actions? If so, how is this inference made? Also, is it possible for G_x to infer that G_y will perform these actions to make G_x's individual plan executable? Third, is it possible for the two agents to mutually believe that these actions will be performed in a cooperative way? We now characterize factors which affect the inference in the collaborative plan construction.

In Section 3 we discussed the role of an agent's rationality in individual planning. In this section we again emphasize the need for rationality of agents collaborating to achieve a goal. If there are several possible collaborative plans, the collaborating agents mutually believe that each agent will always choose the most effective plan with respect to execution cost.

To decide which actions will be performed by each agent in a collaborative plan, we look at three factors: obstacle elimination; workload balancing; and cost effectiveness. We then describe the algorithm for action decision.

(1) Obstacle Detection and Elimination

Unsatisfied preconditions of incomplete operators can be regarded as explicit obstacles to the agent's plan. The agent, say G_x, believes that these conditions have not been met or that he cannot satisfy them by himself. The other agent, say G_y, may also include some of these operators in his individual plan. If he knows that the preconditions do hold or that he can act to make them hold, these operators in his individual pan are complete. This information is also included in the suggestions for collaboration given to G_x. Therefore, after G_x receives the award, he believes that the conditions are satisfied or will be eventually. In the suggestion sent to agent G_y, it is specified that G_x does not believe these preconditions are satisfied. So if G_y can satisfy, i.e. he has a partial plan which satisfies these preconditions, he acts to remove these obstacles for G_x. For instance, in the individual plan P_{g_1} of the example given in Section 2, $\mathsf{open}(Door)$ is an obstacle to agent g_1. So when agent g_2 receives

a collaborative award, he knows that this is an obstacle to agent g_1. Therefore, even though agent g_2 does not perform any other actions in collaboration, he believes that he must perform at least $\mathsf{open}(g_2, d_{23}, T_i, \tau^{g_2}_{open})$. If agent G_y believes that the obstacle precondition will never be satisfied, he must inform agent G_x of that belief.

(2) Balancing Workload among Collaborating Agents

As stated above, a collaborative activity by two agents can be a composite action involving sequential, concurrent, and conjoined parts. If these parts form partial plans which can be performed concurrently, and also if both agents are able to execute those partial plans, then the agents must mutually agree on an allocation of actions which will evenly balance the workload. Performing these partial plans concurrently will reduce the execution time of the collaborating plan.

(3) Cost Effectiveness in Collaboration

In the example given in Section 2, there are two alternative collaborative plans to move block b form room r_1 to room r_3, if agent g_1 and g_2 return individual plans P_{g_1} and P'_{g_2}, respectively. (We ignore the action to open door d_{23} for a while.) These plans, $CP'_{g_1g_2}$ and $CP''_{g_1g_2}$, are as follows:

$$CP'_{g_1g_2} = \{\mathsf{pickup}(g_1, b, T_j, \tau^{g_1}_{pickup}),\ \mathsf{move}(g_1, b, r_1, r_2, T_{j+1}, \tau^{g_1}_{move}),\ \mathsf{move}(g_1, b, r_2, r_3, T_{j+2}, \tau^{g_1}_{move})\}$$
$$T_j + \tau^{g_1}_{pickup} < T_{j+1},\ T_{j+1} + \tau^{g_1}_{move} < T_{j+2}.$$

$$CP''_{g_1g_2} = \{\mathsf{move}(g_2, nil, r_2, r_1, T_{i+1}, \tau^{g_2}_{move}),\ \mathsf{pickup}(g_2, b, T_{i+2}, \tau^{g_2}_{pickup}),\ \mathsf{move}(g_2, b, r_1, r_2, T_{i+3}, \tau^{g_2}_{move}),$$
$$\mathsf{move}(g_2, b, r_2, r_3, T_{i+4}, \tau^{g_2}_{move})\}$$
$$T_{i+1} + \tau^{g_2}_{move} < T_{i+2},\ T_{i+2} + \tau^{g_2}_{pickup} < T_{i+3},\ T_{i+3} + \tau^{g_2}_{move} < T_{i+4}.$$

These two plans are actually incomplete. However, if they were complete, by comparing the execution cost of the common actions in $CP'_{g_1g_2}$ and $CP''_{g_1g_2}$, and summing the cost of each operator, an agent can determine which plan is cheaper to execute. Therefore, in general, by comparing the execution cost of common partial plans, agents can decide which agent should perform the partial plans.

Taking these three factors into account, each agent elaborates his initial individual plan to decide which actions to perform using the algorithm described in Figure 3. As a result, he constructs a collaborative plan. In the algorithm described for agent G_x with collaborating agent G_y, cases **CC**, **CI**, **IC**, and **II** indicate the states of completeness of an operator as it appears in the agent's individual plan P_x and the suggestion for collaboration S_x given to him. (These operators are also *shared operators.*) For instance, if an operator $O_{x(i)}$ in P_x is **IC**, it is incomplete, while the shared operator $O_{y(j)}$ in the suggestion S_x is complete. Note that in the algorithm, the shared operator is written as $O_{x(i)}$ and $O_{y(j)}$.

The elaborated individual plan, obtained by executing the algorithm, is called a collaborative operator and written CP_x. Similarly G_y constructs CP_y. Agents g_1 and g_2 of the example in Section 2 construct collaborative plans CP_{g_1} and CP_{g_2}, respectively.

$$CP_{g_1} = \{\mathsf{pickup}(g_1, b, T_j, \tau^{g_1}_{pickup}),$$
$$\mathsf{move}(g_1, b, r_1, r_2, T_{j+1}, \tau^{g_1}_{move}),$$
$$\mathsf{move}(g_1, b, r_2, r_3, T_{j+2}, \tau^{g_1}_{move})\}$$
$$T_j + \tau^{g_1}_{pickup} < T_{j+1},\ T_{j+1} + \tau^{g_1}_{move} < T_{j+2}.$$

$$CP_{g_2} = \{\mathsf{open}(g_2, d_{23}, T_i, \tau^{g_2}_{open})\}$$
$$T_i + \tau^{g_2}_{open} < T_{j+2}.$$

Combining CP_{g_1} and CP_{g_1} in a proper temporal order which satisfies the given time constraints results in the optimal collaborative plan $CP_{g_1g_2}$ given in Section 2.

1. Let $i = 1$. While $(1 \leq i \leq n)$, do;
   If $O_{x(i)}$ is conjoined , then
      If $O_{x(i)}$ is **II**, then
         Remove unsatisfied preconditions not shared by $O_{x(i)}$ and $O_{y(j)}$.
      Else if $O_{x(i)}$ is **IC** or **CI**, then
         Remove all unsatisfied preconditions from $O_{x(i)}$ and $O_{y(j)}$.
   Else
      If $O_{x(i)}$ is **CC**, then
         Find the same partial plans of successive shared but non conjoined operators,
            $P_{x(i,i+k)}$ in P_x and $P_{y(j,j+k)}$ in S_x, and
         Compute costs $\tau_{P_{x(i,i+k)}}$ and $\tau_{P_{y(j,j+k)}}$ of these partial plans;
            If $\tau_{P_{x(i,i+k)}} > \tau_{P_{y(j,j+k)}}$, then
               Remove $P_{x(i,i+k)}$ from P_x, and
               Reflect the removal upon preconditions of subsequent operators of P_x.
               (See the note about the reflect operation described below)
            Else if $\tau_{P_{x(i,i+k)}} < \tau_{P_{y(j,j+k)}}$, then
               Remove $P_{y(j,j+k)}$ from S_x, and
               Reflect the removal upon preconditions of subsequent operators of S_x.
            Else
               Communicate with G_y to decide who will perform the partial plan.
      Else if it is **CI**, then
         Remove $O_{y(j)}$ from S_x, and
         Reflect the removal upon preconditions of subsequent operators of S_x.
      Else if it is **IC**, then
         Remove $O_{x(i)}$ from P_x, and
         Reflect the removal upon preconditions of subsequent operators of P_x.
      Else if it is **II**, then
         Compute costs of $\tau_{O_{x(i)}}$ and $\tau_{O_{y(j)}}$;
            If $\tau_{O_{x(i)}} > \tau_{O_{y(j)}}$, then
               Remove $O_{x(i)}$ from P_x, and
                  Reflect the removal upon preconditions of subsequent operators of P_x.
            Else if $\tau_{O_{x(i)}} < \tau_{O_{y(j)}}$, then
               Remove $O_{y(j)}$ from S_x, and
               Reflect the removal upon preconditions of subsequent operators of S_x.
            Else
               Communicate with G_y to decide who will perform the partial plan.
   Increment i.
2. Remove all operators including non shared ones which are no longer needed to complete the goal.
   (Operators needed for obstacles elimination are left unremoved.)
3. If incomplete operators are left in P_x, then
   Send RFP to bulletin board(s) listing preconditions of incomplete operators as a goal.

Note: In this algorithm, the reflection of removal of operators upon the preconditions of subsequent operators should be the effect specific to his states, since the other agent will perform the partial plan which achieves some sharable states.

Figure 3: **Individual plan elaboration and collaborative plan construction algorithm**

5 Relation to Other Work

In open distributed multiagent systems, agents are continuously changing with respect to their skills and availability, being born, and dying. The contract-net protocol ([Davis and Smith 83]) provides a way for an agent who needs help to dynamically decompose a task into subtasks and allocate these subtasks to other agents through mutual selection. This protocol provides dynamic and opportunistic control in multiagent systems. However, in a open distributed environment each manager have limited knowledge to draw on for task decomposition and allocation, and this approach may not be effective. The *collaborative awards* proposed in this paper make decomposition and allocation of tasks much more flexible. They allow managers to investigate possible collaboration among potential contractors. Complex tasks which are executed in a subcontract hierarchy in the contract-net, can now be performed by means of collaboration among agents at the same hierarchy level. Our scheme provides a better way to accomplish organizational tasks, since tasks are decomposed and allocated based not on only one agent's point of view. Many collaborative plans are investigated among agents, and the optimal one can be selected and performed.

The generalization of centralized planning techniques to accommodate multiple and distributed centers of planning control is investigated on the basis of Sacerdoti's NOAH planning systems in [Corkill 79]. His paper mainly concerns resolving conflicts in distributed hierarchical planning. The top-down decomposition of a conjunctive goal is used to divide the goal into subgoals. Each subgoal is also assigned to distributed planning elements. We are mainly concerned that fixed top-down decomposition of complex goals and fixed top-down assignment of subgoals are not promising in open systems for reasons described in this paper. Our scheme provides dynamic and opportunistic control for generating collaborative plans for complex goals. Furthermore it allows cooperation - individual plan elaboration and collaboration - among planning elements in plan generation.

Coordinating plans in a system is one significant issue in multiagent planning. [Georgeff 83] addresses the issue of combining individual plans in such a way that avoids interference among the agents. In his scheme, appropriate synchronization actions are inserted in original plans. Our method for mutual planning focuses on collaboration among agents. It emphasizes constructing mutually cooperative plans to collaborate from agents' initial, possibly incomplete, individual plans. These two approaches need to be combined for optimal mutual planning.

Obstacle detection and elimination in plan recognition is first introduced in [Allen and Perrault 80], in order to provide a helpful response in discourse. Allen's paper discusses a method for detecting and eliminating implicit obstacles which can be derived by applying knowledge plan inference rules to each step in the plan. In discourse, problems arise from the fact that the entire plan of the speaker may not be inferred. Allen proposes a couple of specific strategies for controlling inference in obstacles detection. In this paper we discuss how to eliminate both explicit and implicit obstacles when observed by other agents having more complete knowledge about the world. In our scheme, the goals of collaborating agents are known to each member of the group, and agents can detect some implicit obstacles in others' plans.

Rationality of an agent is needed to construct a well organized society of agents, since it allows agents to choose among alternatives under uncertainty. [Doyle 90] emphasizes the need for rationality in reasoning. As he pointed out, not only logical rationality, but also economic rationality is of great importance. Rational agents are essential in multiagent systems because their actions are predictable.

[Rosenschein and Genesereth 85] and [Rosenschein *et al.* 86] discuss deals among rational agents. Remarkably, [Rosenschein *et al.* 86] concludes that it is possible to coordinate decisions without communication using explicit models of values and possible choices of other agents under a variety of strong assumptions. In our scheme, an agent given a suggestion for collaboration dynamically sets up a partial model. Since it provides him with predictions of other agents' decisions, communication among agents can be reduced to some extent.

Martial has investigated how planning agents can positively cooperate in distributed environments

[Martial 90]. Many previous papers on distributed coordinated planning mainly focus on how to resolve conflicts [Corkill 79] [Georgeff 83], however Martial precisely studies situations where a positive effect can be reached as modeled by his favor relation. We also focus on the same aspect of cooperation in terms of collaborative plan construction. In Martial's method, agents broadcast their plans at any time and different levels of abstraction, so that they may refine their plans in a coordinated way. His method is based on the assumption that there is a collection of autonomous intelligent agents which communicate about planned actions ahead of time. In our scheme, the investigation of possible positive cooperation - collaboration - is taken into account, when need for help actually arises. It is basically designed to provide opportunistic collaboration to distributed planning.

6 Concluding Remarks

In this paper, we have presented an organizational scheme of collaboration in multiagent planning systems, and discussed strategies for collaborative plan construction by group members.

Large, multiagent systems can be viewed as open distributed environments. Thus, agents have inconsistent and partial world views. In multiagent cooperative plan construction, several agents mutually generate collaborative plans by inference based on their own beliefs, and partial knowledge about the world. Therefore, mutual planning is confounded by disparities in agents' world knowledge.

In the proposed scheme, an agent who needs help, dynamically organizes a group. He first announces a request for proposals by sending a message to bulletin boards. Agents who read the RFP and who can construct a, possibly incomplete, individual plan for the request, send their individual plans to the manager. Each operator in the plan is associated with a cost estimated by the agent. The manager, then, investigates possible collaboration of potential contractors. If collaboration seems possible, the manager gives collaborative awards along with computed *suggestions for collaboration*. A suggestion for collaboration given to a contractor agent contains: (1) Explicit obstacles of the other collaborating agents which the agent may possibly resolve; (2) Actions which both collaborating agents can perform. Since this sets up a partial model for predicting the other agent's actions, communication among collaborating agents can be reduced. Using this suggestion, along with his initial individual plan and beliefs, each collaborating agent constructs a collaborative plan through inference. In collaborative plan construction, each collaborating agent decides the actions he should perform, the actions the other agent should perform, and the actions both agents do jointly. In the process, each agent takes three factors into account: the elimination of obstacles of other agent; balancing workload among agents; and cost effectiveness.

We are currently working on the following extensions: (1) Extending the algorithm for selecting collaborative awards and computing suggestions for collaboration to include collaboration among more than two agents; (2) Implementing the proposed scheme on a society of SocioAgents; (3) Combining the proposed scheme with an approach which avoids interference among the agents for optimum mutual planning; (4) Incorporating a learning capability into agents, so that successful collaboration can be reutilized again without the overhead of organizing a group.

Acknowledgments

We would like to give great thanks to the other members of Sony CSL for their helpful comments and supports.

References

[Allen and Perrault 80] James F. Allen and C. Raymond Perrault. Analyzing Intention in Utterances. *Artificial Intelligence*, Vol. 15, No. 3, pp.143–178, 1980.

[Corkill 79] Daniel D. Corkill. Hierarchical Planning in a Distributed Environment. In *Proceedings of The Sixth International Joint Conference on Artificial Intelligence.* IJCAI, 1979.

[Davis and Smith 83] Randall Davis and Reid G. Smith. Negotiation as a Metaphor for Distributed Problem Solving. *Artificial Intelligence*, Vol. 20, pp.63–109, 1983.

[Doyle 90] John Doyle. Rationality and its Roles in Reasoning. In *Proceedings of The Eighth National Conference on Artificial Intelligence in 1990.* AAAI, 1990.

[Fikes and Nilsson 71] R. E. Fikes and Nils J. Nilsson. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence*, Vol. 2, No. 3, pp.189–205, 1971.

[Gasser and Huhns 89] Les Gasser and Michael N. Huhns, editors. *Distributed Artificial Intelligence, volume II.* Morgan Kaufmann Publishers, Inc., 1989.

[Georgeff 83] Michael P. Georgeff. Communication and Interaction in Multi-Agent Planning. In *Proceedings of The Third National Conference on Artificial Intelligence in 1983.* AAAI, 1983.

[Halpern and Moses 85] J. Y. Halpern and Y. O. Moses. A Guide to the Modal Logics of Knowledge and Belief. In *Proceedings of Nineth International Joint Conference on Artificial Intelligence*, Los Angeles, CA, August 1985. IJCAI, Inc.

[Martial 90] Frank von Martial. Coordination of Plans in Multiagent Worlds by Taking Advantage of the Favor Relation. In *Proceedings of The Tenth International Workshop on Distributed Artificial Intelligence.* AAAI, 1990.

[Nilsson 80] Nils J. Nilsson. *Principles of Artificial Intelligence.* Tioga Publishing Co., 1980.

[Osawa and Tokoro 90] Ei-Ichi Osawa and Mario Tokoro. SocioAgent: A Society of Rational Speech Actors. In *Proceedings of 7th Conference of Japan Society for Software Science and Technology*, October 1990. also appeared in SCSL-TR-90-011 of Sony Computer Science Laboratory Inc. (in Japanese).

[Rosenschein and Genesereth 85] Jeffery S. Rosenschein and Michael R. Genesereth. Deals Among Rational Agents. In *Proceedings of Nineth International Joint Conference on Artificial Intelligence*, pp. 91–99, Los Angeles, CA, August 1985. IJCAI, Inc.

[Rosenschein *et al.* 86] Jeffery S. Rosenschein, Matthew L. Ginsberg, and Michael R. Genesereth. Cooperation Without Communication. In *Proceedings of The Fifth National Conference on Artificial Intelligence in 1986.* AAAI, 1986.

[Tokoro 90] Mario Tokoro. Computational Field Model: Toward a New Computing Model/Methodology for Open Distributed Environment. In *Proceedings of the 2nd IEEE Workshop on Future Trends in Distributed Computing Systems*, September 1990. also appeared as Technical Report SCSL-TR-90-006.

Appendix: Beliefs and skills of agents in the example

The followings are beliefs and action rules of each agent which appear in the example in Section 2. Details about the formalization of beliefs and action rules are given in Section 3. For simplicity, names of agents are ommitted from the beliefs of each agent. A time point obtained from a virtual clock has a concrete value, however in the following it is parametrized, assuming that for any time points t_i and t_j, if $i \leq i$, then $t_i \leq t_j$.

agent: g_1

belief:

$B(block(b), t_0)$. $B(handempty(g_1), t_1)$. $B(room(r_1), t_6)$. $B(room(r_2), t_7)$. $B(in(g_1, r_1), t_8)$.
$B(onfloor(b, r_1), t_6)$. $B(boundary(d_{1x}, r_1, r_x), t_6)$. $B(\neg open(d_{2y}), t_7)$.
$B(boundary(d_{12}, r_1, r_2), t_7)$. $B(boundary(d_{2y}, r_2, R_y), t_7)$. $B(open(d_{12}), t_7)$.
$B(boundary(D, R_1, R_2) \supset boundary(D, R_2, R_1), t_0)$. $B(hodling(G, Obj) \supset \neg handempty(G), t_0)$.

action rule:

pickup$(G, Obj, T, \tau^{g_1}_{pickup})$
precond: $B(handempty(G), T)$, $B(in(G, R), T)$, $B(onfloor(Obj, R), T)$
effect: $B(\neg handempty(G), T + \tau^{g_1}_{pickup})$, $B(\neg onfloor(Obj, R), T + \tau^{g_1}_{pickup})$,
$B(holding(G, Obj), T + \tau^{g_1}_{pickup})$.

putdown$(G, Obj, T, \tau^{g_1}_{put})$
precond: $B(hodling(G, Obj), T)$, $B(in(G, R), T)$
effect: $B(\neg holding(G, Obj), T + \tau^{g_1}_{put})$, $B(handempty(G), T + \tau^{g_1}_{put})$, $B(onfloor(Obj, R), T + \tau^{g_1}_{put})$.

move$(G, Obj, R_x, R_y, T, \tau^{g_1}_{move})$
precond: $B(in(G, R_x), T)$, $B(hodling(G, Obj), T)$, $B(boundary(Door, R_x, R_y), T)$, $B(open(Door), T)$
effect: $B(\neg in(G, R_x), T + \tau^{g_1}_{move})$, $B(in(G, R_y), T + \tau^{g_1}_{move})$.

trans$(G_x, G_y, Obj, T, \tau^{g_1}_{trans})$
precond: $B(hodling(G_x, Obj), T)$, $B(in(G_x, R), T)$, $B(in(G_y, R), T)$, $B(handempty(G_y), T)$
effect: $B(\neg handempty(G_y), T + \tau^{g_1}_{trans})$, $B(\neg holding(G_x, Obj), T + \tau^{g_1}_{trans})$,
$B(holding(G_y, Obj), T + \tau^{g_1}_{trans})$, $B(handempty(G_x), T + \tau^{g_1}_{trans})$.

agent: g_2

belief:

$B(block(b), t_0)$. $B(handempty(g_2), t_1)$. $B(room(r_1), t_2)$. $B(room(r_2), t_3)$. $B(room(r_3), t_4)$.
$B(in(g_2, r_2), t_5)$. $B(boundary(d_{12}, r_1, r_2), t_3)$. $B(boundary(d_{23}, r_2, r_3), t_4)$. $B(open(d_{12}), t_3)$.
$B(\neg open(d_{23}), t_5)$.
$B(boundary(D, R_1, R_2) \supset boundary(D, R_2, R_1), t_0)$. $B(hodling(G, Obj) \supset \neg handempty(G), t_0)$.

action rule:

pickup$(G, Obj, T, \tau^{g_2}_{pickup})$
similar with that of agent g_1.

move$(G, Obj, R_x, R_y, T, \tau^{g_2}_{move})$
similar with that of agent g_1.

trans$(G_x, G_y, Obj, T, \tau^{g_2}_{trans})$
similar with that of agent g_1.

open$(G, Door, T, \tau^{g_2}_{open})$
precond: $B(in(G, R), T)$, $B(boundary(Door, R, X), T)$, $B(\neg open(Door), \Gamma)$, $B(handempty(G), T)$
effect: $B(open(Door), T + \tau^{g_2}_{open})$.

agent: g_3

belief:

$B(block(b), t_0)$. $B(handempty(g_3), t_1)$. $B(room(r_3), t_7)$. $B(in(g_3, r_3), t_7)$. $B(\neg open(d_{y3}), t_7)$.
$B(open(d_{3x}), t_6)$.

action rule:

trans$(G_x, G_y, Obj, T, \tau^{g_3}_{trans})$
similar with that of agent g_1.

COOPERATIVE PROBLEM-SOLVING GUIDED BY INTENTIONS AND PERCEPTION

Birgit Burmeister
++30-39982-202
bur@b21.uucp

Kurt Sundermeyer
++30-39982-236
sun@b21.uucp

Daimler-Benz AG Research Institute Berlin
Alt-Moabit 91b
W-1000 Berlin 21
Germany
Fax: ++ 30-39982-107

Abstract

We describe a domain independent control structure for cooperating problem solving both at the level of concepts and of realization (system architecture and implementation), and in terms of an example. The concepts are rooted in a generic agent model based upon intentions, behavior and resources of agents. Agents are motivated to act by their long term goals, desires, preferences, responsibilities and the like, within their perceived surrounding, that is other agents and the environment they exist in.

1. Introduction

Although the scope of DAI is not yet clearly defined, some main streams can be distinguished.

- The editors of /BG/ divide DAI into the primary areas 'Distributed Problem Solving' and 'Multi-Agent Systems'. Distributed problem solving "... considers how the work of solving a particular problem can be divided among a number of modules ... that cooperate at the level of dividing and sharing knowledge about the problem and about the developing solution". Multiagent systems are "... concerned with coordinating intelligent behavior among a collection of ... autonomous intelligent 'agents' how they coordinate their knowledge, goals, skills and plans jointly to take action or to solve problems".
- Recently has developed a debate between proponents of 'contemplative' and of 'reactive' systems, e.g. /Br/. In a contemplative system knowledge about other agents and the environment is explicitly represented such that the agent is able to reason how to arrive at some goal. A reactive system simply behaves on a stimulus-response basis within its perceived surrounding.

The point of view taken in this article is predominantly within contemplative multi-agent systems. Thus we deal with modeling scenarios with more or less sophisticated systems ("agents") interacting in some environment. Each agent perceives its surrounding. It has intentions and some degree of autonomy and cooperativeness. To realize intentions an agent needs resources, which in general are limited and have to be shared. An agent eventually undertakes steps to realize intentions. The interaction among the agents consists of their mutual perception and of coordinating activities.

Our aim is to cover a wide span of these kinds of scenarios with general control structures and communication strategies for cooperative problem-solving. The main aspect of this article is a control structure. The concepts are introduced in section 2. They are based on a general agent model. This, as well as the problem solving and cooperation strategy of an agent, is based upon its intentions, resources and behavior. The realization of the concepts is described in section 3 from the point of a system architecture, in which the cognitive skills of an agent are realized as a knowledge-based system, and a test-environment for knowledge-based systems. In order to illustrate the cooperative problem solving concepts we give an example from traffic securing and optimizing systems in section 4. We conclude with an outlook in section 5.

2. Concepts

In this section we give a short overview of our agent model, as described in /Su/. The model is the base of an agent architecture, the treatment of the cognitive skills of an agent, a testbed, and the control structure for problem-solving and cooperation. Moreover we introduce a formal description of the agent model and of the cognitive actions of an agent as it is the thorough underpinning of the realization.

2.1 Agent Model

Overview

We think of a DAI scenario as a set of *agents* which exist in an *environment* and which *interact* with each other and with the environment.

An agent perceives its *surrounding*, i.e. the environment and other agents, acts in accordance with its intentions and needs resources for performing perception or actions.

- As for *perception* we distinguish between whether more than one agent is explicitly involved (*receiving* messages from other agents) or not (*sensing* other agents and/or the environment).
- Whereas perception happens unintentionally and on a continuous basis, *actions* are intended and can actively be planned and executed by an agent to any desired moment.
  We differentiate *cognitive actions* from *effectoric actions*. Cognitive actions of an agent can not be directly perceived by other agents. They only become apparent by effectoric actions they may initiate.
  The effectoric actions are further divided into *sending* and *acting*.
  Sometimes it is convenient to talk of *behavior* as comprising both actions and perception. The complete taxonomy of behavior is shown in Fig.1.

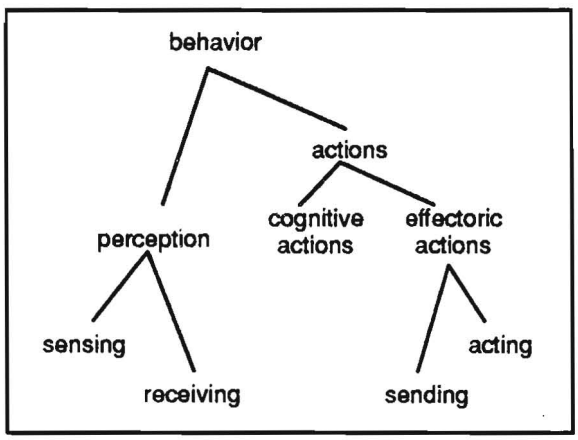


Fig.1: Taxonomy of 'behavior'

- The role of *intentions* has recently been investigated in depth by e.g. /CL/, /W/. In accordance with this work we distinguish long-term intentions, like superior goals, preferences, interests, responsibilities as *strategic intentions* from *tactical intentions* (short- and mid-term intentions, like subgoals, plans, and plan-steps). The difference can be seen in that tactical intentions are directly bound to actions in contrast to strategic intentions.
- The technical term *resources* is used in a very broad sense and covers everything that is needed for executing perception or actions. Thus resources may be divided into sensing resources (physical sensors, the content of buffers, ...), sending and receiving resources (communication hardware, low-level protocols, bandwidth message-queues, ...), acting resources (robot arms, time, space, energy, ...), and cognitive resources (knowledge and belief).

Intentions, behavior and resources are intimately tied together: Every intention is associated with the necessary resources for realizing it, every realized intention is an action, and every type of behavior needs and/or provides its typical resources. This interplay is exploited by other authors too (e.g. /BIP/, /Is/), and more commonly covered under the topic "beliefs, goals, and actions".

The interaction among agents consists of their mutual perception and their coordination of activities: comparison of intentions (to identify goal conflicts and common interests), adjustment of resources (in case of resource conflicts and resource sharing) and synchronization of actions. By our broad usage of the term resources this lastly amounts to the exchange of resources among the agents and among an agent and the environment.

Formal Description

The agent model is formalized in terms of sets, relations and mappings.

As notation for the DAI scenario we introduce Scen = <AG, Env>, where AG is the set of agents and Env the environment. Env and every element Ag_i of AG are pairs themselves. The first element of each pair is the generic description of the object, the other one the description of its actual state. This distinction of descriptions holds throughout for all objects introduced in the sequel.

An agent Ag_i is characterized by the tuple $<PERC_i, ACT_i, INT_i, RES_i>$.
The set $PERC_i$, representing the perceptive behavior of agent Ag_i, is divided as $PERC_i = SENSING_i \cup RECEIVING_i$.
For the actions we denote $ACT_i = CognACT_i \cup EffACT_i$ and $EffACT_i = ACTING_i \cup SENDING_i$.
For each agent Ag_i the respective sets of intentions are INT_i, $StratINT_i$, and $TactINT_i$ with $INT_i = StratINT_i \cup TactINT_i$. The one-to-one correspondence between tactical intentions and actions (realized intentions are actions and every action can be intended) gives rise to an isomorphism between $TactINT_i$ and ACT_i.
The set of resources RES_i is the union of the sets of resources needed for executing the different behavior types: $SensingRES_i$, $ActingRES_i$, $SendingRES_i$, $ReceivingRES_i$, and $CognRES_i$. There is a relation Execute(r,a) with $r \in RES_i$ and $a \in ACT_i$, relating actions to the resources for executing them. By the isomorphism of ACT_i and $TactINT_i$ a similar relation Realize(r,i) holds for $r \in RES_i$ and $i \in TactINT_i$, relating the tactical intentions with the resources for realizing them.

The environment is completely characterized by its resources which it consumes or provides.

All elements of $PERC_i$ and ACT_i are seen as mappings among resources (always referring to the state description of the resources involved).

The elements $sensing_{ia}$ (the further index 'a' numbering the elements in the respective set) of $SENSING_i$ are mappings from the environment and from the acting resources of a subset AG_S of AG (these are those agents within the sensor range of the respective agent) to the sensing resources of AG_i

$$sensing_{ia}: \quad Env \times ActingRES_S \rightarrow SensingRES_i$$

and $receiving_{ika} \in RECEIVING_i$ are mappings from the sending resources of an agent AG_k to the receiving resources of agent Ag_i:

$$receiving_{ika}: \quad SendingRES_k \rightarrow ReceivingRES_i \ .$$

Similarly

$$sending_{ika}: \quad SendingRES_i \rightarrow ReceivingRES_k \ .$$

The elements of $ACTING_i$ are mappings which involve the acting resources of Ag_i, and possibly of the environment, and of other agents from subsets AG_A and $AG_{A'}$ of AG.

$acting_{ia}$: $ActingRES_i \times ActingRES_A \times Env \rightarrow ActingRES_i \times ActingRES_{A'} \times Env$

The cognitive actions of Ag_i only involve resources of this agent. These actions are described in the following section.

2.2 Cognitive Actions

Cooperative problem-solving from our point of view is identical to cognitive actions. Informally it subsumes the tasks of

- analyzing observations (passive recognition of other agents and the environment by sensory means, receiving messages from other agents)
- moulding and reformulating tactical intentions
- checking resources
- considering actions
- preparing messages (determining receivers and formulating questions, demands and informations).

A common treatment of all these different types of cognitive actions would have the advantage of a standard frame of terms. However these tasks seem so different from each other, that a common treatment is unfeasible. The basic distinction between unaware perception and intended actions discussed in section 2.1 and the observation that the different tasks above can be grouped as those analyzing perception, those preparing actions, and those providing the necessary resources, suggests the partition of the cognitive actions as

$CognACT_i = ANALYZE_i \cup PREPARE_i \cup PROVIDE_i$

(In the sequel we drop the index 'i' for the agent, since cognitive actions refer to only one agent, per se.)

Cognitive Actions and Perception

Those cognitive actions which are concerned with the analysis of perceived data, i.e. the elements of ANALYZE, are not treated explicitly by intentions of an agent. The functions 'analyze' (here we drop also the index 'a' numbering the functions) take the actual perceptive resources and turn these into a model of the perceived surrounding (Perc):

analyze: $SensingRES \times ReceivingRES \rightarrow Perc \subset CognRES$.

Cognitive Actions and Effectoric Actions

Those cognitive actions which deal with preparing effectoric actions (PREPARE) and with providing the resources for acting and sending (PROVIDE) are treated explicitly by the tuple <ACT, INT, RES>, since each agent aims to act in a perceived world according to its intentions and on behalf of available resources.

The idea behind our treatment is the following: Each agent has a repertoire of generic actions which it is aware of. Generic actions are partially ordered in the sense that some have recourse to others in form of execution procedures. At the top of this ordering are 'strategic' actions StratACT which only make sense in specific world situations. At the bottom of the ordering are 'primitive' actions PrimACT.

The strategic intentions and the knowledge (or rather belief) of the state of the world (world model WM with PERC $\subset$ WM $\subset$ CognRES) determine the chosen tactical intention of an agent.

adopt: WM x StratINT $\rightarrow$ adoptINT $\in$ TactINT .

Committing to an adopted intention means to follow the partial ordering of the tactical intentions (the ordering being inherited from the ordering of the elements in EffACT) and adopting recursively intentions lower in the ordering

follow: TactINT $\rightarrow$ TactINT

where the domain of the first mapping in this sequence is adoptINT, and the sequence terminates with those tactical intentions which correspond to the actions in PrimEffACT.

In order to successfully follow a tactical intention, that is to have a chance to realize it, the agent needs resources. So following an intention i means to obtain the necessary resources in Realize(r,i). If the resources for an intention are available it is called executable, ExecINT $\subset$ TactINT.

If the agent has the necessary resources at its disposal, the tactical intention can be realized immediately. If resources are not immediately at the agents disposal they possibly may be obtained from the environment or from other agents. This amounts to intending further generic acting or sending processes with their necessary resources, etc. If resources are available, but also needed by others, agents have to negotiate. Negotiation steps are also treated like generic actions. If the resources are not available at all, the agent has to drop its originally chosen intention, to adopt another one, follow it, and so forth.

The mappings in PREPARE are compositions of *adopt* and *follow* mappings,

prepare = adopt $\circ$ follow* $\circ$ [prepare] ,

where various *follow* mappings may be applied according to the recursive process mentioned above and where this possibly has to be interrupted to start another *prepare*. Thus ultimately *prepare* are mappings

prepare: WM x StratINT $\rightarrow$ primexecINT $\in$ TactINT .

Finally each *provide* mapping projects from Realize(r,execINT) those resources r which are needed for the realization of the intention.

3. Realization

In this section we describe the realization of the concepts presented in the previous chapter. The system architecture with its modular structure and the realization of the cognitive skills of an agent as a knowledge-based system are described in sections 3.1 and 3.2, respectively. We further briefly present our test-environment for cooperating knowledge-based systems in section 3.3.

3.1 Agents and their Modules

The features of agents which have been discussed in section 2.1 can be transformed almost uniquely into a modular system architecture with modules COGNITION, responsible for "cognitive actions", SENSORS, responsible for "sensing", ACTUATORS, responsible for "acting", COMMUNICATION, responsible for the connected pair of "receiving" and "sending", and INTENTION representing only strategic intentions, since, as discussed in section 2.2, moulding and revising tactical intentions is counted as cognitive actions.

The model of the DAI scenario is completed by a module ENVIRONMENT. The full architecture is shown in Fig.2.

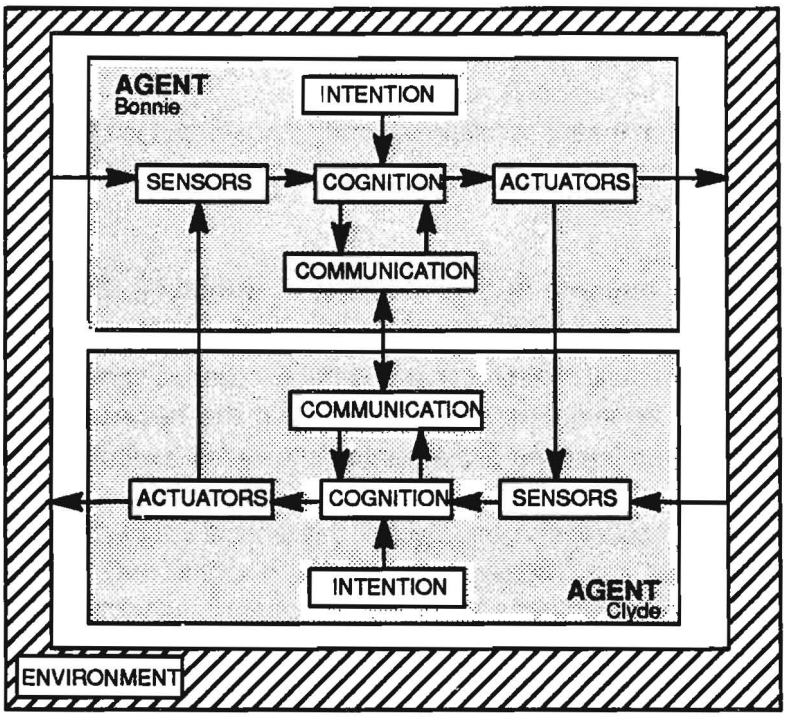


Fig.2: AGENTs and their Modules

The further system details of the modules largely depend on the specific application. Since we are interested in modelling the cognitive skills of an agent we can specify COGNITION if we decide on some paradigm.

3.2 COGNITION as a Knowledge-Based System

We decided to realize each module COGNITION as a knowledge-based system. Besides a knowledge-base and a problem-solving component (as they are standard for isolated knowledge-based systems), it contains a cooperation component.

- The knowledge base KB represents the cognitive resources of the agent.
- The problem-solving component PC performs those cognitive actions which an agent can perform without coordination with other agents.
- The cooperation component CC is responsible for negotiation processes, suitable selection of message types, resource allocation, and the like.

COGNITION has interfaces to the other modules as shown in Fig.3.

Knowledge Base

The knowledge which is explicitly represented in KB is both the generic description of CognRES (generic knowledge) and the actual state of $CognRES_i$ (actual knowledge).

As for structuring the knowledge we found the most natural and efficient way in a tree-like decomposition with a number of composite objects. KB is composed of parts 'self', 'other_agents' and 'environment'. Each of these parts is divided into generic and actual knowledge, and these in turn into knowledge about behavior, intentions and resources (except for 'environment', which is solely described by resources).

- The part 'KB-self-generic-behavior' contains knowledge about the agent's repertoire of generic behavior, except for cognition. (Although this would be an avenue to meta-levels of problem-solving and cooperation we leave this out at present to avoid things like "intending to adopt an intention", or "obtain resources for adopting".)

In our object-oriented implementation the data structure 'generic_behavior' is a super-class of other classes. It defines the slots with the name, an initial condition, an execution procedure, and parameters of a generic behavior.
Only the class 'generic_effectoric_action' contains as an additional slot the tactical intention to which its instances are related (by the isomorphism). Their initial conditions are matched against entries within the knowledge base.
In contrast (but in accordance with the considerations in section 2.2) 'generic_perception' is initialized by either message queues (for 'generic_receiving') or sensor queues (for 'generic_sensing').

- 'KB-self-generic-intentions' contains knowledge both about strategic and tactical intentions. The part with knowledge about strategic intentions relates these to favorable strategic generic behaviors. Knowledge about tactical intentions incorporates knowledge of resources needed for realizing these intentions, that is Realize(r,i).
- 'KB-self-generic-resources' describes each resource by its source (self, others, environment), to which module it belongs and by which process it can be gained. If the source is not 'self' this process is formulated as a negotiation protocol.
- The part 'KB-self-actual' contains the knowledge about the intentions the agent is currently trying to achieve, the behavior it is executing and the resources it has at its own disposal.
- 'KB-others-generic' contains generic knowledge about the typical behavior, intentions and resources of the other agents in the scenario, whereas 'KB-others-actual' is the knowledge about those agents that are actually interacting with the agent. This knowledge is often called knowledge about 'acquaintances',e.g. in MACE /GBH/ or in the actor model.
- Knowledge about the environment is knowledge about the resources provided by the environment, typically ('KB-environment-generic') and actually ('KB-environment-actual').

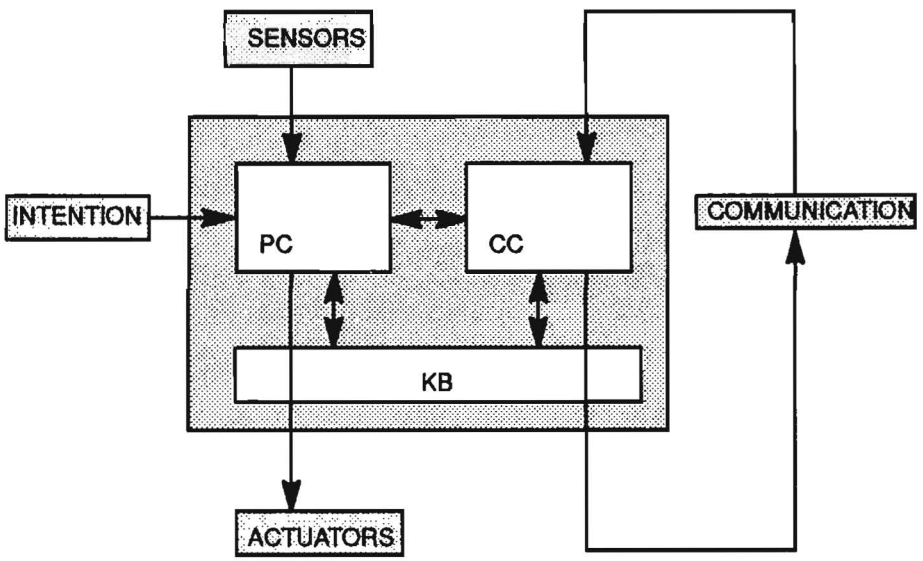


Fig.3: COGNITION and its Interfaces

Problem-Solving Component

The task of PC is the analysis of sensing and the preparation of acting together with the provision of its necessary resources. To simplify the wording we simply write 'actions' for the elements of ACTING, since as discussed, cognitive actions are not treated in this form, and since the preparation of sending is delegated to the cooperation component.
The data flow within the problem-solving component is described as follows (comp. Fig.4).

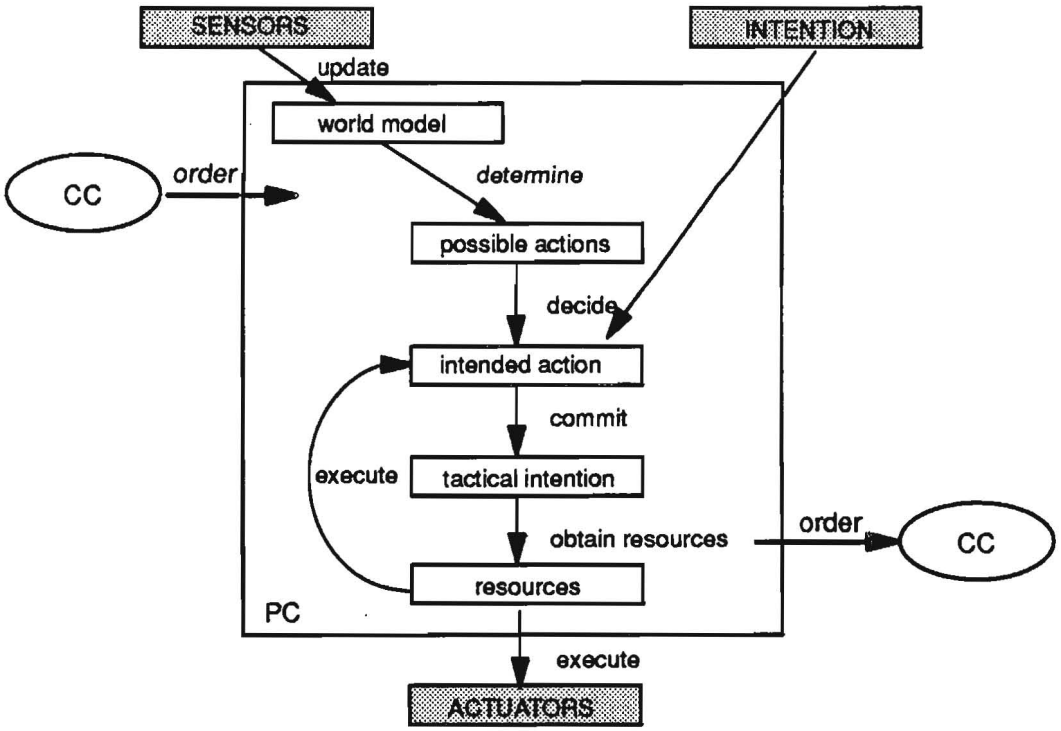


Fig.4: Data Flow within the Problem-Solving Component

1. The agent acts according to a default behavior, which directly derives from its strategic intentions. This behavior is performed as long as nothing else happens, if the performance of a generic action is interrupted or as long no new one has been chosen.
2. By analyzing new sensor data the models for the environment and for the other agents within KB are updated.
3. Possible actions are determined from the set of generic actions by comparing the world model with the initial conditions of the generic actions.
4. If several possible actions exist, a decision for one of them is made by the strategic intentions.
5. The agent commits itself to perform the chosen action.
6. To realize the tactical intention i its necessary resources r from Realize(r,i) are checked:
   - If the necessary resources are immediately present, i.e. if they are at the agents disposal, the adjoined generic action can be executed according to the execution procedure; see step 7.
   - If the necessary resources are not immediately available, the agent aims to get the resources from the environment or from other agents.
     The task to obtain resources from other agents is delegated to the cooperation component CC; see next paragraph.
   - If the necessary resources cannot be made available, the tactical intention is not realizable and a commitment to another generic action is to be made.
7. A generic action is executed by following an execution procedure. In general this leads again to a commitment to a tactical intention of an action lower in the partial order. The execution of primitive_acting directly happens by calling the interface function to ACTUATORS.

Step 2 realizes part of the *analyze* mapping, steps 3 and 4 the *adopt* mapping, and steps 5 to step 7 the *follow* and *provide* mappings, as introduced in sect. 2.2.

Aside from following the seven-step procedure above, PC can be initiated on the level of determining possible actions by an order from CC due to a request by another agent.

Cooperation Component

The two tasks of the cooperation component CC are to obtain resources from and to provide resources to other agents. To fulfill these tasks CC negotiates (and for this communicates) with other agents, i.e. prepares sending actions for COMMUNICATION and analyses receiving messages from COMMUNICATION. According to our attitude of treating effectoric actions different from perception, 'sending' is described in terms of tactical intentions in contrast to 'receiving'.
Negotiation and communication follows protocols, like the contract net protocol /Sm/ or knowledge interchange protocols /CI/. As mentioned before, these protocols are represented as generic_behavior.

CC is initiated by either an order from PC to obtain or provide resources or by an incoming message from another agent for providing resources.

a) If CC is initiated by PC to obtain a certain resource, CC takes the following steps (comp. Fig.5a):
   1. CC looks up the generic description of the resource in KB, where it finds a generic_sending which must be executed to obtain the resource.
   2. This generic_sending is treated just like generic-acting within the problem-solving component, namely by committing to a tactical intention, by checking and obtaining necessary resources and by following an execution procedure, which eventually leads to a primitive_generic_sending, i.e. a function which directly can be executed by COMMUNICATION.
      The resources (address of receiver etc.) are in our first approach assumed to be at the agent's disposal, to prevent a recursive call of CC to itself.

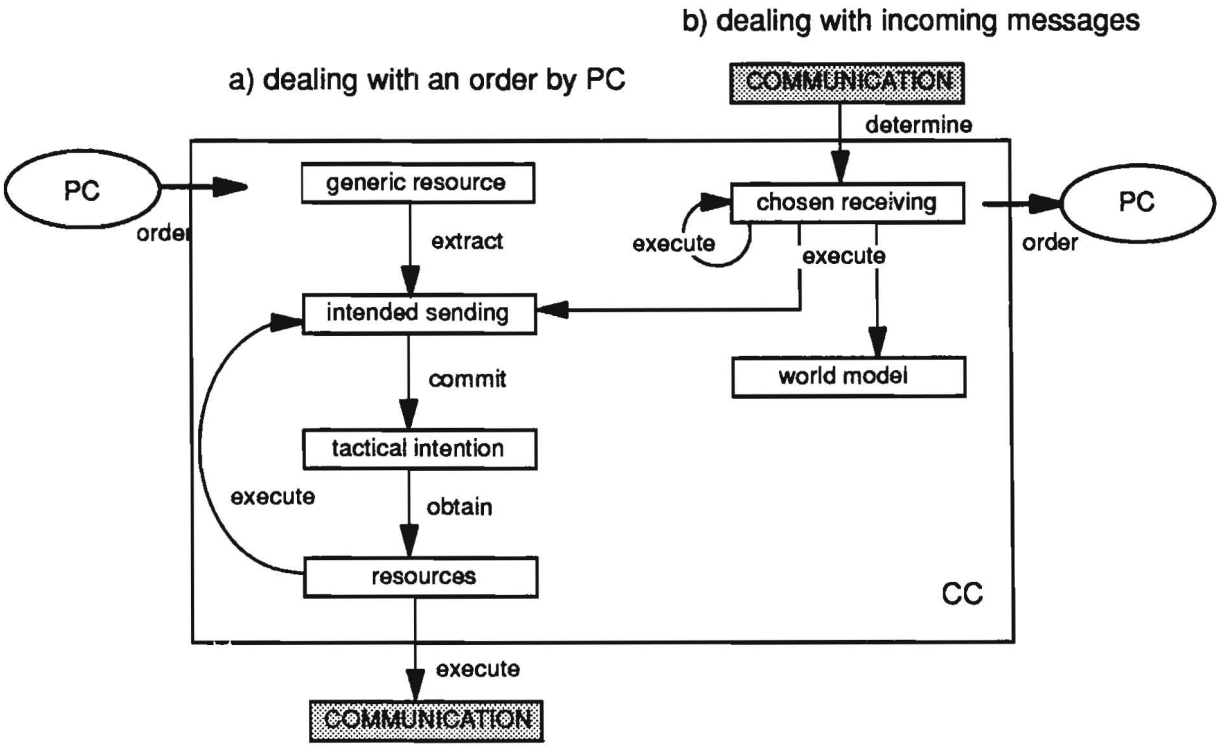


Fig.5: Data Flow within the Cooperation Component

b) Otherwise CC 'observes' the message queue, in order to handle incoming messages (comp. Fig.5b):
   1. There is generic_receiving that deals with incoming messages. Protocols specify special receiving behavior for every message type they include. CC follows the respective execution procedure.
   2. The execution of primitive_receiving can have different effects: The content of the received message can complete or change the world model as part of KB, the received message can initiate (or follow) a negotiation as described in a) or it can determine an acting behavior to be prepared (and executed) by PC.

As basic message types we presently use INFORM (where a reaction of the receiving agent is not expected), QUERY (where the receiver is expected to send an answer), and DEMAND (where an acting behavior of the receiver is expected). In accordance with the work of other authors, e.g. /NT/, we will build dialogue structures from these three types.

3.3 Test-Environment

The knowledge-based systems, representing the cognitive component of the agents involved, are embedded in our "Development And Simulation Environment for Distributed Intelligent Systems" (DASEDIS). DASEDIS supports a realistic simulation and provides instruments for implementing, inspecting, and observing interacting knowledge-based systems.

The architecture of this experimental tool is derived from the modular architecture of an agent. Everything besides the knowledge-based systems, namely the modules SENSORS, ACTUATORS, COMMUNICATION, INTENTION, ENVIRONMENT, is modeled within the simulation component of DASEDIS. These models are exchangeable both within an application as well as for a domain. The simulation can be visualized in the DASEDIS user interface. The visualization is largely determined by the application. The other (application independent) tasks of the interface are the integration and connection of the simulation component and the development component. The development component serves as basis for the implementation of the various components of the knowledge-based systems, for the inspection of the knowledge bases, and for the observation of the problem-solving and the cooperation component.
DASEDIS provides methods for implementing agents, control and communication strategies by basic data structures.

4. Example

Our first application is drawn from the area of traffic securing and optimizing systems. This can be road bound or unbound two-dimensional traffic (cars or ships), or three-dimensional traffic (air planes or space vehicles). We presently concentrate on road-bound traffic in our project COroad. A comparable application is described in /L/.

In this section we introduce a scenario and describe the solution of a simple overtaking maneuver using the problem solving and cooperation mechanisms described before.

4.1 The COroad-Scenario

We started with a very simple scenario, in which two to ten agents (with possibly different performances and intentions) move on a two or three-lane highway. They change lanes and overtake, or they enter or leave convoys. Each agent has partial knowledge about parameters (for example velocities, vehicle type), relative position (distance, lane) and intentions of other agents. Each agent decides with respect to the perceived world and with respect to its strategic intentions of whether it should drive with a certain speed, adopt the speed of another agent, overtake, or interrupt an overtaking maneuver. These correspond to strategic generic behavior "ride", "follow", "overtake", "interrupt-overtaking". The default generic behavior is "ride", which is performed if no other agent is involved.

The vehicle model in ACTUATORS describes the rough geometry of the vehicle, its velocity, its acceleration/braking behavior. Different vehicle types are characterized by their maximal velocity and their power/weight ratio. The simulation functions are "drive", "brake", "accelerate", "change lane". Thus these are the primitive acting types. The driver model in INTENTION describes the attitudes and long term goals of a driver, such as driving cautiously, economically, fast. SENSORS "measure" the relative position of other vehicles. COMMUNICATION simulates the communication between vehicles. ENVIRONMENT describes the road in its topology, its qualities, as well as restrictions due to speed regulations and the like.

The knowledge base KB contains generic knowledge like the "driving school knowledge", models of other typical agents (ordinary cars, trucks etc.) and of the typical environment (two-lane, three-lane highway) as well as actual knowledge like parameters describing the environment, the current own data (velocity etc.), the actual problem solving state with respect to intentions, behavior,, and resources, as well as the state of other agents (this however being incomplete).

The DASEDIS user interface allows the user to input vehicle data and driver's strategic intentions. As output it shows the scenario in a graphical form (vehicle symbols moving on a road drawn on the visualization window) and parameters of selected vehicles.

4.2 An Overtaking Maneuver

Bonnie is a time-saving driver. She likes to drive as fast as possible. If no other agent is within her 'zone of relevance' on the highway she adopts the default behavior "ride" with the maximum power.

```
(def-generic-acting ride
    :init-cond      :default
    :exec           '( (drive :speed (resource vehicle-max-speed) ) )
    :tact-intention 'to-ride )
```

(In italic letters we state in a CommonLISP and CLOS like notation those pieces of generic knowledge which Bonnie uses).

At a certain moment her SENSORS register another agent (Clyde) on the same lane before her and with a decreasing distance. As soon as the distance becomes smaller than a trigger distance Bonnie has to decide what to do. Her problem-solving component determines two kinds of (strategic) generic acting behaviors to be applicable in this situation, namely 'overtake' and 'follow'.

```
(def-generic-acting overtake
   :init-cond       '(and  (same-lane-before-p $vehic-before-me $self)
                           (distance-decreasing-p $vehic-before-me $self)
                           (< (distance $vehic-before-me $self)
                              (trigger-distance)))
   :exec            '(  (change-lane :direction :left)
                        (pass-by)
                        (change-lane :direction :right))
   :tact-intention  'to-overtake )

(def-generic-acting follow
   :init-cond       '(and  (same-lane-before-p $vehic-before-me $self)
                           (distance-decreasing-p $vehic-before-me $self)
                           (< (distance $vehic-before-me $self)
                              (trigger-distance)))
   :exec            '( (drive (resources desired-speed-follow)) )
   :tact-intention  'to-follow )
```

Matching these with her strategic intention to drive time-saving Bonnie decides to overtake:

```
(def-generic-strategic-intention drive-time-saving
   :favorable-behavior   '(overtake) )

(def-generic-strategic-intention drive-economic
   :favorable-behavior   '(follow) )
```

So, Bonnie commits to the tactical intention 'to-overtake'.

```
(def-generic-tactical-intention to-overtake
   :resources       '(ok-to-overtake) )

(def-generic-resource ok-to-overtake
   :provided-where    :others
   :provided-by       $vehic-before-me
   :provided-how      '( (send-demand :content 'stay-on-lane
                                      :receiver $vehic-before-me) ) )
```

The only resource needed for overtaking is an 'ok' by Clyde. This resource is provided by :others and is provided through sending a demand to Clyde that he should stay on his lane. Assuming for simplicity that Clyde answers "ok" to this demand, Bonnie's cooperation component inserts this answer into the knowledge base. As soon as the resource 'ok-to-overtake' is available Bonnie can execute the behavior 'overtake'.

The first action to take (as stated in the :exec-attribute of overtake) is to change to the left lane. A tactical intention 'to-change-lane' is formulated:

```
(def-primitive-generic-acting change-lane
   :exec            'simu:change-lane
   :tact-intention  'to-change-lane
   :params          '(:direction) )

(def-generic-tactical-intention to-change-lane
   :resources       '( new-lane-no range (road new-lane-no range) ) )

(def-generic-resource road
   :provided-where    :environment
   :provided-by       :environment)
```

Here Bonnie needs as resource a certain range on the new lane to be free. Assuming that in checking this resource she finds in her own actual knowledge (in the part :environment), that the road on the new lane is free (the knowledge about the environment coming from her SENSORS). So the execution procedure of 'change-lane' can be followed. Since it is a simulation function, it is directly executed in ACTUATORS.

The next step in the execution procedure of 'overtake' is 'pass-by'.

```
(def-generic-acting pass-by
   :exec            '( (drive :speed (resource desired-speed-pass-by) ) )
   :tact-intention  'to-pass-by
   :params          '(:direction) )

(def-generic-tactical-intention to-pass-by
   :resources       '(desired-speed-pass-by) )
```

```
(def-generic-resource desired-speed-pass-by
   :provided-where   :self
   :provided-by      :cognition)
```

So in order to pass-by, Bonnie needs to know the speed by which she wants to pass by. Assuming a flying overtaking (and a flat highway) this is her current speed, knowledge of which is in her own KB. She can execute 'pass-by', which is finally to drive with a given speed.

```
(def-primitive-generic-acting drive
   :exec             'simu:drive
   :tact-intention   'to-drive
   :params           '(:speed) )

(def-generic-tactical-intention to-drive
   :resources        '(  desired-speed-drive
                         (> vehicle-max-speed desired-speed-drive) ) )

(def-generic-resource desired-speed-drive
   :provided-where   :self
   :provided-by      :cognition)

(def-generic-resource vehicle-max-speed
   :provided-where   :self
   :provided-by      :actuators)
```

The only resource of drive is that this speed is not allowed to be than the maximal speed the vehicle can drive. Drive, being a primitive acting behavior, is executed by ACTUATORS.

The last step of overtaking is to change back to the right lane. Again the resource 'road' is checked, as long as the road on the right lane is not free, Bonnie keeps on driving on her current lane, when she has passed by Clyde the resource 'road' is available again and she can change to the right lane. Bonnie has overtaken Clyde. Afterwards she returns to the default behavior "ride" with her favorable speed.

What is demonstrated in this example is the interplay between the selection of a (strategic) generic behavior, the forming of the corresponding tactical intention, the checking and obtaining of resources and finally the execution of the behavior in a recursive manner until a primitive behavior, i.e. a simulation function is reached.

5. Conclusion and Outlook

The work described in this article is part of our COSY project /BS/. The aim within COSY (COoperating SYstems) is to arrive at a systematics for the design of cooperating systems. We pick up the loose ends from existing theoretical and empirical research results in DAI and investigate concepts in carefully directed experiments. The concepts are implemented and evaluated in order to find out control structures and communication strategies most appropriate for large classes of applications.

For the purpose of testing and comparing existing results and for refining and extending them a very general agent model and a very broad concept of cooperative problem-solving is needed.

As argued elsewhere /Su/, our agent model being based on intentions, resources and behavior, covers other approaches like state-, actor-, role-, and organization-oriented ones.

Also the data and control flow presented in this article is meant to set a general frame for specific types of cooperative problem solving.

- The control structure in the problem-solving component allows to mix methods of classical planning (where resources are to be seen as STRIPS-like preconditions) with script-based planning (where generic actions are tied to stereotyped situations), which allows a quick solution for complex problem-solving.
- By techniques to be described in a forthcoming publication, we are able to give reactive abilities to the cooperative planning process.

- The control flow within the cooperation component can subsume different cooperation strategies. We want to demonstrate and utilize this by forming various cooperation strategies as "prefabricates" of the development component in DASEDIS.

The advantage of our approach should be seen in that the original paradigm of knowledge-based systems, namely the separation of domain knowledge from a general problem solving procedure, is extended to interacting knowledge-based systems including cooperation.
Lastly, by the basic assumption that agents are motivated to act by some sort of intention influenced by their perceived surrounding gives the conceptual framework a good chance to investigate the interplay of goals, belief and actions and its relation to ability, organizations and roles.

References

/BG/ A.H.Bond, L.Gasser (eds.): "Readings in Distributed Artificial Intelligence", Morgan Kaufmann, 1988

/BIP/ M.E.Bratman, D.J.Israel, M.E.Pollack: "Plans and resource bounded practical reasoning", Comput. Intell. 4 (1988) 349-355

/Br/ R.A.Brooks: "Intelligence without representation", Artif. Intell. 47 (1991) 139-159

/BS/ B.Burmeister, K.Sundermeyer: "COSY: A Project for the Methodology of Multi-Agent Systems", Draft Proc. CKBS, Univ. Keele, Oct.90

/CI/ J.A.Campbell, M.P.D'Iverno: "Knowledge Interchange Protocols", in: Y.Demazeau, J.P.Müller (ed.), Decentralized A.I. (Proc. MAAMAW '89), Elsevier/North-Holland, 1990, pp. 63-80

/CL/ P.R.Cohen, H.J.Levesque: "Intention is Choice with Commitment" Artif. Intell. 42 (1990) 213-261

/GBH/ L.Gasser, C.Braganza, N.Herman: "MACE: A Flexible Testbed for Distributed AI Research", in: M.N.Huhns (ed.), "Distributed Artificial Intelligence", Pitman & Morgan Kaufmann, 1987

/Is/ T.Ishida: "CoCo: A Multi–Agent System for Concurrent and Cooperative Operation Tasks", in: M.Benda (ed.), Proc. 9th Workshop on Distributed Artificial Intelligence, 1989, pp.197-213

/L/ P.Levi: "Verteilte Aktionsplanung für Autonome Mobile Agenten", in: K.v.Luck (ed.), "Künstliche Intelligenz", Informatik Fachberichte 203, Springer, 1989

/NT/ Ch.Numaoka, M.Tokoro: "Conversation among Situated Agents", in: M.N.Huhns (ed.), Proc 10th Workshop on Distributed Artificial Intelligence, chapter13

/Sm/ R.Smith: "A Framework for Distributed Problem Solving", UMI Research Press, 1979,1981

/Su/ K.Sundermeyer: "Modellierung von Szenarien Kooperierender Akteure", in: H.Marburger (ed.), German Workshop on Artificial Intelligence: GWAI-90, Springer, Berlin, 1990, pp.11-18

/W/ E.Werner: "Social Intentions", Proc. ECAI-88, pp.719-723

A Multi-Agent Analogical Representation for Physical Objects*

Luca Maria Gambardella and Marc Haex

IDSIA, Istituto Dalle Molle di Studi sull'Intelligenza Artificiale
Corso Elvezia 36 - CH - 6900 Lugano
Phone: +41 91 22 88 81 Fax: +41 91 22 89 94
Email: luca@idsia.uu.ch marc@idsia.uu.ch

Abstract

The topic of this paper is a representation model for solid objects used for physical simulation purposes and for planning in a robot assembly system. The system combines analogical representation and multi-agent modelling, using a bottom-up representation for objects based on analogical agents. We call these agents analogical because they are mapped into, interact with and reason directly on the workspace representation, which is a discrete grid. The agents contain local geometrical and physical constraints and they cooperate to satisfy them while moving in the direction of an external force and interacting with other objects in the workspace. An emergent functionality of the simulation of a block moving in a complex environment is the solution to various stability problems.

1 Introduction

Simulation is a frequently used technique in many fields. In planning systems it is a valuable method to check whether the execution of the planned actions will lead to a successful state without having to try them in the real environment. This paper deals with the simulation of physical objects, i.e. objects moving according to some force and colliding with other objects in the environment. Consider for instance an object falling onto the edge of a table: it will hit the table, rotate around some pivot touch point and fall further down until it hits another object or the ground. In this paper we will present a system modeling this kind of behaviour in 2 dimensions with polygonic solid shapes. We will describe a bottom-up representation for these objects, consisting of autonomous agents, and show how the global behaviour emerges from the interaction of these agents. We will call these agents analogical because they are mapped into, interact with and reason directly on the workspace representation.

This kind of analogical simulation has previously been applied to other physical systems like liquids [DKS91] [GM89] and strings [GGM89]. The use of analogical representation is fostered by the nature of the problem. Simulating complex physical systems using exclusively symbolic information would result in low accuracy when detailed spatial knowledge is needed.

A key example of an analogical simulation program is WHISPER, described in [Fun80]. It is able to detect and simulate instabilities in a blocks world using diagram representations. WHISPER

*This research is supported by Swiss National Project NFP23 "Automatic Assembly based on Artificial Intelligence". The project is a collaborative effort of Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (Lugano), Institut de Microtechnique Ecole Polytechnique Federale de Lausanne and Institute de Mathematiques et Informatique Universite de Neuchâtel.

has similar functionalities to those of our system but uses centralized high-level reasoning on low-level distributed analogical representation to create the envisionment of object configurations. Our system tries to avoid this global centralized control by distributing among the constituent agents the necessary local behaviour to obtain a correct global result. We will obtain the same result, without explicitly describing the movement of an object; the required functionality emerges from internal communications and interaction with the environment. The importance of the use of analogical representations in autonomous agent organizations is explained in [Ste89].

The next section describes the different kinds of agents that our representation consists of. The third one explains how agents cooperate to obtain the desired behaviour. This is followed by a discussion of the main characteristics, limits and possible applications of the analogical agents approach.

2 Agent Architecture

Before explaining the different kinds of agents, it is important to notice that the underlying workspace is a discrete grid; this means a two dimensional array of cells. Each object occupies a number of cells according to its size and shape. Objects are considered to be two-dimensional, solid, non-elastic polygons. The cells making up the contour of the polygon contain two kinds of agents, namely node agents on the nodes of the polygon and contour agents on the edges between the nodes. These two kinds of agents are organized in a two-level hierarchy in which the contour agents are subordinated to the node agents in the sense that they serve as analogical sensors and their position depends on the geometrical information contained in the node agents. Node agents contain the necessary local rules that determine the geometrical shape and the non-elasticity property of the objects. For reasons of efficiency, the cells that fill up the polygon remain empty. A third kind of agent which is not a real part of an object, but which is physically attached to it is the force agent. It represents the

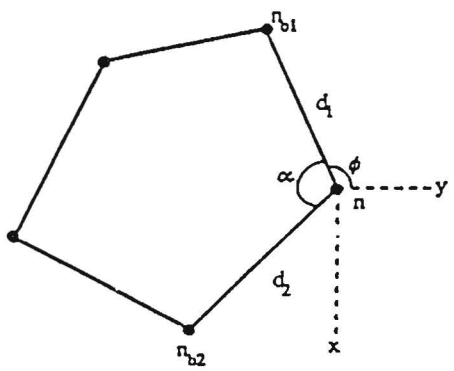


Figure 1: A node agent and its geometrical information.

qualitative force acting on the object and will play an important role in the simulation behaviour of the agents. By a qualitative force, it is meant that it only gives an indication of the direction of the force. Notice that the granularity of the workspace grid influences the accuracy of our model, because it determines the unit of the size that an agent can move. The following paragraphs will describe these agents, giving their knowledge and positional constraints.

2.1 Node Agents

Node agents are positioned on the angular points of the polygonal object. The necessary geometrical information about the object is distributed over these agents. Each node agent n contains the following information slots (see Figure 1):

- A position (x, y) denoting a cell in the workspace grid.
- Links to the two neighbouring node agents: n_{b1} and n_{b2}.
- The distances d_1 and d_2 from n to resp. n_{b1} and n_{b2}.

- The angle α between the two neighbouring node agents.
- The orientation ϕ of the latter angle. This is a variable.
- A link to the two direct contour agents of n, as described in the next paragraph.

d_1, d_2 and α are set at creation time and remain constant all the time. They represent the constraints for a node agent and they are used to describe the geometrical shape of the object. The constant angle α expresses the non-elasticity constraint for the object and d_1, d_2 tell that the object is not extensible. The angle ϕ describes the orientation of the fixed angle in the workspace. At each time step the angle and the distances between n and its two neighbour agents have to be equal to α and to d_1 and d_2. Each node agent checks its constraint by asking the two neighbour node agents for their position and from computing the actual distances d_1', d_2' and angle α'. When these constraints are not satisfied the agent can correct them by changing its position or by asking n_{b1} or n_{b2} to adapt their position in order to obtain the correct value for d_1', d_2' and angle α'. Another constraint is that each cell normally contain a single agent (i.e. a node or contour agent), which implies that the node agent will always check a cell before it tries to occupy it.

2.2 Contour Agents

Contour agents are positioned on the edges between the node agents. Each contour agent c contains the following information slots (see Figure 2):

- A position (x, y) denoting a cell in the workspace grid.
- A link to the two neighbouring contour agents c_{b1} and c_{b2}.
- A link to the two node agents n_1 and n_2 at the ends of the edge on which c is positioned.

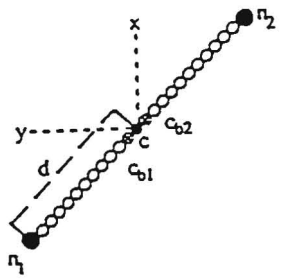


Figure 2: A contour agent and its geometrical information.

- A distance d to one of the two node agents mentioned in the previous slot.

The constant d represents a positional constraint for the agent. Each time the object moves, the contour agents have to recompute their new position using the constant distance d and the links to the node agents. When it tries to change position it will make sure that it does not occupy a cell already occupied by another object. The purpose of contour agents is to act as analogical sensors, i.e. to detect contact between the object and other objects represented as filled cells in the workspace.

2.3 Force Agent

The force acting on an object is represented by a force agent, which is positioned at the cell containing the point of application of a force. For instance, for an object falling under gravity this will be the center of mass. The knowledge of a force agent f is contained in the following slots (see Figure 3):

- A position (x, y) denoting a cell in the workspace grid.

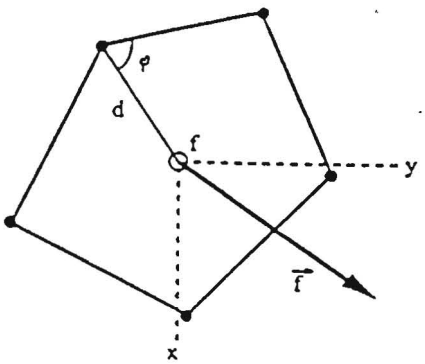


Figure 3: A force agent and its geometrical information.

- A qualitative force vector $\vec{f}$ denoting the direction of the force.
- A constant angle ϕ and distance d denoting the relative position of the force agent to the object.

The force agent's positional constraint is relative to that of the node agent and is contained in the constants ϕ and d. After each timestep the actual values for this angle and distance are computed and if necessary the agent's position is corrected. The force vector can be either static or can be changed by an external controller. Also, in order to consider multiple moving objects, a force propagation protocol between colliding obstacles could be considered.

2.4 The communication protocol

In order to satisfy their constraints and their goal, which we will describe in the next section, communication between the agents will be necessary. Between node agents and contour agents this occurs according to an actor protocol, which means that agents can only communicate directly with those agents that they know about, and that communication is done by message passing. These communication abilities are called links in the previous paragraphs. Notice that between the node agents a doubly linked circular list exists, as for the contour agents. Node agents have links with their direct neighbouring contour agents, which implies that by message passing each node or contour agent can reach every other node or contour agent. Contour agents can use a shortcut link to the node agents of their edge, for performance reasons. The force agent can communicate with every other agent and vice versa.

3 Agent Behaviour

In this chapter we will describe the behaviour of the agents to obtain global movement of an object according to a force applied to it. We will consider one object moving in the workspace containing static obstacles. The point at which the force is applied is considered to be the center of mass of the object. Forces applied at points other than the center of mass result in a torque and rotational movement, even in free space.

3.1 Unconstrained Translation

When a force is applied to an object a force agent is created and attached to it. Now, the goal of the node agents is to try to move in the direction of the force. The node agents will try to occupy the neighbouring cell in the workspace grid according to the required direction, consequently the contour agents will change their position in order to remain on the correct edge position. If every agent successfully changes position, i.e., does not try to occupy a filled cell, the result will be a one cell translation of the object. The positional constraints of the node agents will remain satisfied and no complicated communication will be needed to resatisfy them and the force agent will start a new movement instruction.

3.2 Stability Problem

When an agent tries to occupy a cell that is already occupied by another object, it will not change position. If this happens for one or more agents the agent configuration will not conform to the object initially modeled. At this point the agents will have to negotiate to satisfy their local constraints and to obtain a new configuration conforming to the initial geometry and the movement caused by the force. Because we deal with only one moving block in a workspace, containing only static obstacle blocks, we can distinguish two situations. The first is that the moving object is completely blocked by the others. The second possibility is that the object is only partially blocked and it will start rotating around a pivot touch point, which is the touch point closest to the center of mass. This is the so-called stability problem. We will show how we obtain this global behaviour, without really having the high-level notion of stability or rotation around a pivot point, but by cooperation among the agents in the bottom-up representation. The general idea is that two agents will be selected, one on each side of the force vector (see Figure 4 and 5), from which the reconstruction protocol will be initiated.

These two agents are called *selected* and are found in the following way. First each contour agent which was unsuccessful in trying to occupying its desired cell, looks to see whether both its neighbours are in the same condition. If this is not the case the agent knows that it is the last one in a series of touching agents (and thus candidate for a pivot point).

In Figures 4.a and 5.a these are the contour agents which are marked black.

These candidate pivot points communicate with the force agent and compute their distance from the force vector. For each side of the force vector the agent having the shortest distance is marked as *selected.* In Figures 4 and 5 this leads to the marking of the contour agent which is labeled f_1.

If this results in two *selected* agents a reconstruction protocol is started. In the case of only one

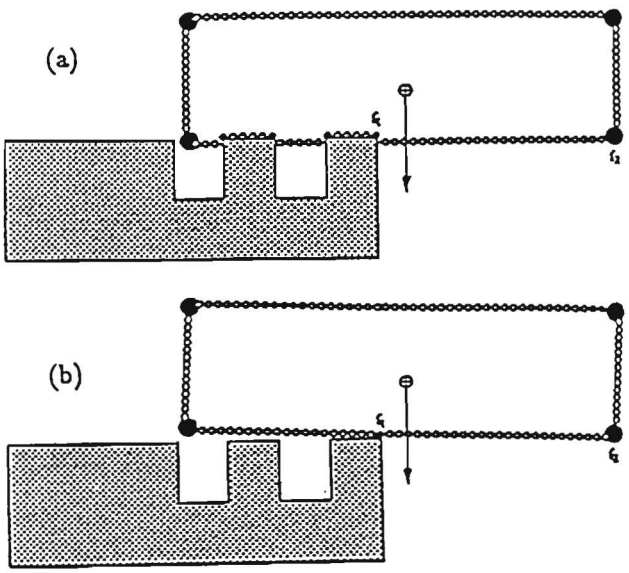


Figure 4: An unstable situation. 4.a Shows an agent configuration after a collision with a static obstacle. Agents f_1 is first chosen to be the *selected* agents, Node agent f_2 is chosen to be the second *selected* agent. 4.b shows the resulting configuration after the reconstruction.

selected agent the nearest node agent at the other side of the force vector is chosen to be the second *selected* agent (Node agent f_2 in Figure 4).

These two agents determine how the positions of the rest of the agents are corrected. This is done in the following way: the *selected* agents communicate to their neighbours to change their position in a way to satisfy the constraints. These in turn do the same with their neighbours. When the loop is closed all constraints are satisfied and a new movement can be started. Notice that the first case of finding two *selected* contour agents, each on one side of the force vector, agrees with a stable situation while the second case means an unstable situation. Examples of resulting configurations after this correction are shown in Figures 4 .b and 5 .b.

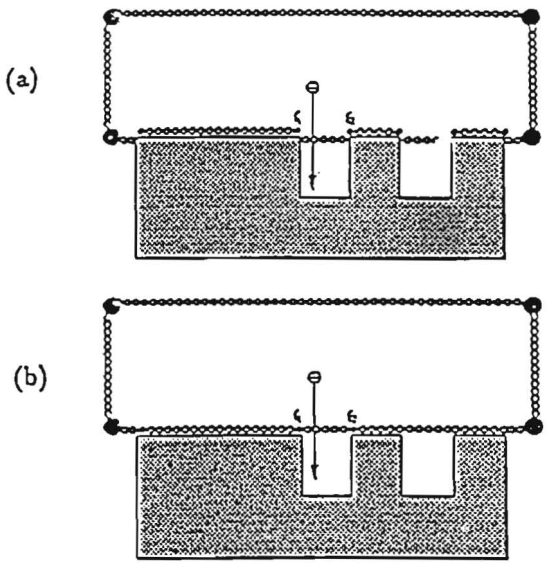


Figure 5: A stable situation. 5.a Shows an agent configuration after a collision with a static obstacle. Agents f_1 and f_2 are chosen to be the *selected* agents. 5.b shows the resulting configuration after the reconstruction.

4 Characteristics, Applications and Limits of the System

The main difference between our multi-agent approach and high-level approaches, like the WHISPER system by Funt [Fun80] , is that we do not explicitly code the global physical behaviour, but it emerges from the interaction between agents. In WHISPER a high level reasoner first checks a blocks configuration for instabilities, choses a pivot point and simulates the rotation explicitly. Our system does not have an idea of instability or rotation, agents always apply the same behaviour of moving in a certain direction and if necessary recover from an abnormal situation. In order to distinguish between a stable or unstable configuration the global behaviour must be interpreted externally by looking at the workspace grid or internally by monitoring the behaviour of individual agents. For stability, it is sufficient to observe the movement of the agents. This explains the need for an interpreter module to extract information relating to the status of a simulation system which uses distributed representation and control. In [Gam91] the use of an analogical string simulation in an automatic assembly system applying a planning, simulating and interpreting loop is described.

A possible application of this physical block simulation in robot assembly could be the simulation of an object following a path, defined by the planner and represented as a series of forces in the workspace. The system could monitor the object, evaluate the success of the result using as a basis for recovery planning or execution of the plan in the real environment.

For this kind of application the current functionality will be sufficient. For other applications it will be necessary to cover more complex functionalities like velocity, acceleration object surface properties and other dynamic features of a physical object. To be able to take these features into account for simulation the current qualitative knowledge of the agents needs to be extended with a more quantitative one.

5 Conclusions

We have described a model of physical objects for simulation purposes which relied on autonomous agents, bottom-up descriptions, constraint satisfaction and the use of local and analogical information.

The result of the simulation of a moving block in a complex environment implicitly solves the stability problem, not by global reasoning, but by cooperation between autonomous agents.

Analogical representations are used to represent both the workspace and the changes that are made in it.

We mixed this strength of analogical representations with the power of autonomous agent systems which lies in the capabilities of agents to cooperate and communicate to satisfy their local constraints.

An implementation of the model serves as a basis for further research. One goal is to expand the

model to cover multiple moving objects and propagation of forces. Another goal is to use the model in robot assembly, in which we simulate a grasped object moving in its workspace.

References

[DKS91] Jo Decuyper, Didier Keymeulen, and Luc Steels. A qualitative model for the behavior of liquids in daily-life circumstances. In *First European Workshop on Qualitative Reasoning About Physical Systems*, Genova, Italy, January 1991.

[Fun80] Brian V. Funt. Problem-solving with diagrammatic representations. *Artificial Intelligence*, 13(3):201–230, 1980.

[Gam91] Luca Maria Gambardella. Simulation and planning with multiple representations. In *AI, Simulation and Planning in High Autonomy Systems*, Cocoa Beach, Florida, April 1991.

[GGM89] Luca Maria Gambardella, Francesco Gardin, and Bernard Meltzer. Analogical representation of naive physics. In *Second Workshop on Qualitative Physics*, Paris, France, 1989.

[GM89] Francesco Gardin and Bernard Meltzer. Analogical representations of naive physics. *Artificial Intelligence*, March 1989.

[Ste89] Luc Steels. Cooperation between distributed agents through self - organisation. In Yves Demazeau and J.P.-Müller, editors, *Decentralized A.I., Proceedings of the First European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, Cambridge, England, August 1989.

Variable Coupling of Agents to their Environment: Combining Situated and Symbolic Automata

Extended Abstract

George Kiss
The Open University
Walton Hall, Milton Keynes
Mk7 6AA
England
Email: gr_kiss@vax.acs.open.ac.uk

The paper identifies *generality* and *power* (processing work per unit time) as two major but conflicting requirements for the design of autonomous intelligent agents. A separation of these two concerns leads to the notion of a *variable degree of causal coupling* between parts of a mechanism and its environment in terms of space and time. Recent controversies surrounding sub-symbolic processing, symbol-grounding, situated agents and reactive architectures can all be interpreted as manifestations of the pressure towards power by *close coupling* to the environment. Classical AI approaches based on symbolic processing, planning, general problem solving methods, the use of logic can all be interpreted as manifestations of the pressure towards generality by loose coupling or *decoupling* from the environment. An implementation strategy for variable coupling can be a layered architecture, where the decoupled higher layers support generality and the close coupled lower layers support power.

The consequences of these distinctions for the choice of representations and processing strategies is discussed and illustrated.

Some historical precedents and current implementation efforts towards such architectures are reviewed.

It is postulated that the topmost layer of such an architecture continues the layering indefinitely by having reflexive capabilities. It is also assumed that the distributed structure of this layer also supports the availability of "common information" at each of its components.

It is then shown that this layer is associated with phenomena of the agent's self, consciousness, subjectivity and "free" will.

The availability of common information at each component of the layer enables coordinated action and thus produces the unitary nature of the agent's self.

The role of consciousness is to support generality by being a modality-independent representation system in the architecture. Representations within this layer correspond to the subjective meaning extracted from the incoming information.

If there are nonlinearities present, the reflexive processing is capable of chaotic modes of behaviour. It is proposed that the unpredictability of agent action and hence the impression of free will are due to the fact that chaotic processes act as generators of information. This resolves the ontological conflict between determinism and free agent action. Chaotic processes are deterministic but are informationally decoupled and hence opaque, unpredictable, from an observer's point of view.

Toward an Architecture for Adaptive, Rational, Mobile Agents[1]

Innes A. Ferguson

Computer Laboratory
University of Cambridge,
Cambridge CB2 3QG, UK
Tel.: +44 223 334421
Fax: +44 223 334678
E-mail: iaf@cl.cam.ac.uk

July 7, 1991

Abstract

It is becoming widely accepted that neither purely reactive nor purely deliberative control techniques are capable of producing the range of behaviours required of intelligent agents in dynamic, unpredictable, multi-agent worlds. We present a new architecture for controlling autonomous, mobile agents – building on previous work addressing reactive and deliberative control methods. The proposed multi-layered control architecture allows a resource-bounded, goal-directed agent to react promptly to unexpected changes in its environment; at the same time it enables the agent to reason predictively about potential conflicts by constructing and projecting theories which hypothesize other agents' intentions.

The line of research adopted is very much a pragmatic one. A single, common architecture has been implemented which, being heavily parameterized, allows an experimenter to study functionally- and behaviourally-diverse agent configurations. A principal aim of this research is to understand the role different functional capabilities play in constraining an agent's behaviour under varying environmental conditions. To this end, an experimental testbed comprising a simulated multi-agent world has also been constructed. Some preliminary experience with the new control architecture is described.

[1]This work was supported by a Bell-Northern Research Postgraduate Scholarship and a CVCP Overseas Research Student Award. I would also like to thank William Clocksin and Julia Galliers for their helpful advice and support, and Barney Pell for many fruitful and enjoyable discussions.

1 Introduction

In order to survive and thrive in complex, real-world domains, future robotic agents will need to be made considerably more robust and adaptive than they are at present. Such domains (e.g. factory floors or space stations) are likely to be populated by multiple agents, each pursuing any number of goals. Because agents have incomplete knowledge about the world, it is inevitable that some of these goals will conflict. In real-world domains agents typically perform complex tasks requiring some degree of attention to be paid to computational resource bounds, temporal deadlines, and the impact their shorter-term actions might be having on their longer-term goals. On the other hand, time never stops or slows down for agents to deliberate upon all possible courses of action for every world state. Intelligent agents will require a range of skills to respond promptly to unexpected events, while simultaneously being able to carry out pre-programmed tasks and resolve unexpected conflicts in a timely and efficient manner. Not surprisingly, it is becoming widely accepted that neither purely reactive nor purely deliberative control techniques are capable of producing the range of robust, flexible behaviours desired of future intelligent agents.

In this paper we present a new multi-layered architecture for controlling autonomous, mobile agents or TOURINGMACHINES which combines capabilities for producing a range of reactive and deliberative behaviours in dynamic, unpredictable domains. This new approach is influenced on the one hand by recent work on reactive and behaviour-based agent architectures [Bro86, Fir87, Kae87], and on the other by more traditional AI endeavours such as planning, diagnostic theory formation [PGA86], resource-bounded reasoning [PIB87], and belief and intention modelling [Bra87, GLS87, PIB87].

Our research adopts a fairly pragmatic approach toward understanding how complex, dynamic environments might constrain the design of agents and, conversely, how different functional capabilities within agents might combine to generate different behaviours. To evaluate the TOURINGMACHINE architecture we have implemented a multi-agent simulation testbed. By varying parameters constraining agents' functional capabilities (e.g. sensing characteristics, attentional powers, degree of reactivity, world modelling powers) or parameters characterizing the environment itself (e.g. number of agents and obstacles, ratio of cpu time to simulated-world time), we can study a number of tradeoffs vis-à-vis how much reacting, planning, and predicting resource-bounded agents should be doing in order to behave rationally with respect to their goals.[2] In many ways, our approach to evaluating agent designs resembles the empirical approaches used in the Phoenix [CGHH89] and Tileworld [PR90] projects.

In our example domain we consider one or more agents, each with the task of following a different route from some starting location to some goal location within certain time bounds. Each agent starts with some geographical knowledge of the world (e.g. locations of paths and path intersections), but has no prior knowledge regarding other agents' locations or goals or static obstacles it might encounter along its route. An agent can communicate its intentions to turn or overtake by signalling – much like a driver does in a car – and can only consume up to some fixed number of computational resources per unit of simulated world time. Before discussing specifics of the TOURINGMACHINE architecture, its implementation, and its simulation testbed, we consider some important requirements for intelligent agency.

[2]The definition of rational behaviour used here is borrowed from Pollack *et al.* [PIB87] and corresponds to "the production of actions that further the goals of an agent, based upon [its] conception of the world."

2 Intelligent Agency

In recent years there has been considerable growth of interest in the design of intelligent agent architectures for dynamic, unpredictable domains. One popular design approach – whose resulting architectures we'll call *deliberative* – attempts to endow agents with sophisticated control by embedding in these a number of general AI capabilities such as means-end reasoning, epistemic modelling [PIB87], plan recognition [Woo90], or natural language understanding [VB90]. Influenced principally by the fruits of classical AI planning research, deliberative architectures have been designed both to handle complex goals (e.g. those involving action-at-a-distance, resource constraints, or multiple agents) and to operate flexibly in unpredictable or novel situations (e.g. by performing contingency planning or analogical reasoning). This generality, however, exacts a price; by virtue of having to maintain complete, up-to-date world models, deliberative architectures can be resource-intensive and are usually slow at making critical decisions in real-time situations.

Breaking with the traditionally held belief that "complex" architectures are required to produce intelligent agent behaviours, a number of *non-deliberative* (e.g. reactive [Fir87], situated [AC87, Mae90], and behaviour-based [Bro86, Kae87]) architectures have recently been proposed. These architectures are characterized by a more direct coupling of perception to action, increased decentralization of control, and relative simplicity of design. Because they perform localized search, the time spent deciding which action to effect in any given situation can be minimized. At the same time, however, these architectures run the risk of generating sub-optimal action sequences *precisely because* they operate with minimal memory or state information [Fir87]. Also, because non-deliberative agents are essentially *hardwired* to effect a particular action sequence in each given situation, they can be ineffective when confronted with situations which are either novel or which do not provide immediate access to the complete set of environmental stimuli needed for determining subsequent action sequences. Indeed, to date, there has been little evidence to suggest that pure non-deliberative architectures are capable of handling multiple, complex, resource-bounded goals in any sophisticated manner [Kir91, GLS87, Mae90]. Like their deliberative cousins, non-deliberative agents will require that their environments be reasonably cooperative if they are to achieve their goals satisfactorily [Bro86].

Operating in the real world means having to deal with multiple events at several levels of granularity – both in time and space. So, while agents must remain reactive in order to survive, some amount of strategic or predictive decision-making will be required if agents are to handle complex goals while keeping their long-term options open. Agents, however, cannot be expected to model their surroundings in every detail as there will simply be too many events to consider, a large number of which will be of little or no relevance anyway. What is required, in effect, is an architecture that can cope with uncertainty, react to unforseen events, and recover dynamically from poor decisions. All of this, of course, on top of accomplishing whatever tasks it was originally programmed for.

3 Touring Machines

For almost all practical purposes, an autonomous robotic agent must be *adaptive* – it must be capable of carrying out its intended goals in dynamic, unpredictable environments. To do this, we believe, the agent must be capable of exhibiting a range of different behaviours. First, it will need to be reactive to deal with events which it might not have had sufficient time or resources to consider. Secondly, since the agent's main task, in our case, will

be to get from some starting location to some goal location in some specified time, it should be capable of rational, resource-bounded, goal-directed behaviour. And thirdly, since it will inhabit a world populated by other entities (about which very little will be known in advance) it must be able to reason about what events are taking place around it, determine what effect these events could have on its own goals, and, where possible, predict what is likely to happen in the near future so as to be better informed when choosing and effecting subsequent actions. Because these skills have such disparate characteristics and requirements, the most sensible way of realizing them, it would seem, is as separate activity-producing behaviours in a layered framework. We have adopted this approach in designing and implementing TOURINGMACHINES.

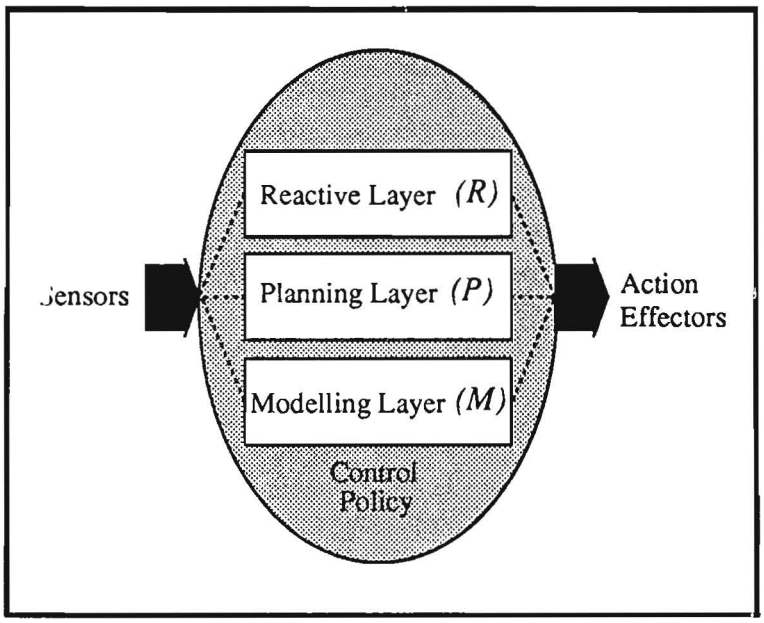


Figure 1: The TOURINGMACHINE architecture.

TOURINGMACHINES comprise three concurrently-operating, independently motivated, activity-producing layers: a *reactive* layer $\mathcal{R}$, a *planning* layer $\mathcal{P}$, and a reflective-predictive or *modelling* layer $\mathcal{M}$ (see Figure 1). Each models the agent's world at a different level of abstraction and each is endowed with different task-oriented capabilities. The TOURINGMACHINE framework is, in fact, *hybrid*, as it may incorporate several functional or *horizontal* faculties within a given task-achieving or *vertical* layer. For example, hypothetical reasoning and focus of attention are both realized in layer $\mathcal{M}$.

The main principle behind vertical decomposition is to create activity-producing subsystems each of which directly connects perception to action and which can independently decide if it should or should not act in a given world situation. Frequently, however, one layer's proposed actions will conflict with those of another: a layer is an *approximate* machine and thus its abstracted world model is necessarily incomplete. Because of this, layers need to be mediated by an enveloping *control policy* (Figure 1) if the agent, as a single whole, is to behave appropriately in each different world situation.

Implemented as a combination of inter-layer message-passing and context-activated control rules, the control policy's mediation enables each layer to examine data from other layers, inject new data into them, or even remove data from the layers. (The term *data*

here covers sensed input to and action output from layers, the contents of inter-layer messages, as well as certain rules or plans residing within layers.) This has the effect of altering, when required, the normal flow of data in the affected layer(s). So, for example, the reactive rule in layer $\mathcal{R}$ to prevent an agent from straying over lane markings can, with the appropriate control rule present, be overridden by layer $\mathcal{M}$ should the agent embark on a plan to overtake the agent in front of it.

Inputs to and outputs from layers are generated in a synchronous fashion, with the context-activated control rules being applied to these inputs and outputs at each synchronization point. The rules, thus, act as filters between the agent's sensors and its internal layers, and between its layers and its action effectors. Mediation remains active at all times and is largely "transparent" to the layers: each layer acts as if it alone were controlling the agent, remaining largely unaware of any "interference" (either by other layers or by the rules of the control policy) with its own inputs and outputs. The overall control policy is such that while striving to service the agent's high-level tasks (e.g. `exit-path`) it is sensitive also to its low-level, high-priority goals (e.g. `avoid-collision`).

The TOURINGMACHINE layered framework is strongly influenced by Brooks' *subsumption* architecture [Bro86]. This comprises several concurrently-operating, task-achieving behaviours which are implemented as fixed-topology networks of finite-state machines along with various registers and timers. Layers communicate via fixed-length messages over "wires" and are mediated by *suppression* and *inhibition* mechanisms which can alter the flow of inter-layer messages to produce the correct action for the situation at hand.

Besides several technical differences, the main distinction between the two architectures is that TOURINGMACHINES store and manipulate explicit representations of, among other things, beliefs, desires, and intentions in order to perform such cognitive tasks as reflection and prediction (see below). Brooks' agents have not to date been used to solve such high-level tasks, and it's not at all clear whether his architecture could be scaled up indefinitely without ever resorting to the use of internal representations [Kir91]. Aspects of the TOURINGMACHINE framework also bear some resemblance to the 2-layered (roughly $\mathcal{R}$ and $\mathcal{P}$) Phoenix architecture [CGHH89]. The following sections describe each layer in some more detail.[3]

3.1 Layer $\mathcal{R}$ (reactive)

The purpose of this layer is to provide an agent with fast, reactive capabilities for coping with events it hasn't previously planned for or modelled. A typical event, for example, would be the sudden appearance of some hitherto unseen agent or obstacle. Layer $\mathcal{R}$ provides the agent with a series of rules for avoiding obstacles, walls, kerbs or other agents, and for preventing it from straying over path lane markings. For example, the two rules for avoiding collisions with other agents are:

```
rule-4: if is-in-front(Other, Observer) and
           speed(Other) < speed(Observer) and
           separation(Other, Observer) < Front.Threshold
        then
           reduce-speed-by(Observer, speed(Observer) - speed(Other))

rule-5: if is-behind(Other, Observer) and
           speed(Other) > speed(Observer) and
```

[3] Due to space restrictions much detail will, in fact, be omitted and presented elsewhere [FerIP].

```
    separation(Other, Observer) < Rear.Threshold
then
    increase-speed-by(Observer, speed(Other) - speed(Observer))
```

where `Front.Threshold` and `Rear.Threshold` are parameters associated with the agent `Observer`. As we shall see below, an agent can be made variably reactive or inert by choosing appropriate values for these (and other) parameters.

Rules are stimulated *solely* and directly by input they receive from the agent's sensors. When a given rule fires, an appropriate action (e.g. `accelerate` or `turn-wheel`) is immediately sent to the agent's effectors.[4] Clearly, actions effected at this level cannot be guaranteed to be rational since rules are memoryless and fire on the agent's sensory information alone. Consequently, each time a reactive rule fires, layer $\mathcal{M}$ (modelling) must be flagged (sent a message by layer $\mathcal{R}$) so that it can assess whether the resulting unplanned state change will require further processing. In particular, layer $\mathcal{M}$ will need to determine if any actions effected by layer $\mathcal{R}$ are likely to prevent the agent from achieving its planned tasks.

3.2 Layer $\mathcal{P}$ (planning)

The purpose of this layer is to generate and execute plans. Since an agent's main task typically involves relocating to some destination within certain pre-specified time bounds, it makes sense for the agent to do some amount of forward planning (e.g. `locate-destination`, `calculate-cruise-speed`). However, since the agent is very likely to encounter other entities unexpectedly, complete, detailed plans are undesirable if replanning is to be kept to a minimum. Layer $\mathcal{P}$, therefore, is realized as a linear, hierarchical, partial planner which can interleave plan formation and execution, and defer committing to specific subplan execution methods or temporal orderings of subplans until absolutely necessary. Also, since TOURINGMACHINES have limited computational resources, the planner is designed so that its operation can be pre-empted and its state suspended for subsequent use. The plan elaboration scheme employed is akin to the partial elaboration method of PRS [GLS87] and the lazy skeletal expansion scheme used in Phoenix agents [CGHH89]. In essence, we take Bratman's view [Bra87] that plans are useful for constraining the amount of subsequent deliberation an agent will need to perform.

The planner manipulates and instantiates template plans or *schemas* which it retrieves from a *schema library* (Figure 2). Schemas are procedural structures consisting of a body, a set of preconditions, a set of applicability constraints (e.g. temporal ordering), a set of postconditions, and an associated cost in terms of computational resources. Schemas are either *primitive* or *composite*. Primitive schemas can either submit physical actions to be effected (e.g. `turn-wheel`, `signal-left`) or perform various arithmetic or geometric calculations (e.g. `calculate-stopping-distance`). Composite schemas trigger library searches and subplan expansion. The planner also has access to a database of topological facts about its task domain.

The planner uses a fixed, combined depth-first and best-first search strategy for constructing *single-agent* plans. Apart from occasionally generating sensory acts to determine the location of, say, a fixed landmark, the planner remains largely "unaware" of what's

[4] Several reactive rules could fire simultaneously but only one is allowed to submit its corresponding action; currently the rule triggered by the (spatially) nearest environmental stimulus is chosen. Other selection policies may be considered in the future.

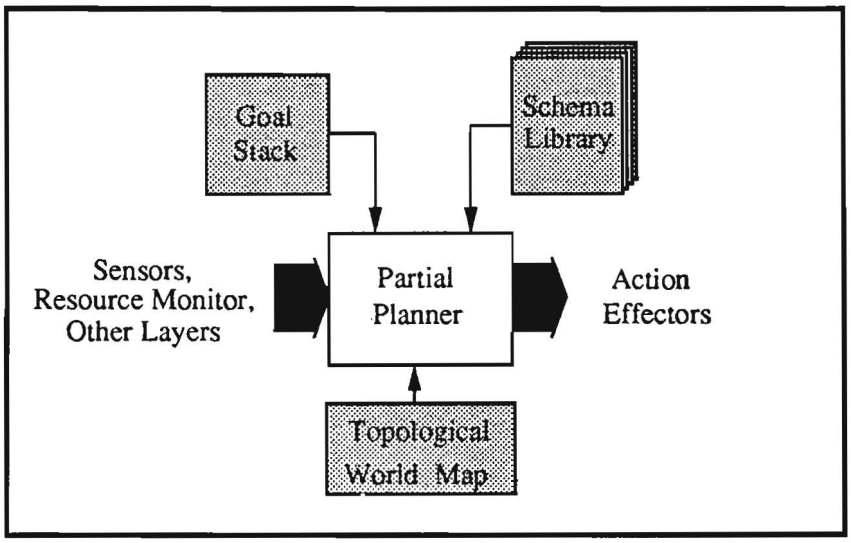


Figure 2: Layer $\mathcal{P}$.

going on around it. In particular, it does not consider what other agents are doing, this task being left to layer $\mathcal{M}$ which, in effect, is the only part of the agent that has any *reasoned* view of what other events are taking place in the world. So, while the planner is capable of some limited backtracking (e.g. to try an alternative execution method if the chosen one has failed or to try to re-satisfy an applicability constraint), initiation of *dynamic (re-)planning* (e.g. `overtake-agent`) is the responsibility of layer $\mathcal{M}$. Layer $\mathcal{P}$, then, is able to take on new goals and abandon old ones if layer $\mathcal{M}$ so dictates. In this manner, layer $\mathcal{P}$ keeps abreast of changes in the agent's environment.

3.3 Layer $\mathcal{M}$ (modelling)

The main purpose of layer $\mathcal{M}$ is to provide an agent with reflective and predictive capabilities. The agent realizes such capabilities by constructing *models* of world entities, including itself, which it uses as a platform for explaining observed behaviours and making predictions about possible future behaviours.[5] The potential gain in this approach is that by making successful predictions about other entities' activities the agent should be able to detect potential goal conflicts earlier on. This would then enable it to make changes to its own plans in a more effective manner than if it were to wait for these conflicts to materialize. Goal conflicts can occur within the agent itself (e.g. the agent's projected time of arrival at its destination exceeds its original deadline or the agent's layer $\mathcal{R}$ effects an action which alters the agent's trajectory) or in relation to another agent (e.g. the agent's trajectory intersects that of another agent).

Other functions made available to the agent through this layer (see Figure 3) include a heuristic focus of attention module for creating closures within which to perform inferencing and a goal conflict detection/resolution facility for dealing with intra- and inter-agent conflicts. Like every module in the TOURINGMACHINE architecture, each function in layer $\mathcal{M}$ is resource-bounded, thus ensuring a degree of reactivity in the agent as a whole.

[5] We assume TOURINGMACHINES can readily identify various physical properties of world entities such as type, size, Cartesian location, speed, acceleration, orientation, and communicated information. This concords with most other simulated agent environments [CGHH89, DM90, PR90, SH88, VB90, Woo90].

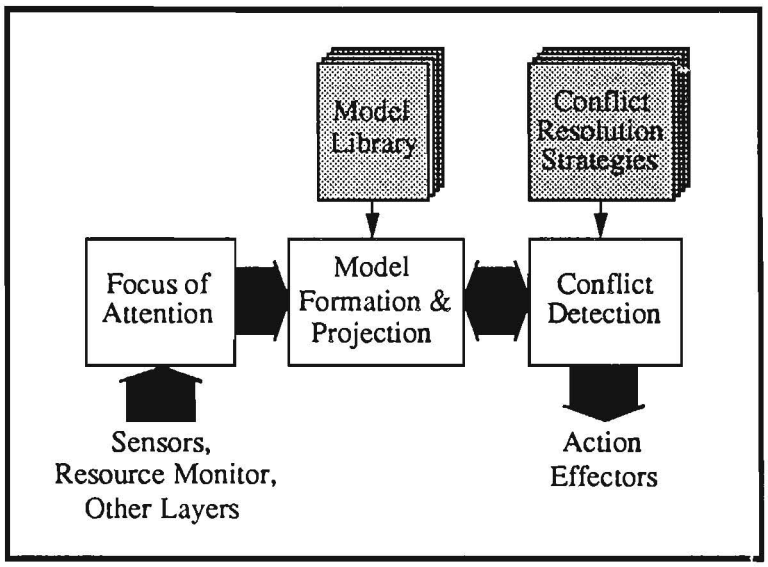


Figure 3: Layer $\mathcal{M}$.

The structures used by an agent to model an entity's behaviour are time-indexed 4-tuples of the form $\langle C, B, D, I \rangle$, where C is the entity's *Configuration*, namely, (x, y)-location, speed, acceleration, orientation, and signalled communications; B is the set of *Beliefs* ascribed to the entity; D is its ascribed list of partially-ordered goals or *Desires*; and I is its ascribed plan or *Intention* structure.[6] The models used by an agent are, in fact, filled-in instances of model templates which the agent obtains from a library (Figure 3). While all templates have the same basic 4-way structure, they can be made to differ in such aspects as the depth of information that can be represented (e.g. a particular template's B component might not permit nested beliefs), initial default values provided, and cost. The last of these will subsequently be taken into account each time the agent makes an inference from the chosen model.

Reasoning from a model of an entity essentially involves looking for *discrepancies* between the entity's *actual* behaviour and that *predicted* by its model or, in the case of a self-model, between the agent's actual behaviour and that *desired* by the agent. Predictions are formed by temporally projecting those parameters that make up the modelled entity's configuration vector C, in the context of the current world situation and the entity's ascribed intentions. Noticing a discrepancy between actual and predicted (or desired) behaviours, however, need not on every occasion force the agent into a wholesale revision of its "faulty" model. This is because associated with each of the parameters of a model's C-vector are upper- and lower-bounds whose sizes can be chosen by the testbed user. The agent doing the modelling, then, will become "aroused" only if the entity's observed configuration parameters fall outside the corresponding C-vector bounds in its model of the entity. Clearly, different settings for these parameter bounds will affect both the amount of environmental change perceptible to the agent and the amount of time the agent will need to spend revising its models. Studying such tradeoffs in TOURINGMACHINES is a focus of

[6] Plan ascription or recognition has been realized in TOURINGMACHINES as a process of *scientific theory formation* which employs an abductive reasoning methodology similar to that of the Theorist default/diagnostic reasoning system [PGA86].

current study. Achieving the optimal level of sensitivity to environmental change has also been recognized as a critical issue in Sanborn and Hendler's Traffic World system [SH88] and – through the use of plan-monitoring *envelopes* – in the Phoenix project [CGHH89].

4 Experimental Testbed

To validate TOURINGMACHINES, we have implemented our control architecture in SICStus Prolog and are experimenting with it in a simulated 2-dimensional world occupied by, among other things, other TOURINGMACHINES, obstacles, walls, paths, and assorted information signs. World dynamics are realized by a discrete event simulator which incorporates a plausible world updater for enforcing "realistic" notions of time and motion, and which creates the illusion of concurrent world activity through appropriate action scheduling. Other processes handled by the simulator include a facility for tracing scenario parameters, a statistics-gathering package for agent performance analysis, and several text and graphics windows for displaying output.

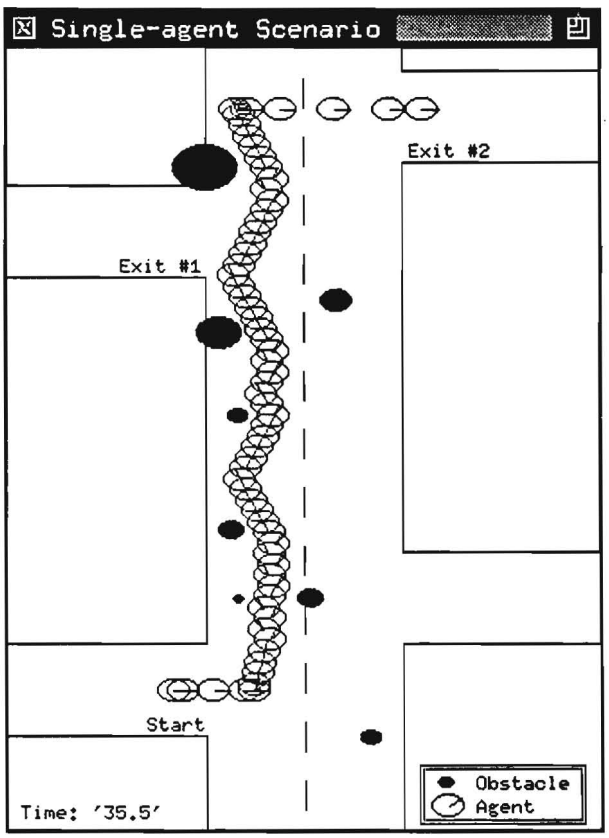


Figure 4: Graphical output from the testbed showing a scenario involving one agent and several obstacles.

Our testbed also provides a scenario definition facility which allows us to generate scenario instances from a fairly rich collection of *agent-* and *environment-level* parameters. So, for example, we can configure a TOURINGMACHINE to be variably reactive by altering parameters defining such things as the distribution of computational resources within its three control layers, the amount of forward planning it performs, the sensitivity of its

reactive rules, or the frequency with which it senses or models the world. In a similar fashion, we can experiment with the TOURINGMACHINE'S tolerance to environmental uncertainty by adjusting its sensing horizon, by tightening its initial goal deadline, by populating its world with many other fast-moving agents, or by varying the ratio of cpu to simulated world time used in the scenario. This last one affects the amount of time the TOURINGMACHINE has to deliberate between clock ticks.

The TOURINGMACHINE testbed has been designed to enable controlled, repeatable experimentation and to facilitate the creation of diverse agent scenarios for subsequent user analysis. Based on some very early tests, we are satisfied that our agents can behave robustly in the presence of unexpected obstacles while successfully accomplishing time-constrained, relocation-type goals (see Figure 4). But this is just the beginning. Ultimately, through the design and analysis of more complex scenarios, we hope to gain more insight into the *behavioural ecology* – to use Cohen's terminology [CGHH89] – of TOURINGMACHINES. In other words, we are interested in studying, and eventually discovering general rules that describe, the relationships and tradeoffs that exist between an agent's design (in other words, the particular configuration of its functional capabilities and knowledge sources), its environment, and the repertoire of demonstrable behaviours that the agent is capable of. So, for example, we are interested in understanding how well a given TOURINGMACHINE configuration might perform across a range of environments and also how the behaviours of different configurations of TOURINGMACHINES compare when placed in a single common environment. Criteria with which to evaluate the performance of our agents have already been identified and include, among others, resource consumption and utilization, wasted planning effort (e.g. amount of backtracking or replanning required), number of successful/unsuccessful actions effected, ratio of successful to unsuccessful model-based explanations or predictions, number of model revisions performed, and delay in arriving at a target destination.

5 Conclusions

We have presented a new, robust control architecture for resource-bounded, goal-directed, mobile agents operating in dynamic environments. Our layered, activity-producing architecture integrates both deliberative and non-deliberative control features enabling a TOURINGMACHINE to produce a range of reactive, goal-oriented, reflective, and predictive behaviours as demanded by the agent's goals and environmental situation. This empowers agents to deal with events and tasks at different levels of granularity (e.g. avoiding collisions, accomplishing complex goals, predicting world behaviour). We have also briefly described a feature-rich simulation testbed within which we have started to study design-behaviour-environment tradeoffs.

By using a highly parameterized, layered architecture we have benefited greatly in terms of our effort to design, implement, and test different agent configurations. Our experience so far has demonstrated that TOURINGMACHINES can be configured to behave "sensibly" in dynamic environments. The work presented here is ongoing: future work will include functionally extending our current implementation (e.g. adding more agent plans and model templates, enhancing inter-layer control), as well as experimenting with multiple, heterogeneous agents in diverse environments. We believe this will provide us with important clues about how best to design adaptive, rational, autonomous agents.

References

[AC87] Philip E. Agre and David Chapman. Pengi: An implementation of a theory of activity. In *Proceedings Conference of the American Association for Artificial Intelligence*, pages 268–272, 1987.

[Bra87] Michael E. Bratman. *Intention, Plans, and Practical Reason.* Harvard University Press, Cambridge, MA, 1987.

[Bro86] Rodney A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–23, 1986.

[CGHH89] Paul R. Cohen, Michael L. Greenberg, David M. Hart, and Adele E. Howe. Trial by fire: Understanding the design requirements for agents in complex environments. *AI Magazine*, 10(3):32–48, 1989.

[DM90] Edmund H. Durfee and Thomas A. Montgomery. A hierarchical protocol for coordinating multiagent behaviours. In *Proceedings Conference of the American Association for Artificial Intelligence*, pages 86–93, 1990.

[FerIP] Innes A. Ferguson. *Touring Machines: An Architecture for Adaptive, Rational, Mobile Agents.* PhD thesis, Computer Laboratory, University of Cambridge, Cambridge, UK. (In preparation.)

[Fir87] James R. Firby. An investigation into reactive planning in complex domains. In *Proceedings Conference of the American Association for Artificial Intelligence*, pages 202–206, 1987.

[GLS87] Michael P. Georgeff, Amy L. Lansky, and Marcel J. Schoppers. Reasoning and planning in dynamic domains: An experiment with a mobile robot. Technical Note 380, SRI International, Menlo Park, CA, April 1987.

[Kae87] Leslie Pack Kaelbling. An architecture for intelligent reactive systems. In M.P. Georgeff and A.L. Lansky, editors, *Reasoning about Actions and Plans - Proceedings 1986 Workshop*, pages 395–410. Morgan Kaufmann Publishers, Inc., Los Altos, CA, 1987.

[Kir91] David Kirsh. Today the earwig, tomorrow man? *Artificial Intelligence*, 47:161–184, 1991.

[Mae90] Pattie Maes. Situated agents can have goals. *Robotics and Autonomous Systems*, 6(1&2):49–70, 1990.

[PGA86] David L. Poole, Randy G. Goebel, and Romas Aleliunas. Theorist: A logical reasoning system for defaults and diagnosis. Research Report CS-86-06, University of Waterloo, Waterloo, Ont., February 1986.

[PIB87] Martha E. Pollack, David J. Israel, and Michael E. Bratman. Toward an architecture for resource-bounded agents. Technical Note 425, SRI International, Menlo Park, CA, July 1987.

[PR90] Martha E. Pollack and Marc Ringuette. Introducing the Tileworld: Experimentally evaluating agent architectures. In *Proceedings Conference of the American Association for Artificial Intelligence*, pages 183–189, 1990.

[SH88] J. Sanborn and J. Hendler. A model of reaction for planning in dynamic environments. *International Journal of Artificial Intelligence in Engineering*, 3(2):95–102, 1988.

[VB90] Steven Vere and Timothy Bickmore. A basic agent. *Computational Intelligence*, 6(1):41–60, 1990.

[Woo90] Sharon Wood. *Planning in a Rapidly Changing Environment.* DPhil thesis, University of Sussex, Brighton, UK, 1990.

Eco-Problem-Solving model: Results of the N-Puzzle

Alexis DROGOUL[1], Christophe DUBREUIL[2]

Abstract

Eco-Problem-Solving (EPS) is a new approach to problem solving based on the paradigms of Distributed Artificial Intelligence and founded on interacting agents. We show the way to decomposing the n-puzzle problem into EPS agents, the behaviors with which they are provided and some general mechanisms. Then, we show how simple interactions between agents can lead to the solving of the problem of the edges. We prove that our solving method is guaranteed always to find a solution if there is one. We also prove that the method is more than complete and becomes decidable. Evidence of completeness and decidability are formulated. Finally, we propose some empirical results to show that the EPS implementation of the n-puzzle can effectively solve significantly larger problems than have previously been solvable using traditional heuristic search methods.

1 Université Pierre et Marie Curie - LAFORIA
Tour 46-0 - 2ème étage
4, Place Jussieu 75252 PARIS Cedex 05 FRANCE
drogoul@laforia.ibp.fr

2 CERT-ONERA
B.P. 4025
2, Avenue Ed. Belin 31055 Toulouse Cedex FRANCE
dubreuil@tls-cs.cert.fr

1) Introduction

Our aim in this paper is to show that Eco-Problem-Solving, based on the paradigms of Distributed Artificial Intelligence and interactive agents, is able to solve any size of n-puzzle without planning.

Section 2 approaches n-puzzle vis à vis classical planning. The model of Eco-Problem-Solving is presented in Section 3. Section 4 describes the agents involved in the n-puzzle solving and Section 5 algorithms used in their behaviors. In order to see exactly how it works, Section 6 shows the solving of the standard problem of the edges, thanks to snapshots of the system in progress. The completeness and the decidability of our method are proved in Section 7. Then, Section 8 presents empirical results for several sizes of n-puzzle.

2) N-Puzzle and Heuristic Search

N-puzzle consists of a square frame containing N square tiles and an empty position called the "blank". Authorized operations slide any tile adjacent to the blank into the blank position. The task is to rearrange the tiles from some random initial configuration into a particular designed goal configuration.

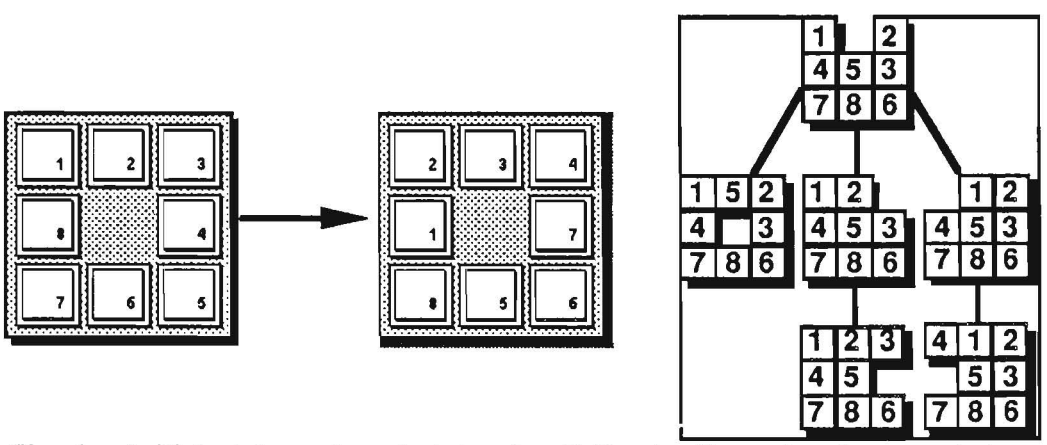


Fig. 1 - Initial state and goal state of a N-Puzzle; Example of a search tree

N-Puzzle is the common example of a problem that requires the use of heuristic search algorithms to be solved. In fact, the set of operators, the initial and goal states can be easily defined as well as the entire set of states of the problem.

A* [7] is the best known of these algorithms. Many of its implementations use the Manhattan Distance function for estimating the relative merits of different states of the puzzle relative to the goal state. A* has the property of always finding an optimal solution to the n-puzzle, given a fine heuristic function. But it needs in practice both exponential space and time to run, so its applicability is restricted to relatively small problems (8-puzzle for A*, 15-puzzle for IDA* [10]).

To overcome this drawback, latest approaches have sacrificed solution optimality for the benefit of a limited search horizon, in order to solve larger puzzles than have previously been solvable using A*. The major items in terms of results are the Real-

Time-A* and the Learning-Real-Time-A* algorithms [11]. It seems now possible to solve as far as the 24-puzzle in a reasonable time (i.e. less than a human).

But even these works appear to be limited with respect to the greater sizes of n-puzzles. First, the search horizon to obtaining good solution lengths (in terms of moves of the tiles) seems exponentially to increase with the size of the puzzle (92 states on average for the 8-puzzle, 2622 for the 15-puzzle). Secondly, both A* and RTA* need to be specially adapted to each size of puzzle, given the fact that the heuristic functions are not necessarily the same.

Considering these limits, we think it is now time to question the heuristic search paradigm and to explore other ways of solving.

3) Eco-Problem-Solving (EPS) model

This model is based on the paradigm of "computational eco-systems" [8]. Problem solving is seen as the production of stable states in a dynamic system, where evolution is due to the behaviors of simple agents. A problem is then defined by a population of interacting agents.

EPS is twofold: a domain independent kernel where behaviors of the agents are described and a domain dependent application where their actions are coded. We already used this to solve various AI problems (e.g. cubes world, hanoï towers) [4].

EPS agents are actor-based and use Agha's model of continuations [1]. An agent possesses another agent as goal and acquaintances. It only takes decisions from its local informations, without knowing about any global state of the world. It has a simple behavior which can be compared to a basic "biological" pattern: satisfaction, flight.

Note that our approach differs from connectionism: our agents are not statically linked together and they behave independently. It also differs from other distributed approaches, such as "distributed planning" [5] or "planning for multiple agents" [9] where solution is obtained by coordination of the agents local plans. The actions of our agents follow the three principles below[3]:

a) The will to be satisfied: *This corresponds to the description of the goal. A function in the kernel called TrySatisfaction handles it. This function calls two domain-dependent actions: doSatisfaction (if the agent can be satisfied) or satisfactionAggression (if it must attack other agents to seek satisfaction). A satisfied agent does not seek satisfaction anymore, unless it is provided with a new goal.*

b) The will to be free: *Before trying to act, an agent has to be free. Freeing itself consists in attacking its jailers (acquaintances that prevent it from acting) and in telling them to flee.*

c) The obligation to flee: *Fleeing is the answer to an attack. It makes the agent change its position in the problem to avoid conflicts. The function flee handles it and leads to two domain-dependent actions: doFlee (if there is a way to flee) or fleeAggression (if it must attack other agents to flee). A flee message is often supplied*

[3] Depending on the characteristics of the problem, agents may have additional knowledge about their environment and domain-dependent behaviors that are not described in the kernel.

with a constraint (another agent) given by the attacker. The fleeing agent, then, will not have the possibility to attack this constraint.

A problem in EPS is defined by describing its initial state (the acquaintances of the agents are initialized) and its final state (a goal is given to each agent) [6]. The allocation of a goal to an agent creates a slave-master relationship between the agent and the goal. The agent becomes a *dependency* of its goal and is at the same time the *master* of its own *dependencies. Dependencies* will see their satisfaction only after that of their goal (when an agent has reached its goal it informs its *dependencies* by sending them a TrySatisfaction message).

N-Puzzle has been implemented under the EPS kernel called EcoTalk. It is based on the kernel defined by Jacques Ferber [3], and Actalk, a language of actors under Smalltalk-80 [2]. Each of the problem entities is represented by a class of actors which has the actions seen above. Their ancestor is a class named EcoAgent which defines the kernel methods (so all the agents will inherit these behaviors).

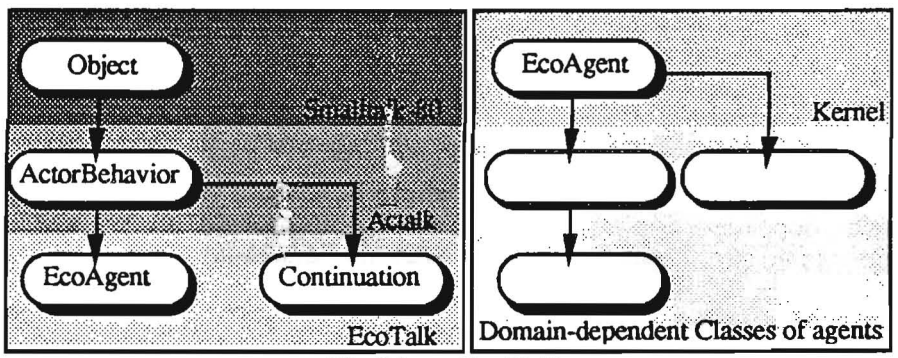


Fig.2A - EcoTalk

4) N-puzzle Agents

4.1) Decomposition

N-puzzle has been decomposed into three different types of agents: N+1 EcoSquares, N EcoTiles and the EcoPuzzle. Squares are the locations on which tiles move. Problem solving begins by giving each agent its goal and acquaintances, and asking the EcoPuzzle to seek satisfaction.

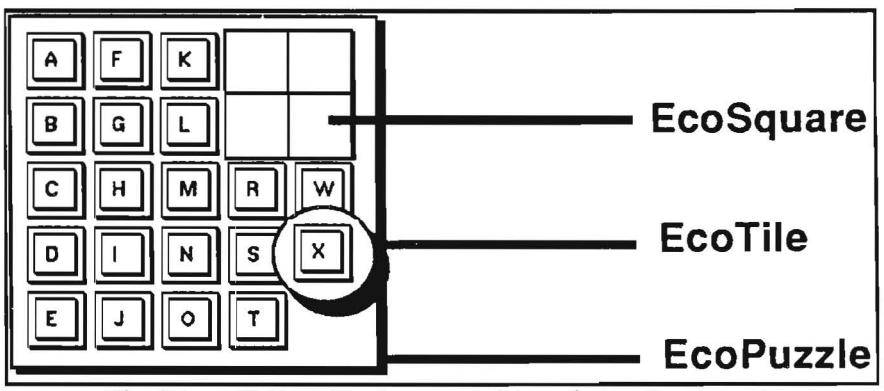


Fig.2B - N-Puzzle decomposition into Eco-agents

4.2) Behaviors of the EcoSquares

The acquaintances of an EcoSquare are: the tile lying *on* it (its *on*) and the squares adjacent to it (its *adjacents*). An EcoSquare is always satisfied and not able to flee. It can free itself by sending a "flee" message to its *on*. An EcoSquare can also be locked or unlocked, depending on the behavior of its *on* .

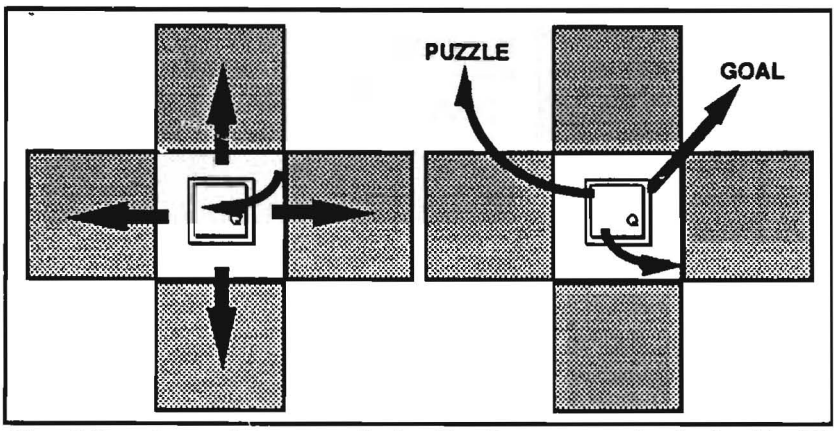


Fig. 2C - Acquaintances of the EcoSquares and of the EcoTiles

4.3) Behaviors of the EcoTiles

The acquaintances of an EcoTile consist in the EcoSquare on which it lies (its *under*), the square on to which it has to move (its *goal*) and the *puzzle*. Its satisfaction is called incremental. The agent searches among the *adjacents* of its *under* the nearest unlocked to its *goal* . If the square is not free, the tile tells it to free itself and moves on it. When freeing it, an EcoTile can transmit two possible constraints to the tile that will have to flee: its *goal* (if it is an edge square), or the *under* of the previous satisfied tile (in order to preserve its satisfaction).

The flight behavior of a tile consists in searching the nearest square to the blank, different from the constraint and unlocked, among the *adjacents* of its *under* . If the *goal* of the tile is found among them, it is prioritarly chosen. If no suitable squares are found, the tile takes the nearest one to the blank and tells the *puzzle* to unlock all the squares. The tile tells the chosen square to free itself and moves on to it.

A tile locks its *under* when it tells another square to free itself or when it becomes satisfied. That means it gives indirect information to the other agents whose meaning is: "I am already attacking or satisfied, so do not attack me unless no other choice can be made". This square is unlocked when the tile moves.

4.4) Behavior of the EcoPuzzle

The acquaintances of the EcoPuzzle are the squares, the tiles and a *goal* . It follows satisfaction behavior that consists in determining in which order the tiles will try to satisfy themselves and telling the first tile in this list to do it (see Section 5.1 for details).

A puzzle also possesses general mechanisms such as the computation of the distance between two squares, or the ability to lock/unlock its lines and columns. The next section presents some of these because of their importance in the solving.

5) General Mechanisms

5.1) Serializing the attempts of tile satisfaction

Tiles cannot satisfy themselves together at the same time. A sole blank location does not allow two tiles to move concurrently. Therefore, it is necessary to serialize their attempts of satisfaction. A relationship is provided by EPS between an agent and its *goal*. Unfortunately, the goals of the tiles are not relative (a tile) but absolute (a square). Consequently, there are no direct slave-master relationships between the tiles. An indirect slave/master relationship between the tiles is then generated by creating a relationship between the squares (this creation has been realized using EPS [4]).

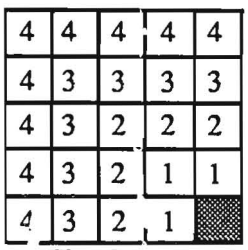


Fig. 3.A

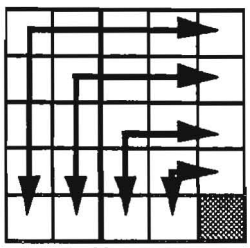


Fig. 3.B

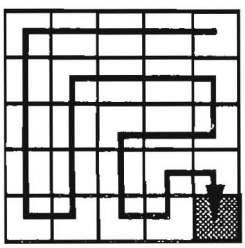


Fig. 3.C

All the squares are ordered in a list by their distance to the blank in the final state (3.A). Squares at the same distance are chained together (3.B). Then (3.C), these lists are linked up.

When the ordered list of squares has been obtained, the next step consists in giving the right goals to the puzzle and the squares (The future blank location is not in the list). The *goal* of the first square becomes the puzzle. The *goal* of each square is the tile whose *goal* is the previous square in the list. The *goal* of the puzzle is the tile whose *goal* is the last square in the list.

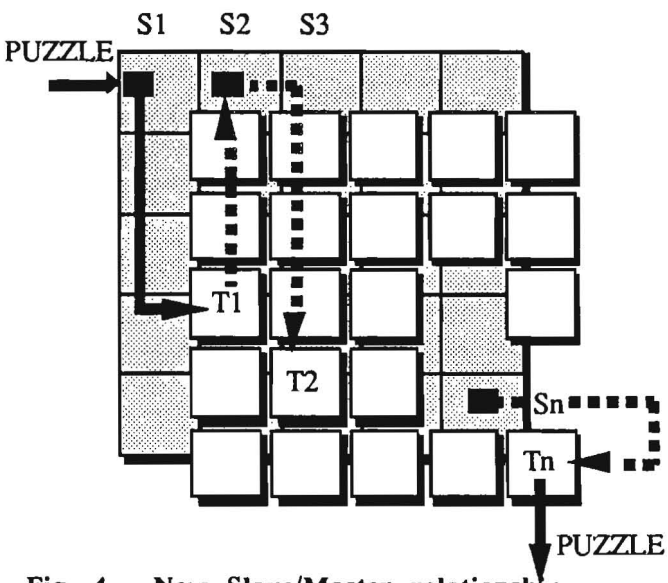


Fig. 4 - New Slave/Master relationship

The puzzle has received the message TrySatisfaction. It is already satisfied and so informs its dependencies that they can satisfy themselves. The first square, S1, whose goal was the puzzle, is also satisfied (a square is always satisfied) and informs the tile T1 that it can try to satisfy itself. Once satisfied, this tile informs in turn its dependencies (i.e. the square whose goal was T1), and so on. When Tn, the last tile, has been satisfied, it informs the puzzle that it can satisfy itself again. The loop has been completed and the solving stops.

5.2) Distance computation

The algorithm used for calculating the right distance between squares is based on the Manhattan distance algorithm and Voronoï's diagram [12]. A distance is calculated between a target-square and a list of start-squares. The target is asked by the puzzle to

generate a wave by transmitting the value 1 to its *adjacents* . This wave is then transmitted to their *adjacents* with a value increased by one. Squares cannot transmit the wave to locked neighbors (the wave breaks on "obstacles") and already valued squares. Once the start-squares have been reached, the wave stops.

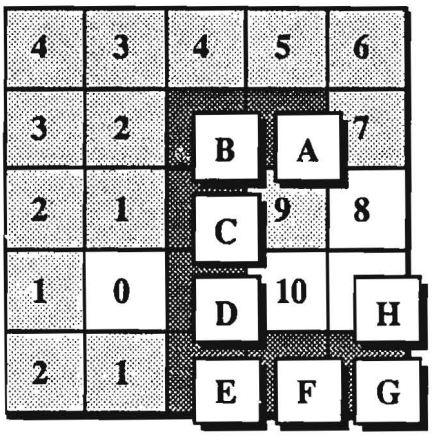


Fig.5A - Distance Computation

Tile A has attacked Tile B, which has attacked Tile C, etc... And Tile H has been asked to flee. Before fleeing, Tile H chooses a square on which to move. This square must be the nearest of its square's adjacents to the blank. Tile H then asks the puzzle to calculate the distance between these squares and the blank. The wave generated by the blank square breaks on the squares whose tiles are already fleeing. Once the two squares adjacent to Tile H have been reached, the puzzle gives them back ordered by their value to the tile which will choose the square whose "value" is the smallest.

This system can be seen as a sort of gradient whose value equals the distance at which it has been generated. It allows tiles to know the right distance between two squares in terms of "real" moves and illustrates the appeal of square locking.

The complexity of this method is in the worst case in O(n), where n is the number of unlocked squares. As a matter of fact, it is easy to see that an unlocked square only transmits one value to its neighbors and can not be reached again by the wave.

5.3) Locking columns and lines

The EcoPuzzle also possesses another mechanism that allows it to lock definitively an entire line/column of the puzzle when all the agents making up this line/column are satisfied. When an EcoTile has reached its goal, it just asks the puzzle to verify if its line/column can be locked. It may be locked if it is on the border of the puzzle or if an adjacent line/column has already been locked. The agents (EcoTiles and EcoSquares) making up this line/column are killed and this line/column will no longer be disturbed by the other agents. The purpose of this mechanism is to reduce dynamically the size of the puzzle during the solving in order to accelerate the distance calculations. For instance, a 24-puzzle becomes a 15-puzzle once a line and a column have been locked.

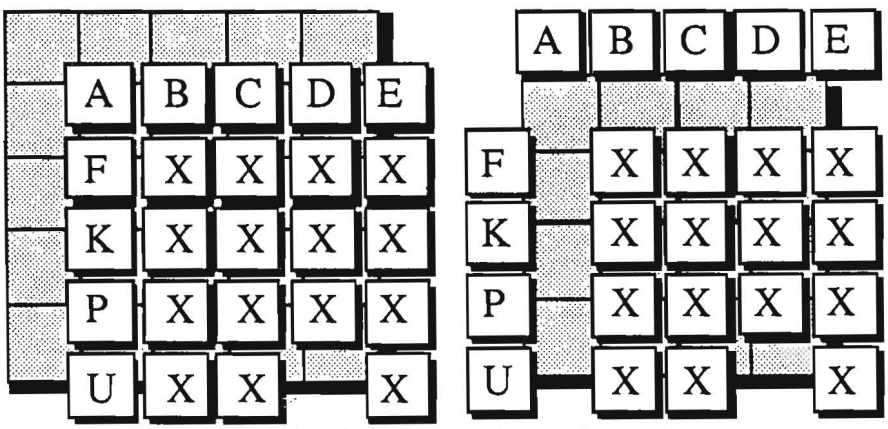


Fig.5B - Locking a line and a column

6) Emergence of complex behaviors

The underlying paradigm of our solving system is to make the solution of a problem emerge from local interactions. As an instance of emergence, the solving of the problem of the edges is described in this section. This problem is as follows: how can a tile whose goal is an edge of the puzzle satisfy itself without disturbing the satisfied tiles that make up the line or the column including this edge ?

General planning systems apply special heuristics to overcome this difficulty. In contrast, EPS solves it without requiring any special behaviors and the solution simply emerges from the interactions between the tiles and the constraints they transmit when attacking other agents. Figure 6 shows how it works in a 24-puzzle.

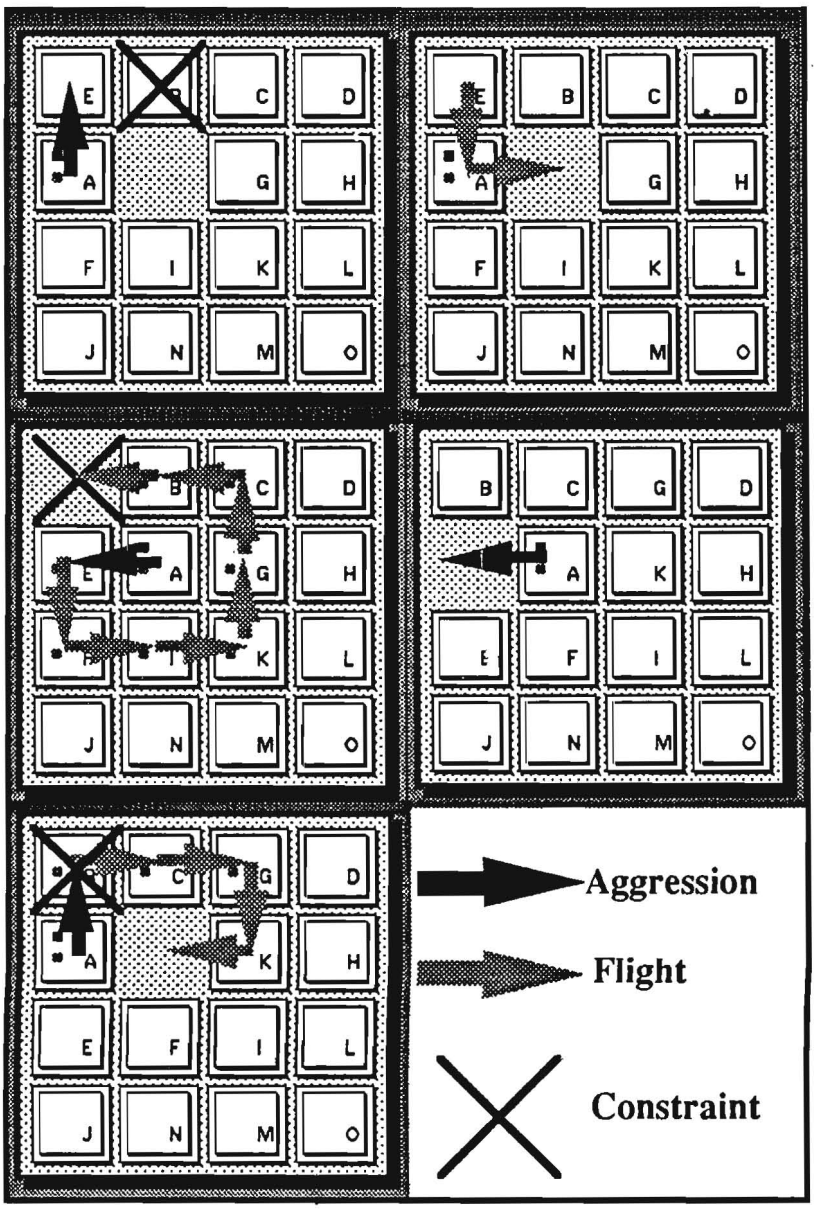


Fig. 6 - The problem of the edges

Tiles B,C and D are already satisfied and A tries to satisfy itself. Therefore, it attacks E with the square of B as constraint (B is the previous satisfied tile). E then attacks A in order to flee (it cannot attack B) and tells the puzzle to unlock all the squares (All its adjacent squares are locked).

A flees on the blank and tries to satisfy itself again by attacking E with its goal as constraint.

E attacks F, which attacks I, which attacks K and so on up to B. B then flees on the blank (the previous constraint was only valuable for E), followed by all the fleeing tiles.

A moves on the blank and attacks B which lies on its goal. The transmitted constraint is the goal of A (because B is no longer satisfied). B attacks C, which attacks G (C does not flee on the blank but on its goal). G attacks K. K flees on the blank, followed by all the fleeing tiles. And A can now satisfy itself by moving on its goal.

7) Completeness and decidability

7.1) Completeness

The proof of the completeness of our problem solving method for any size of n-puzzle will be made in three steps: first, proving the fact that, given a puzzle whose dimensions are N x N (N > 1), we can reduce it to a (N-1) x (N-1) puzzle. Secondly, proving that we can solve the 1-puzzle (1 x 1). Finally concluding on the proof that we can solve any size of n-puzzle with the same agents.

We assume that the serialization of the squares is the same as in Figure 1, Section 5.1. We also assume that the lines and columns are numbered from 1 to N, starting from the blank position in the final goal state. Thus, the first tiles that will try to satisfy themselves will be those making up the line N.

Theorem 1: ***The N-1 first tiles of line N can satisfy themselves without being shifted by an external attack.***

Proof: *The first tile can satisfy itself and lock its square without disturbing any other tiles. Let us now assume that the i^{th} tile ($1 < i < N-1$) has satisfied itself and locked its square without disturbing the previously satisfied ones. Can the $(i+1)^{th}$ tile reach its goal without shifting the previous tiles ?*

A tile can be pushed out of its square for two reasons: (1) An adjacent tile tries to satisfy itself and attacks it; (2) An adjacent tile tries to flee and attacks it. But a tile on a locked square can never be attacked (see Section 4.3). So the i previously satisfied tiles cannot be shifted unless their squares are unlocked.

Squares can only be unlocked when a fleeing tile has found no place to flee because all its adjacent squares were locked (Section 4.3). Can this happen? The answer is no, thanks to the distance computation; a tile never attacks a square that leads to a dead-end because the blank or the goal would not be reachable from this square (Section 5.2).

Moreover, as the goal of the $(i+1)^{th}$ tile is not a corner (only the first and the N^{th} tiles have corners as goals), it never transmits any constraints to the tiles it attacks. So a fleeing tile is always able to choose a square from which the blank is reachable and never asks the puzzle to unlock all the squares.

We have proved that, given i satisfied tiles on line N ($1 < i < N-1$), the $(i+1)^{th}$ tile can satisfy itself without disturbing them. We have also showed that the first tile can satisfy itself without disturbing any other. The proof of the theorem by a simple induction on i is straightforward.

Theorem 2: ***When the N^{th} tile of the line N seeks satisfaction, it may destroy some previous satisfactions but puts them back in place when reaching its goal.***

Proof: *When trying to reach its goal, the N^{th} tile can meet three situations: (1) Its goal is blank and the N^{th} tile is adjacent to it: it just moves on it without destroying any satisfactions.(2) Its goal is occupied by another tile and the N^{th} tile is adjacent to it: this is the situation described in Section 6. Some satisfactions are destroyed but, when the tile moves on its goal, it necessarily attacks the $(N-1)^{th}$ tile. As a fleeing tile primarily moves on its goal when it is adjacent (see Section 4.3), the $(N-1)^{th}$ tile attacks the (N-*

1)th square (which can make the (N-2)th tile flee at its turn on its goal, and so forth until all the satisfactions have been re-established). (3) Its goal is blank and there is another tile T between the Nth tile and its goal: the Nth tile attacks T with the blank as constraint. T cannot flee because the other square adjacent to the blank contains the (N-1)th tile and is locked. It then asks the puzzle to unlock all the squares and attacks the tile below the (N-1)th tile. Whatever occurs after that does not matter because we are back to situation 2.

These two theorems prove that, given a N x N puzzle, we are able to build the line number N. The same proofs, once columns and lines have been exchanged, show that we are also able to build the column number N.

Section 5.3 explains that entirely satisfied lines or columns can be definitively locked and no longer used in the problem. Then, we lock line N and column N and consider their agents as dead. And doing this creates a new puzzle, whose dimensions are (N-1) x (N-1). So the system can always reduce the solving of an N x N puzzle to the solving of an (N-1) x (N-1) puzzle, whatever N > 1 may be .

1-puzzle is easy to solve with our system. It is decomposed into an EcoPuzzle and an EcoSquare. The solving begins by sending the puzzle the message TrySatisfaction. The puzzle tries to create the list of ordered squares, but, as no squares other the blank location can be found, obtains an empty list. Thus, it stops. It has found the solution.

An N x N puzzle (N > 1) can be solved only if the (N-1)x(N-1) puzzle (obtained by locking a line and a column of the previous one) can be solved and the 1 x 1 puzzle is solved. The method is then correct for solving any size of n-puzzle.

Consequently, whatever may be the size of the puzzle to be solved, the behaviors of the agents do not have to change.

7.2) Decidability

The other important consequence of this proof is the decidability of the solving method: It finds the solution when there is one and stops with acceptance. Where there is none, it also stops but with no acceptance. How does it stop ?

Theorem 3: ***If a satisfied tile is attacked, and if the goal of the tile that seeks satisfaction is not an edge, the puzzle is not solvable.***

Proof: *We assume that the puzzle is greater than 2. Theorems 1 & 2 prove that satisfied tiles cannot be shifted unless the goal of the current tile that seeks satisfaction is an edge. So, if a satisfied tile is attacked while the goal of the current tile is not an edge, there is something wrong with the puzzle.On the other hand, a wrong initial state should generate loops among the tiles. Loops can only occur between tiles that seek satisfaction and make the other one flee at each turn. But it would necessarily mean that one of them has reached its goal (two tiles cannot seek satisfaction concurrently). And that is the situation to which theorem 3 applies.*

Pratically, a way to stop when this situation is met is as follows: Everytime a tile seeks satisfaction, it informs the puzzle of the *current* goal. Everytime a satisfied tile is attacked, it asks the puzzle if the *current* goal is an edge. If not, the tile does not answer the attack and the solving stops, leaving the puzzle in a wrong state.

8) Empirical Results

EPS system is able to solve very large puzzles. The n-puzzle has been tested on the 8 up to the 168-puzzle. Some of them were as yet unsolved because of the cost of the computation. Figure 7 shows the average solving times and the average solution lengths in terms of moves.

All the results have been registered with the EcoTalk kernel written in Smalltalk-80 release 2.5, running on a Macintosh IIfx. They have been obtained by making the average over a hundred random problem instances of each size.

It is important to notice that we do not look for optimality in solving. The lengths of the solutions obtained for the 8 and 15 puzzles in terms of moves are approximatively a little more than the double of the optimal ones (respectively 22 and 53 moves, as computed by A*). While no practical techniques exist for computing optimal solutions for greater puzzles, we cannot conclude on their performances. But the system solves them in a reasonable time.

Size	Average time (in seconds)	Average number of moves	Average moves per tile
8	1,5	51	6,4
15	5,6	133	8,9
24	18,5	298	12,4
35	49,3	525	15,0
48	91,2	802	16,7
63	160,5	1155	18,5
80	300,4	1712	21,4
99	484,8	2273	22,9
120	665,7	2830	23,6
143	830,4	3132	21,9
168	1020,5	3624	21,5

Fig. 7 - Performances of the EPS implementation of the n-puzzle

The other interesting fact is that the average number of moves a tile needs to be satisfied increases very slowly and even decreases for the biggest sizes of n-puzzles. This means that the quality of the solution in terms of moves increases with the size of the puzzle.

9) Conclusion

We present a new approach to Distributed Problem Solving. We show that the EPS model, based on agents provided with satisfaction and flight behaviors, solves the n-puzzle problem without using a heuristic search approach, but as the result of the interactions between agents.

Then, we prove that the solving method generated by EPS is correct, complete and decidable for any size of n-puzzle ($n > 2$). That means the solving always stops: with the solution if there is one and in a wrong state if there is none.

Finally, we present empirical results that demonstrate that the EPS implementation is effective at solving larger problems than have previously been solvable with heuristic search algorithms because combinational explosion has been drastically reduced.

We are currently working on extensions of the n-puzzle problem, and our purpose will be to show that a few changes should allow the actual implementation to solve any kind of n-puzzle: any frame (not necessarily a square), any number of blank locations.

Acknowledgements

The EPS implementation of the n-puzzle problem have been produced in collaboration with Jacques Ferber. We are sincerely grateful to Anne Collinot and Eric Jacopin for the valuable comments they have provided in drafting this paper.

References

[1] **G. Agha** "Actors - A model of Concurrent Computation for Distributed Systems"
MIT Press, 1986.

[2] **J.-P. Briot** "From Objects to Actors, study of a limited symbiosis in Smalltalk-80"
LITP Report 88-58RXF, September 1988.

[3] **C. Delaye, J. Ferber & E. Jacopin** "An interactive approach to problem solving"
in Proceedings of ORSTOM'90, November 1990.

[4] **A. Drogoul & C. Dubreuil** "EPS Implementations of classical AI problems"
LAFORIA Technical Report, LAFORIA 1990 (in French).

[5] **E.H Durfee, V.R. Lesser, D.D. Corkill** "Cooperation through communication in a distributed problem solving network"
in "Distributed Artificial Intelligence", M. Huhns (Ed) Pitman Publishing,1987.

[6] **J. Ferber & E. Jacopin** "The Framework of Eco problem solving"
in Proceedings of MAAMAW 90, page 103-114, 1990.

[7] **P.E. Hart, N.J. Nilsson and B. Raphael**, "A formal basis for the heuristic determination of minimum cost paths"
IEEE Trans. Syst. Sci. Cybern. 4, 1968.

[8] **B.A. Hubberman & T. Hogg**, "The behavior of Computational Ecologies"
in "The Ecology of computation", B.A. Huberman, Ed. North Holland, 1988.

[9] **M.J Katz & J.S. Rosenschein**, "Plans for multiple agents"
Workshop on Distributed Artificial Intelligence (Preliminary Papers), Lake Arrowhead USA, 1988.

[10] **R.E. Korf**, "Depth-First iterative-deepening: An optimal admissible tree search"
in Artificial Intelligence 27, 1985.

[11] **R.E. Korf**, "Real-Time Heuristic Search"
in Artificial Intelligence 42, 1990.

[12] **M.I. Shamos & D. Hoey** "Closest point problems"
in Proceedings of the 16th IEEE Symposium on Foundations of Computer Science"

Exploiting Emergent Behaviour in Multi-Agent Systems

Peter Wavish
Philips Research Laboratories,
Redhill, Surrey, England, RH1 5HA
wavish@prl.philips.co.uk

Abstract

This paper addresses the question of how to exploit emergent behaviour in the design of multi-agent systems. The method advocated is to design the individual agents so that they maintain symbolic representations of emergent behaviour which can then be used as a basis for building higher level behaviours. The paper falls into four parts: a description of behaviour-based agents, a discussion of emergent behaviour, a description of a methodology for developing agents which exploit emergent behaviour, and a practical example of the application of these ideas to the development of a simulated multi-agent system.

1. Introduction

Emergent behaviour is an issue in multi-agent systems in two respects. Firstly, the behaviour of a system or group of agents is emergent from the behaviour of individual agents [e.g. Steels, 1990]. Secondly, the architecture of an individual agent can itself rely on emergent behaviour [e.g. Brooks, 1989]. In either case emergent behaviour is to be valued as an efficient and robust way of producing behaviour.

A central problem with designing systems with emergent behaviour is to produce the desired emergent behaviour in the first place. Existing methods include careful design [Brooks, 1989] and machine learning [Maes & Brooks, 1990]. In this paper, however, we will set aside the problems involved in creating emergent behaviour and concentrate instead on how the emergent behaviour can be made use of in the design of multi-agent systems.

The problem is that emergent behaviours, in their nature, have no symbolic representation and so cannot easily be used to build higher level behaviours. This is particularly the case for agents which are designed and implemented entirely as sets of interacting symbolic representations of behaviour.

The paper is organised as follows. Section 2 describes behaviour-based agents and the behaviour representation language we have developed for implementing them. Such agents consist of a set of symbolic representations of behaviour which correspond to their actual behaviour as perceived by the designer.

Section 3 starts with a definition of emergent behaviour. It then shows how the problem referred to above can be overcome by creating and maintaining, within the agent, new symbolic behaviours which are maintained so that they correspond to the emergent behaviours and so can be used as a basis for building further behaviours.

Section 4 presents a methodology for developing systems of behaviour-based agents which maintain internal representations of emergent behaviour. This methodology is based on the use of a simulation of the multi-agent world.

Finally, section 5 describes the application of this methodology in a simulated mobile robot domain.

2. Behaviour-based agents

Whereas much Artificial Intelligence work is based on the explicit representation of knowledge, our approach to multi-agent systems is based on the explicit representation of *behaviour* [Brooks, 1985]. This is because we believe that the activity of an agent is produced primarily by the interplay between the agent and its environment, not by reasoning processes (such as planning) occurring entirely within the agent [Suchman, 1987; Agre & Chapman, 1987; Rosenschein & Kaelbling, 1986]. By representing behaviour explicitly it is possible to model the agent, its environment and their causal interactions within a single coherent framework.

One of the consequences of representing behaviour explicitly is that, knowing the *current* behaviour of a system, it is possible to predict its immediate *future* behaviour. Repeatedly predicting future behaviour in this way gives rise to a *simulation* of the system. This makes it possible to design programming languages whose source code consists of declarative symbolic representations of behaviour and whose mode of execution is discrete event simulation. We have designed a programming language which works in this way called ABLE (Agent Behaviour LanguagE) and a real-time variant of it called RTA (Real Time ABLE) [Wavish & Connah, 1990; Graham & Wavish, 1991; Wavish 1991]. Both languages are summarised in figure 1.

From a programming language perspective, atomic and simple behaviours correspond to simple facts in PROLOG, licences and schemas are different kinds of forward chaining production rule, functions correspond to PROLOG predicates, and worlds correspond to partitions of the temporal database. The main distinctions between ABLE and traditional production rule languages are that rules (i.e. licences, schemas and functions) execute in parallel, they are dynamic and can be arbitrarily nested, and they can be time-annotated with the times for which components of the condition must be present and the durations for which newly created behaviours must persist.

From a representational perspective, atomic behaviours usually represent objects or agents, simple behaviours usually represent attributes of objects (the way they are currently behaving), licences and schemas represent different kinds of causal links between behaviours, and worlds represent different realities which are largely causally independent. The overall computation mimics the way that cause and effect appear to operate in the real world. Changes propagate independently and concurrently in a way that depends on the structure of the symbolically represented behaviour through which the changes are propagating. Interactions between different trains of cause and effect depend critically on their timing relative to the time-line of the simulated world.

Atomic behaviour *agent1*

Simple behaviour *agent(agent1)*

Licence – predicts independent behaviour

agent(A) & say(B,U)/0.1 →
hear(A,say(B,U))/0.5

Schema – predicts dependent behaviour

agent(A) ⇒
{
object(A),
hear(A,say(B,U))/0.2 & agent(B) →
look_at(A,B)
}

Function – defines 'virtual' behaviour

together(A,O) ⇐
(at(A,P) & at(O,P))/0.1

World – self-contained set of behaviour

goal(agent1,goal_world:at(agent1,place3))

Figure 1: Behaviour constructs in ABLE and RTA

The existing ABLE interpreter is quite slow, and this has in the past prevented us from running multi-agent simulations such as [Connah & Wavish, 1990] and [Hickman & Shiels, 1991] in real time. This is a problem both because it prevents the designer from making free use of the simulation for trying out ideas, and because it restricts the class of real agents which can be implemented in ABLE. To overcome this problem, we have defined a compilable variant of ABLE called RTA (Real Time ABLE) which is fast enough to operate in real time. RTA texts are compiled into C code which can then be compiled either for the host workstation or for a target microprocessor. The compiled code is typically hundreds of kilobytes in size and executes at 10,000 events per second on a typical workstation. Its internal time-line is locked to the real time clock of the computer. RTA is therefore effective both for simulating agents in real time and for actually implementing them.

The particular way that RTA is restricted compared with ABLE is that the set of possible behaviours is finite and is determined at compile time. In practice this means that licences and schemas cannot contain unbound variables, so building and decomposing structure at run time is not possible. The effect of this from the programmer's viewpoint is to force a style of representation similar to that used in Pengi [Agre & Chapman, 1987]. For instance,

the representation of another agent saying something is not, as it is in ABLE, the simple behaviour:

```
hear(agent1,say(agent2,"hello"))        % agent1 hears agent2 say "hello"
```

because variables would be needed to decompose this structure. Instead, while agent2 is actually uttering and agent1 is hearing, the following set of behaviour will be active in agent1:

```
{
hear(agent1,"hello"),                   % agent1 hears "hello" being said
see(agent1,agent2),                     % agent1 is looking at agent2
see_speaking(agent1)                    % agent1 sees something speaking
}
```

In other words the binding between the components of a complex behaviour is provided not by their incorporation into a single data structure but by their co–occurrence at a particular moment in time. An agent encoded in RTA can be viewed as a large, fixed set of such behaviours, of which only a small subset will be active at any one time. These behaviours are linked together by a network of logical operators and delays which determine how the behaviours are interrelated. In fact, agents are represented by the RTA compiler as asynchronous digital logic circuits, where behaviours are implemented as registers, logical operators as logic gates, and delays as monostables. This level of representation is however normally hidden from the designer, in contrast with hardware–oriented approaches such as [Agre & Chapman, 1987].

When agents of this kind interact with each other or with their environment in general, their overall behaviour, and also the behaviour of the system of which they form part, emerges from interactions between their explicitly represented behaviours. This emergent behaviour is not represented directly, nor is it represented by any simple function of the behaviours that *are* actually represented. In the next section we focus on this kind of emergent behaviour.

3. Emergent Behaviour

A good introduction to emergent functionality is given in [Steels, 1991]. For the purposes of this paper, however, we will define emergent behaviour in a way which is relevant to the concerns of a designer programming in ABLE or RTA. Such a designer has a vocabulary of behaviour descriptions V1 with which behaviour of a system can be described, and a much smaller subset V2 which is actually used in the written texts describing and ultimately generating the behaviour of the system. *Emergent behaviour* is that behaviour which is produced by behaviour describable by V2, but which is not itself describable by V2, although it is describable by V1.

To see how this definition is applied, consider the emergent walking behaviour of Brook's six–legged robot [Brooks, 1989]. Walking is not easily describable in terms of the explicitly represented behaviours (such as moving a leg forward) from which it emerges, but it is

immediately obvious to the observer that the robot is exhibiting walking behaviour and 'walking' is already part of the designer's vocabulary.

Stable emergent behaviour is valuable both because it can be produced much more economically than explicitly programmed behaviour and because it is typically very robust. The problem with emergent behaviour within a behaviour programming framework is that because, by definition, it is not explicitly represented, it is difficult to make use of it to build higher level behaviours. For example, the designer may wish to add another behaviour to the six-legged robot which depends on (or controls) whether it is walking or not, but if there is no explicit representation of walking behaviour, this cannot easily be done.

The basis of a solution to this problem is for the designer to add a new symbolic behaviour SB1 that represents the real emergent walking behaviour EB1. This behaviour must be maintained so that it is "on" when the real walking behaviour exists and is "off" when real walking behaviour is absent. It is then possible to add further behaviour B2 which *apparently* depends on the real emergent behaviour EB1 but is *actually* causally linked to its symbolic representation SB1. This state of affairs can be depicted as follows:

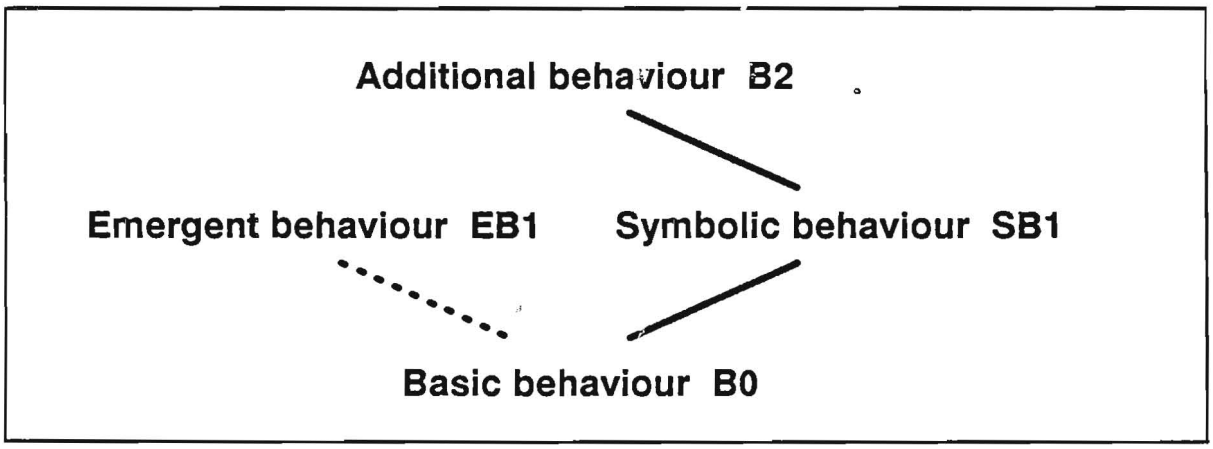


Figure 2: Maintaining a symbolic representation of emergent behaviour

In figure 2, bold lines represent causal links between behaviours which may run in either (or both) directions, and the broken line represents emergence. The emergence link is also bi-directional in the sense that emergent behaviour EB1 couples back to affect the basic behaviours B0 which produce it (for instance, the emergent walking behaviour of Brooks' robot coordinates the individual move-leg-forward behaviours). Notice that there is no direct link between EB1, the emergent behaviour, and B2, the additional behaviour which is apparently built on top of it.

The diagram glosses over the distinction between real behaviour and symbolic representations of behaviour at the level of basic behaviour B0. The *symbolic* behaviour B0 represents only a subset of the *real* behaviour B0. This in practice restricts the causal links that can be established by the designer between B0 and SB1. However, in order to simplify the following discussion, it will be assumed that there is a direct correspondence

between real behaviours and their symbolic representations at this level, so we can think of B0 in terms of either real or symbolic behaviour as appropriate.

The strategy for building B2 on top of SB1 by causally linking it to SB1 only works if the symbolic behaviour correlates well with the emergent behaviour EB1, i.e. one does not exist without the other. Since the causal and emergent relations shown in the diagram are bi-directional, there are two ways of achieving this:

1) the basic behaviours B0 and the corresponding emergent behaviour EB1 already exist, and the symbolic behaviour SB1 is caused by the behaviours in B0 which indicate the presence of EB1.
2) the symbolic behaviour SB1 indirectly gives rise to emergent behaviour EB1 through its direct causal links to the set of basic behaviours B0 which create the conditions for emergence.

In either case, the behaviours EB1 and SB1 will be correlated with each other, but in the first case the emergent behaviour 'drives' the symbolic behaviour, whereas in the second case the symbolic behaviour 'drives' the emergent behaviour. Although we will focus on these two cases, there is also a third case which is the superposition of the first two cases, in which the emergent behaviour EB1 and the symbolic behaviour SB1 both drive each other and so are locked together in a mutually supportive loop.

In the first case, the symbolic behaviour SB1 needs to be turned "on" when the emergent behaviour starts, and turned "off" when it finishes. The designer's task is to determine what particular configurations of basic behaviours B0 are associated with the existence of the emergent behaviour EB1, and to devise a network of behaviour which allows the basic behaviours B0 to determine the state of SB1 appropriately. Relevant behaviours in B0 include those which are known to give rise to the emergent behaviour and those which indicate its presence.

In the second case, the emergent behaviour EB1 needs to be produced when the symbolic behaviour SB1 is turned "on", and removed when it turns "off". The designer's task is to find some subset of basic behaviours B0 which, in conjunction with other existing basic behaviours, will give rise to the emergent behaviour EB1, and to devise a network of behaviours which allows the state of the symbolic behaviour SB1 to determine the existence of the appropriate basic behaviours.

So far, we have implicitly been considering the behaviour of a single agent (in other words, even though the emergent behaviour will normally depend on the effect of the agent's environment, it is predominantly the behaviour *of the agent itself*). In the typical multi-agent situation, however, the emergent behaviour EB1 may be largely *outside* the agent itself and may emerge from the concurrent activity of a number of agents. The symbolic representation SB1, however, is still located *within* the agent (because that is the only place where the designer can actually put symbolic representations). Figure 3 shows the relation between the agents and its environment.

A further difference is that the set of basic behaviour B0 is effectively split into the subset AB0, which is the basic behaviour of the agent, and the subset WB0, which is the basic

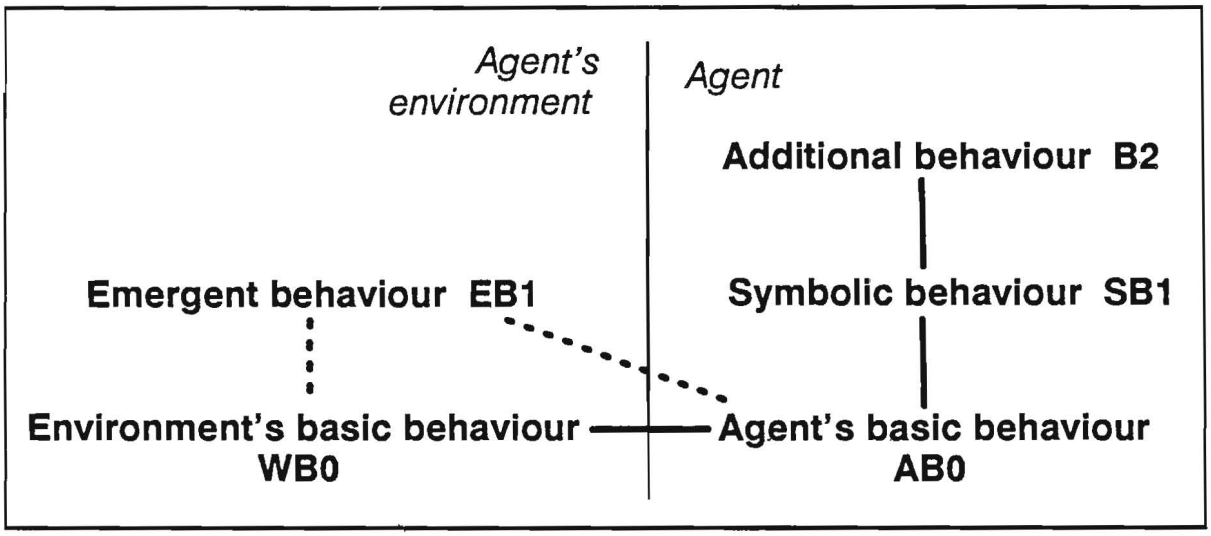


Figure 3: The relation between the agent and its environment

behaviour of its environment or world. These two subsets are however causally linked. This makes the designer's problem of maintaining a high correlation between the symbolic behaviour SB1 and the real emergent behaviour EB1 much more difficult, because the agent has direct access only to AB0, and not to WB0, although it is the union of these sets that is responsible for the existence of EB1.

In the first case we have been considering, where the emergent behaviour EB1 drives the symbolic behaviour SB1, the agent's basic behaviour is influenced by the emergent behaviour both directly and indirectly through WB0. Typically nothing can be done to strengthen the direct link, so in order to improve the correlation between EB1 and SB1 the indirect link through WB0 must be strengthened. This can be done either by adding sensors to the agent or by increasing its level of activity so that it interacts more frequently with behaviours in WB0. In either case, the whole process of maintaining SB1 in correspondence with EB1 can be regarded as a process of *perception*.

In the second case, where the symbolic behaviour SB1 drives the emergent behaviour EB1, the basic behaviour of the agent AB0 is part of the behaviour in the world from which the emergent behaviour EB1 emerges. The basic behaviour of the agent AB0 also affects behaviour in its environment WB0 which also contributes to the emergence of EB1. The designer must consider both paths in order to determine how the emergent behaviour EB1 is to be produced. The whole process of creating and maintaining emergent behaviour in the world according to the state of the symbolic behaviour SB1 can be regarded as *action*. It should be remembered, however, that the point of this action is not just to affect the behaviour of the world, but to make it correspond to the agent's representation of it.

We have presented a picture of how an agent can maintain its internal representation of emergent behaviour in correspondence with real emergent behaviour in its environment. The purpose of the internal representation is to provide a 'handle' on the emergent behaviour so that the agent can behave appropriately in its presence. There are at least

three general ways in which the emergent behaviour may be related to the agent's current interests and hence to its behaviour:

1) **The emergent behaviour is beneficial** with respect to the agent's current interests, in which case the agent's behaviour is designed to support the continued existence of the emergent behaviour.
2) **The emergent behaviour is damaging** with respects to the agent's current interests, in which case its behaviour is designed to suppress the emergent behaviour.
3) **The emergent behaviour is neutral** with respect to the agent's current interests, in which case the agent may make use of its representation of the emergent behaviour to ensure that its behaviour in general is appropriate to the presence of the emergent behaviour.

This completes our account of emergent behaviour. The central idea is to maintain a correspondence between real emergent behaviour in the world and a symbolic representation of that behaviour in the agent. The technical problems are to do with maintaining that correspondence. The reason for taking the trouble to solve these technical problems is that it then becomes possible to build higher levels of behaviour apparently (but not really) on top of the emergent behaviour. In the next section we will put forward a methodology for developing multi-agent systems in this way.

4. Methodology

This picture of the relation between emergent behaviour and its symbolic representation within the agent gives rise to a particular way of developing multi-agent systems based on simulation of the system. The designer depends heavily on the simulation to reveal spontaneously occurring emergent behaviour, to test out ideas for creating emergent behaviour, and to optimise the causal pathways within the agent that maintain the correspondences between the real emergent behaviour and its symbolic representation.

The overall development cycle is edit, compile, run, and observe what happens. Four interesting situations can be singled out:

1) **Expected and wanted emergent behaviour is present.** Adjust the causal chain of behaviour WB0 – AB0 – SB1 to optimise the correlation between EB1 and SB1 and to optimise the stability of EB1 with respect to changes in AB0. This keeps the system 'in tune'.
2) **Expected and wanted emergent behaviour is absent.** Check and if necessary redesign the causal chain of behaviour SB1 – AB0 – WB0 so that AB0 and WB0 provide the right conditions for EB1 to emerge. This makes SB1 available for future use.
3) **Unexpected emergent behaviour is present.** Create a new causal chain of behaviour EB1 – WB0 – AB0 to maintain the new symbolic behaviour SB1. This makes SB1 available for future use.
4) **Unwanted emergent behaviour is present.** Design a new causal chain of behaviour SB1 – AB0 – WB0 which removes the conditions necessary for the emergent behaviour EB1 to exist. This suppresses EB1 whenever it occurs.

As the development progresses, the vocabulary of behaviour available will become extended by the process of maintaining representations of newly observed behaviours and so provide an increasingly rich base for creating further new behaviour. It should perhaps be pointed out that this process results in the agent acquiring explicit representations of behaviours which are *attributed to the system by the designer*. While this is not necessarily the optimal way for a simple agent to view the world, it is much easier for the designer if the agent and the designer share a common conceptualisation of what is going on.

This process may be compared with the original subsumption architecture approach [Brooks, 1985] where once a layer of behaviour is defined, its implementation is frozen. In the present approach, the implementation of any behaviour may be freely changed as the development progresses provided it remains correlated with the real behaviour it represents as perceived by the designer. This ensures that higher levels of behaviour can be built on a solid foundation. In fact, as higher levels of behaviour are built, lower levels tend to go 'off tune', and it is normally necessary to review their design from time to time.

5. An example

This methodology has been tried out in a simulated mobile robot domain. One robot, the 'dog', has the task of herding five other robots, the 'sheep' into a 'pen' consisting of a number of obstacles arranged in a partial ring, as shown below. The obstacles can be rearranged to create different problems for the agents. A 'shepherd' agent whose position can easily be controlled by the user is also provided.

Each robot can move in any of eight directions, and has two means of sensing, corresponding roughly to sight and touch. Both the robot's touch surface and its field of view are divided into eight sectors, in each of which it senses either the presence or the absence of an object. In the case of sight, the class of object to which the sight sensor is sensitive can be predetermined. This means that when the dog is visually sensing the sheep, it can tell where in its visual field the sheep are but cannot for example identify individual sheep or even tell how many sheep there are. In addition to this, the dog can see whether it is facing the shepherd or not (in other words, whether the shepherd is in any of the three forward facing visual sectors).

Much of the basic behaviour of the dog and sheep is concerned with dealing with collisions and visually seeking objects. Two behaviours of the dog will be singled out here because they correspond to the two ways of maintaining the correspondence between the emergent and the symbolic behaviours described in the previous section. The first behaviour is that of being trapped, which illustrates how a behaviour which is apparent to the designer has a behaviour representation maintained for it which is then used to influence further behaviour of the dog. The second behaviour is that of the dog driving the sheep across the field, which is built from a pair of emergent rounding-up behaviours, and illustrates how emergent behaviour is produced in response to symbolic behaviour.

The trapped behaviour of the dog occurs when the dog is surrounded by obstacles which block its forward progress. Notice that the trap itself is usually perceived as a trap by the

human observer who sees the configuration of the obstacles, but it does not exist as an object within the simulation, and is not perceived as such by the dog. A *blocked* behaviour for the dog is maintained so that it exists when the dog is being touched on any of its front three faces (touching behaviour for each of the dog's eight faces is maintained by software which computes the distance and relative orientation of objects within the simulated two dimensional world). The normal behaviour of the dog in response to being blocked is to turn in the appropriate direction until it is no longer blocked. If the dog happens to be surrounded by obstacles, it will repeatedly move forward, become blocked, turn, move again and so on. *trapped* behaviour is maintained so that it starts to exist when the dog has been repeatedly blocked for some period of time, and ceases when the dog has *not* been blocked for some time.

It can be argued that being-trapped behaviour, as described, is not emergent, on the grounds that it is (almost) compositionally related to touching and being-blocked behaviours. It is therefore worth saying why we consider it to be emergent. We are distinguishing in this paper between the vocabulary of behaviour which is used to generate the behaviour of the agents, and the considerably larger vocabulary available to the designer. It is a matter of fact that when the dog is surrounded by obstacles and is repeatedly being blocked, turning and moving forward again, it is strongly perceived (by human observers) as being trapped even though there is no symbolic representation of its being-trapped behaviour. The behaviour observed is not describable within the vocabulary so far used by the designer which includes touching, being blocked, and turning. Being trapped is related to other concepts such as escaping which are at a different conceptual level.

Having defined and maintained a symbolic behaviour *trapped* which correlates well with the perceived being-trapped behaviour, it becomes easy to program the dog so that it behaves appropriately. For example, if the dog is visually seeking an object when it becomes trapped, it will tend to stay in the corner of the trap nearest its objective, so preventing a search for a way out of the trap. *trapped* behaviour is therefore used to suppress goal-seeking behaviour of this kind so that the dog is free to escape from the trap. A further behaviour of the dog in this situation is to start howling, so drawing the attention of another agent (such as the human observer) to its predicament.

A different way of viewing *trapped* behaviour is that it is one component of the dog's overall model of its situation, and it is maintained by what is normally termed a process of perception. This is broadly correct, but it is the activity of the whole dog in turning, moving and becoming blocked that is used to maintain *trapped* behaviour, so this is much closer to Neisser's account of perception [Neisser, 1976] than to the traditional information processing model.

Having described how a symbolic representation of behaviour is maintained to correspond with spontaneously occurring emergent behaviour, we will now examine emergent behaviour which is driven by its symbolic representation. Figure 4 shows the tracks of the dog and the five sheep as the dog rounds up the sheep and drive them towards the shepherd.

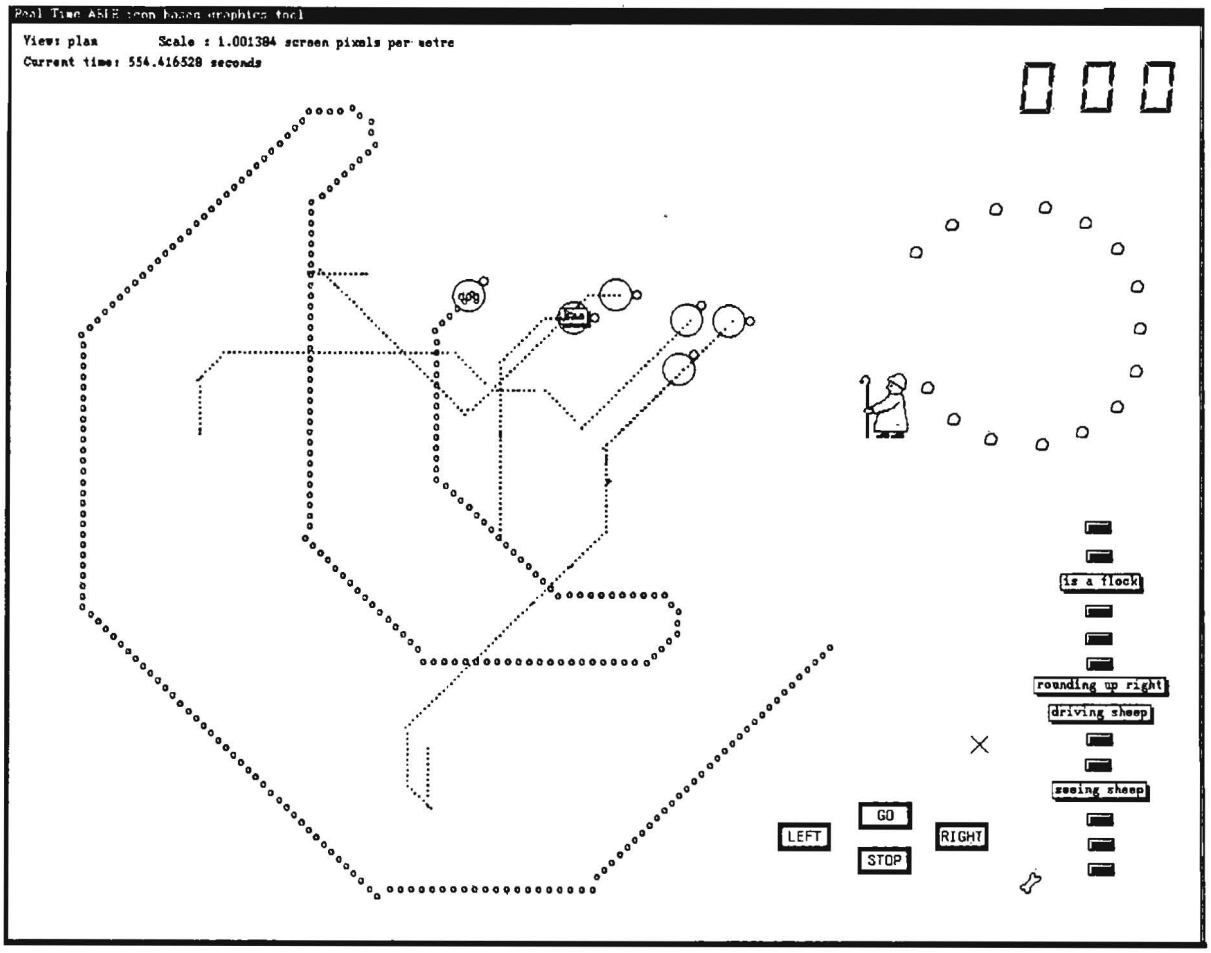


Figure 4: The dog rounding up and driving the sheep

The behaviour *drive_sheep* corresponds to the dog's ability to drive the sheep towards the shepherd. This is built compositionally from two complementary behaviours, *round_up_sheep_left* and *round_up_sheep_right*. The overall operation of driving sheep is to *round_up_sheep_left* until the dog is facing the shepherd on one side of the flock, and then *round_up_sheep_right* until the dog is facing the shepherd on the other side of the flock. These behaviours take place alternately, with the dog weaving to and fro behind the flock, so that the flock as a whole is driven towards the shepherd. The coherence of the flock during *drive_sheep* behaviour is derived from the *round_up_sheep_left* and *round_up_sheep_right* behaviours, both of which have the effect of making the sheep gather into a flock.

The two rounding up behaviours are emergent (in the sense used in this paper) from the behaviours of the individual agents. The sheep run away from the dog when it gets too close, and the dog, by running round the flock in circles, tends to drive the sheep towards each other. The dog's circling behaviour is very simple: in *round_up_sheep_left* behaviour it changes direction so as to keep its front visual sector free of sheep and its front left visual sector showing sheep. The resulting emergent behaviour of the sheep, that they tend to gather together into a flock, is very stable, but depends on parameters of the basic

behaviour of both the dog and the sheep, for example their relative speeds and certain timing considerations.

In this particular case, it is a single agent, the dog, who initiates and maintains these behaviours, so they are represented symbolically as behaviours of the dog. These behaviours produce relatively low level behaviours of the dog which are linked (by causal processes in the world) to similarly low-level behaviours of the sheep. The behaviour of gathering together into a flock is emergent from all of these low-level behaviours.

Just as the maintenance of the symbolic behaviour *trapped* to conform with the observable being-trapped behaviour was viewed as perception, the way the real rounding-up behaviour is produced in accordance with the symbolic *rounding_up_sheep_left* behaviour can be viewed as action. But whereas traditional accounts of action deal with changes of state of objects in the environment of the agent, action in this case involves a collective activity of a group of agents which is sustained by the behaviour of the individual agents. During this activity, a new entity (the flock of sheep) comes into being whose existence is maintained by the activity of the agents. The agent is then further able to act on this emergent entity by driving it across the field.

An alternative way of viewing the behaviour of the dog is to ascribe to it the *belief* that it is trapped or the *objective* of moving the sheep towards the shepherd. The internal state of the dog contains components corresponding to this, namely the *trapped* and *drive_sheep* behaviours. The dog is not however designed to conform with any logic of beliefs and action. Relations between components of its internal state are determined by the causal processes which maintain its internal behaviour representations consistent with real behaviours in the world. It is interesting that this process results in behaviour which is naturally describable in intentional terms, and we hope to investigate the way that intentional behaviour can be made to emerge from concrete behaviour in future work.

6. Conclusions

The main point of this paper is to describe a practical methodology for building agents out of emergent behaviour centred around the notion of maintaining symbolic representations of the emergent behaviour. This methodology has been tested by developing a simple multi-agent demonstrator.

The discussion in this paper has concentrated mainly on an individual agent embedded in a multi-agent world. A natural extension to this work would be to consider groups of agents such that the individual agents all have internal symbolic representations of the emergent behaviours of the group. This would appear to provide a good basis for supporting cooperative behaviour.

Acknowledgement

This work was done in collaboration with David Connah, Michael Graham and Steve Hickman.

References

[Agre & Chapman, 1987]
[Philip E. Agre and David Chapman, "Pengi: an Implementation of a Theory of Activity", Proc. 6th National Conference on Artificial Intelligence, Morgan Kaufmann, Los Altos, CA, 1987.

[Brooks, 1985]
Rodney A. Brooks, "A Robust Layered Control System for a Mobile Robots", Journal of Robotics and Automation, Volume RA-2, Number 1.

[Brooks, 1989]
Rodney A. Brooks, "A robot that walks: Emergent behavior from a carefully evolved network", Neural Computation I (2), 1989.

[Connah & Wavish, 1990]
D.M. Connah and P.R. Wavish, "An Experiment in Cooperation", Proceedings 1st European Workshop on Modeling an Autonomous Agent in a Multi-Agent World, ed. Y. Demazeau and J-P Muller, Elsevier Science Publishers B.V. (North-Holland), Spring 1990.

[Connell, 1989]
Jonathan Connell, "A Colony Architecture for an Artificial Creature', MIT Artificial Intelligence Laboratory Technical Report No. AI-TR 1151, August 1989.

[Graham & Wavish, 1991]
Michael Graham and Peter Wavish, "Simulating and Implementing Agents and Multiple Agent Systems", Proceedings of the 1991 European Simulation Multi-Conference, Copenhagen, June 1991.

[Hickman & Shiels, 1991]
S.J. Hickman and M.A. Shiels, "Situated Action as a Basis for Cooperation", in Decentralised Artificial Intelligence 2, Proceedings of the 2nd European Workshop on Modelling Autonomous Agents in a Multi-Agent World, Paris, 1991.

[Maes & Brooks, 1990]
Pattie Maes and Rodney A. Brooks, "Learning to Coordinate Behaviors", Proc. 8th National Conference on Artificial Intelligence, AAAI Press / The MIT Press, 1990.

[Neisser, 1976]
Ulric Neisser, "Cognition and Reality", W.H.Freeman & Co., San Francisco, 1976.

[Rosenschein & Kaelbling, 1986]
Stanley J. Rosenschein and Leslie Pack Kaelbling, "The synthesis of digital machines with provable epistemic properties", in Proceedings of the Conference on Theoretical Aspects of Reasoning About Knowledge, ed. Joseph Halpern, Monterey, CA, 1986.

[Steels, 1990]
Luc Steels, "Cooperation between Distributed Agents through Self-Organisation", Proceedings of the 1st European Workshop on Modelling Autonomous Agents in a Multi-Agent World, ed. Y. Demazeau and J-P Muller, Elsevier Science Publishers B.V. (North-Holland), Spring 1990.

[Steels, 1991]
Luc Steels, "Towards a Theory of Emergent Functionality", in "From Animals to Animats", Proceedings of the 1st International Conference on Simulation of Adaptive Behaviour, ed. Jean-Arcady Meyer and Stewart W. Wlison, Bradford Books, 1991, pp. 451-461.

[Suchman, 1987]
Lucy Suchman, "Plans and Situated Actions: The Problem of Human-Machine Communication", Cambridge University Press, 1987.

[Wavish, 1991]
P.R. Wavish, "Real Time ABLE", Philips Research Laboratories Annual Review 1990, 1991.

[Wavish & Connah, 1990]
P.R. Wavish and D.M. Connah, "Representing Multi-agent Worlds in ABLE", PRL Technical Note No. 2964, October 1990.

A Distributed Artificial Intelligence View on General Purpose Vision Systems

Olivier Boissier Yves Demazeau

Laboratoire LIFIA/IMAG - 46 av Félix Viallet
F-38031 Grenoble Cx - FRANCE

tel +33 76574745
fax +33 76574602
boissier@lifia.imag.fr

tel +33 76574604
fax +33 76574602
demazeau@lifia.imag.fr

Abstract

A General Purpose Vision System (GPVS) is an open and domain-independent system that is able to build, maintain, and use an internal representation of the external world from data provided by physical sensors such as cameras. The experience in constructing the *Vision As Process* (VAP) and SATURNE systems at LIFIA using a Distributed Artificial Intelligence (DAI) approach has enabled us to compare, from a DAI point of view, these two systems with other major existing GPVS's. This work represents the first large-scale comparison between GPVS's at the module and module-integration levels using a DAI formalism. This has lead us to the identification of several basic common features within the systems studied. Furthermore we think these common features are essential for the construction of a GPVS, namely : 1/ the distribution of the knowledge representation provides *level-agents* that denote the same knowledge at a given level of representation 2/ the distribution of the knowledge processing provides *focus-agents* which realize the system's "focus of attention" 3/ intersection of level-agents and focus-agents determines the active *basic-agents* that explicitly or implicitly constitute the system at a given time. The description of GPVSs in terms of basic agents is a novel feature of our approach and may also be applicable to other domains such as robotics.

1. Introduction

A General Purpose Vision System (GPVS) is an open and domain-independent system that is able to build, maintain, and use an internal representation of the external world from data provided by physical sensors such as cameras. This representation is built in order for example to be used by other systems acting in the same external world but lacking in perceptual capabilities. Such systems may interact with the vision system through an exchange of data or of goals the vision system has to satisfy. Its internal activity is realized by the chaining of specific processes it contains, or in collaboration with other systems with which it interacts. The experience in constructing the *Vision As Process* (VAP) and SATURNE systems at LIFIA using a Distributed Artificial Intelligence (DAI) approach, has enabled us to compare, from a DAI point of view, these two systems with the major existing GPVS's. After the definition of the vocabulary and concepts we use in the field of DAI, we try to identify the main concepts involved in the building of a GPVS, and we propose a methodology to analyse the whole system, as well as its functioning. Then we will also show that the DAI approach allows a unified description of all GPVSs while from a computer vision perspective such systems are generally extremely difficult to compare. We finally use the DAI formalism in section 6 to describe one of the GPVSs developed at LIFIA, the VAP system (*Vision As Process*), built in the framework of the European ESPRIT Basic Research Project BRA3038.

2. Distributed Artificial Intelligence

The description of a system using a DAI formalism can be made according to two points of view [BOI 89] [DM 90] : 1) a *macro level* taking into account the building blocks of the system

as individuals having knowledge, limited resolution abilities and interacting together. At this level we consider the system as a *society of agents*. The degree of homogeneity of the society changes depending of the degree of homogeneity or heterogeneity of the agents that build it as well as the number of these agents (from few to many). 2) a *micro level* considering the system or each building block as a whole entity that has a problem to solve with its knowledge and its resolution abilities. In this case, we focalise our attention on the *agent*. The grain of the agents covers the range from coarse to fine.
In DAI all systems may be represented using these two levels of description. We will use the agent and society models figured in [DM 90]. Additionally we use concepts from current work carried out at Grenoble in the framework of the PLEIAD group [GDA 91].

2.1. Agent

Agents are the dynamic entities acting in the world. The effect of their actions is perceived in it by the production of events that correspond to modifications of the environment, and to communication acts. Agents can vary in complexity, from ants to robots. We mean by *agent* an entity that acts *rationally* and *intentionally* according to its own goals and to the current state of its knowledge. We say that agents are *autonomous* or exhibit intelligence if they are capable of flexibility and adaptiveness, of setting up their own goals based on their interests and of achieving these goals by efficient actions.

An agent can be split into two main parts. The first one, which is a static aspect, defines the architecture of the agent. This aspect, which commonly termed *knowledge representation*, deals with the definition of the types of knowledge available to the agent and how it is represented. The second aspect is the dynamic processing that takes place on the agent's architecture. We call this aspect the *processing methods*.

Static aspect

The type of knowledge an agent has which is relevant to its existence in a society may be divided into :

- explicit and abstract *world representation* in which the agent lives.
- *abilities* : what an agent can offer to others; topics of interest, what an agent is interested in, and its representation of other agents' abilities.
- explicit and abstract representation of the problems or *goals* that the agent has to solve,
- *plans* to be executed,
- choices or *decisions* taken.

Dynamic aspect

Processing methods are the way a dynamic aspect is added to this static structure, these may be divided into :

- *reasoning capabilities* including communication planning, detection of incoherences, integration (combination of data), use of data, reasoning on the others' knowledge and behaviours,
- *choice and decision making* mechanism, or decision capabilities.

The *control* consists of the transformation and the inclusion in an agent's processing methods of global constraints due to the society of agents. It also includes inherent constraints of the processing methods that are used within each agent.

2.2. Society of Agents

Taking the same way of description for the society the agents build, we have a static and dynamic aspects.

Static aspect

The society of agents is organized according to a network that can be heterarchical, hierarchical, or market-like. This exhibits *links* of communication that can be a priori defined between two agents. Basic *interactions* deal with what is exchanged using these links : knowledge, goals, plans or choices. Agents can know each others and exchange data namely - *direct communication* - or they can communicate without knowing each other by posting and taking data according to predefined characteristics - *indirect communication* -. These three features statically define the game rules, or what can be done inside the society.

Dynamic aspect

We have to define the manner and the moment to use the network, given the game rules that have been settled. Given what is effectively exchanged on the network - knowledge, goals, plans or choices - the links between agents are used either for simple communication or for control influence. Deduced from the content of the exchange and from the eventual protocol of communication that can take place between the agents, we can speak of cooperation, competition or cohabitation.
The control at this level deals with the settling and the regulation of the different data exchanges described above. It can be distributed over the agents in case of entire autonomous agents or may also be centralised somewhere in one agent.

3. General Purpose Vision Systems

The research dealing with the construction of a GPVS is generally concerned with : 1) design of explicit *models* of a visual problem domain, 2) *methods for extracting features* from a representation of the perceived scene (image, 2D description, 3D description for example), 3) *methods for matching* these features to models by using a suitable *control structure*. A GPVS has to be able to perform a non trivial set of visual tasks using a set of such models and methods in an imperfectly known environment. This definition leads us to require that such a system must be an *open system* in the sense that new functionalities to solve new kinds of problems can be easily added to it. Moreover it has to be a *domain-independent system* to be used in any environment.
In the late 70's, several proposals for General Vision architectures were reported [HR 78]. From this time onward few researchers have tried to bring together a large set of methods covering all the processing chain. Almost all research has been focused on smaller parts of the whole problem. Each time, in order to solve a particular subset of problems, it was assumed that solutions to other related subproblems were known. Few attempts were made to build domain-independent vision systems. Almost all systems were well suited for a specific domain, such as SPAM [MWH 87] for airport scenes or MESSIE [GGM 89] for aerial images.
More recently new efforts were made to integrate visual modules or processing methods in the context of a General Vision System. The first class of GPVSs are aimed at processing single static images from several types of image properties (SCHEMA [DCB 89], SATURNE [DEM 86] [DEM 90], SKIDS [ABH 89]). Others introduce in addition a *Focus Of Attention* dimension on the control of the perceptual process. Such systems incorporate several aspects : spatial, temporal, and semantic. Moreover an added dynamic dimension corresponding to the use of sequences of images, dealing with the tracking of objects, puts such systems in the trend of Active Vision [AWB 87]. The MEDUSA [ALO 90], Rochester system [CM 87][BAL 89][WB 90][BAL 91], Vision As Process (VAP) [CCE 89] [CG 90], are the main systems following such an approach.

4. Analysing General Purpose Vision Systems

The main problem of visual perception is the interpretation of an enormous amount of inherently ambiguous symbolic and numeric data. To simplify the representation it is natural to

define intermediate *levels of representation* between the information coming from the sensors and the final description of the scene in the context of the system's goals. On another hand the structuration of the whole processing taking place in a GPVS appears also necessary. Indeed such systems are implicitly structured by the global goals they have to satisfy and by the designers themselves. Finally, the functioning of GPVSs exhibit two main *behaviours*. The first is a *preattentive* or *understanding* behaviour consisting of a continuous and permanent activity driven by a set of intrinsic goals. The second behaviour, *resolutive* or *recognition* , is much more discrete and punctual since it is driven by the satisfaction of extrinsic goals.
The current trend in the construction of GPVSs consists of using general principles from the design of complex systems, namely to divide the system into different components. The splitting use the two structurations presented above. The resulting components are then regrouped or integrated inside the system by the addition of a communication language and an interaction protocol. An *horizontal splitting* enables us 1/ to structure the levels of representation, 2/ to identify the basic transformations between levels and 3/ to install both the preattentive and resolutive behaviours. A further *vertical splitting* enables 1/ the definition of the focus of attention of the system and to predefine privileged vertical links within the system, 2/ the identification of the basic agents of the system and 3/ the envisionment of a common internal structure of these standard agents. These points will be discussed in more details in the following sections.

4.1. Horizontal Splitting into levels of representation (level-agents)

The levels inherited by the horizontal splitting are those that were clearly identified in the 80's [HR 78]. A great deal of work has been done on the several proposed different horizontal splittings. However from the point of view of the knowledge representation, there is still a strong need to discover what are the most suitable representations at a given level. This is particularly the case at the higher levels. Setting this horizontal splitting on a GPVS defines different entities in the system that we call *level agents*.
The different levels of representation are used in GPVSs for organizing the large amount of information with which they are faced. The informations - data and goals - are transformed and then gathered in subsets according to several criteria : degree of abstraction on the shape (*abstraction*), expression in different spatial and temporal systems of reference (*decentration*) [MAR 82] [DEM 86]. Abstraction is a classical notion that we will not discuss here [HR 78] [MAR 82]. Decentration has two aspects : a spatial and a temporal one. The spatial aspect consists of expressing the features at each level according to different points of view : viewer-centered, image-centered, object-centered or scene-centered. These points of view are used because of their suitability for the processing taking place on the representation in this reference frame. The temporal aspect deals with how features evolve with time. As we go up in levels of representation - with higher decentration - this sensitivity to time decreases. For instance, the edges belonging to an object observed in the scene exhibit a high degree of temporal instability. However the edge information at a given time maybe sufficient to maintain the object hypothesis within the system. As a consequence of these definitions, decentration is made up of only the spatial aspect in systems that do not include the maintenance of the scene representation over time.

4.2. Vertical Splitting according to Foci of Attention (focus-agents)

Despite of this first structuration, it now appears that the obtained level-agents are too coarse-grained to be considered as basic processing units. There appears to be no general structure for each level agent that could actively produce the entire description of the scene at a given level. As a consequence, there is a even stronger need of structuring again these level-agents, splitting them into several subparts. We propose here some key concepts for both the inter comparison of GPVSs as well as for analysing them namely, the *Focus of Attention* dimension. In fact, a GPVS is implicitly vertically structured by the global goal it has to satisfy and by the designer himself. That is to say, at a given moment, the processing of the system is focused either on a

location, on an object, on a task, or on a feature - the focus of attention -. More drastically, we can observe that this focalisation of the processing exists at each level of representation on which the system is distributed. This leads to an observable vertical splitting of a GPVS into groups according to a focus of attention at a given moment. These resulting groups can physically exist in the system such as SATURNE , or can be dynamically defined by grouping the different activated subparts of the system at a given moment along the levels of representation, as for example in the VAP system. We call these groups the *focus-agents*.

4.3. The society of Agents : the static aspect

Basic Agents : The horizontal splitting provides *level-agents* that denote the same knowledge at a given level of representation. The vertical splitting provides *focus-agents* that denote the same focus of attention through the levels. The intersection of these two kinds of agents defines the active basic units that constitute the system at a given time. These are what we call the *basic agents*.

Network : Both splittings define too the different links of interactions between the basic agents. This defines what we call the interaction network. In GPVSs we have mainly two kinds of links : horizontal ones that allows the basic agents to exchange informations within the same level of representation and vertical ones that links basic agents belonging to the same focus agent. This latter kind of link favours obviously control influences inside the society between basic agents because of the focus of attention attached to the concept of focus agent. The levels of representation are the main interaction media of the society.

4.4. The society of agents : Dynamic aspect

Basic Interactions : Among the different informations that are manipulated by a GPVS, we can distinguish two subsets :

- informations used on the *same* level, concerned with the *enrichment* aspect if produced inside a basic agent. They can give rise also to interactions that we call *communication* if they resulted from an exchange between basic agents inside the same level agent.
- informations leading to the production of new data on *an other* level. This is the result of either, a *perceptual* act, interaction between basic agents that do not belong to the same level agent, or an *inference* that is internal to a basic agent.

Vertical Knowledge Interchange Protocols : We will illustrate the knowledge interchange protocols only with the perception aspect. However, we have to take in mind that a knowledge interchange protocol also exists for the communication aspect. This intra-level protocol is currently poorly studied. For instance it exists in the SCHEMA system where objects' schemas within the same high level of representation communicate through a blackboard. This protocol is less well defined than the vertical one dealing with the perception. The process of perception has been for a long time considered as the skilled combination of two functions : *segmentation* and *interpretation* acting on the lower and higher levels respectively. This combination was realised by a feedback from the high level processes to the lower levels. Representation levels were refined as the *predict and verify* processing mode from one level to another [LUX 85]. It can be also formulated as : given a model in a representation at one level and data in an other representation at another level, is there an instance of the model in this latter representation. The *prediction* phase corresponds to the transformation of the data issued from a representation into more abstract data representation. The *verification* phase consists of using knowledge or models of this more abstract representation in order to match it with the incoming data. This phase can also trigger a feedback to the lower level in order to confirm the hypotheses. This feedback takes place after the transformation of the verification data in the lower levels of representation (*projection* step).

On this basic cycle takes place a an other interaction protocol that uses it. Tsotsos [TSO 89, TSO 90] defines the visual search taking place in the visual processing, as having two aspects : 1) a *bottom-up* - data driven - that consists of transition from lower levels to upper, 2) a *top-down* - goal-directed - corresponding to transitions from upper to lower levels. In the bottom-up case, the goals are either unknown in advance either known but not used except to determine when the search ends. The top-down case makes use of goals to assist in optimizing the solution to the problem. In fact, depending on the complexity of the task, the large amount of manipulated data, an exclusively data driven resolution could lead to a combinatorial explosion. This mixture between both modes of internal functioning - bottom-up and top-down - has led to the emergence of many resolution cycles depending on the extend of this mixture [TSO 87][RJ 88]. These are for example the perceptual cycle by Crowley [CRO-90] (figure 1) or Kanade [KAN 80] that explicitly presents the two main levels used (figure 2). For an excellent review of the different cycles the reader is reffered to [TSO 87] or [LUX 85]. This wide range of cycles in GPVSs shows that the structuring of the processing of a GPVS is quite a difficult problem.

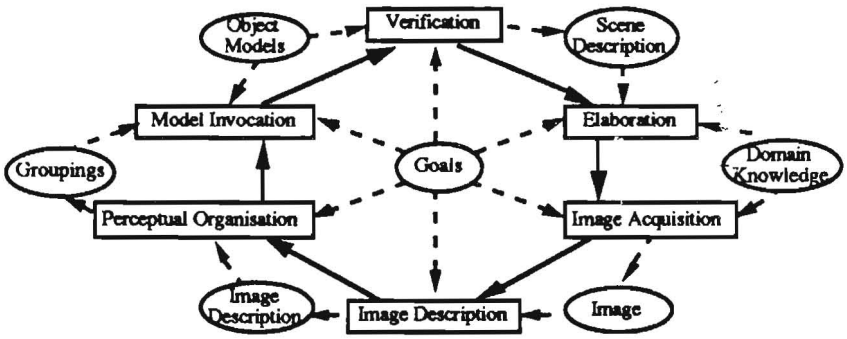


Figure 1 "The VAP cycle [CRO 90]"

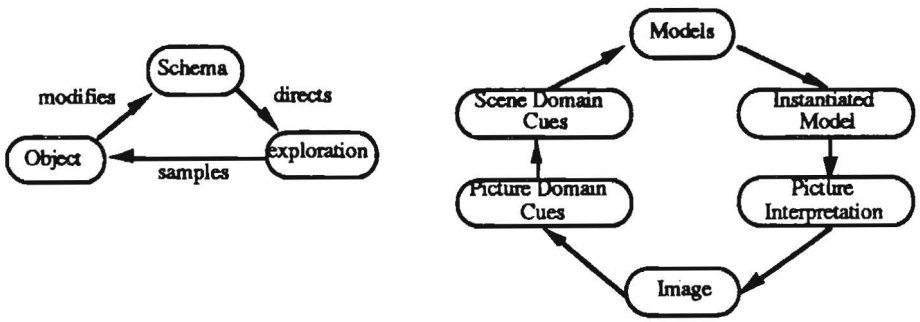


Figure 2 "The Perceptual Cycle from [RJ 88] and Kanade's cycle [KAN 80]"

Behaviour of the society

The global processing that takes place within a GPVS consists of a cycle of a close interaction between two main behaviours : the resolutive and preattentive ones.

Resolutive behaviour : The *resolutive* behaviour is a discrete and punctual functioning of the GPVS. It is driven by the satisfaction of extrinsic goals, such as the answering to a request coming from a robot arm to locate some particular object. As far as the resolutive behaviour of

the system is concerned, the vision task can be described as a search one : look for an instance of one model in one set of data.

Preattentive behaviour : However, a GPVS must be open to the external world, it is observing. So it must permanently pay attention to the external events or data in a *preattentive* mode. This second main behaviour of a GPVS consists of a continuous and permanent activity driven by a set of intrinsic goals. Here again, the kind of unexpected events or data to which the system is sensitive can either be explicitly programmed inside the system either result from the data, satisfying some implicit goal such as *understand.*

Mixed behaviour : A GPVS must be adaptative and flexible, allowing it to reconcile these two types of behaviours. Even though the resolutive behaviour seems to be much more top-down while the preattentive behaviour seems to be more bottom-up. As a consequence each basic agent will have the task of either resolving a particular subproblem, resolutive aspect, either to execute their own processing independently of any external goal, preattentive mode. Both behaviours are implemented using the predict and verify loop between the several levels of representation that are addressed while solving the current problem. They are implemented in addition with a combination of top-down and bottom-up processing.

4.5. The agent : static and dynamic aspects

To enable such behaviours, GPVSs have to incorporate in their basic agents different functionalities as well as several kinds of knowledge, analytical, geometrical and physical. In addition to such knowledge of the real environment that is classically found in such systems, temporal, relational, functional and communicational knowledge of the behaviours of the basic agents within the systems has also to be included. These last knowledge sources come directly from multi-agent studies. Basic agent's internal structures are specific to each GPVS. So, no standard description of such a structure is given in this paper. As an example, in the last section, we will describe the VAP system in more details providing such kind of a description.

5. Review of General Purpose Vision Systems with these Analysis Principles

5.1. The SCHEMA system

The Image Understanding Architecture [WLH 87] (figure 3) is a general Vision System whose goal is the recognition and the localisation of objects in the scene together with their relationships. This system uses an architecture the principle of which lies in the decomposition of the vision process into different layers and processes that are hierarchically organized. Physically the system is organized as three sets of processors that are respectively dedicated to : image processing, manipulation of events and features extracted from the image, and symbolic processing. This system is specialized in static scenes and does not incorporate the means to control its sensors.

Horizontal splitting : The SCHEMA system does not have an explicit definition of levels. The levels are continuous in the sense that they depend on the amount of detail used, and on the knowledge organization needed inside the object model Schema by the different processing methods. Therefore the number of levels in the system depends on the object model that is considered. Nevertheless on a large scale three levels may be distinguished in this system : image, intermediate level, scene description - equivalent to the scene interpretation of the VAP system -. The intermediate level has a variable extension : it embodies the levels between image and symbolic scene interpretation in the VAP System.

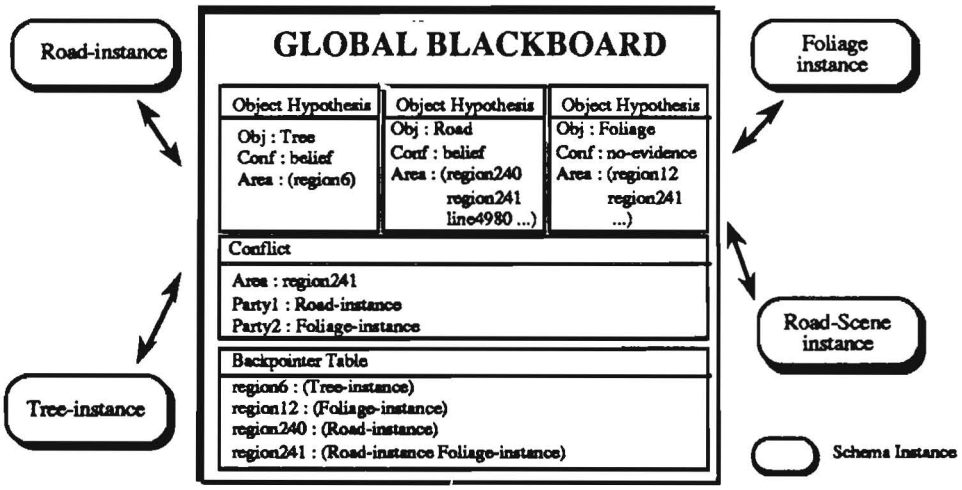


Figure 3 "The SCHEMA System [DCB 89]"

Vertical Splitting : In the SCHEMA system, the focus-agents are issued from a *Object model* splitting (figure 10). There exists dynamically a focus-agent for each object that can exist in the scene. The representation levels are not explicitly taken into account. A focus-agent corresponds to a processing cone covering all the representation levels. This cone is dynamically generated by the use of inter-level processing methods communicating between them inside the cone on the representation level they need.

Basic Agents : As previously mentioned, the basic agent of this system are the object models organized within a blackboard.

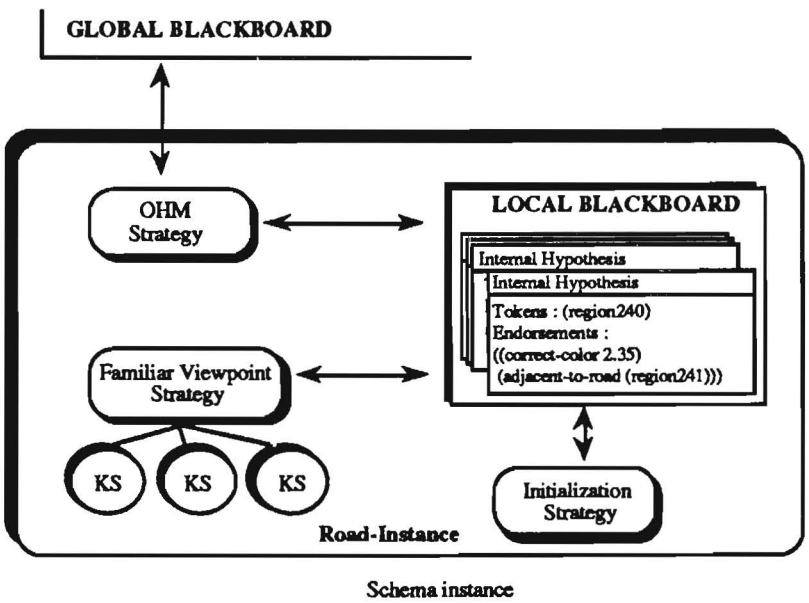


Figure 4 "Agent Model of the SCHEMA System [DCB 89]"

Behaviours : In the SCHEMA system, the functioning mode is resolutive. The system is initiated with an hypothesis that the different agents have to verify. The SCHEMA system can also be described as a Distributed Problem Solving system in the sense that each agent is built to satisfy a particular goal.

Each time an hypothesis on the existence of an entity has to be made, the Schema dedicated to the interpretation of this entity is triggered. The architecture used is a Blackboard in which the triggered Schemas interact according to two modes : cooperation and competition.
The processing methods building a focus-agent do not communicate directly with the processing methods of an other focus-agent. The only exchange of data that takes place is on the higher level through the blackboard between the basic agents.

Internal Structure : Within the SCHEMA system, one agent is a specialized vision system that can recognize one type of entity. Several general vision processing methods coexist in the system. A set of strategies makes it possible to apply or to choose the Knowledge Source that has to be activated according to the conditions in which the interpretation of the entity has to take place. The internal activity of each agent as well as each focus-agent includes the prediction-verification cycle (figure 4).
At the highest level the deductive knowledge is organized in Knowledge Sources or *Schemas* each consisting of one or more dedicated strategies relevant to the interpretation of particular entity such as "tree", "road", "foliage". This modularity principle is inherited from the incremental development of the Knowledge Base [DCB 87]. On the other levels there are procedures called by the different Schemas.

5.2. The SATURNE system

The main objectives of the SATURNE system, currently under construction, is the development of the notion of levels of representation based on the two principles abstraction and decentration. This system is mainly composed of a number of *Shape-from methods* that are able to communicate with each other and with other intelligent agents (figure 5). The current limitation of the system lies in the fact that it is conceived to ensure an instantaneous passive understanding without taking into account the temporal dimension. The architecture itself is independent of the Computer Vision application field since it is suited to the integration of other robotics or AI modules.

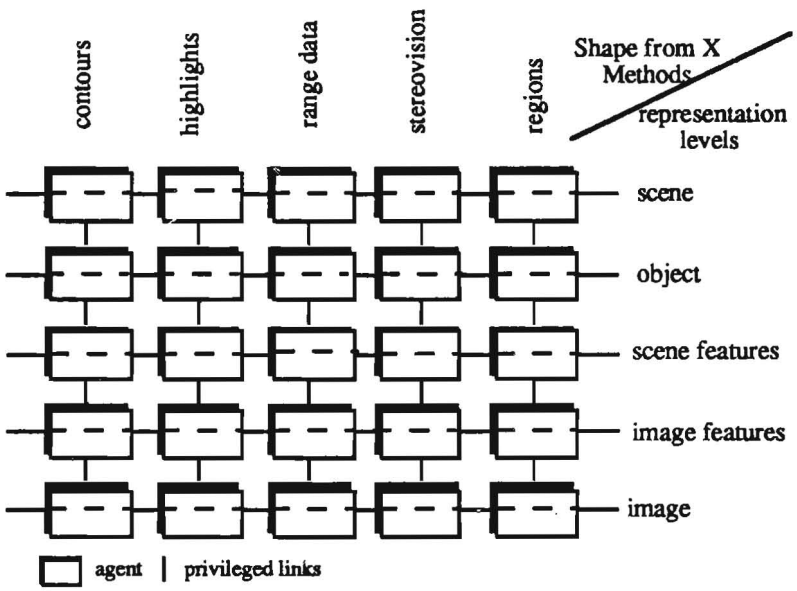


Figure 5 "The SATURNE System"

Horizontal splitting : The SATURNE system has explicit levels of representation on which it makes use of a distributed representation. In the SATURNE system, we have five levels that

are Image, Image Features, Scene Features, Object and Scene. This structuration is mostly inspired by D. Marr's levels [MAR 82] : Image, Primal sketch, 2,5 sketch, 3D Model.

Vertical Splitting : The SATURNE System uses an explicit vertical splitting based on the features (contours, highlights, shading, texture, motion, stereo) on which the system can be focussed. This decomposition is due to Marr [MAR 82]. Each focus-agent includes the basic agents that are specialized in the processing of the same feature (figure 10). Focus-agents can be explicitly represented and constitute the built-in shape-from methods in the system. The currently available focus-agents include : shape-from contours, shape-from highlights, shape-from range data and a grasping module. Other shape-from methods : shape-from shadows, shape-from shading, shape-from motion, shape-from texture, are scheduled for construction and integration. The action field of such an agent covers the set of representation available along a specific property of the image.

Basic Agents : A basic agent is defined as the intersection of a representation level and a Shape-from method. Moreover, they explicitly constitute the agents that compose the society at its finest grain. Every basic agent is specialized in representing and processing a certain kind of feature at a given level of representation.

Behaviours : The design and the basic functioning of this system - understanding : preattentive and permanent - is characteristic of a multi-agent system. The second behaviour - *recognition* : resolutive and punctual -, more directed by an extrinsic goal, is much more typical of a Distributed Problem Solving system. Recognition behaviour is an extension and a particular use of the multi-agent architecture through the implementation of particular decision capabilities for each of the shape-from modules.
Perception is provided by basic agents of the same focus-agent while communication is available with agents of the same level-agent. Links of communication between the basic agents are dynamically set according to some instantaneous goal provided by an external agent. It is hoped that the system can be used in any environment, under any condition which results would not be affected, but the global functioning of the system, using general shape-from methods as focus-agents for all the goals may be hard to adjust especially if we want it to be robust.

Internal Structure : The basic agents of the SATURNE system are called intentional ones (figure 6) [DM 90]. At a given level, the agent has a representations of the world called its *knowledge* . This knowledge can be inherent or acquired through *perception* (at a lower level) or *communication* (at the same level) with other agents. To communicate with a lower or higher level, the agent transforms its representation into the target representation. Each reception of information is assumed to be communicated to the receiver at its level of representation. *Goals* are abstracted from the observation of the behaviour of the agent. These goals do not need to exist explicitly within the agent. In the context of its knowledge and goals, an agent can be thought of as having to consider a set of *possible solutions* if the goal is to solve a problem. An agent does not need to be able to derive all the possible solutions but only a part of these depending on its *reasoning capabilities*. When various possible solutions are potentially applicable, a decision must be made among them to choose the best one - the choice - from the point of view of the agent.

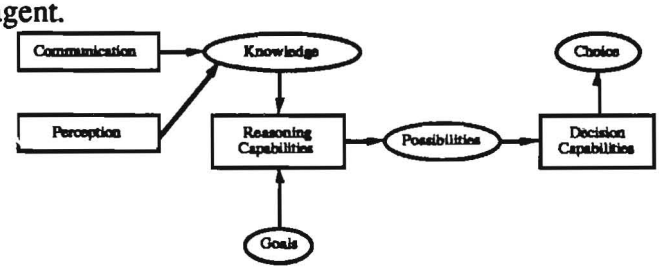


Figure 6 "Model of an Agent in SATURNE"

5.3. The "Vision As Process" system

The aim of the Vision As Process project [CCE 89] is to investigate *focus of attention* techniques for the control of the perceptual process in an integrated vision system (figure 7). The major effort in this project is aimed at system integration. The characteristics of such a system are active sub-systems such as movable cameras operating in a dynamic environment under real time constraints. These characteristics have a major effect on the control of the system. The first version of this system, currently under construction, is composed of the following modules : Camera Control Unit, Image description processes operating at multiple resolutions, Processes for extracting 3D description of the scene from sequences of images, Process for dynamically maintaining a symbolic description of the scene using information from the other processes and a priori knowledge about the scene. All these modules were developed independently with the aim of integration using a skeleton, the SAVA system [DCB 90][BC 91]. The control of the system is the task of a supervisor module. The operating cycle reproduces the traditional approach taken in building a GPVS : close interaction between a given module and its direct successor.

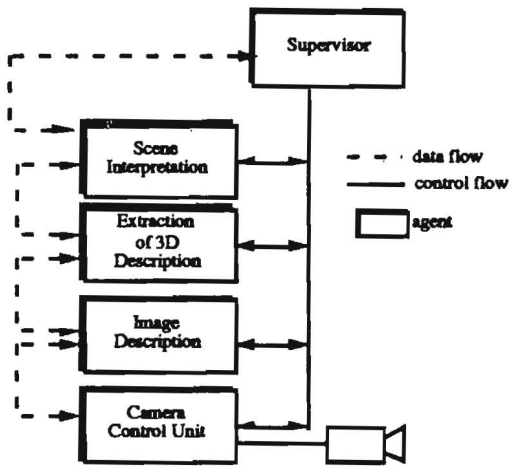


Figure 7 "The VAP system [CG 90]

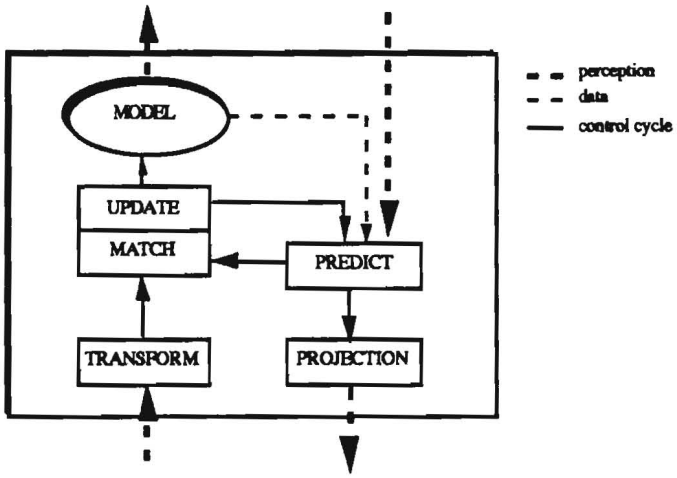


Figure 8 "The VAP Agent Model"

Horizontal splitting : The VAP systems uses explicit levels of representation. The four used levels are : image, 2D image description, 3D image description, Symbolic Scene Interpretation.

Vertical Splitting : A focus-agent is defined as being composed of the union of the processing units that transform a representation into a successive one for a given Region of Interest (ROI). The ROI is the spatial area, temporal slice and its semantic bucket - set of objects or features - within the focus agent processes and builds its representation (figure 10).

Basic Agents : At each moment of the processing a basic agent can be dynamically defined : this is the level-agent to which is affected the ROI expressed at this level of representation. In this way, we have on the same level a dynamic definition of several homogeneous basic agents corresponding each one to a different ROI. Currently, just one agent of this kind is active at a given time on a given level. We envision a parallel processing mode at each representation level in which several ROIs will be defined. This feature may for example be used for the tracking of features or objects in the scene as in [CKB 90] or [TM 89].

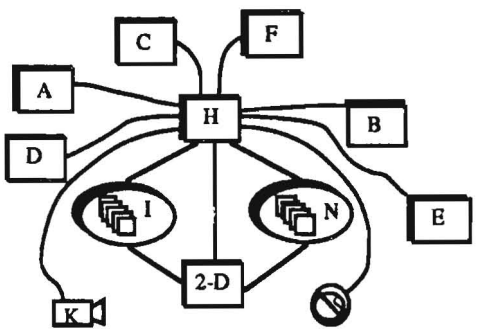
Behaviours : The system exhibit mainly a resolutive behaviour : its activity is directed towards the satisfaction of goals given to the supervisor. This mode of functioning is illustrated in figure 1 using the cycle described in [CRO 89].

No communication between basic-agents is currently settled because of the impossibility of having more than one basic-agent at the same time on the same level of representation. In some way, the communication between basic-agents occurs when it integrates results provided by a previous Region of Interest, stored on the level of representation. The perception aspect is the major mode of interaction between basic agents within the same focus agent.

Internal Structure : In VAP, the functioning of the system is based on the cycle Match-Update-Predict added to the internal cycle inherited from the Prediction and Verify [LUX 85]. It incorporates also a temporal prediction that enables the system to integrate its results over time (figure 8). At the input level of the agent the transformation function makes it possible to go from one representation to another at a higher level. At the output level the verification phase is accompanied by a projection function that makes it possible to go from one representation to another one at a lower level. As a consequence, every agent is able to transform the representation at a lower level into another one at its level and reciprocally.

5.4. The MEDUSA system

MEDUSA is an active vision system (figure 9) which is built with an active camera system, inertial sensors, a hand which is visible from the camera and a sub-system that allows it to move around in the environment. This system works on multiple images accumulating them while moving. There are two main data structures used by every component of the system : images and normal flow fields extracted from the images. A first module is dedicated to the task of the extraction of normal flow fields from the sequences of images. A second module of the system is a central controller which has a global view of the resolution, and controls all the activation and execution of the other modules within the system. Apart from these two modules the other components of MEDUSA are task dedicated. For instance, one module is able to determine if something is moving independently of the system and is able to locate it. The task of another module will be to detect objects getting closer to the system.



2-D : From the series of images finds a series of normal flow fields
A : answers the following question for Medusa : Is there anything moving independently of me ? Yes or No ? And if Yes is it in the image ?
B : Answers the following question : Is this moving object getting closer of me ? Or which places in the image correspond to parts of the scene which I am getting closer to ?
C : performs the task of keeping the moving object in the center of the visual filed by appropriately rotating the eye of Medusa.
...
I : Images
N : Normal Flow Fields
K : Active Camera
S : Inertial Sensor
H : the Head of Medusa
Agent

Figure 9 "The MEDUSA System [ALO 90]"

Horizontal splitting : The MEDUSA system seems to lack of homogeneous number of levels. Levels of representation seems to be gathered inside modules and depend also on the

need of the different methods used inside each one. This description could be incomplete or erroneous due to the partial information we currently have on this system.

Vertical Splitting : Aloimonos considers vision problems as being composed of two particular tasks that interact without being completely independant of each other : localisation and description. This dichotomy is developed by introducing the purposive vision concept [AS 89, ALO 90] : the vision system has to be cut in different modules according to the basics goals the system has to satisfy. Thus, the MEDUSA system is vertically splitted along *Task* focus of attention criteria (figure 10). A focus-agent in Medusa is a particular instanciation of shape-from method to satisfy a dedicated task.

Basic Agents : Basic-agents are the result of intersection of task-dedicated focus-agents and level-agents. However, focus-agents are built with a fixed set of basic agents that are linked rigidly.

Behaviour : The aim of such a system is to organize its processing methods along the different tasks it is able to process such as obstacle detection or object tracking. The behaviour of the system seems to be able to include both modes, resolutive and preattentive, event though the purposive vision concepts tends to use it in a resolutive way.
In this centralized system, no communication takes place between modules. Only the controller can communicate with the others : it is the main channel of communication allowing it to have a global view of the state of the system and a control on its evolution.

Internal Structure : As previously said for MEDUSA, several processing focus-agents are built in MEDUSA by using the task similarity criteria to merge the different processing units. The links between the units is fixed. It is not clear if these processing units are duplicated into the several focus-agents or if they are shared, defining by this way some agents. The lack of information does not able us to tell about the internal structure of the basic units.

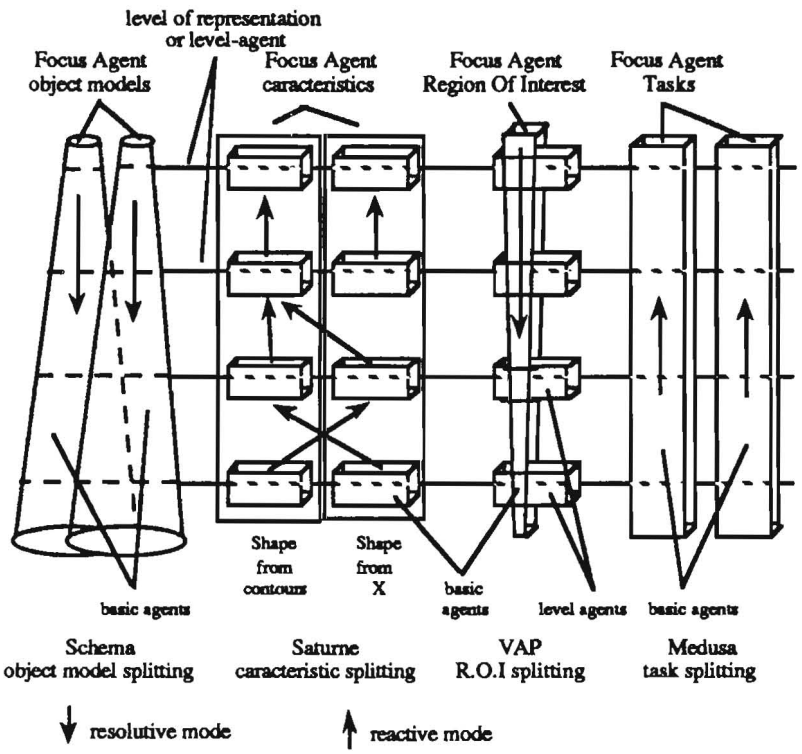


Figure 10 "Vertical Splitting of the GPVS"

6. A DAI Description of the VAP System

We are currently finishing the first step of the "Vision As Process" Project. In this section, we will describe what has been realized. We will discuss also some of the extensions that are planned to upgrade the system in the light of the first realized experiments. The VAP system is currently built with five heterogeneous modules, plugged inside the same system. Their integration is made in order to allow the system to actively work in a dynamic environment. We have set an additional constraint on this system, that is to have loosely coupled agents. We also integrate an other parameter related to the movement of the sensors themselves or due to events occuring in the scene. Aloimonos [AWB 87] showed that this additional parameter can increase the robustness of the vision methods. It can be used for example to decrease the uncertainty of an interpretation or to constrain problems. However, it has also some drawbacks and especially for the movement of sensors, it introduces an additional control parameter : the system has to decide where to look and to execute the move of the sensors.
The VAP system is written in C and Lucid Common Lisp 4.1. We have developed the framework in which agents are plugged in C under Sun OS Version 4.1. This communication skeleton SAVA [DCB 90] allows the agents to run on different machines that are Sun 3/260 or Sun 4 Sparc Station by using the Socket mechanism in Unix BSD 4.3. The Man Machine Interfaces are written in X Windows v 11R4 using the MOTIF programming system.

6.1. The agents

As defined above, a basic agent is a particular instanciation of a level agent with a region of interest. So It is a group of processing methods that acquire and improve the scene description on a given level of representation. The level-agents of our system are (figure 7)

- *Camera Control Unit* : execution of the moves, the focus, vergence and zoom of the camera
- *2D Image description* : build the 2D description of the image in terms of edges.
- *3D scene description* : maintenance and building of a three dimensional object models.
- *interpretation* : building of a symbolic scene description
- *supervision* : control of the system according to goals fixed by the user.

Static Aspect

Each agent has a different representation - understanding - of the same phenomenon that constitutes the scene itself. Each agent translates a perception of the scene in the representation at one level. The agent will then send it to an other agent. Depending of the agent, the representation is built from models of primitives of the domain such as edges, perceptual groupings, or geons [BIE 85]. These models represent prototypical concepts of the domain that have to be instanciated by the data sensed by the agent. Concepts are organised along a specialization hierarchy and a composition one [TSO 87]. The aim of an agent's representation is to describe the perceived world. As a consequence, we have spatial, temporal and functional links between the different primitives used by the agent.

The knowledge present in each agent can be regrouped in three classes :

- the *model* : information produced by the agent itself. It is the scene description at the representation level on which the agent acts. Relationships are linking the objects together.
- the *perceptual data* : information belonging to an other agent's model and sent to the agent. They are expressed in the sender's representation. They are used by the agent to build its own representation.
- the *control data* : These are parameters set defining the behaviour of the agent. A main control parameter consists of the *Region Of Interest* - the spatial area, temporal slice and the semantic bucket, in which to process -. These parameters are also thresholds for the processing, timeouts and so on. The control data are splitted in two subsets : one dealing with the *internal* behaviour constraining the internal processing. In this subset we find the way to

add dynamically new tasks to the agent according to different goals during a certain number of processing cycles. The other subset gathers all the parameters defining the *external* behaviour of the agent in the society, especially its sensitivity to the exchange of messages with other agents. This latter aspect allows to set a variable bandwidth of communication for each agent. This will be done in order to study the importance and impact of communication on the internal processing. Currently we are also studying the way to add to one agent : 1) knowledge about the other agents, 2) capabilities of the agent such as to provide images, particular features, objects and 3) field of interest of one agent. This latter aspect deals with the explicit expression of the agent's needs. Further work has to be done to explicitly express the interests and competences of an agent in order to allow some adaptability of the behaviour.
Control parameters are not defined only by default in the agent itself, but may be settled also by agents from the upper levels.

Dynamic Aspect

The prediction-verification cycle is present in each agent with a more or less degree of explicitness. The prediction phase produces hypotheses through a transformation operation on the perceived data. The *verification* phase tries to verify constraints directly on these hypotheses or to search for further perceived data to confirm them. This leads to the generation of goals for other agents consisting of requests for finding or verifying the existence of hypotheses in their model.
Our system is able to act in a dynamic environment. So it has to handle data evolving in time. This is done by defining in each agent a cycle allowing to do temporal prediction on the hypotheses and to update its model with these new hypotheses. The old ones in the model which predicted position matches the new ones allow to track the different features in the representation and to maintain the description of the scene in the agent.

The basic steps of the cycle added on the prediction-verification set are :

- *match* : matching of the predictions with the new incoming data.
- *update* : new data in the model are added and their confidence factor is updated. When this confidence factor is to low, the data is removed from the model in order to keep the size of the model reasonable. This constraint of keeping always a small amount of data to process comes from the realtime aspect of our project.
- *predict* : use of a temporal behaviour model of the data in order to do a prediction on their future location.

Each agent has an incomplete perception of the other agent because of the difference of processing time scale. This effect is due to the fact that each VAP basic agent belongs to a different level agent, having thus a time decentration on its representation.
On the figure (figure 8), the prediction step of the cycle and the verification step of the processing cycle are merged in the same box, identified by *predict*.. Communication of data involves a change in the levels of representation. So, there exist a *transformation* phase that translates data from a given level into understandable terms for the upper level. The inverse transformation is the *projection* phase.

Currently, the existing agents do not have a lot of explicit decision abilities. For instance, they are unable to change their Region Of Interest on their own. This change is driven by the state of the representation and by the goals of the system. But given this goal, the ROI is refined and changed according to the new representation. To modify its behaviour without setting an incoherent processing mode in the system, the agent needs to have some global idea of the resolution. We have seen that the verification on the data produced by an agent is made by the upper agent that have the models and knowledge to look for more data or detect incoherences.

6.2. The Society of Agents

Static Aspect

From the data point of view, everybody can communicate to everybody as soon as the agent is able to process the information it receives.
From the control point of view, the agents are organized along a hierarchy. The supervisor controls the highest modules, that is the symbolic interpretation module. Currently, the upper agent defines the ROI for a lower agent but also the different resolution parameters such as thresholds or constants for its internal processing.
Inspired by the preattentive mode of the SATURNE system, further work involves settling a heterarchical control so that the system will be much more suited to react to the events. We project to investigate the way to distribute the control in each agent, allowing it to define itself the reasoning parameters it needs for processing the data. By the already explicit representation of control data in one agent, this will lead to add some reasoning capabilities to reason on its own behaviour.

Dynamic Aspect

Communication acts are currently very primitive in the system. This is mainly due to the fact that the society is composed of heterogeneous agents that do have their specific well-defined tasks and that the society is hierarchically organised. As a consequence, we only defined a set of requests to allow agents to ask each other for some information having certain kinds of features. For instance, the 2-D description module can be asked to discover some perceptual groupings present in the scene in a given Region of Interest. This perceptual grouping could be to find orthogonal edges and which one of their extremities is located at the same place.

The communication acts is splitted in two parts :

• *data request* dealing with the data exchange. They allow the definition of instantaneous goals for an agent through the asking for data satisfying a set of constraints in its model. The process of answering gives rise either to a specific processing method in the agent (perceptual grouping for example), either to a simple interrogation of the internal model issued from its usual behaviour. The set of messages is :

• Find : looks for a particular item already identified, according to a list of matching parameters for the primitive. Answers by giving the list of identity of primitives feeting the matching parameters.

• Get : Given an id, it will return the set of parameters that defines the primitive.

• Verify : It will seek to match a specified primitive to the current contents of the model.

• Put : It allows an external agent to change the internal model of an agent by putting some primitives in it.

• *control request* dealing with the specification or adjustments of the control parameters of the other agent or requests for having information on it : Region of Interest, Parameters of Extraction, Processing Time, Behaviour. Such requests affects the decision capabilities of the agent as well as some of its reasoning capabilities. As an example, a behaviour request corresponds to the fact that one agent orders to another one to regularly communicate some kind of information at a certain frequency and during a certain while, without being further requested to do so. This installs some preattentive behaviour inside the agent. The set of messages is :

• Get : gets the value of the specified parameter.

• Set : Sets the specified parameter to the specified value

The current existing communication is a direct one. The success of the cooperation between these agents is strongly dependent from the communication protocol that is settled in the society. Further work is being performed to install a strong communication protocol. This will mainly be done in order to test several modes of cooperation.

7. Conclusion

In this paper, we have identified a number of important concepts involved in the construction of a GPVS. We have also proposed a methodology for the structuring of the whole system into standard modules, as well as for its functioning. This has been possible because of the existence of the VAP system that constitutes a testbed for experimenting with these concepts. The current release of VAP is not sufficiently open or flexible. The ideas expressed in this paper will contribute to redefine and improve both the VAP and SATURNE systems, particularly with respect to a common organisation and communication structure. Both systems are de facto quite different at the levels of the basic and focus agents. Nevertheless, we think the needs in communication of data and control requests are the same.
We believe that the DAI analysis of the currently on-going GPVSs projects we have presented is one of the first attempts to compare these systems using this approach. We will go on our work by analysing other systems such as the Rochester one [BAL 91] or Skids [ABH 89]. More generally, DAI can be used as a tool to analyze any complex systems. DAI is also a natural description for the conceiption of complex systems and we will keep on working on a general methodology for the construction of GPVSs. Computer Vision field is a particular application area for AI studies. Therefore we think that the concepts inherited from our Vision Problems, like the Focus of Attention, or Preattentive or Resolutive Behaviours, can probably be extended at the AI level to help the modelling of Autonomous Agents in a Multi-Agent World, independantly of the type of the agent.
In the light of the experience gained from developing the VAP and SATURNE systems, we are convinced that the DAI field approach is an excellent aid to both the integration of visual modules, and the study of control of perception in such systems.

8. Bibliography

[ABH 89] A. Ayoun, C. Bur, R. Havas, N. Touitou, J. M. Valade, "A Real Time Perception Architecture : the Skids Machine", in Multi Sensor Fusion and Environment modelling, Toulouse France, October 1989

[ALO 90] J. Y. Aloimonos, "Purposive and Qualitative Active Vision", Workshop on Active Vision European Conference on Computer Vision, April 1990

[AS 89] J. Y. Aloimonos, D. Shulman, "Integration of Visual Modules, an extension of the Marr Paradigm", Academic Press, 1989

[AWB 87] J. Y. Aloimonos, I. Weiss, A. Bandyopadhyay, "Active Vision", International Conference on Computer Vision, pp 35--54, 1987

[BAL 89] D.H. Ballard, "Reference Frames for Animate Vision", in IJCAI, pp 1635--1641, 1989

[BAL 91] D.H. Ballard, "Animate Vision", in Artifial Intelligence, february 1991.

[BBF 77] D.H. Ballard, C.H. Brown, J.A. Feldman, "An approach to Knowledge Directed Image Analysis", in IJCAI, pp 664--670, 1977

[BBF 78] D.H. Ballard, C.H. Brown, J.A. Feldman, "An approach to Knowledge Directed Image Analysis", in Computer Vision Systems, Hanson and Riseman Editors, 1978

[BC 91] S. Berthet, V. Caviggia, "SAVA 2.0", Technical Report, June 1991

[BIE 85] I. Biederman, "Human Image Understanding : Recent Research and a Theory", Computer Vision, Graphics, and Image Processing, 32, pp 29--73, 1985

[BOI 90] O. Boissier, La Coopération entre Systèmes à Base de Connaissances", Research Report, RR811-I-IMAG-96-LIFIA, LIFIA-IMAG, France, February 1990

[BRO 86] R. Brooks, "A robust Layered Control System for a mobile robot", IEEE journal of Robotics and Automation, Vol RA-2, 1, 1986

[CCE 89] J. L. Crowley, A. Chehikian, J.O. Eklundh, J. Kittler, J. Illingworth, G. Granlund, J. Wiklund, E. Granum, H. I. Christensen, "Technical Annex for ESPRIT Basic Research Action 3038, Vision As Process", Aalborg, March 1989

[CG 90] H.I. Christensen, E. Granum, "Initial Control Specification", IR.E.1.1. VAP Internal Report, 1990

[CKB 90] A. Califano, R. Kjeldsen, R. M. Bolle, "Data and Model Driven Foveation", Computer Vision Pattern Recognition proceedings, pp 1--7, 1990

[CM 87] D.J. Coombs, B.D. Marsh, "ROVER : A prototype Active Vision System", TR 219, University of Rochester, August 1987

[CP 84] J. L. Crowley, A. C. Parker, "A representation for shape based on peaks and ridges in the difference of low-pass transform, IEEE PAMI, march, 1984

[CRO 89] J. L. Crowley, "Knowledge, Symbolic Reasoning and Perception", pp 501-515, in Intelligent Autonomous Systems, T. Kanade, F.C.A. Groen, L.O. Hertzberger editors.

[DCB 90] C. Discours, J.L Crowley, O. Bernard, F. Charton, "Specification of Skeleton system for Demonstrator", DR.G.1.1, VAP Internal Report, 1990

[DCB 89] B. A. Draper, R. T. Collins, J. Brolio, A. R. Hanson, E. M. Riseman, "The SCHEMA System", International Journal of Computer Vision, vol 2, pp 209--250,1989

[DCB-87] B. A. Draper, R. T. Collins, J. Brolio, J. Griffith, A. R. Hanson, E. M. Riseman, "Tools and Experiments in the Knowledge Directed interpretation of Road Scenes", DARPA Image Understanding Workshop, pp 178--193, 1987

[DEM 86] Y. Demazeau, "Niveaux de représentation pour la vision par ordinateur. Indices d'image et indices de scene", Thèse INP Grenoble, 1986

[DEM 90] Y. Demazeau, "A multi agent approach to integration of visual modules", European Working Week on Vision, Heraklio, September 1990.

[DM 90] Y. Demazeau, J.P. Müller, "Decentralized Artificial Intelligence", in Decentralized AI, Elsevier North Holland, July 1990

[DM 91] Y. Demazeau, J.P. Müller, "Internal and External descriptions of intentional or reactive architectures", in Decentralized AI 2, Elsevier North Holland, to appear

[GDA 91] Grenoble Working Group on DAI, Internal Working Paper, 1991.

[GGM 89] P. Garnesson, G. Giraudon, P. Montesinos, "Messie : Un système multi spécialistes en vision. Application à l'interprétation en imagerie aérienne, Technical Report Inria number 1012, April 1989

[HR 78] A. R. Hanson, E. M. Riseman, Computer Vision Systems, Academic Press, 1978

[KAN 80] T. Kanade, "Region Segmentation : Signal vs Semantics", Computer Graphics and Image Processing, 13, pp 279--297, 1980

[LUX 85] A. Lux, "Algorithmique et controle en vision par ordinateur, thèse d'état, INPG Grenoble, septembre, 1985

[MAR 82] D. Marr, "Vision", Freeman, San Francisco, 1982

[MIN 85] M. Minsky, "The Society of Mind", Simon and Shuster, New York, 1985

[MWH 87] D. M. McKeown, Jr. Wilson, A. Harvey, "Automating Knowledge Acquisition for Aerial Image Interpretation", DARPA, Image Understanding Workshop, pp 205-226, February 1987

[RJ 88] A. R. Rao and R. Jain, "Knowledge representation and Control in Computer Vision Systems, IEEE Expert, pp 64--79, spring 1988

[TER 83] A. Terry, "The CRYSALIS Project : hierarchical control of production systems", Technical Report, HPP-83-19, Computer Science Dept, Stanford University, 1983

[TM 89] C.L. Tan, W.N. Martin, "An analysis of a distributed multiresolution vision system", in Pattern Recognition, vol 22, 3, pp 257--265, 1989

[TSO 87] J. K. Tsotsos, "Image Understanding", The Encyclopedia of Artificial Intelligence", S. Shapiro and D. Eckroth editors, Wiley and Sons, 1987

[TSO 88] J.K. Tsotsos, "A Complexity Level Analysis of Vision", International Journal of Computer Vision, vol 1, num 4, pp 303--420, 1988

[TSO 89] J. K. Tsotsos, "The complexity of Perceptual Search Tasks", IJCAI, pp 1571--1577, 1989

[TSO 90] J. K. Tsotsos, "Active vs. Passive Visual Search : which is more efficient", technical report RBCV-TR-90-34, University of Toronto, September 1990

[WB 90] S. D. Whitehead, D.H. Ballard, "Learning to Perceive and Act", Technical Report, Rochester University, 1990

[WLH 87] C. C. Weems, S. P. Levitan, A. R. Hanson, E. M. Riseman, "The image understanding architecture", DARPA Image Understanding Workshop, pp 483--496, 1987

Real-Time Performance of Intelligent Autonomous Agents

Anne COLLINOT[1]
LAFORIA / IBP
Université Paris VI
Tour 46-00 - 2ème étage
4, Place Jussieu
75252 Paris Cedex 05
France

collinot@laforia.ibp.fr
(1) 44-27-43-32

Barbara HAYES-ROTH
Knowledge Systems Laboratory
Stanford University
701 Welch Road, Building A
Palo Alto, CA 94304
U.S.A.

bhr@sumex-aim.stanford.edu
(415) 725-0506

Abstract

In a multi-agent world, several agents act simultaneoulsly, competitively or cooperatively. In many situations, an intelligent autonomous agent must interact with the other agents or the physical environment in real time. Because it cannot predict all the events that will occur in the physical environment or result from other agents reasoning, it must notice and control its responses to unanticipated events. However, insuring execution of the best possible operation conflicts with meeting deadlines, especially as the event rate and the number of known operations increase. Rater than engineer agents to meet deadlines under particular parameter values, we aim to build autonomous agents that control their reasoning so as to guarantee real-time performance despite increases in parameter values. We propose a satisficing algorithm. To control response time, it triggers only a limited number of operations and interrupts triggering to execute the best one available whenever it triggers a "good enough" operation or a deadline occurs. To insure that it can execute high-priority operations when interrupts occur, it uses dynamic control plans to trigger operations roughly "best-first." In this paper, we describe the satisficing algorithm, informally analyse the behavior of an agent under this algorithm, and present experimental results.

---

[1] The research was conducted while Anne Collinot was a Post-Doctoral Fellow at Stanford.

1. The Problem

In a multi-agent world, several agents act simultaneously, competitively or cooperatively. In many situations, an intelligent autonomous agent must interact with the other agents or the physical environment in real-time (e.g., [Decker and Lesser, 1990; Howe et al., 1990]). Because it cannot predict all of the events that will occur in the physical environment or result from other agents reasoning, it must notice and respond to important unanticipated events. On the other hand, because it has limited resources, the agent must be selective in its responses, so as to achieve its most important goals. In general, the *utility* of an agent's behavior is a function of the *criticality* of the events to which it responds and the *value* of its responses to them. Moreover, because other agents or physical processes in the environment have their own temporal dynamics, the value of an agent's response to an event depends not only on its response *quality* (the correctness of the response and perhaps other features such as completeness or precision), but also on its response *latency* (the delay between occurence of the event and the response). Different situations may impose different constraints on response latency. We focus on deadlines, including both soft deadlines, whose violation reduces response value incrementally, and hard deadlines, whose violation reduces response value directly to 0.

In addition to being individually challenging, these requirements conflict. In particular, identifying and choosing the best among all possible operations (commonly called the "match process" and "conflict resolution" [Forgy, 1982]) conflicts with meeting deadlines because it entails unbounded response latencies. A given event can trigger (satisfy the conditions of) multiple reasoning operations and a given operation can be triggered by multiple events. If n is the number of events the agent notices and k is the number of operations the agent knows, in the worst case, the time to trigger all executable operations is $O(nk)$. Each of these operations must be rated so that the best one can be chosen. If m is the number of executable operations triggered and r is the number of rating criteria, the time spent rating and choosing among these operations is $O(mr)$. This is not acceptable in agents that must produce high quality responses to important events in real time.

In this paper, we address this problem in the context of autonomous deliberative agents, that is agents that reason about their actions to achieve goals. Deliberative architectures [Corkill et al., 1982; Erman et al. 1980; Georgeff and Lansky, 1987; Hayes-Roth, 1985; McDermott and Forgy, 1978; Newell, 1973] iteratively enumerate possible responses to new events and execute the best one. When augmented with multiple processors [Gupta et al., 1989; Laird et al., 1987], they match events to all known operations in parallel and guarantee bounded cycle time. The use of multiple processors permits to solve the immediate problem of bounding response latencies for a particular application by engineering the agent architecture for associated values of the two complexity parameters. This approach handles anticipated increases in these parameters with additional processors, but the demand for processors has the same complexity as latency has on a single processor. It cannot handle unanticipated increases in event rate or number of known operations. But, in any practical multi-agent world, we have to assume some limit on an agent's computational resources and a reasonable probability that event rate or number of known operations occasionally or eventually will exceed its resources.

A more flexible approach is to apply the concept of "anytime algorithms" [Dean and Boddy, 1988]. On each reasoning cycle, the triggering of reasoning operations is interrupted and the best available one is executed whenever a deadline occurs, thereby guaranteeing bounded latency. The longer triggering is allowed to continue, the higher the expected value and the longer the latency (both up to some maximum) of the operation. Thus, an agent can make strategic trade-offs between response quality and latency. In particular, it can sacrifice quality as necessary to bound latency while event rate and number of known operations increase. The question is, of course, how will

necessary sacrifices in quality impact utility? As we shall see, strategic interruption of triggering is a key feature of the proposed "satisficing algorithm". With additional features, however, it gives better performance than an anytime algorithm with respect to both quality and latency of response.

In this paper we describle a "satisficing approach" and report experimental results of its performance. The satisficing algorithm is designed to permit intelligent real time control of reasoning within a deliberative agent architecture. As we shall see, real time control of reasoning plus several additional features enable an agent to execute high quality operations in bounded time, despite increases in environmental complexity and number of known operations.

2. A Satisficing Approach

Our approach replaces an exhaustive search for the optimal operation on each cycle with a non-exhaustive search for a satisficing operation. To bound latency, it triggers only a limited number of operations and interrupts triggering to execute the best operation available when either it finds one that is "good-enough" or a deadline occurs. To insure that it can execute high-priority operations (high-quality responses to high criticality events) with short latency, it uses dynamic control plans to trigger operations roughly "best-first". In this section, we describe our agent architecture and the satisficing algorithm.

2.1 The Agent Architecture

We assume a deliberative agent architecture comprising asynchronous systems for perception, reasoning and action [Hayes-Roth, 1990]. The perception system senses the environment, labels each event by relevance, criticality, and urgency, and gives it an overall priority. It orders events in the reasoning system's input buffer by priority [Washington and Hayes-Roth, 1989; Washington et al. 1990]. The reasoning system iterates the satisficng algorithm (discussed below): (a) it uses perceived and internally generated events, along with the current control plan, to trigger a limited number of reasoning operations roughly best-first; and (b) when interrupted by a good-enough operation or a deadline, it executes the highest priority triggered operation with respect to the current control plan. Executed operations produce reasoning results, modify the control plan, or place intended actions (including communication actions) in output buffers. The action system retrieves intended actions from the output buffers and executes them in the environment.

Control plans are central to the architecture. They focus an agent's perception of the environment, enable it to coordinate opportunistic and goal-directed reasoning, and guide its execution of high-priority reasoning operations in bounded time under the satisficing algorithm. Accordingly, we briefly explain control plans and their use, with illustrations from Guardian, an experimental agent that monitors simulated intensive-care patients [Hayes-Roth et al., 1989].

A control plan is a data structure comprising a number of decisions, each of which describes a class of operations to be performed during some time period. For example, given an observation of high PIP (peak inspiratory pressure) at time t4, Guardain makes decision D7 in Figure 1, "Quickly react to high PIP." Based on knowledge of different types of operations and events, it evaluates the degree to which each subsequently triggered operation matches D7. Thus, it prefers to execute "quick" operations (e.g. associative, rather than model-based reasoning) that "react to" (diagnose or correct) problems related to the high PIP, until those problems are solved at t5.

An agent generates control decisions with general control reasoning operations within the basic reasoning cycle. Consider this example:

Name: Urgent-Reaction
Trigger: Critical Observation *O*
Prescription: Quickly react to *O*
Criticality: Criticality of *O*
Goal: Diagnose problems related to *O* are corrected

This operation is triggered and its parameter, *O*, is instanciated whenever the perception system delivers an observation with high criticality (such as high PIP). When executed, it generates a control decision favoring "quick" reasoning operations that "react to" *O*, as in D7, and gives it the same criticality of *O*. The decision is deactivated when its goal is achieved, namely that all diagnosed problems related to *O* have been corrected.

Using a small set of general operations to generate a variety of specific decisions, an agent constructs control plans that are appropriate to its situation and changes those plans as the situation changes [Hayes-Roth, 1985; Johnson and Hayes-Roth, 1987]. At each point in time, the agent executes the highest-priority triggered reasoning operation that matches one of the active control decisions, with its priority being the product of degree of match and criticality of the decision.

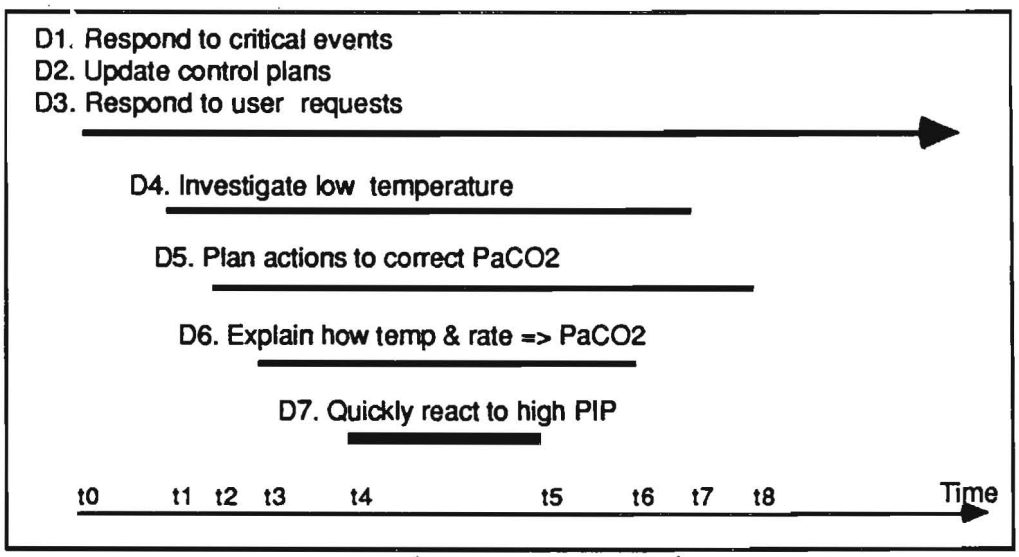


Figure 1. Illustrative Guardian control plan. Horizontal lines signify active time interval and criticality of decisions.

(a) At time t0, Guardian has made three moderately critical control decisions, D1-D3, to respond to critical events, update control plans, and respond to user requests. During t0-t1, it executes triggered reasoning operations that match any of D1-D3, ignoring others.

(b) At t1, Guardian perceives the patient's low temperature, a moderately important abnormality. It executes a triggered control operation to introduce D4, favoring investigation of this problem. It begins to execute triggered operations that diagnose the low temperature (post-operative status), predict its course (spontaneous warming), and infer consequences (low, but rising partial pressure of CO2 in the arterial blood (PaCO2)). Because newly perceived events are delivered asynchronously to its event buffers, Guardian remains sensitive to new events and can execute triggered operations that match D1-D3.

(c) At t2, Guardian infers that the patient currently has low PaCO2, which will rise as temperature rises, a moderately important abnormality. It executes a triggered control operation to introduce D5, favoring planning of corrective actions. It begins to execute triggered operations that plan corrective changes to breathing rate, while continuing its diagnosis, prediction, and causal inference and remaining sensitive to new events.

(d) At t3, Guardian perceives a request for explanation of its predictions that low temperature will cause low PaCO2. It executes a triggered control opeartion to introduce D6, favoring construction of the requested explanation. It begins to execute triggered operations for explanation, while continuing its other tasks and remaining sensitive to new events.

(e) At t4, Guardian perceives the patient's high PIP, a critical abnormality with a deadline on the order of minutes. It executes a triggered control operation to introduce D7, favoring quick reaction to the high PIP. It executes triggered operations that diagnose the underlying problem (pneumothorax, a hole in the lung that allows inspired air to escape into the chest cavity, preventing subsequent inflation of the lung) and recommend a corrected action (insertion of a chest tube). During this interval it ignores less critical ongoing tasks, but remains sensitive to possibly critical new events.

(f) At t5, Guardian completes diagnosis and correction of problems underlying the high PIP and executes a triggered control operation to deactivate D7. It resumes its interrupted tasks. When they are completed at t6, t7, and t8 it deactivates D6, D4, and D5 in a similar fashion and continues executing triggered operations that match D1-D3.

2.2 The Satisficing Algorithm

Now let us turn to the focus of this paper, an agent's use of its control plans to trigger and execute high-priority operations in bounded time. Figure 2 shows the proposed *satisficing algorithm.* As discussed below, control plans play a key role in the algorithm.

The satisficing algorithm uses the current control plan to trigger and prioritize a limited number of executable operations roughly best-first by selecting a limited number of events and operation types for consideration best-first.

The satisficing algorithm considers a limited number of events, which it retrieves best-first from its buffer. Most events are placed in the buffer by the perception system, which uses the current control plan to prioritize them. For example, under D7 in Figure 1, Guardian's perception system gives high priority to observations of PIP and to events of the same type, "breathing measurements." Some reasoning operations executed under D7 instruct the perception system to give high priority to other relevant types of events, for example interpretations of lung x-rays. Events generated by reasoning operations also are prioritized. The satisficing algorithm ignores some events (possibly even high-priority events) that overflow the buffer worst-first or are not yet retrieved when an interrupt occurs.

The satisficing algorithm considers a limited number of known operation types, using the current control plan to retrieve them best-first from memory. For example, under D7, Guardian gives high priority to: operations of the specified type, "react," including its subtypes, "diagnose" and act;" and "associative" operations, which are "quicker" than the alternative "model-based" operations. For a given event, the satisficing algorithm retrieves operations best-first, stopping when it retrieves one that is "bad-enough," that is, does not match its control decision well enough. It also ignores some operations (possibly even high-priority operations) that are not yet retrieved when an interrupt occurs.

Taking these two factors together, the satisficing algorithm attempts to trigger high-priority operations with high-priority events before attempting to trigger lower-priority operations with lower-priority events. For example, under D7, Guardian first tries to trigger "associative diagnosis" and "associative action" operations with PIP events. Given the possibility of partial matches between executable operations and control decisions (some degree of match to some number of variables in some number of control decisions), the algorithm triggers and prioritizes operations roughly, but not always exactly, best-first. It inserts and orders executable operations in a limited-capacity agenda, from which the best available operation is chosen for immediate execution whenever an interrupt occurs. Most executable operations (possibly even high-priority ones), eventually overflow the agenda as newly triggered, higher-priority operations are inserted.

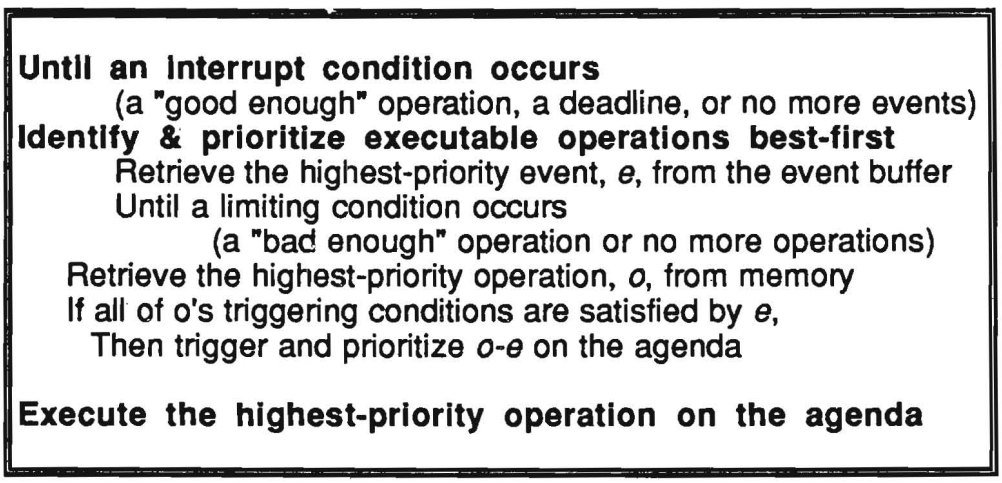


Figure 2. The Satisficing Algorithm

The satisficing algorithm uses interrupt conditions in the current control plan to interrupt its triggering of operations and immediately execute one that is "good enough" or the "best available."

Interrupt conditions are of three types:

(1) If a newly triggered operation is "good-enough" with respect to the current control plan, the agent executes it. For example, under D7, an associative diagnosis operation triggered by the observed high PIP would perfectly match a highly critical control decision and, therefore, be good enough for Guardian to execute immediately. *With interruption by a "good-enough" operation," the algorithm puts a floor under the quality of reasoning and, within that constraint, reasons as fast as possible.*

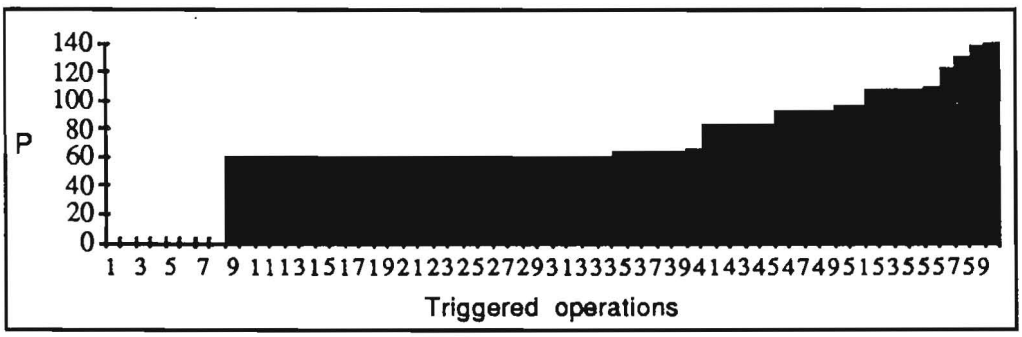
(2) If a deadline occurs, the agent executes the "best available" operation. For example, under D7, Guardian might execute a model-based operation for diagnosing the high PIP if an associative operation were not triggered within a few seconds. *With interruption by a deadline, the algorithm puts a ceiling on the latency of reasoning and, within that constraint, reasons as well as possible.*

(3) If all events and operations are processed, the agent executes the "best available" operation. This is like an exhaustive search for the optimal operation, but takes longer with the satisficing algorithm than with an algorithm optimized for that purpose. In addition, the satisficing algorithm may lose critical events through buffer and agenda overflows. *With exhaustive processing, the agent reasons as well as possible, regardless of latency.*

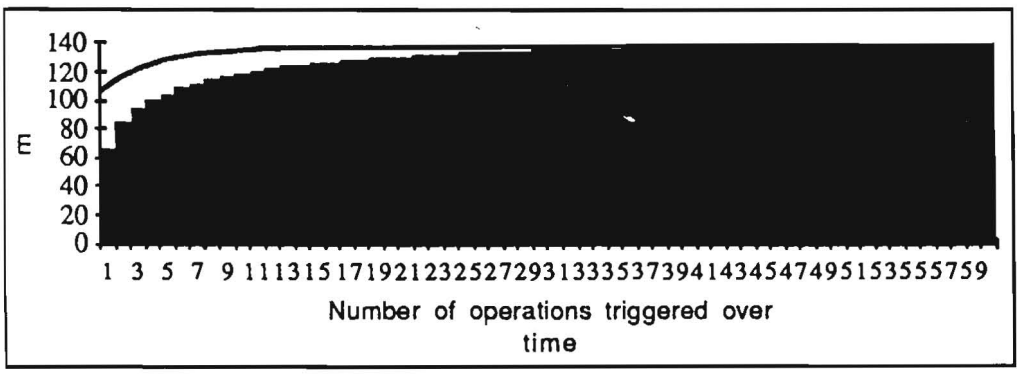
Although different interrupt conditions are better under different circumstances, we offer two general observations. First, an agent should avoid using interrupt (3) because it has high cost and uncertain benefit. Second, interrupts (1) and (2) make a powerful combination because they allow an agent to reason as fast as possible at a criterial level of quality, compromising quality only when necessary to meet deadlines.

3. Behavior under the Satisficing Algorithm

For a given pattern of events and repertoire of operations, an agent's behavior under the satisficing algorithm is determined by its control plan and interrupt conditions. Consider two classes of control plans. *Non-discriminative control plans* have low to moderate criticality and match many potential operations. Therefore, they give comparably low priorities to many operations and provide no obvious criterion for "good enough" operations. By only weakly constraining selection of events and known operations, they permit only roughly best-first triggering of operations.Very discriminative control plans have high criticality and match few potential operations. Therefore, they give distinctively high priorities to few operations and easily identify "good-enough" operations. By strongly constraining selection of events and known operations, they permit strongly best-first triggering of operations. Obviously, these two classes represent a continuum. But they present an interesting contrast, as shown in the examples from Guardian below. For comparison to the satisficing algorithm, we also consider: (a) an exhaustive algorithm that uses an efficient Rete-like match process [Forgy, 1982] and executes the best triggered operations; and (b) an anytime algorithm that uses the same algorithm, but interrupts triggering to execute the best available operation on deadline.



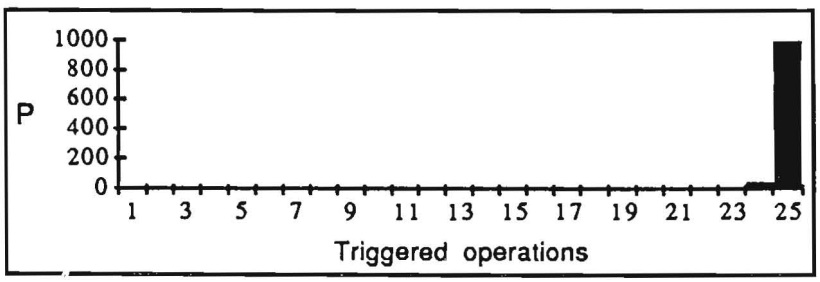
Distribution of priorities on cycle 96.



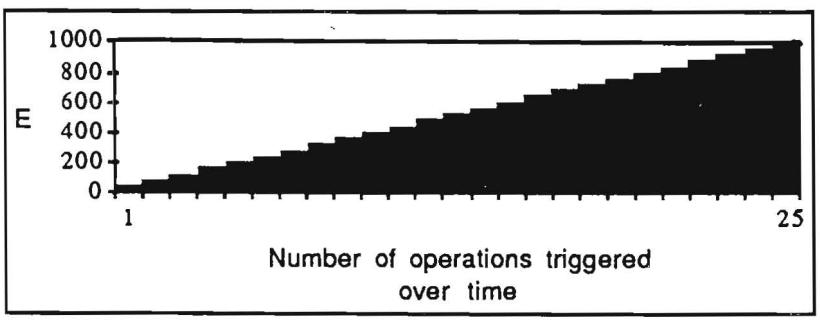
Expected value of best available operation on cycle 96.

Figure 3. The case of a Non-Discriminative Control Plan. Top panel: Distribution of priorities among all possible triggered operations on this cycle. Bottom panel: Expected value of best available operation over time during triggering in random order (shaded curve) or in roughly best-first order (unshaded curve).

Consider the case of a *non-discriminative control plan*, such as Guardian's decision D4: "Investigate low temperature." D4 is only moderately critical. It perfectly matches many potential operations (e.g., associative or model-based operations for diagnosis, prediction, or causal inference triggered by low temperature) and partially matches many others, giving comparably low priorities to all of them. For example, in the top panel of Figure 3, D4 gives a relatively flat distribution of low priorities to the 60 operations that could be triggered on reasoning cycle 96 of a typical run. The exhaustive algorithm triggers all 60 operations and executes the best one. However, its latency is long and exceeds any reasonable deadline. The anytime algorithm does better. In the bottom panel of Figure 3, the shaded curve shows how the expected value (in this case, reflecting the actual value) of the best available operation increases over time during triggering. When a deadline occurs, the anytime algorithm executes the best available operation. Thus, it gracefully trades response quality for latency. However, because the function has a low asymptote and approaches it rapidly, continued triggering has low, rapidly decreasing marginal utility. The satisficing algorithm does better. by ordering its triggering of operations, the satisficing algorithm produces an expected value function (unshaded curve in Figure 3) that has a higher intercept and reaches asymptote earlier. For any reasonable deadline, the best available operation has higher value under the satisficing algorithm than under the anytime algorithm. (As shown below, the time spent ordering the triggering of operations is less than the time saved by limiting the number of operations triggered.) Although triggering is only roughly best-first and it is not obvious how to identify a "good enough" operation, the satisficing algorithm can interrupt triggering early with its own internal deadline and stil guarantee execution of an operation with near asymptotic value. *Given a non-discriminative control plan and interruption by short internal deadlines, the satisficing algorithm makes graceful trade-offs within near-asymptotic quality and latency bounds.*



Distribution of priorities on cycle 49.



Expected value of best available operation on cycle 49.

Figure 4. The Case of a Very Discriminative Control Plan. Top Panel: Distribution of priorities among all possible triggered operations on this cycle. Bottom panel: Expected value of best availabale operation over time during triggering in random order (shaded curve) or in roughly best-first order (horizontal line).

Consider the case of a *very discriminative control plan*, such as Guardian's decision D7: "Quickly react to high PIP." D7 is highly critical, matches only a few potential operations (associative operations for diagnosis or action triggered by high PIP), and gives them distinctively high priorities. For example, in the top panel of Figure 4, D7 gives a very high priority to exactly one of the 25 operations that could be triggered on reasoning cycle 49 of a typical run. The exhaustive algorithm triggers all 25 operations and executes the best one. However, its latency is long and exceeds any reasonable deadline. The anytime algorithm does not necessarily do better. In the bottom panel of Figure 4, the shaded curve shows that the expected value of the best available operation increases roughly linearly over time during triggering. When a deadline occurs, the anytime algorithm executes the best available operation. However, in this case, the linear increase in expected value does not reflect a similar increase in actual value of the best available operation. In fact, actual value is a step function whose point of discontinuity is unknown. Thus, the anytime algorithm trades expected response value, but perhaps not actual value, for latency. The satisficing algorithm does better, as shown by the horizontal line in Figure 4. Because D7 is very discriminative, the satisficing algorithm triggers the best possible operation first. Interrupting triggering for an obviously "good-enough" (very high priority) operation, it executes the best possible operation immediately. *Given a very discriminative control plan and interruption by "good-enough" operations, the satisficing algorithm optimizes both the quality and latency of response.*

4. Experiments

To verify that the satisficing algorithm behaves as intended in a realistic domain, we evaluated it in Guardian. The predictions are that, despite increases in event rate and number of known operations, Guardian will: (a) trigger and choose operations for execution in constant time; (b) respond to most, if not all, critical events correctly; (c) respond faster with a more discriminative control plan; and (d) produce a high utility behavior.

4.1 Method

We used the monitoring scenario discussed above, enacted by our patient simulation in real-time. There are four critical events. Perceived low temperature, predicted low PaCO2, and perceived request for explanation are moderately critical and require response. Perceived high PIP is highly critical and requires quick response. In all cases, complete response entails many reasoning cycles. Other observations of twenty patient variables and a variety of internally generated inferences have low criticality and permit response, but do not require any. Critical events occur in the same order in every simulated run through the scenario. However, normal variability introduced by the simulation and uncontrollable variation in network communication times cause the exact timing of events to vary somewhat.

We manipulated the two complexity factors, event rate and number of known operations. In experiment 1, we held the number of known operations constant at k=39 and manipulated the rate of perceived events entering the reasoning system's buffer to be: 1r, 2r, 4r. and 8r events per second, r=.15. In experiment 2, we held the event rate constant at r=.15 events per second and manipulated the number of known operations to be: 1k, 2k, 4k, and 8k, k=39. In experiment 3, we manipulated both variables: 1r-1k, 2r-2k, 4r-4k, 8r-8k.

As a standard of comparison, we evaluated the exhaustive algorithm in another version of Guardian. This gives a measure of the actual computational cost of higher values of the complexity factors and, therefore, the magnitude of the satisficing algorithm's achievement in circumventing that cost. In fact, with event rates 4r and 8r, response time under the exhaustive algorithm is too long to complete the scenario. Therefore, in experiments 1 and 3, we used a maximum rate of 3r for the exhaustive

algorithm only. Except for their control algorithms, the two versions of Guardian are identical. They contain knowledge and reasoning operations necessary to perceive events from the simulator and to construct and follow the control plan in Figure 1. They contain other knowledge and reasoning operations that have low priority during this scenario. (In earlier experiments, we evaluated a version of Guardian using the anytime algorithm described above. However, the trade-offs between response quality and latency were too precipitous to give good overall performance.)

4.2 Results

Figure 5 shows "agenda time," the time to trigger and choose an operation on each reasoning cycle, for different experimental conditions. The left column shows times for the highly critical event, which presents the strongest demand for high quality, bounded-time response and for which Guardian makes its most discriminative control decision. Each point plotted in these graphs is averaged over the 21 reasoning cycles Guardian uses to diagnose and initiate corrective action for the high PIP problem. The right column shows results for reasoning about the three moderately critical events, for which Guardian makes less discriminative control decisions. Each point in these graphs is averaged over the approximately 65 reasoning cycles Guardian uses to reason about these events.

The most important result is the relation of agenda time to the three manipulations. With the exhaustive algorithm, as one would expect, agenda time for both highly and moderately critical events is a steep linear function of event rate, a shallower linear function of number of known operations, and a second-order function of the two variables together. As mentioned above, beyond an event rate of 3r, agenda time is too long to complete the scenario. By contrast, as predicted, *with the satisficing algorithm, agenda time is constant regardless of event rate, number of known operations, or the two variables together.*

The satisficing algorithm sometimes ignores moderately critical events in order to meet deadlines for highly-critical events. For example, in Experiment 3, condition 8r-8k, the perceived high PIP happened to occur shortly after the perceived request for explanation. While Guardian immediately responded to the high PIP, a proliferation of highly critical new perceived and internally generated event caused the explanation request to overflow its event buffer. As a result, Guardian could not trigger explanation operations even after the high PIP problem was corrected. This occurred in only a few conditions with high event rates and random temporal clustering of critical events. Although it is a reasonable kind of trade-off, we are exploring architectural mechanisms to minimize the loss of critical events. In general, as predicted, *the satisficing algorithm always responds correctly to the highly critical event and nearly always responds correctly to the moderately critical events.*

It is worth noting that the satisficing algorithm is comparatively fast. In all conditions, agenda time is substantially shorter for the satisficing algorithm than for the exhaustive algorithm. Thus, the computational cost of triggering operations best-first is much less than the computational savings form limiting the number of operations triggered. Although this result is implementation-sensitive, it is likely to hold generally because the exhaustive algorithm already has been optimized, while the satisficing algorithm has not.

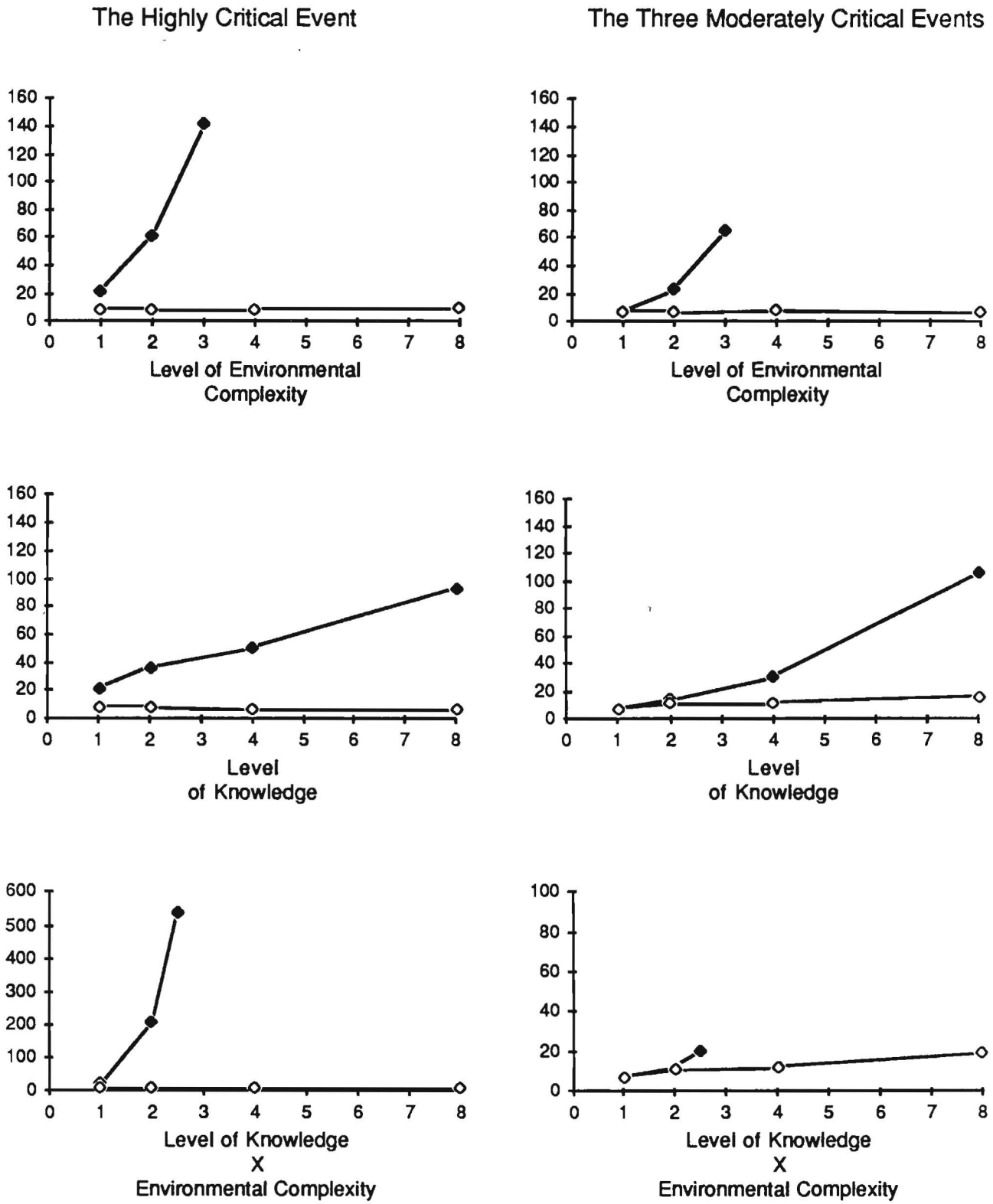


Figure 5. Experimental Results. Average agenda time (in seconds) under the exhaustive algorithm (shaded dots) and the satisficing algorithm (unshaded dots) for different levels of environmental complexity (event rate), knowledge (number of known operations), and the combination. Note change of scale for agenda time in the bottom two graphs.

In addition, as predicted, *the satisficing algorithm responds faster with a more discriminative control plan*. In these experiments, Guardian has a more discriminative control plan for the high PIP event than for the other critical events. In all cases, the exhaustive algorithm actually produces longer agenda times for the high PIP event than for the others. This is because the high PIP event is accompanied by a flury of events, independent of our manipulation, all of which the exhaustive algorithm considers. By contrast, the satisficing algorithm produces shorter times for the high PIP event than for the others. It ignores low-priority events, no matter how many occur and, as suggested by our informal analysis, a very discriminative control plan enables the satisficing algorithm to identify and execute the best possible operation immediately, while less discriminative control plans entail more triggering time.

Finally, as predicted, *the satisficing algorithm maintains high utility despite increases in event rate and number of known operations*. Recall that an agent's utility is a function of the criticality of the events to which it responds and the value (quality and latency) of its responses. Under any reasonable combining function, Guardian's utility under the exhaustive algorithm is low. Although it eventually produces the correct response to every key event, its latencies are excessive, especially for the highly-critical high-PIP event with its short deadline and life-threatening consequences. By contrast, Guardian's utility under the satisficing algorithm is high in all conditions because it naturally favors correct, timely responses to critical events. It always responds immediately to the highly-critical high-PIP event and responds promptly to most other critical events as well.

5. Conclusions

The satisficing approach is designed to enable an intelligent autonomous agent to guarantee real-time performance despite increases in environmental complexity and number of known operations. Three factors determine its actual effectiveness: a good satisficing algorithm, an effective control plan, and appropriate interrupt conditions. The approach also relies on architectural mechanisms, such as the perceptual process, which prioritizes and filters events, and the reasoning system's limited-capacity event and agenda buffers, which strictly bound the number of events and operations under consideration at any point in time. The present results provide a proof of concept using a simple version of the algorithm and control plans and interrupt conditions previously developed for Guardian. In our experiments, the satisficing approach maintains high utility by responding correctly to highly and moderately critical events in constant time, despite substantial increases in event rate and number of known operations. Only in extreme cases (co-occurence of several critical events in the context of a high overall event rate) must it ignore a moderately critical event in order to give a timely response to a highly critical event.

In ongoing research, we are investigating formal properties of control plans and interrupt conditions and their implications for behavior of agents under the satisficing cycle. We also wish to investigate the use of control plans for controlling communication [Bouron, 1991] among several agents. In order to guarantee real time performance, intelligent autnomous agents must also adapt their communication activity so as to meet deadlines associated with the agents goals. Knowing about control plans of other agents should enable an agent to decide what to communicate at appropriate times.

Acknowledgements

This research was supported by NASA contract NAG 2-581 under DARPA Order 6822 and Boeing Computer Services contract W289988. The Guardian system was developed in collaboration with Adam Seiver, Richard Washington, David Ash, Rattikorn Hewett, Luc Boureau, and Angel Vina, with additional funding by NIH contract 5 P41 LM05208-18 and EPRI contract RP2614-48. We thank Edward A. Feigenbaum for sponsoring the work in the Knowledge Systems Laboratory.

References

Bouron, T. COMMAS: A communication model for multi-agent systems. *Proceedings of the 1991 European Simulation Multiconference*, Denmark, 1991.

Corkill, D.D., Lesser, V.R., and Hudlicka, E. Unifying data-directed and goal-directed control: An example and experiments. Proceedings of the National Conference on Artificial Intelligence, 143-147, 1982.

Dean, T., and Boddy, M. An analysis of time-dependent planning. *Proceedings of the Seventh National Conference on Artificial Intelligence, 1988.*

Decker K., and Lesser V.R. A scenario for cooperative distributed problem solving. *Proceedings of Tenth International Workshop on Distributed Artificial Intelligence*, 1990.

Forgy, C.L. RETE: A fast algorithm for the many pattern/many object pattern matching problem. *Artificial Intelligence*, 19:17-32, 1982.

Erman, L.D., Hayes-Roth, F., Lesser, V.R., and Reddy, D.R. The Hearsay-II speech-understanding system: Integrating knowledge to resolve uncertainty. *Computing Surveys* 12:213-253, 1980.

Georgeff, M.P., and Lansky, A.L. Reactive reasoning and planning. *Proceedings of the Sixth National Conference on Artificial Intelligence*, 1987.

Gupta, A., Forgy, C., and Newell, A. High-speed implementations of rule-based systems. *ACM Transactions on Computer Systems*, 7:119-146, 1989.

Hayes-Roth, B. A blackboard architecture for control. *Artificial Intelligence*, 26:251-321, 1985.

Hayes-Roth, B. Architectural foundations for real-time performance in intelligent agents. *The Journal of Real-Time Systems*, 2:99-125, 1990.

Hayes-Roth, B., Washington, R., Hewett, M., Hewett, R., and Seiver A. Intelligent monitoring and control. *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, 1989.

Howe, A.E., Hart, D.M., and Cohen P.R. Addressing real-time constraints in the design of autonomous agents. *The Journal of Real-Time Systems*, 2:81-97, 1990.

Johnson, M.V., and Hayes-Roth, B. Integrating diverse reasoning methods in the BB1 blackboard control architecture. *Proceedings of the Sixth National Conference on Artificial Intelligence*, 1987.

Laird, J.E., Newell, A., and Rosenbloom, P.S. SOAR: An architecture for general intelligence. *Artificial Intelligence*, 33, 1987.

McDermott, J., and Forgy, C. Production system conflict resolution strategies. In Waterman, D.A., and Hayes-Roth, F. (eds), *Pattern-Directed Inference Systems*, Academic Press, 1978.

Newell, A. Production systems: models of control structures. In Chase W.G. (ed.), *Visual Information Processing*, Academic Press, 1973.

Washington, R., and Hayes-Roth, B. Managing input data in real-time AI systems. *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, 1989.

Washington, R., Boureau, L., and Hayes-roth, B. Using knowledge for real-time input data management. *Technical report*, Stanford University, 1990.

Conversation for Organizational Activity

(A Panel on The Dynamics of Knowledge and Organization in Multi-Agent Systems)

Chisato Numaoka
Sony Computer Science Laboratory Inc.
Takanawa Muse Building,
3-14-13 Higashi-gotanda, Shinagawa-ku,
Tokyo, 141 JAPAN
TEL: +81-3-3448-4380 FAX: +81-3-3448-4273
E-mail: chisato@csl.sony.co.jp

Abstract

In this paper I address the issues of how an organization such as a group or a hierarchical society can be formed and how organizational knowledge (e.g. a contract) can be acquired through conversation among situated agents. A special interest group (SIG) has the most fundamental style of every organization. Through conversation, situated agents can join any SIG and act as a member in order to individually profit from group activity. On the basis of SIGs, we can formulate markets and hierarchies which are organizations of different styles. Conversation is key to the organizational activity of situated agents in these organizations.

1 Introduction

Various in management science (e.g. [Malone 87, Marschak and Radner 72, Baligh and Richartz 67]) use the concepts of *market* and *hierarchy* as organizational structures in human economic activity. The action of each human in an economic community is based on a payoff function for the organization to they belong. Situated agents [1] with goals are similar to humans in the sense that both pursue activities for their own benefit. Here, I investigate how an agent forms various useful organizational structures using conversation, and what benefits agents get from belonging to these organizations.

In our analysis, both the market and the hierarchy are represented as *special interest groups* in which every member benefits from the group. The master/slave relationship is fundamental to the hierarchy, while the market is based on a suppliers/buyers relationship. Furthermore, the hierarchy pursues benefits for the organization whereas the market provides a field in which each member can pursue its own benefits. The rest of this paper describes characteristics of three organizational structures (viz. SIG, market, and hierarchy) and conversational activity of agents in these organizations.

2 Well-formed Plans

Situated agents perform physical actions based on logical plans. Plans can not only be constructed by a situated agent itself but also be affected by information received through conversation with other agents. The most fundamental plans are called *well-formed plans*. Here I list up well-formed plans used in this paper:

- Atomic plans, whose goals are to perform any primitive action of functional action units.
- (SERIAL WFP_1 ... WFP_n): to perform WFP_1, ..., and WFP_n serially.
- (NDOR WFP_1 ... WFP_n): to perform WFP_1, ..., or WFP_n non-deterministically.
- (TEST F($?V_1$,...,$?V_n$)): to test whether or not there is F in a belief and, if exists, to bind objects to variables (this sentence is not so clear. what F?) ($?V_1$,...,$?V_n$).

Other constructs of WFP are as follows:

- (SKIP) $\stackrel{\text{def}}{=}$ (TEST true).
- (ABORT) $\stackrel{\text{def}}{=}$ (TEST (NOT true)).
- (CASE F WFP_{true} WFP_{false} WFP_{unknown}) $\stackrel{\text{def}}{=}$

  (NDOR (SERIAL (TEST F) WFP_{true})
  (SERIAL (TEST (NEG F)) WFP_{false})
  (SERIAL (TEST (UNKNOWN F)) WFP_{unknown})

- (IF-THEN-ELSE F WFP_1 WFP_2) $\stackrel{\text{def}}{=}$

  (NDOR (SERIAL (TEST F) WFP_1)
  (SERIAL (TEST (NOT F)) WFP_2)).

[1] Situated agents are autonomous and self-contained agents who can reflect their behavior on situations.

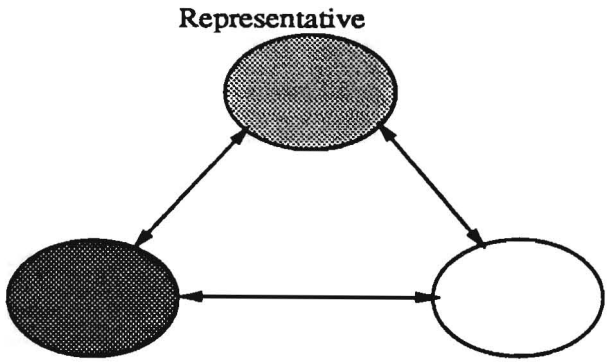


Figure 1: Organizational Structures (1): Special Interest Group

- (IF-THEN F WFP) $\stackrel{\text{def}}{=}$

  (IF-THEN-ELSE F WFP (SKIP)).

- (DO-FOR-ALL F WFP): As long as there is F which is not tested, perform WFP. (should be "an F", maybe? i still don't understand what "F" is.)

3 Special Interest Group

In this section, I formally define the characteristics of a special interest group (henceforth SIG), an organization in which every member has the same interests. A SIG can be viewed as a conceptually bounded field through which each member of the group can share information with others and can cooperate with others to do a job (See Figure 1). In order for all members to be equal, restrictions, such as group specific rules, will be given equally to all members.

A SIG is a *mapping function* from a group name defined by an agent to a set of agents. A SIG has a set of restrictions on information acquired through group communication. The followings are definitions related to SIG:

Definition 3.1 (Special Interest Group (SIG)) *A SIG is a mapping function from an agent identifier and a group name (defined by a representative of the SIG) to a set of agents and a set of restrictions on the SIG.*

Definition 3.2 (Representative of an SIG) *A representative of an SIG is an agent who defines a framework for the SIG. A representative is not privileged within the group, and its only special role is to tell the group name and restrictions to agents wanting to join the SIG.*

An agent must have the following information to join an SIG:

- *The group name of the SIG,*
- *Protocols for joining the SIG, and*
- *Services provided by the SIG.*

Obviously, if an agent does not know the advantages of being a member of a SIG, it does not have any reason to join that SIG. Thus, knowing services provided by a SIG is an important matter. Every member of an SIG has a right to vote on new members. In addition, the representative of the SIG has the authority to make a final decision based on the voting. Requirements for joining a SIG are defined by the representative and given to the members.

3.1 Group Formation

Expressing a Desire to Join an SIG When an agent wants to join an SIG, it has to make an utterance to introduce itself to the representative. This utterance must include an agent's identifier A_i. Suppose the protocol for joining an SIG is (TRUE join (Agent-ID A_i)) and the identifier of the representative is R. Thus, this utterance should be:

(INFORM A_i R (TRUE Join A_i) R)

Voting The representative gives voting plans to every member of the SIG. We may do this as follows:

```
(INFORM A B
  (TRUE (WFP (Voting ?agent-id)
      (SERIAL
        (TEST (TRUE Preference ?agent-id ?x))
        (TEST (TRUE Preference-Average ?y))
        (IF-THEN-ELSE (< ?x ?y)
          (INFORM B A (TRUE against) A)
          (INFORM B A (TRUE for) A)))))
  B).
```

Here, *Preference* is a relation which shows the degree of preference for an agent assigned to variable *?agent-id* and *Preference-Average* is a relation to show an average value in the preference spectrum. This utterance means that if representative A REQUESTs including a formula *(Voting agent-id)* to agent B then agent B should say *(TRUE against)* if its preference is less than average or *(TRUE for)* otherwise.

When representative A believes that B knows how to vote, the voting procedure is as follows:

Step 1 *Request (Voting agent-id) to every member of the group. Agent-id indicates the agent identifier of an agent who wants to join the SIG is assigned.*

Step 2 *According to the well-formed plan of the composite action* (Voting agent-id), *every member Informs A (TRUE for) or (TRUE against).*

Step 3 *A decides based on the vote and its own judgment, whether the agent can join the SIG.*

Notification of New Members A notification of approval is send by the representative to the new agent using INFORM as follows:

(INFORM R A_i
  (TRUE Group-members (Group-name name)
    Member's-list
  A_i).

This notification comes in the form of a members' list. In addition, the new member is also told voting plans.

Leave an SIG When an agent wants to leave an SIG, it does this according to the protocol of leaving for that SIG. The agent sends to another member of the SIG with the group name of the SIG indicated as Interpreter. [2] An example is:

(INFORM A_i A_j
  (TRUE Leave A_i (Group-name gname)) A_j).

3.2 Group Communication

Here I give a rule describing the FORWARD action in the case that a group name is indicated as the Interpreter.

Rule 3.1 *If the Interpreter of an utterance is a name of group, that utterance is FORWARDed to every member of the group,* gname.

According to this rule, the above utterance will be forwarded to each member of the group, *gname*. When an agent receives this utterance, it deletes the name of the agent who wants to leave *gname* from the SIG's group member list. After all, members of the SIG receive the utterance, services supplied by the SIG are not available to the departed agent. [3]

Every group communication follows Rule 3.1. That is, notification is given to a group by specifying the group name as Interpreter.

Principle 3.1 *Every member of a group has the opportunity to receive all information sent to the group.*

This principle is a direct application of the definition of INFORM.

[2] Interpreter is a special concept we introduced in Situated Conversation Model [Numaoka 90]. In Situated Conversation Model, an agent who should interpret an utterance is not always a hearer of the utterance and Interpreter indicates an agent who should interpret utterances finally. This is shown in each INFORM, QUERY, or REQUEST formula as its last argument.

[3] [Maruichi *et al.* 89] has shown that an environment which facilitates speaking to unspecified hearers is useful for group communication. Since an environment us a kind of group and the environment knows all members of the group, if an agent makes an utterance to the environment, the environment can send the utterance to all group members on behalf of the speaker. In Situated Conversation Model, this task is achieved without environment by situated agents knowing the notion of group.

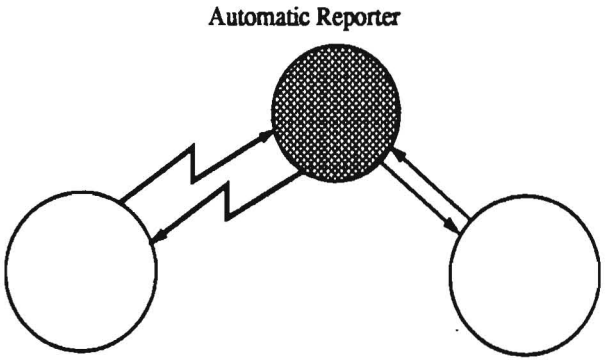


Figure 2: An Automatic Reporter

3.3 Shared Resources among SIG Members

In an SIG, members share resources, including situated agents and well-defined objects supplied by the representative of the SIG. Some shared resources may have to used in an exclusive manner due to internal restrictions. For example, an agent of type *telephone* can be accessed by only one user at a time, so there must be a mechanism to guarantee exclusive access. Telephones are, of course, unnecessary in Situated Conversation Model since a situated agent can always receive messages. However, the telephone answering machine is a good approximation of the mechanism agent uses to receive messages. Therefore, necessary is proxy of an automatic reporter such as a telephone answering machine which notify a current status of an agent and inform a request of the agent to senders of messages to it (see Figure 2). The notion of an insensitive actor [Agha 86] is useful for this purpose. An insensitive actor is one which is insensitive to incoming messages until it is sent a become communication specifying a replacement behavior. [4] Since whenever a message is received by an actor, it executes the request as soon as possible, if an actor does not want to accept messages while performing a task, it assigns another actor to deal with the incoming messages.

How can we implement an agent with this ability in Situated Conversation Model? Consider the example of a Japanese SIG having telephone agents capable of connecting with telephone agents in another SIG in the U.S. Agents who want to phone the SIG in the U.S. must belong to an SIG in Japan with this ability. Thus, the representative of an SIG must have the ability to inform SIG members of the agent identifiers of phone agents able to connect with telephone agents in the U.S.

The agent identifier of such an agent should identify itself as a receptionist. That is, there is an automatic

[4] In the actor model, the actor can explicitly specify the behavior it would like for its next task. The behavior specified is called replacement behavior.

reporter contained in the OBJECT level of this agent. An automatic reporter is a well-defined object without a META level. Its abilities are:

- To forward received utterances to an agent, and
- To make an utterance specified by the agent when it is in an insensitive state and not to forward received utterances. [5]

Thus, an automatic reporter can change its role according to the sensitivity of an agent. For example, if a telephone is occupied, it changes its state from sensitive to insensitive and reports this change to its automatic reporter. This frees the agent from being bothered with incoming utterances.

3.4 An Example SIG

In this section, I give an example of a SIG for sharing a meeting room. Invariants [6] for the meeting room are as follows:

Invariant 1 *Only one agent at a time can use the meeting room, R.*

$\forall$ (LAND (TRUE occupies A_i R)
(TRUE occupies A_j R))
where i $\neq$ j.

Invariant 2 *The room is utilized whenever there is a request.*

Invariant 3 *Chances to use the meeting room are fairly given to every member of the SIG.*

For this invariant, the meeting room agent follows the definition and principle given below:

Definition 3.3 (Fair Assignment of a Room) *If it is an agent's turn to use the room and it requires the room, the room is assigned to that agent. If agent A_i requires the room when it is not his turn, it is assigned the room if and only if it is guaranteed that no agent between the current agent and agent A_{i-1}, in cyclic order of the group member's list, requires the room.*

Principle 3.2 *A meeting room agent with a group member's list in cyclic order can guarantee that the room is assigned to members of the SIG as fairly as possible.*

Suppose $(A_1, \ldots, A_i, \ldots, A_j, \ldots, A_n)$ is the group member's list. There are three cases.

[5] Whether the automatic reporter stores these received utterances or not should be defined by the designer of the agent.

[6] We use the term Invariant for indicating the formula which cannot remain in any belief space. That is, if a formula which is contradictory to the formula represented by an invariant appears in any belief space, then any truth maintenance mechanism will cause any physical action in order to resolve the situation.

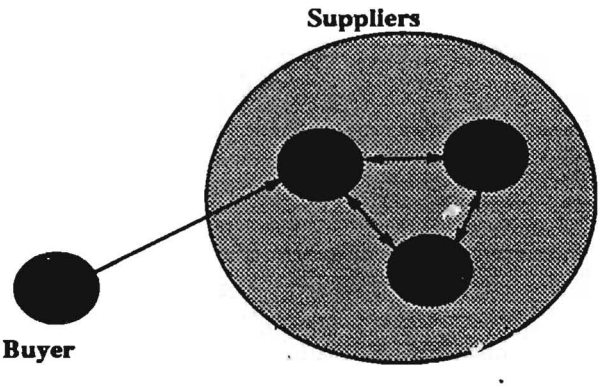


Figure 3: Organizational Structures (2): Market

Case 1: It is A_i's turn, and it requests the room. A_i is assigned the room, and the next turn goes to A_{i+1}.

Case 2: It is A_i's turn, and A_j requests the room. The meeting room agent asks agent i through j-1, in order, whether any of them need the room. If not, the room is assigned to A_j and A_i takes A_j's place in the order. If some agents do need the room, the agent closest in line to A_i is assigned the room, and A_i then takes the other agent's place in the order.

Case 3: The same as Case 2 but i is exchanged for j.

4 Market

A market is an another kind of organization. In a market, there are suppliers, task processors for various types of tasks, and buyers, agents needing the services of any supplier. In a sense, the buyers are "product managers." They know all suppliers and can choose the best one for their needs. In a market, buyers contract some task directly with suppliers whose products they need (See Figure 3).

The relationships of master with slaves and buyers with suppliers differ in that, in the former, the master assigns tasks to slaves based on their ability, in the latter, a buyer assigns tasks based on which supplier can best do the task not simply which supplier has the ability. The master in a hierarchy is the coordinator of an organization as well as being a part of that organization. A buyer in a market, however, is the user of an organizational functionality and is not necessarily part of the organization.

In a market, a group of suppliers is modeled as an SIG. Therefore, we can say a market is represented by a pair consisting of a buyer and an SIG. The advantage of a market is that if a new agent is developed with a better implementation, the benefit of the agent is available for all members of a SIG, a group of suppliers, immediately when the agent joins the SIG.

4.1 Example: Contract Net Protocol created through Conversation

The Contract Net Protocol is a protocol for assigning tasks to suppliers called *contractors* using *bidding* [Davis and Smith 83]. In this protocol, users are called *managers* and a manager begins a contract process by broadcasting an *announce* message. Every contractor receiving an announce message replies with a *bid* message to the manager if it has the ability to perform the task indicated in the announce message. The manager then selects a contractor from the bits received and it sends a *contract* message to the selected contractor.

Following is an implementation of Contract Net Protocol on a market. A group of suppliers is defined to be an SIG. Let its name be *construct* ($\in \mathcal{N}$). The SIG construct should contain certain requirements for members of the SIG. Suppose these requirements are as follows:
(Primitive Actions)

(Estimate Spec): An action for estimating the cost for a construction specified by *Spec.*
(Construct Spec): An action for perform a construction specified by *Spec.*

(Internal States)

estCost: The estimated cost of a construction.
estTerm: The estimated term of a construction.
possible(Spec): a formula created by the result of an action *(Estimate Spec).*

The SIG representative establishes these requirements, which every member of the supplier SIG must satisfy. Therefore, a designer of a supplier must satisfy the representative agent's specifications.

Another possibility is to assume that every supplier in the SIG construct has these abilities and can reply to questions about its internal states. The Contract Net Protocol for a market whose suppliers form SIG constructs is implemented with conversational action protocols as follows: [7]
(Defining ANNOUNCE and BID using Conversation)

```
(INFORM A B
  (TRUE (WFP (Announce ?spec)
    (SERIAL
      (TEST (TRUE Self-ID ?c))
      (Estimate ?spec)
      (IF-THEN-ELSE (TEST (TRUE Possible ?spec))
        (INFORM ?c A (TRUE Bid dueCost dueTerm) A)
        (ABORT)))))
  (Group-name construct)).
```

By performing this action, B forwards the utterance to all members of the SIG. As a result, they will know the interpretation of utterance *(Announce spec).* In the above utterance, ?c is replaced by the agent identifier of the hearer receiving the utterance. If an estimation fails, using ABORT causes the action corresponding to well-formed plan *(Announce spec)* to fail, and the agent utters a failure REPORT.

(ANNOUNCE and CONTRACT a Construction) After performing the INFORM action, A REQUESTs B, a supplier of SIG *construct*, to announce a construction specified by *spec* as follows:

```
(REQUEST A B (Announce spec)
             (Group-name construct)).
```

B then forwards the utterance to all members of the SIG *construct.* Then, A will wait for bids and choose a contractor based on those bids. It then utters the following REQUEST that ?c constructs a building as described in the full specification, *fullSpec*:

```
(REQUEST A ?c (Construct fullSpec) ?c).
```

These examples show that conversational action protocols are enough strong to express any other protocol, including Contract Net Protocol, if an organization can be modeled as an SIG, hierarchy, or market.

[7] Here, the authors presuppose that buyer A knows supplier B, a member of the SIG construct.

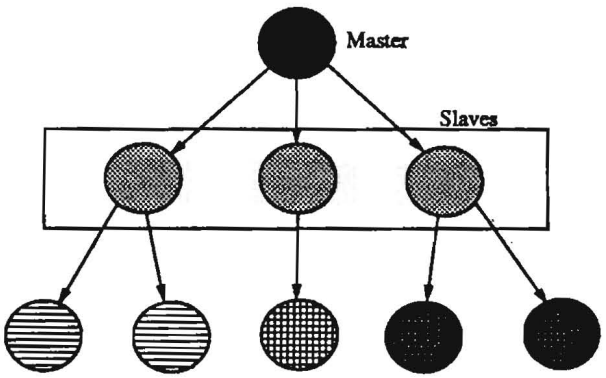


Figure 4: Organizational Structures (3): Hierarchy

5 Hierarchy

A hierarchy is designed as a kind of SIG (See Figure 4), which has a name, a representative, and some other original group members. A representative is called a *master* and other members are called *slaves.* Here is a master/slave relation. This is the relative relation for a hierarchy. Thus, a slave may be a master of other hierarchy. A hierarchy is an organization to deal with large-scale tasks. Since it is difficult for an agent to perform a large-scale task, the task is divided into smaller tasks. These tasks are assigned to slaves. In order to monitor the status of tasks, a manager is required. This is a master agent. A master has an authority controlling slave agents and must

know all functionalities of its slave agents to assign subtasks. Since a hierarchy is an SIG, an agent can join the hierarchy. A master also has an authority to decide to allow a new agent to join by its own preference.

The duty of a slave is to comply with every request its master makes. For example, after beginning an assigned task, if the slave is required to change some parts of the task, it must obey even if the changes mean it will have to do the task over again. An agent designed as a slave must know all the actions performed by its master of the hierarchy. It must use these actions to define how the master can use it so that the master can understand how the slave performs tasks.

The advantage of a hierarchy is that when a master gives part of an organizational goal, and gives permission to use the resources of the hierarchy, to each slave, it forces the slaves to give their best effort to achieve these organizational goals. The biggest problem in hierarchy is that the master must know in advance how many agents he will require. Because of this, a market system, which ensures finding agents appropriate to a given task, is often a better system.

5.1 Example: Recruiting Agents

A hierarchy is an example of distributed problem solving, decomposing a problem into smaller problems and assigning each to a slave. If there is no slave available to be in charge of a task, the master recruits a slave by issuing an invitation. However, since the number of agents the master knows is limited, it can ask the agents it does know to forward the invitation message to all the agents they know. This is not just a simple FORWARD action, so the master must first use the following INFORM to define how to distribute the invitation:

```
(INFORM A B
  (TRUE (WFP (Recruit ?message)
    (SERIAL
      (TEST (TRUE Self-ID ?a1))
      (DO-FOR-ALL (TRUE Agent-ID ?a2)
        (SERIAL
          (TEST (TRUE WFP (Recruit ?msg) ?body))
          (INFORM ?a1 ?a2
              (TRUE WFP (Recruit ?msg))?body
              ?a2)
          (REQUEST ?a1 ?a2
              (Recruit message)
              ?a2)))
      ;;RECRUITING PLAN)))
  B).
```

In this action, *DO-FOR-ALL* is a well-formed plan, for all agents ?a2 whose agent ids B knows, perform the action shown in a formula *(SERIAL ...)*. *TEST* finds a formula which matches the given formula including variables, and *Self-ID* is an agent own identifier. After issuing the INFORM above, master A takes the following action:

(REQUEST A B (Recruit message) B).

message contains all that is required to carry out the task.

6 Conclusion

In this paper I have classified organizational structures into three categories: Special Interest Group, market, and hierarchy. Situated agents can use these organizations through conversation in Open Systems. In addition, I have discussed the relationship between conversation and these structures.

A special interest group (SIG) is a fundamental organization for implementing both hierarchies and markets. A SIG allows its members to share information and/or resources. In a market, a buyer is designed to select the best of many possible suppliers for a job. New suppliers are designed to do tasks better than the existing suppliers in a market. In a sense, all agents represent suppliers in a market. If a supplier is developed by redesigning an existing supplier, it can join a market by joining with the original supplier. Then the original supplier will notify buyers to the new supplier. A hierarchy is for systematic organizations where a master coordinates sub-organizations that autonomously carry out tasks under the direction of a master of the sub-organization.

An agent can simultaneously belong to various kinds of organizations and, as a result, must play various roles. This means a situated agent has to work within the restrictions of various kinds of organization. The analysis of this problem is left for future work.

References

[Agha 86] Gul Agha. *ACTORS: A Model of Concurrent Computaion in Distributed Systems.* The MIT Press, 1986.

[Baligh and Richartz 67] H.H. Baligh and L. Richartz. *Vertical Market Structures.* Allyn and Bacon, Boston, 1967.

[Davis and Smith 83] R. Davis and R.G. Smith. Negotiation as a metaphor for distributed problem solving. *Journal of Artificial Intelligence*, Vol. 20, No. 1, pp.63–109, 1983.

[Malone 87] Thomas W. Malone. Modeling Coordination in Organizations and Markets. *Management Science*, Vol. 33, No. 10, pp.1317–1332, 1987.

[Marschak and Radner 72] J.G. Marschak and R. Radner. *Economic Theory of Teams.* Yale University Press, 1972.

[Maruichi *et al.* 89] T. Maruichi, M. Ichikawa, and M. Tokoro. Modeling Autonomous Agents and Their Groups. In *Proceedings of European Workshop on Modeling an Autonomous Agent in a Multi-agent World*, August 1989.

[Numaoka 90] Chisato Numaoka. *A Conceptual Framework for Modeling Conversation in Open Distributed Systems.* PhD thesis, Keio University, December 1990.

A MODEL FOR BELIEF REVISION IN A MULTI-AGENT ENVIRONMENT

Aldo Franco Dragoni
University of Ancona, Computer Sciences Institute
via Brecce Bianche 60131 ANCONA (Italy)
tel. +39-71-2204832; fax +39-71-2204474
e-mail luca@anvax2.cineca.it

INTRODUCTION.

In a Multi-Agent setting it became necessary to enlarge the traditional concept of Belief Revision. For detecting contradictions and identifying their sources it is sufficient to maintain informations about *what* has been told; but to "solve" a contradiction it is necessary to keep informations about *who* said it or, in general, about the source where that knowledge came from. We can take as certain the fact that an agent gave an information, but we can take the given information only as a revisable assumption. The Belief Revision system can't leave the sources of the informations out of consideration because of their relevance in giving the additional notion of "strength of belief" [Galliers 89]. In fact, the reliability of the source affects the credibility of the information and vice-versa. It is necessary to develop systems that deal with couples <assumption, source_of_the_assumption>. In [Dragoni 91] we propose a system that moves in this direction. Here we give a short description of that system. In part one we describe the agent's knowledge processing structure with a particular characterization of the "Assumption Based Belief Revision" concept; in part two we outline the project of an embedded device that enables the overall system to deal with couples <assumption,source>.

1.1 PRELIMINARIES.

By "Belief Revision" we mean the process of detecting contradictions, identifying the assumptions from which they came out and *readjusting the knowledge base to remove the contradictions*. Beliefs are assumed to be expressed as sentences of first order logic stored in the agent's memory. There are two kinds of sentences: those introduced as assumptions and those deductively derived as logical consequences of the assumptions. We need an Assumption Based Truth Maintenance System [De Kleer 86]. We use the following modified version of the *Supported Wff* of Martins and Shapiro [Mar-Shap 86] (the rationalities for the multi-agent topic are in part two):

SWM=<A,OSO,OS1,..,OSn,OSE,RS>

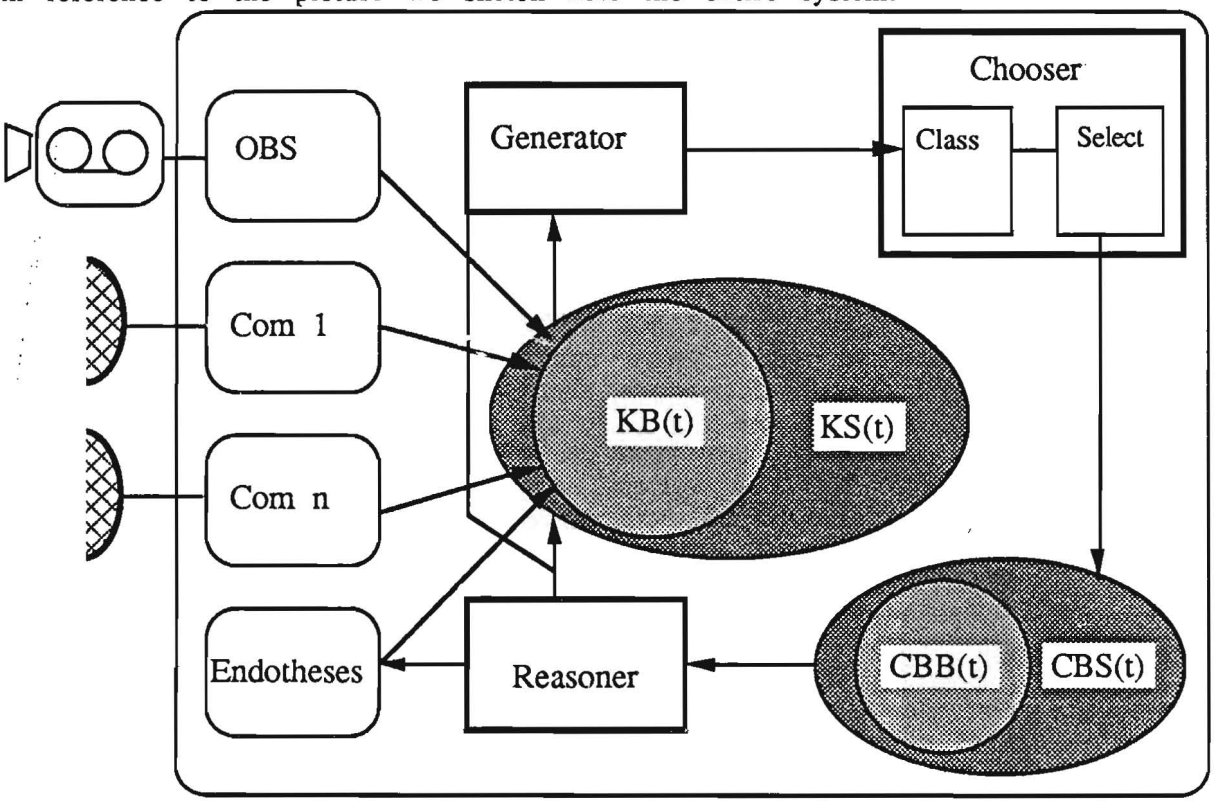
where *A* is an F.O.L sentence; among the assumptions really used in the derivation of A *OSO* contains those whose source is an *observation*, *OSi* contains those whose source is a *comunication* received from the agent *i* and *OSE* contains those introduced ipothetically by the agent himself; $OS = OSO \cup \Sigma_i OS_i \cup OSE$ is the *Origin Set* of A; *RS* is the *Restriction Set*; it contains all the sets of assumptions that unioned with the OS produce a strongly-inconsistent set (see below).

An *assumption* is an SWM whose OS contains only the SWM's sentence. We define *contradiction*, a couple of SWMs $<A,OS_1,RS_1>$ and $<\neg A,OS_2,RS_2>$. The set $OS_1 \cup OS_2$ from which has been derived the contradiction is defined to be a *strongly-inconsistent* set. A *weakly-consistent* set is a not strongly-inconsistent one.

The *Knowledge Base* KB(t) is the set of all the assumptions introduced by the Reasoner (see below) up to t. The *Knowledge Space* KS(t) is the set of all the sentences deductively derived from KB(t) by the Reasoner up to t. A *Belief Base* BB(t) is a subset of KB(t) such that it is weakly-consistent and it is *maximal with respect to KB(t)* (if augmented with whatever else assumption of KB(t) it becomes a strongly-inconsistent set). The *Belief Space* $BS_{BB(t)}(t)$ joined with a Belief Base BB(t) is the set of all the sentences derived from BB(t) up to t.

1.2 THE BELIEF REVISION SYSTEM.

With reference to the picture we sketch here the entire system.



1.2.1 The REASONER.

Its essential task is to clock simulated time providing the assumption of a new SWM in KB(t), or the deduction of a new SWM in KS(t). The first activity is intended to model forms of plausible reasoning [Davis 90] (abduction, induction, default reasoning [Gen-Nils 87] etc.); the second activity is intended to model the limited deductive ability of a real reasoning agent. $KB(t) \subseteq KS(t)$ because assumptions are logical consequences of themselves. No sentences will ever be removed from KS(t). We call *Current Belief Base* CBB(t) the particular Belief Base chosen by the Chooser (defined below) as the preferred one. We call *Current Belief Space* the set $CBS(t)=BS_{CBB(t)}(t)$. The intended meaning for CBS(t) is to be the most believable and maximal piece of knowledge actually available for the reasoning agent. Probably, for best results, it would be preferable to limit at CBS(t) (instead of at the entire KS(t)) the input of the Reasoner (as depicted in the picture) but we see no serious advantages to be so drastic.

1.2.2 The Belief Bases' GENERATOR.

For our purpose, it would be very desirable for the agent's belief base CBB(t) to be consistent. Unfortunately, practical FOL-based systems have to restrict themselves to consider only limited forms of Consistency because of the indecidibility of the validity problem. Previously defined Weak-Consistency is our limited form of Consistency. It seems also desirable for an agent to use as more informations as possible in its reasoning. Hence our choice to impose maximality for CBB(t). Notice that this maximality is intended with respect to all the assumptions in KB(t) and not, as usually, with respect of all the sentences of the Language; this is because we give no importance to the sentences not already introduced in the memory. Each event is a clock pulse for the Context Generator. It searches all over KS(t) for a contradiction If it succeeds it records the discovery of the strongly-inconsistent set and redefines the Situation S(t) of all the Belief Bases in KB(t) (see the *Updating Restriction Set* rule in [Mar-Shap 87] for details).

1.2.3 The CHOOSER.

After the discovery of a new contradiction the agent is in a position to revision its beliefs. It is not the case to select which belief is to be thrown away to remove the contradiction, but, quite more generally, to choose which is the *new preferred Belief Base* among them in S(t). This is what we mean by "Belief Revision" and this is the task of the Chooser; it is an appropriate machine that takes S(t) as input and gives the new preferred Context CBB(t) as output. We think the Chooser as the cascade of two components: the Classifier and the Selector. The Classifier takes KB(t) as input an gives as output the list of all the assumptions in KB(t) ordered according to some specific criteria. The Selector takes as input the list passed from the Classifier and the situation S(t) passed by the Generator and gives as output CBB(t).

This system shows both *foundational* and *coherence* nature. From [Galliers 89]: "Foundation theory considers new beliefs are only to be added on the basis of other justified beliefs, and beliefs no longer justified are abandoned", this is the case of beliefs corresponding to the deductively derived sentences which are in CBS(t) and are added only on the basis of the set of assumptions CBB(t); "Coherence theory represents a conservatism whereby justification is only a requisite condition of believing if there is a special reason to doubt a belief", this is the case of beliefs corresponding to the assumptions whose permanence in CBB(t) is only due to their being not strongly-inconsistent with the others in the same Belief Base; the assumptions in a Belief Base are there because there isn't a valid reason for their not being there.

In order to produce the list of the assumptions, the Classifier needs not only some credibility-importance criteria to judge them but also a strategy to manage those criteria. These could in fact be used at least in two different ways:

a) they could be sorted according to their importance (the importance of the criteria themselves) and used in cascade as selective filters on the assumptions, or
b) it could be assigned a weight to each one of them according to their importance and then they could be used as tests score on the assumptions, reporting the degree with which they are satisfied.

We have developed some algorithms based on the first (non-numeric) strategy.

2.1 THE MULTI-AGENT SETTING.

We distinguish at least three kinds of sources:

- *perception* (typically vision) gives a first direct information about the state of the world (objects' and agents' spatial positions etc.) and about spatial events or occurring actions;
- *communication*: each agent is able to exchange informations employing a certain physical channel and appropriate communication protocols; agents are not necessarily sincere and competent;
- *reflection*: for the sake of realism we admit the presence of assumptions engendered by some forms of hypothetical reasoning internal to the agent; we call them *Endothesis* and we discuss below a problem with them.

In addition we see the presence of "a priori" assumptions that, for our purpose, could be though as innate to the agent; they represent, typically, the rules (causal or not) of the knowledge domain under consideration but we think them as not removable, therefore, not assumptions at all.

2.2 SOME CRITERIA FOR JUDGING ASSUMPTIONS IN A MULTI-AGENT SETTING.

The following is a proposal list.

1. Assumptions derived from observation are stronger then those derived from communication. Observation is taken as a sort of Super-Agent.

1b. Assumptions derived from communication which are in contrast with sets of assumptions all derived from observation have no strengh at all.

2. The sources multiplicity confirms the assumption.

3. The more the conflicts with other assumptions, the weaker the assumption.

4. The OSs of SWMs with the same wff confirm each others because of their mu-

tual coherence.

Two criteria modelling psychological attitudes.

5. *Belief Conservativeness*: it is stronger the assumption supporting more SWMs.

6. *Goals Conservativeness*: it is stronger the assumption supporting more goals (assuming a planner working on the system).

It is important to consider also the reliability of the agents.

7. The less reliable the agent who made a communication, the weaker the assumption derived from it . We could estimate the agent's reliability by:

- Self-Inconsistency (he made communications mutually inconsistent)
- average of Inconsistency of the assumptions derived from communications received from that agent with respect of all the other assumptions derived from observations in KB(t) (we choose to not consider the conflicts with other assumptions in order not to punish competency).

We emphasize the importance of the 4th criterion in giving a prize to coherence. Let $Th(K_t)$ represent a scientific theory based on a set of assumptions K_t not derived from observation and let K_0 be a set of assumptions all derived from observation; if there is a couple of SWMs in KS(t), named $\mathcal{S}$ and $\mathcal{T}$, where $wff(\mathcal{S})=wff(\mathcal{T})$, $OS(\mathcal{S})=K_t$ and $OS(\mathcal{T})=K_0$, then K_0 could be intended to represent the experimental evidences of the scientific theory so that it is justified its reinforcement over K_t.

The 7th criterion could be seen as a preprocessor that gives a weight to each assumption derived from communication. The criteria 1-4 should be able to manage these weights.

2.3 THE ENDOTHESES.

Realistic situations require much complex treatments. Among other considerations, we endorse the need for assumptions "internal" to the agent; we think them as auxiliary beliefs, functional to the reasoning process which is going on. That's the rationality for the Endotheses. They could be the result of the application of some sort of plausible inference rules, it may be "induction" (i.e. from α and β infer $\alpha \rightarrow \beta$) or "abduction" (i.e. from β and $\alpha \rightarrow \beta$ infer α) or a non-monotonic default rule. These Endotheses are treated normally by the Generator; that is, an Endothesis α belongs to every Belief Base not containing a subset that is strong-inconsistent with α. This implies that an Endothesis α hypothetically derived from a set of assumptions $\{\alpha_1,..,\alpha_i\}$ in a Belief Base, can as well belong to other Belief Bases not containing $\{\alpha_1,..,\alpha_i\}$. This could seem strange but it is in accord with the Principle of Positive Undermining [Harman 86]: the lack of justification is not a good reason to remove a belief; we think that this principle is more appropriate for beliefs plausibly derived from a set of assumptions than it is for beliefs which are logical consequences of the set of assumptions. The real problem with the Endotheses is that it isn't clear how they are to be treated by the Chooser; because of the arbitrarity with which they can be introduced we can't fix criteria for the Chooser to process them. The problem is that their introduction is again a casual element in the reasoning story and we are no more favourably disposed towards such casualness. We've been well disposed towards the casualness inherent the story of the introduction of assumptions *from the outside* of the agent (by communication and observation). They are regarded as "interrupts" to be processed, and the change of the deductive theory following the arrive of one of them is justified by the real change of the agent's cognitive state. We've been not so well disposed towards the casualness inherent the story of the derivations of new SWMs in KS(t). However, if the Inconsistency of a set is revealed in its being strong-inconsisten depends on the story of the deductions made by the Reasoner. But Strong-Inconsistency itself does not depend on that story. So, if the property that defines a Belief Base is Weak-Consistency (not Consistency) then we have nothing to worry about the casualness of the deductions.

But now, we are not well disposed towards the casualness inherent the story of the introduction of assumptions *from the inside* of the agent (the Endotheses). It could be objected that these events too could be regarded as interrupts changing the agent's

cognitive state and we've just accepted the fact that casual elements internal to the Reasoner activity (deductions) can affect the agent's cognitive state. However, our resolution is to reduce this casualness simply *giving the lowest importance to the Endotheses when subjected to the Chooser's processing.*

CONCLUSIONS.

This paper deals with the concept of Assumption Based Belief Revision in a Multi-Agent environment, that is how to consider also the *sources* of the information (i. e. who gave it) in the general belief revision process. We have briefly presented:

a) a rather innovative general framework for assumption based Belief Revision

b) some abstract criteria to deal with an agent's knowledge base built upon observations, internal hypotheses and several other agent's informative contributions.

The former topic is based on

- choosing a new preferred Belief Base versus removing the beliefs causing the contradiction,
- achieving our defined Weak-Consistency versus achieving Consistency

The latter topic covers:

- the definition of very general criteria to associate each agent advise (or agent himself) with an implicit credibility factor
- the discussion of a criterion to judge the (our defined) Endotheses, that is assumptions derived internally to the agent
- the discussion of strategies that use these criteria to compute the new preferred context

The overall system exhibits an enviably anthropomorphous behaviour.

WHAT IS MISSING.

This research belongs to a Multi-Agent planning project, but several examples show that the system fits as well in police investigations or detective stories. The system could also be seen as a module in expert systems regarding not well established knowledge with multiple experts contrasting contributions. The real limitation of this approach is that it doesn't reason about *why* an agent gave an information; we don't take explicity in count the *intention* [Cohen 90] of the agents or their dependence relations [Castelfranchi 91] as usefull elements to judge their utterances. The key element is only consistency with the observation. It's too little for some pourpose.

REFERENCES.

[Castelfranchi 91]: Cristiano Castelfranchi & Maria Miceli & Amedeo Cesta, Dependence Relations among Autonomuos Agent, in this proceedings.

[Cohen 90]: Philip Cohen & Jerry Morgan & Martha Pollack, Intentions in Communication, The MIT Press, Cambridge, Mass., 1990.

[Davis 90]: Ernest Davis, Representation of Commonsense Knowledge, Morgan Kaufmann Publisher, san Mateo, Calif, 1990

[De Kleer 86]: Johan de Kleer, An Assumption Based Truth Maintenance System, Artificial Intelligence 28, pp. 127-162, 1986.

[Dragoni 91] Aldo Franco Dragoni, A Model for Belief Revision in a Multi-Agent Environment, Tech. Rep., Computer Sciences Inst., University of Ancona (Italy) 1991.

[Galliers 89]: Julia Rose Galliers, Modelling Autonomous Belief Revision in Dialogue, Tech Rep. Cambridge University Comp. Lab., Cambridge (England), 1989.

[Gen-Nils 87]: Michael Genesereth, Nils Nilsson, Logical Foundations of Artificial Intelligence, Morgan Kaufmann Publisher, san Mateo, Calif, 1987

[Harmann 86]: G. Harman, Change in View - Principles in Reasoning, The MIT Press, Cambridge, Mass., 1986.

[Mar-Shap 86]: Joao P. Martins, Stuart C. Shapiro, Theoretical Foundations of Belief Revision, in: J.Y. Halpern ed., *Theoretical Aspects of Reasoning about Knowledge*, (Morgan Kaufmann, Los Altos, CA, 1986).

[Mar-Shap 87]: Joao P. Martins, Stuart C. Shapiro, A Model for Belief Revision, Artificial Intelligence 35 (1), pp. 25-79, 1988.

On Being Responsible[1]

N.R.Jennings
Dept Electronic Engineering,
QMW, University of London,
Mile End Road,
London E1 4NS
UK.
n_jennings@eurokom.ie

1 Joint Action: An Introduction

Within a multi-agent environment, there are numerous possibilities for the way in which community members can operate and interact. The aim of this paper is to produce a behavioural framework for social interactions in which groups of (semi-)autonomous agents decide they wish to solve a particular problem together and then collaborate to attain the desired state (this activity will be referred to as group/joint problem solving). The framework defines prerequisites for joint action and also how agents should behave (both in their own problem solving and with respect to other group members) once they have agreed to participate in joint problem solving.

Typically in a community of autonomous agents, one of the primary motives for such joint action is when no individual within the community is capable of achieving a desired objective alone; only by combining with others in a structured manner can the target be reached (contrast this with objectives such as load balancing). It is typically a reciprocal process in which participating agents augment their objectives and problem solving to comply with those of others - hence it is a fairly sophisticated form of cooperation. It requires greater knowledge, awareness and reflection by an agent both with respect to its own problem solving objectives and about their compatibility with the objectives of others (contrast with task and result sharing [1]).

Joint action, by definition, requires an objective the group wishes to achieve - it is the "glue" which binds the team together. As a consequence of the autonomous nature of the agents, each team member will only participate if it can derive some benefit from the interaction. This differs from the more traditional approach in which agents are assumed to enter into interactions merely if they are requested to do so (i.e. they are benevolent [2,3]). However merely having a common objective is not sufficient for obtaining a collective goal - agents also need to agree upon a means of reaching the target state. Imagine trying to lift a heavy object if the lifting positions of the participants and the relative forces exerted by each participant had not been agreed upon beforehand! Agreeing such a solution may be done in an incremental fashion, nevertheless such a solution must be agreed at a certain level of abstraction. As activity progresses the solution may be refined or modified to better fit prevailing circumstances. Having a common objective and solution means participant's actions can be phrased in terms of "doing their bit" [4]. In the collaborative lift example, an agent lifts the object at one end as a means of contributing to the group objective of lifting the object.

Previous work on collaborative problem solving [4,5,6,7,8] has been deficient because it failed to describe the complete mental state of the participants. This state, which we call **joint responsibility**, internalises for each team member the notion of being in a group. It provides beliefs about how others

1. The work described in this paper has been partially supported by the ESPRIT II project P2256 (ARCHON) whose partners are: Krupp Atlas Elektronik, JRC Ispra, Framentec, Labein, QMW, IRIDIA, Iberduero, ERDC, Amber, Technical University of Athens, University of Amsterdam, Volmac, CERN and University of Porto.

will act, both in performing their intentions and in participating in collaborative interaction per se. Such guidelines are especially important in dynamic and complex environments in which agents' aims and objectives are likely to alter during the course of a cooperative interaction. In addition to providing behavioural guidelines joint responsibility defines the pre-requisites which need to be satisfied before joint action can commence. In the remainder of this paper the notion of joint responsibility as a pre-requisite for joint action is developed. Section two describes an example problem domain in which joint problem solving by a team of autonomous agents is beneficial to all participants and also enhances the quality of the output to the user. Section three introduces, using a logic-based formalism, the notions of joint responsibility and provides pointers to where previous work in this field has been deficient.

2 Fault Recovery in Electricity Distribution Networks

The domain upon which the principles related to joint action will be illustrated is that of fault recovery in electricity distribution networks[2]. The described scenario involves three pre-existing agents each of which has a set of clearly defined goals and is capable of sophisticated problem solving in its own right. The agents, together with an indication of the message flows between them, are shown below.

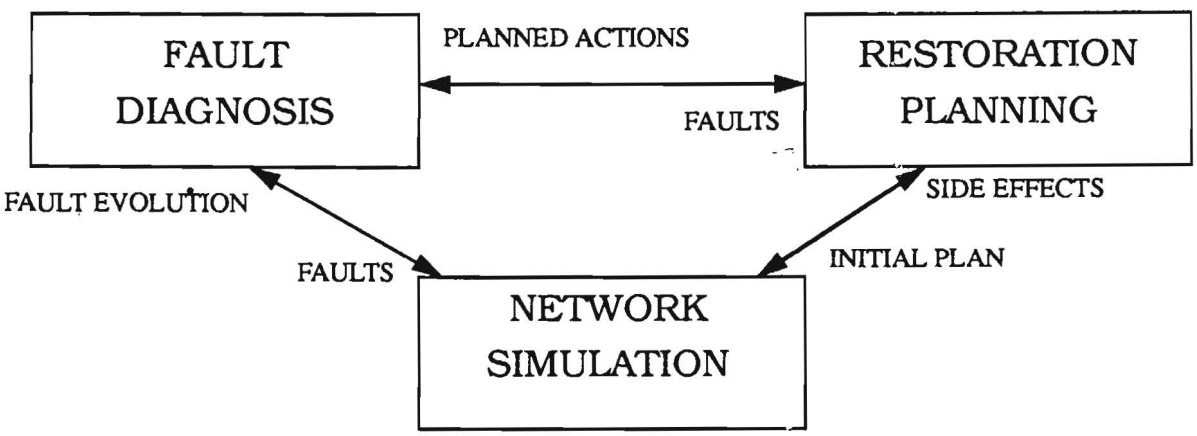


Figure 1

The fault diagnosis agent's role in this example is to indicate to the other two agents that a fault has been detected in the network. The restoration planning agent (RA) is responsible for constructing a maintenance plan once faults have been detected - such a plan will instruct the operator to perform certain sequences of operations in a well defined order. The network simulation agent (NSA) is capable of running and explaining what-if simulations of the network based on the settings of certain key parameters. The joint action which can be instantiated between the NSA and RA is in the area of producing restoration plans whose actions will not cause further parts of the network to fail (i.e. "sensible" restoration plans). As a standalone system, the plans suggested and devised by the RA may lead to further faults as the specified operations may result in overloading a currently working component, causing it to fail. However if the RA's tentative restoration plan is sent to the NSA then its effects can be predicted, problem areas highlighted and the RA informed. If major problems are identified and the RA decides to significantly revise the original restoration plan then there may be further interaction with the NSA before an acceptable plan can be generated. If only minor modifications are made, or the RA deems the highlighted risks acceptable, then the restoration plan may be slightly altered and because of the relatively minor nature of the changes no further interaction is undertaken with the NSA.

2. The example is loosely based on an ARCHON application [9,10].

3 Joint Responsibility

Joint responsibility defines a behavioural framework for participants who wish to engage in collaborative problem solving. It defines the conditions which need to be satisfied before joint action can be initiated and a code of conduct specifying how agents should react when the joint action becomes unsustainable. Joint responsibility will be defined using a logical formalism, similar to that described in [11]. This formalism has the usual connectives of a first order language (^ AND, v OR, ~NOT) - as well as operators for propositional attitudes. BEL(x, p) and GOAL(x, p) mean agent x has p as a belief and a goal respectively, MB(x, y, p) that x and y mutually believe p. Dynamic logic constructs are also used: $\Box$p means p is always true and $\Diamond$p that p will eventually be true. p?;a means "action a with p holding initially", and analogously for a;p?. As stated previously, this analysis will be exemplified using two agents (the restoration agent (RA) and the network simulation (NSA)) for simplicity, although it can of course be applied to communities of arbitrary size.

3.1 Common Goals and Joint Persistence

The first step to achieving joint action is that a group of two (or more) agents realize that they have a common objective (intention[3]) and that this intention can only (best) be fulfilled by collaborating with others. Once this is mutually believed by all participants, a common goal exists and each individual participant becomes committed [11] to achieving that objective. However as Levesque et al. point out this is not a sufficiently sturdy foundation upon which robust joint action can be based [8]; it is particularly fragile if agents intentions change (i.e. they reach a state in which they are no longer committed to attaining the common objective). To rectify these problems, they propose the notion of joint persistent goals (JPGs) [8] in which groups of agents become jointly committed to a common aim. The properties of JPGs can best be illustrated using an example. Suppose the RA and the NSA have established a JPG of producing a "sensible" restoration plan and then at some later stage one of the agents (say the NSA) no longer desires this objective (because the user has asked it to run a what-if question on the network as a very important task). Should it simply drop the common goal without informing the RA?, meaning that the RA will be left waiting indefinitely. Clearly not! Therefore in the interests of robust group problem solving, a JPG requires that the NSA adopts the goal of informing the RA of its change of intention. Thus, JPGs define the conditions under which a joint commitment to a goal can be dropped and also how participants should act when they find themselves in such a situation.

3.2 Solution Commitment

Contrary to the claims of Levesque et al. [8], having a JPG is not sufficient for obtaining joint action. JPG's only specify that agents have a common desire to reach a target state, they do not specify how agents are to reach this state. Agreeing upon a means of reaching the state is nearly as important as the desire to reach the state itself. Therefore although NSA and RA may be able to agree that they want to produce a restoration plan together, unless they can agree upon a means of achieving this then joint action will not follow. In some circumstances, such agreement may be impossible because of the autonomous nature of the agents involved; both agents are likely to have several objectives at any one time and these must be balanced with the desire to produce a sensible restoration plan. If they have insufficient resources to devote to the problem or it conflicts with their other intentions then it may be impossible for them to converge upon the necessary common plan even though they share a common objective.

At this stage we are not concerned with the mechanisms used for achieving the common solution[4] (eg one agent may derive the plan and pass it to others who obediently adopt it or the participants may compile

3. Intentions have been ascribed a variety of differing meanings (eg [11, 12, 13])- within this context they specify a desired or target state **without** consideration of how that state is to be attained.

separate plans and then coordinate them, [14,15]); rather we are concerned with the fact that they must agree upon the **principle** that a common plan is needed to tackle the joint problem.

Before this intuition can be formalised, a simple plan representation language needs to be defined. In a multi-agent environment not only do the actions to be performed need to be specified, but also the agent which will perform the action needs to be given [16]. In order to stress that several agents are working together towards a common objective it is convenient to represent intentions in terms of the agents which will work together to achieve them, rather than the other way around. Therefore the fact that a set of agents $\{\alpha_1,..,\alpha_n\}$ will work together in order to try and reach state σ will be represented as follows: $< \{\alpha_1,..,\alpha_n\}, \sigma >$. Let the set of all agents existing in the environment be denoted by A; unless stated to the contrary, all groups of agents are just a subset of the members of A.

In any complex environment, intentions will typically be composed of sub-intentions which are themselves decomposable - the solution graph for σ being represented by Σ_σ. The nodes without successors (when the graph has been fully expanded) correspond to atomic units of activity which are executable by individual agents. The various stages of intention "execution"[5] can be expressed as follows:

$$\text{EXECUTE/EXECUTING/EXECUTED}(< \{\alpha_1,..,\alpha_n\}, \sigma >, \Sigma_\sigma)$$

which respectively mean that Σ_σ will be executed next, is being executed now or has been executed, for the purpose of attaining σ by agents $\{\alpha_1,..\alpha_n\}$. Underlying this definition is the assumption that at least one team member (or a subset of them) is (are) capable of realising the constituent sub-intentions and that team members will not attempt actions which they cannot execute to some degree.

Typically the solution of a joint action will require actions which need to be coordinated (i.e. a relationship exists) with those of other agents and some which can be executed independently of the activities of other agents. Solutions are therefore likely to contain interrelated components:

$$\Sigma_\sigma = \{\sigma_1\ \Re_{1,2}\ \sigma_2,\ \sigma_3\ \Re_{3,4}\ \sigma_4,....\}$$

$\Re_{1,2}$ defines the relationship between σ_1 and σ_2[6]. This will usually be temporal (eg BEFORE, AFTER, SIMULTANEOUSLY) although it may also express constraints. Such relationships are an integral component of the solution specification and if they are not satisfied then the desired objective cannot be guaranteed by solution Σ_σ. Hence fulfilling an intention means reaching the desired state and satisfying any relationships which exist between that intention and others[7]:

$$(\forall <\{\alpha_w ... \alpha_x\}, \sigma_i> \in \Sigma_\sigma)\ (\exists <\{\alpha_y ... \alpha_z\}, \sigma_j> \in \Sigma_\sigma)\ \Re_{i,j} \supset$$
$$\text{MB}(\{\alpha_w ... \alpha_x\}, \Re_{i,j})?;\ \text{EXECUTE}(<\{\alpha_w ... \alpha_x\}, \sigma_i>, \Sigma_{\sigma i})$$

To illustrate this formalism, the intention of producing a sensible restoration plan can be expressed as follows: σ = < {RA, NSA}, SENSIBLE-RESTORATION-PLAN> and one solution for achieving this is:

Σ_σ = {<RA, TENTATIVE-RESTORATION-PLAN> ***BEFORE***
<NSA, CHECK-PLAN-FOR-OVERLOADS>

<NSA, CHECK-PLAN-FOR-OVERLOADS> ***BEFORE***
<RA, REFINE-RESTORATION-PLAN>}

---

4. A common solution should be understood in a broad context, it is intended to embrace both agreeing upon a strategy or plan for coming to a common set of actions as well as the set of actions themselves.
5. Execution in this context corresponds to expansion of those nodes without successors if a node is expandable or processing of atomic units if not.
6. $\Re_{1,2} \neq \Re_{2,1}$
7. Typically fulfilling relationships is a process requiring communication and synchronization between the responsible agents. However details of how this is achieved are beyond the intended scope of this paper

The success of a solution in reaching its desired objective is the final component of the plan representation language:

• ACHIEVE($<\{\alpha_1,..,\alpha_n\}, \sigma>, \Sigma_\sigma$) $\Leftrightarrow \neg\sigma \wedge$ EXECUTE($<\{\alpha_1,..,\alpha_n\}, \sigma>, \Sigma_\sigma$); σ?

meaning if Σ_σ is executed next σ, which did not hold before this sequence of actions, will hold as a direct consequence of performing the specified actions.

It is now possible to express the first pre-condition for joint action, namely that the participants must agree upon the principle that a common solution is needed if the objective is to be achieved:

NEED-COMMON-SOLUTION ($<\{\alpha_1,..,\alpha_n\}, \sigma>$) $\Leftrightarrow$
MB($\{\alpha_1,..,\alpha_n\}$, $\Diamond \exists\Sigma_\sigma$ EXECUTE($<\{\alpha_1,..,\alpha_n\}, \sigma>, \Sigma_\sigma$) $\vee \Box\neg\sigma$)

This notion that joint action requires a common solution is also expressed in the work of Grosz and Sidner [5,6]. In addition to the shortcomings of their work expressed in [4,16], their formulation is also lacking in the following key areas:

- it does not explicitly address the problem of interrelated intentions (actions)
- it does not specify how team members should react if (for whatever reason) they believe the plan is no longer appropriate for reaching the common objective
- at no stage is one common solution adopted for attaining the common objective, it merely states that there is mutual belief about a means of achieving the desired state
- it is possible for agents which are unable to contribute anything to the overall objective to be involved in the common plan

All of these shortcomings are tackled by the joint responsibility definition; the first by the plan formulation language chosen and the last in the next section. This section concentrates on the second and third points, describing a framework which specifies under what conditions an agent can become uncommitted to the agreed plan and what actions should be undertaken in such circumstances.

There are several circumstances in which the commonly agreed plan may be inappropriate, illustrations are taken from the sensible restoration plan example:

- the plan's objective may already hold

  eg the NSA may calculate that the proposed tentative plan will cause no additional problems in the network - therefore the joint objective of producing a sensible restoration plan has already been met and no additional work is required

- following the plan steps may no longer result in the desired state (due to changed circumstances, for example) - ***INVALID*** plan

  eg the NSA is informed by the diagnosis agent that the network status has changed substantially since the simulation to judge the effect of the tentative restoration plan was started. This means the analysis produced will be out of date (it is missing important new information) and hence it is impossible to tell whether the restoration plan is safe or not without redoing the simulation. It will not be worth redoing the simulation because the tentative restoration plan will be completely altered to take the new information into account, therefore following the agreed solution may not produce the desired result.

- one of the specified plan steps may no longer be achievable - ***UNATTAINABLE*** plan

  eg the operator (who has higher priority than the RA) requests the NSA to run a very time-consuming simulation. Carrying out this user request means that the NSA will not be able to meet its

agreed action of analyzing the tentative plan (in computational terms the notion of "never" corresponds to at least for the foreseeable future).

- one of the plan steps which should have been performed has not been performed, or a relationship between plan steps has not been upheld - ***VIOLATED*** plan

  eg the RA has sent a tentative plan to the NSA which has run a simulation and highlighted potential problem areas. However because a major incident has occurred on the network and many more faults have been generated the RA decides that rather than refining the tentative plan it is better to try and generate a new plan because the information will be that much more up-to-date. In this case, the RA has violated the agreed plan by not performing the refinement task.

These notions can be formalised in the following manner:

- INVALID $(<\{\alpha_1,..\alpha_n\}, \sigma>, \Sigma_\sigma) \Leftrightarrow \Box\sim$ACHIEVE$(<\{\alpha_1,..,\alpha_n\}, \sigma>, \Sigma_\sigma)$
- UNATTAINABLE $(<\{\alpha_1,..\alpha_n\}, \sigma>, \Sigma_\sigma) \Leftrightarrow (\exists <\{\alpha_w,..,\alpha_x\}, \sigma_i> \in \Sigma_\sigma) \Box\sim\sigma_i$
  where $[\{\alpha_w,..,\alpha_x\} \subseteq \{\alpha_1,..,\alpha_n\}]$
- VIOLATED $(<\{\alpha_1,..\alpha_n\}, \sigma>, \Sigma_\sigma) \Leftrightarrow \sim$EXECUTED$(<\{\alpha_1,..,\alpha_n\}, \sigma>, \Sigma_\sigma)$

If the joint plan is to be successful, all participants must be committed (i.e. endeavour to perform actions they are obliged to). However there are circumstances in which it would be rational for an agent to stop being committed - this includes those outlined above and the case in which one (or more) of the other team members is no longer committed. Unless these conditions prevail, an individual agent (α) remains committed (I-COMMIT-CONDS) to solution Σ_σ as a means of achieving σ:

I-COMMIT-CONDS$(\alpha, <\{\alpha_1,..\alpha_n\}, \sigma>, \Sigma_\sigma) \Leftrightarrow$ $[\alpha \in \{\alpha_1,..,\alpha_n\}]$
BEL$(\alpha, \sim\sigma) \wedge$
BEL$(\alpha, \sim$INVALID$(<\{\alpha_1,..\alpha_n\}, \sigma>, \Sigma_\sigma)) \wedge$
BEL$(\alpha, \sim$UNATTAINABLE$(<\{\alpha_1,..\alpha_n\}, \sigma>, \Sigma_\sigma)) \wedge$
BEL$(\alpha, \sim$VIOLATED$(<\{\alpha_1,..\alpha_n\}, \sigma>, \Sigma_\sigma)) \wedge$
BEL$(\alpha, (\forall\alpha_i \in \{\alpha_1,..,\alpha_n\}$ I-COMMIT-CONDS$(\alpha_i, <\{\alpha_1,..\alpha_n\}, \sigma>, \Sigma_\sigma)))$

Once an agent becomes uncommitted to a solution (for any of the reasons outlined) it cannot simply ignore its future responsibilities or carry on as if nothing had happened. Rather it must endeavour to inform other team members that it is no longer committed to the solution. When other agents receive this message a re-planning phase may be initiated or the overall objective may be altered, however we are not concerned with this behaviour here. It is now possible to formalize our intuitions about how individuals within the team should act once a common solution has been derived and agreed upon - this behaviour will be called individual solution commitment (ISC):

ISC$(\alpha, <\{\alpha_1,..\alpha_n\}, \sigma>, \Sigma_\sigma) \Leftrightarrow$ $[\alpha \in \{\alpha_1,..,\alpha_n\}]$
UNTIL $\sim$I-COMMIT-CONDS$(\alpha, <\{\alpha_1,..\alpha_n\}, \sigma>, \Sigma_\sigma)$,
$(\forall <\{\alpha_w,..\alpha_x\}, \sigma_i> \in \Sigma_\sigma) \wedge (\alpha \in \{\alpha_w,..\alpha_x\}) \supset$ $[\{\alpha_w,..,\alpha_x\} \subseteq \{\alpha_1,..,\alpha_n\}]$
BEL$(\alpha, \Diamond$EXECUTE$(<\{\alpha_w,..\alpha_x\}, \sigma_i>, \Sigma_{\sigma i})) \wedge$
EXECUTE$(<\{\alpha_w,..\alpha_x\}, \sigma_i>, \Sigma_{\sigma i}))$
WHEN GOAL$(\alpha,$ MB$(\{\alpha_1,..\alpha_n\}, \sim$I-COMMIT-CONDS$(\alpha, <\{\alpha_1,..\alpha_n\}, \sigma>, \Sigma_\sigma)))$[8]

8. UNTIL p,q WHEN r: until p is true, q will remain true. When (if) p becomes true, r will become true

From this definition it is apparent that a group member will try and fulfil its obligations specified in the agreed solution whilst it is still committed to that solution as a means of achieving the desired result - on becoming uncommitted it endeavours to inform others of this fact. This behaviour ensures that whenever the chosen solution is unsustainable every effort is made to ensure that all team members are made aware of this fact so that computational effort is not wasted.

Combining the results of this section there are clearly two facets concerned with actions for achieving a target state: there is the principle of agreeing to the need for a common solution and also a definition of how group members should behave once such a solution has been agreed upon. These two components can be joined together into a single proposition called solution commitment:

$$\text{SOLUTION-COMMITMENT}(<\{\alpha_1,..,\alpha_n\}, \sigma>) \Leftrightarrow$$
$$\text{MB}(\{\alpha_1,..,\alpha_n\}, \text{NEED-COMMON-SOLUTION}(<\{\alpha_1,..,\alpha_n\}, \sigma>)) \wedge$$
$$(\forall\, \alpha_i \in \{\alpha_1,..\alpha_n\}\ \text{ISC}(\alpha_i, <\{\alpha_1,..\alpha_n\}, \sigma>, \Sigma_\sigma))$$

Returning to our example, this means the RA and the NSA have to agree upon the principle that a common plan is needed for producing a sensible restoration plan. Once such a solution has been agreed, both agents will endeavour to do their parts (eg the RA will generate a tentative plan, the NSA will highlight any potential problem areas and the RA will then refine its tentative plan based on this information). They will continue to do this until either the task is completed satisfactorily or one of them finds the agreed solution unsustainable or one of them discovers the other is no longer committed to the solution.

3.3 Contributions

One attribute completely missing from all descriptions of joint actions is that of group minimality; to be included in a group an agent must be able to contribute something positive to that group's activity. This facet differs from previous attributes in that it is pragmatic rather than conceptually essential. In our restoration plan example, the joint action is between the RA and the NSA and as the scenario is described it makes no sense for any other agent (eg the fault diagnosis agent) to be involved because it is unable to carry out useful problem solving in the context of producing a sensible restoration plan. As we have alluded to, coordinating group problem solving may be a time consuming activity and because cost is proportional to the size of the group it makes sense to only include those agents which carry out activities beneficial to the group's objectives.

There are two ways in which an agent can contribute to the attainment of a group goal: it can perform an act which is part of the agreed solution (positive contribution) or it may refrain from performing an action which would interfere with the agreed solution (non-negative contribution). Imagine a team of agents trying to stack blocks B1, B2 and B3 - a positive contribution could be putting B2 onto B1, a non-negative one not unstacking B2. However due to space limitations we will only consider positive contributions.

Firstly we need to define exactly what it means for a sub-intention to contribute to the attainment of an intention. At this stage, as no definite solution has been agreed upon, a sub-intention can be said to contribute to the overall objective if it is a component of any solution (however inefficient or cumbersome) which reaches the target state:

$$\text{CONTRIBUTES}(<\{\alpha_j,..,\alpha_k\}, \sigma_i>, <\{\alpha_1,..,\alpha_n\}, \sigma>) \Leftrightarrow$$
$$(\exists\, \Sigma_\sigma\ \text{ACHIEVE}(<\{\alpha_w,..,\alpha_x\}, \sigma>, \Sigma_\sigma) \wedge <\{\alpha_j,..,\alpha_k\}, \sigma_i> \in \Sigma_\sigma$$

where $[\{\alpha_w,..,\alpha_x\} \subseteq \{\alpha_1,..,\alpha_n\}]$ and $[\{\alpha_j,..,\alpha_k\} \subseteq \{\alpha_w,..,\alpha_x\}]$.

The first step in being involved in group problem solving activity is for the individual to believe that it is capable of offering something to the group. Once an individual is convinced of this fact, it then has to convince others that it's inclusion will be to the group's benefit. Concentrating on the former, an agent is

capable of contributing to the group goal if it can solve to a sub-intention which is a component of a potential overall solution:

$$\text{CAN-CONTRIBUTE}(\alpha, \langle\{\alpha_1,..,\alpha_n\}, \sigma\rangle) \Leftrightarrow$$
$$\text{ACHIEVE}(\langle\{\alpha_w,..,\alpha_x\}, \sigma_i\rangle, \Sigma_{\sigma i}) \wedge \qquad [\{\alpha_w,..,\alpha_x\} \subseteq \{\alpha_1,..,\alpha_n\}]$$
$$\text{CONTRIBUTES}(\langle\{\alpha_w,..,\alpha_x\}, \sigma_i\rangle, \langle\{\alpha_1,..,\alpha_n\}, \sigma\rangle) \qquad [\alpha \in \{\alpha_w,..\alpha_x\}]$$

The ability of being able to contribute to the attainment of a goal is useless, unless the individual actually intends to participate in the problem solving process:

$$\text{WILL-PARTICIPATE}(\alpha, \langle\{\alpha_1,..,\alpha_n\}, \sigma\rangle) \Leftrightarrow \qquad [\alpha \in \{\alpha_1,..\alpha_n\}]$$
$$\Diamond\text{SOLUTION-COMMITMENT}(\langle\{\alpha_1,..,\alpha_n\}, \sigma\rangle)$$

As stated above, an agent will only be admitted into joint problem solving activity if all group members are firstly convinced that the agent is capable of contributing to the objective and secondly that they believe it will actually participate:

$$\text{MAY-CONTRIBUTE}(\alpha, \langle\{\alpha_1,..,\alpha_n\}, \sigma\rangle) \Leftrightarrow \qquad [\alpha \in \{\alpha_1,..\alpha_n\}]$$
$$\text{MB}(\{\alpha_1,..,\alpha_n\}, \text{CAN-CONTRIBUTE}(\alpha, \langle\{\alpha_1,..,\alpha_n\}, \sigma\rangle)) \wedge$$
$$\text{MB}(\{\alpha_1,..,\alpha_n\}, \text{WILL-PARTICIPATE}(\alpha, \langle\{\alpha_1,..,\alpha_n\}, \sigma\rangle))$$

This is a conservative approach to setting up groups of cooperating agents, in that all other members must agree to the participation of others at this early stage (even before a solution has been developed). A more liberal approach is to weaken this condition and allow agents to participate in the subsequent solution development phase on the basis that they alone believe they can contribute.

3.4 Responsibility At Last!

We are now in a position of being able to draw together all the work specified in this section and fully describe the complete mental state which a group of agents must adopt if they are to jointly solve a common problem together:

$$\textit{JOINT-RESPONSIBILITY}\ (\langle\{\alpha_1,..,\alpha_n\}, \sigma\rangle) \Leftrightarrow$$
$$\textit{MB}\ (\{\alpha_1,..,\alpha_n\}, \textit{JPG}(\langle\{\alpha_1,..,\alpha_n\}, \sigma\rangle)) \wedge$$
$$\textit{MB}\ (\{\alpha_1,..,\alpha_n\}, \textit{SOLUTION-COMMITMENT}\ (\langle\{\alpha_1,..,\alpha_n\}, \sigma\rangle)) \wedge$$
$$\textit{MB}\ (\{\alpha_1,..,\alpha_n\}, (\forall\ \alpha_i \in \{\alpha_1,..\alpha_n\}\ \textit{MAY-CONTRIBUTE}(\alpha_i, \langle\{\alpha_1,..,\alpha_n\}, \sigma\rangle))$$

4 Conclusions

The work presented in this paper is a synthesis and extension of previous work in the fields of multi-agent planning and joint intentions and provides a foundation upon which robust and sophisticated collaborative problem solving can be based. The notion of joint responsibility offers, for the first time, a sufficient definition of the conditions which must be satisfied before joint action can begin and also defines how participants in such actions should behave whilst problem solving. It defines the conditions under which commitment to solutions can be dropped and what an agent should do when it finds itself in a position in which the solution is unsustainable. This type of mental state and behavioural description is important if robust and sophisticated cooperation is to succeed in dynamically changing environments.

References

[1] R.G.Smith & R.Davis, (1981), "*Frameworks for Cooperation in Distributed Problem Solving*", IEEE SMC, 11, 1, pp 61-70

[2] J.S.Rosenschein & M.R.Genesereth, (1985), "*Deals Among Rational Agents*", IJCAI, pp 91-99.

[3] J.R.Galliers, (1989) "*The Positive Role of Conflict in Cooperative Multi-Agent Systems*", Proc MAAMAW 1989.

[4] J.R.Searle, (1990), "*Collective Intentions and Actions*", in Intentions in Communication, (eds P.R.Cohen, J.Morgan & M.E.Pollack), pp 401-416, MIT Press

[5] B.J.Grosz & C.L.Sidner, (1990), "*Plans for Discourse*", Intentions in Communication, (eds P.R.Cohen, J.Morgan & M.E.Pollack), pp 417-444, MIT Press

[6] K.E.Lochbaum, B.J.Grosz & C.L.Sidner, (1990), "*Models of Plans to Support Communication*", Proc AAAI 90, pp 485-490.

[7] R.Tuomela & K.Miller, (1988), "*We-Intentions*", Philosophical Studies 53, 367-389.

[8] H.J.Levesque, P.R.Cohen & J.H.Nunes, (1990), "*On Acting Together*", Proc AAAI, 94-99.

[9] C.Roda, N.R.Jennings & E.H.Mamdani, (1990), "*ARCHON: A Cooperation Framework for Industrial Process Control*", in Cooperating Knowledge Based Systems 1990, (ed S.M.Deen) pp 95-112, Springer Verlag.

[10] N.R.Jennings, "*ARCHON*: An Architecture for Cooperating Systems" in Artificial Intelligence and Simulation of Behaviour Quarterly, Special Issue on Distributed AI, 76.

[11] P.R.Cohen & H.J.Levesque, (1990), "*Persistence, Intention and Commitment*" Intentions in Communication, (eds P.R.Cohen, J.Morgan & M.E.Pollack), pp 33-70, MIT Press

[12] E.Werner, (1989), "*Cooperating Agents: A Unified Theory of Communication & Social Structure*", in Distributed Artificial Intelligence Vol II, (eds Gasser & Huhns), pp 3-36.

[13] M.E.Bratman, (1990), "*What is Intention?*" Intentions in Communication, (eds P.R.Cohen, J.Morgan & M.E.Pollack), pp 15-33, MIT Press

[14] J.S.Rosenschein, (1982), "*Synchronization of Multi-Agent Plans*", Proc AAAI, 115-119.

[15] E.H.Durfee & V.R.Lesser, (1987), "Using Partial Global Plans to Coordinate Distributed Problem Solvers", Proc IJCAI 1987, pp 875-883.

[16] J.R.Hobbs, (1990), "*Artificial Intelligence and Collective Intentionality*", Intentions in Communication, (eds P.R.Cohen, J.Morgan & M.E.Pollack), pp 445-460, MIT Press.

Towards a Semantics of Desires

George Kiss

HCRL
The Open University
Milton Keynes
England
Tel: +44 908 652568
Fax: +44 908 653169
Email: gr_kiss@uk.ac.open.acs.vax

Han Reichgelt

Department of Psychology
University of Nottingham
Nottingham
England
Tel: +44 602 484848
Fax: +44 602 590339
Email: han@uk.ac.nott.psyc

Abstract

As part of an effort to define a unified formal semantics for beliefs, desires and action, this paper sketches a model theory for the axiological aspects of agent theory: hedonic states, likes, goals and values. It pays particular attention o modelling the intensity of likes. The main intuition underlying the model theory is that the axiological aspects of agent theory can be modelled through computational generalisations of physical dynamics. Computational analogues of force, mass and potential are offered.

Introduction

An important part of agent theory appears to be the notion of desires. Several formulations of agent theory have adopted beliefs, desires and intentions as a set of basic notions (the so-called BDI models). However, to our knowledge, so far relatively little has been said explicitly in the AI literature about a theory of desires (Cohen and Levesque, 1985 and in press; Moore, 1985a; Kiss, 1988; Shoham, 1989).

This paper takes some initial steps towards the explicit formulation and formalisation of such a theory. We concentrate on axiological issues, covering hedonic states, likes, goals and values (Kiss, 1988, 1990).

Among the many issues surrounding desires, we select the question of the *intensity* of the attitude of liking for detailed treatment. We think that likes are not the only attitudes that have an intensity aspect. It is common to talk about the strength of beliefs too. We hope to extend our approach to those other attitudes as well in the future.

Differences between the intensities of likes are often called preferences in the literature of decision theory, economics and psychology. Preferences are usually taken as the primary, primitive, notions in the sense that preferences are directly manifested in the choices made by an agent. Likes are therefore treated as *relative, comparative attitudes*. Few disciplines enquire into the mechanisms that might determine such choices and it is usually assumed that it is preferences that are directly available to the agent. Absolute values of liking are usually recovered from behaviourally expressed preferences by some analytical computations from these preferences.

We would like to proceed in the opposite direction and take absolute likes as primary and use these to determine preferences. Our intuition is that an agent has representations of how far it likes various things and when faced with a choice, compares the intensities of its likes to compute a preference. This need not of course exclude mechanisms of context dependence and interaction effects.

Our longer-term research objective is to formulate a unified formal semantics for beliefs, desires and action and to lay foundations for implementation work. This short paper has limited aims. Our main concern is to refine the set of intuitions which were outlined in Kiss (1988, 1990, 1991) and sketch the model theory for a modal logic of liking. The model theory also lends itself for a treatment of modal epistemic operators in the style of Halpern and Moses (1990), although in this paper we will restrict ourselves to axiological issues. We also defer the definition of the syntax and semantics of the logical language, the statement of axioms and the derivation of theorems for another paper.

The main intuition we wish to convey is that the axiological aspects of agent theory are best interpreted in terms of concepts that are computational generalisations of physical dynamics. Demazeau (1991) similarly argues for the importance of concepts borrowed from dynamics for agent design. Traditionally in physics dynamics deals with changes of state in a system and with the causes of these changes, usually conceptualised as forces. Modern developments have turned dynamics into a more abstract area of study, as we shall briefly sketch below.

We propose that dynamics has a natural place in agent theory, since that theory is vitally concerned with (mental) states, their properties, and with the dynamics of sequences of changes in mental state. The interpretation of knowledge and belief as a state of the agent has recently been gaining ground (Rosenschein, 1986; Halpern and Moses, 1985). There have been increasing efforts also in forging a link between knowledge and action, (Moore, 1985b; Cohen and Levesque, in press), thereby introducing a dynamic element, because of the changes caused by action. While these authors have been concerned with agent dynamics, they have not attempted to link their logics to dynamical systems in the physical sense. In this paper we hope to do so and we propose a model theory which is very close to actual dynamic systems. Apart from the fact that we believe that this model theory gives us an intuititvely more appealing way to take about agent-theoretic notions, we also believe that the model theory is close enough to actual physical systems to allow specifications formulated in our logic to be directly implemented.

The rest of the paper is organised as follows. We first review some relevant concepts from abstract dynamics. Next, we discuss how agent-theoretic concepts can be interpreted in such terms. Finally, we formulate computational generalisations of

physical concepts like potential, force, velocity, etc., and indicate how they can provide a framework in which to interpret axiological concepts in agent theory.

Concepts of Abstract Dynamics

The main concepts of recent developments in dynamics deal with the structure of state spaces. Abstractly, the theory can be formulated in terms of functional iteration. The functions which define dynamical systems map states in the state space into other states in the same state space. They are also called mappings or maps. The main concern of the abstract theory is with the asymptotic behaviour of iterative mappings. The iteration of a function is a discrete process. If the process is continuous, the description is often given in the form of differential equations to describe the behaviour of the solution over time.

In a geometric interpretation, the iterative process maps points into points. The points correspond to the states of the process. The process is then said to go through a trajectory or orbit of points. The main concern of dynamics is to understand the nature of all trajectories of a system and to classify them as moving to a fixed point, being periodic, asymptotically periodic, etc. We shall now turn to an informal summary of some of these concepts. For more detail, see, for example, Abraham and Shaw (1981), Devaney (1986), Thompson and Stewart (1986) or Cvitanovic (1984). Cvitanovic also contains an extensive bibliography. The field is developing very rapidly under the designation of chaos theory, which is a specialised branch of dynamics.

The *state space* of a system is generally a topological surface (manifold) on which the possible states of the system are located. This can be just three-dimensional space, or some curved surface, for example, like a doughnut (torus).

It is normally assumed that there is a *force vector field* acting at all points of the state space. This vector field determines the *dynamics* of the system by constraining the *trajectories* to certain directions at each point of the state space. When typical or many trajectories of the system have been drawn, we get a *phase portrait* of the system.

Closed trajectories produce *cyclic behaviour*. Trajectories can otherwise take many shapes, like spirals, straight lines or any kind of curve.

The focus of interest is in the *asymptotic behaviour* of trajectories. *Limit sets* of state spaces are sets of points towards which the trajectories move asymptotically. Limit sets may be solitary points, or cycles, or more complicated distributions of points. Limit sets which are solitary points, are called *fixed points*.

Fixed points of functions are points x for which $f(x)=x$. That is, the fixed points are mapped into themselves by the function. Fixed points are important in dynamics, because they correspond to equilibrium (steady) states of systems. Once a system has somehow got to a state which is a fixed point, it will not move from that state under the iteration of the function f.

It is of interest to ask how a system may get to a fixed point. The simplest case is that the system may start from an initial state that is a fixed point, and there will be no further change. More interestingly, trajectories starting at other states may lead to a

fixed point after a number of transitions. In such cases we say that the fixed point *attracts* the trajectory. The set of states from which trajectories lead to an attractive fixed point are called the *basin of attraction* of the fixed point. It turns out that a fixed point is attractive if the slope (derivative) of the function f is less than 1 at the fixed point. The magnitude of the slope characterizes the strength of the attractor: the greater the strength, the faster the trajectory approaches the fixed point.

A periodic point is a generalisation of the concept of the fixed point to the case when a trajectory cyclically visits a point after every n iterations of the function f.

If the iteration is run backwards, trajectories would appear to diverge from an attractive fixed point. In this situation the fixed point is called a *repellor*. Such fixed points correspond to unstable equilibria in physical systems. Slight disturbance from the equilibrium starts the system on a trajectory leading away from the equilibrium state. Conversely, attractive fixed points correspond to stable equilibria.

Agent Attributes and Dynamics

We now briefly review how to interpret the agent-theoretic concepts of interest in this paper in terms of abstract dynamics.

Compositionality.

We assume that complex agents are architecturally compositional, both structurally and behaviourally. The complex agent structure is produced by assembling simpler component elements. Complex agent behaviour is produced through the (often nonlinear) interactions between the simpler component behaviours. Concurrency, parallelism and distributed systems become important issues.

The agent as controller.

We assume that the agent acts as a controller with respect to the world state. The agent exerts control by taking actions[1]. We include "doing nothing" as an agent action. Taking an evolutionary point of view, we assume that ultimately this control is in the interest of fitness for survival. Fitness for survival is dependent on the existence of certain world states, or on keeping them within permissible bounds. We assume that environmental events produce disturbances in the agent's internal state by causal effects conveyed through inputs. Agent action attempts to counteract such disturbances. An agent can control the world state either by changing its internal state or by attempting to change the external state. For example, the agent may change its beliefs or it may locomote to another location.

---

[1]There have been numerous discussions in philosophy on the exact definition of the notion of an agent action. Rather than attempt a definition of our own, we will simply rely on the intuitions of the reader. Agent actions are those actions that an agent performs qua agent, and over which he or she has direct and voluntary control (or at least has the impression). Thus, actions like making a decision, deciding to raise one's arm are agent actions, whereas such actions as sneezing or reflex actions in the physiological sense of the word are not.

We want to distinguish between a system's natural dynamics (might also be called the free dynamics) which is operating when the agent is executing the "null" action, and the constrained dynamics that results from the composition of the free dynamics with the control dynamics produced by the *non-null* agent actions. The distinction is motivated by recognising that only some events in the world are produced by agent actions.

Axiological aspects of agents

Axiological issues are concerned with the directional nature and asymptotic behaviour of agent dynamics. The teleological (goal-directed) nature of agent behaviour is one of the central examples of such issues. In terms of dynamic system theory, the dynamics can be described in terms of the movement of the system state *towards* stable equilibrium states and *away from* unstable equilibrium states. Teleological agent behaviour is to be identified with movement towards stable equilibria which are in this sense *preferred* states of the system: we shall say that the agent "*likes*" to be in these states. Aversive agent behaviour is to be identified with movement away from unstable equilibria which are in this sense *disliked* by the agent. In the terminology of dynamic system theory, these states are *attractors* and *repellors*. Unstable equilibria arise mainly through competition between attractors and represent boundaries between the basins of attraction of those attractors. Attractors and repellors determine the direction of movement, i.e. the direction of agent action. It is natural to interpret the pro- and anti-attitudes of agents with this kind of directionality. We assume that due to the physiological structuring of living organisms attractors and repellors are created in their behavioural space. By analogy, it should be possible to create attractors and repellors in non-living computational systems through appropriate construction or programming.

A related point of view is found in optimisation theory. In this approach the main underlying idea is that the states and trajectories of a dynamic system are governed by some principle that can be expressed mathematically as finding the stationary value (usually maximisation or minimisation) of an "objective" (or goal) function. There is a great deal of work on the application of such optimality principles to evolutionary, ecological, economic and behavioural processes. We wish to look upon this approach in the same spirit and regard the extrema of the objective function as specifications of the attractors and repellors of the state space. In its application to the description of behavioural or economical processes the objective function is usually called utility. Note that utility is here a descriptive aspect, revealed by the observation of behaviour. In other applications to evolutionary processes the objective function is taken to be fitness for survival. We are of course more concerned with individual agent behaviour and hence with utility in this paper. In summary, from the viewpoint of optimality theory, the agent is maximising utility.

In a utilitarian framework utility would be some function of hedonic states, i.e. pleasure and pain. One might speculate that pleasure and pain are related to fitness and have been incorporated in the architecture of organisms to make available to the individual some state variable that can be used as an indicator of fitness. Such an

interpretation would not be unnatural in the case of pain as an indicator of damage and hence loss of fitness and pleasure as an indicator of health and hence of maximisation of fitness. For the time being, we adopt this utilitarian framework and assume that the agent is maximising a hedonic function.

The *values* of an agent correspond to global (high-dimensional) attractors and repellors of the composite dynamics. We think of values as global attractors which may never be reached or closely approached by trajectories, due to the topological structure of the state space created by the competition between them. In complex agents explicit representations of values form a value system.

A *goal* of an agent corresponds to a local (low-dimensional) attractor in a basin of attraction of the composite dynamics. We think of goals as attractors which are reached or closely approached by nearby trajectories.

To support our intuitions, we wish to use a mechanical analogy. According to this analogy the intensity of a desire (liking) should correspond to some abstract "force of attraction" acting on the agent, producing acceleration of state change.

Similar conceptual frameworks have already been used in mechanical engineering and in robotics (see Koditschek, 1989 for a review). In mechanics it is well known that the total energy of a dissipative system (expressed by the Hamiltonian) will monotonically decrease and will be asymptotically stable. A known technique in robot control engineering is to use feedback control which amounts to following the gradients of total energy. This technique has been used for robot arm control. Direct utilisation of the potential field has been used for path planning with obstacle avoidance in mobile robots (Barraquand and Latombe, in press).

In our mechanical analogy too, the forces would be derived from a potential field and the agent is assumed to follow the gradients of the potential. From the point of view of optimisation theory, the objective function is used as the potential. The description of such a potential therefore amounts to the specification of a goal which is the asymptotically stable equilibrium state of the agent. We can also represent a *value system* in this analogy as additional potentials, with opposite sign, superimposed on the potential created by the goal. In the robot navigational applications such potentials are used to represent obstacles to be avoided while moving towards the goal state. In our analogy these obstacles correspond to elements of the value system, expressed as "prohibitions". The analogy is reasonable in the light of value systems often being expressed in the form of prohibitions (laws, regulations, etc). Presumably positive values (obligations) could always be re-expressed in a negated form.

Model theory

In this section we review some of the fundamental concepts that we need for our model theory: space, time, state, and process (trajectory). Our formulation draws on and extends previous work by Rosenschein and Kaelbling (1986) on agents as situated automata and by Halpern and Moses (1990) on knowledge in distributed systems. Our main concern is the formal characterisation of a process (or trajectory). For this, we need formal notions of time, space and state. We describe each in turn briefly.

Time is analysed as consisting of a set of instants T and a total ordering relation $<$ over T[2].

Space will be regarded as a set of locations L. We shall not assume any specific topology over L, but wish to partition L into subsets, which we shall call *systems*. *Agents* are considered as special types of system. Whereas normal systems may overlap, agents never overlap. That is, there is a set of locations which are part of exactly one agent, and which are not part of any other system. We call such locations *agent locations*. All other locations are called *non-agent locations*.

States are defined as functions from locations to data values. We assume that for every location l in L, there is a set of data values D_l that this location can take. We distinguish between global and local states as follows. Global states are functions which assign to every location l a data value from the appropriate set D_l. Given a set of locations L, we define GS^L to be the set of possible global states. Clearly, if the number of locations is n then GS^L can be regarded as an n-dimensional space. If g is a global state, then $g(l)$ denotes the data value assigned to location l by g.

A local state is a function which assigns appropriate data values to a subset *Loc* of the set of locations, *i.e.* to a system. The set of all possible local states over *Loc* is denoted LS^{Loc}.

Processes are defined as temporal sequences of states. Since the concept of state is tied to that of space through locations taking on data values, it is natural to regard processes as occupying a spatio-temporal region. Following Rosenschein and Kaelbling (1986), we capture these intuitions in two steps. First, at each instant in time a process can be regarded as occupying a set of locations. Second, each occupied location takes on a specific data value. We thus have two functions. The first is a function from T to subsets of L determining the occupied locations, while the second assssociates data values with these locations. We can thus generalise the notion of state to processes. The state of a process at time t is determined by the set of locations occupied at t and their data values.

Just as we did with states, we distinguish global and local processes. Global processes are temporal sequences of global states. Thus, global processes occupy, and assign data values to, all locations at every instant in time. Halpern and Moses call such a global process a "run" of a system. A global process or run can be regarded as one possible way the world can unfold over time, or a "possible world". Formally, a run is a function from T into GS^L. We denote the set of all runs by R and an individual run by r. Then $r(t)$ gives the state of the run r at t, and $r(t)(l)$ gives the data value of location l assigned by the run r to l.

Local processes occupy only a subset of L at each instant in time and are thus a sequence of local states. Local processes can also be thought of as subprocesses of

---

[2]Nothing hinges on this definition of time. In particular, we also could have adopted an instance-based definition of time. However, this would have made the remaining presentation considerably more complicated. Given that this paper is not primarily concerned with a temporal logic, we have opted for the simpler temporal ontology.

aglobal process. Formally, the spatial region occupied by a local process is a functionπ from T and R into *Powerset*(L). The state of the process is then given by a function $s(\pi,r,t)$, which is a set of location data-value pairs: $\{<l, r(t)(l)> \mid l \in \pi(r, t)\}$. This notation emphasizes that the data values of the local process depend on the run of which it is a subprocess.

The foregoing define processes in a very general way. For many applications simpler special cases are sufficient and are conceptually easier to handle. For the purposes of the rest of this paper we introduce *fixed-location processes*, which occupy the same locations at every instant in time. Thus, fixed-location processes do not move spatially and the only change that takes place at successive instants of time is that the fixed locations take on different data values.

Under this picture, complex agents are simply fixed-location processes, where all the the locations involved are agent locations.

Our model theory can accommodate the notion of an "accessible world", which allows us to construct a modal logic of beliefs along similar lines to Halpern and Moses (1990). A process π only occupies a subset of the set of all locations at time t.. It is therefore possible for different runs to assign the same state to π at t. We shall call such runs *alternative runs* with respect to π at time t. Thus, the runs r and $r'$are alternative runs with respect to process π at time t if $s(\pi,r,t) = s(\pi,r',t)$. If we identify a process with an agent situated in the world, then we can regard alternative runs as different states of affairs which are indistinguishable as far as the state of the agent is concerned. This construction gives us a way to interpret the epistemic operator in our logic.

Transition functions

In order to calculate a run of a system, we need to define a transition function for each location. A transition function for a location l is simply a function which calculates the next data value of l based on its present value and the data values of other locations. Although we do not wish to put any strong constraints on which locations can influence the next value of location l, typically the new state of location l will depend not on all locations in the L but only on a subset of them. Cellular automata are typical in this respect: the transition function for each location takes as input only the values of the immediate neightbours of that location. For logic circuits too, the transition function is usually defined as a function of a small subset of specified locations. Interestingly, in the latter case, the present value of the location is irrelevant for computing its next value.

By defining a transition function for each individual location, and allowing a location's transition function to take as input the values of other locations as well, we can reformulate the usual picture of a distributed, concurrent computing system in our proposal. In such a system, the component processes interact with each other through constraint relationships, implemented as message or signal passing. Connectionist architectures can be seen as an example. In this case, the messages are values, usually in the real number or boolean data domains. Under our proposals, we simply see interaction between location l and l' as indicating that the transition function for l has the value of l' as one of its inputs, and vice versa.

Earlier, we distinguished between agent locations and non-agent locations. We can now distinguish between three types of agent locations. The first distinction depends on whether an agent location's transition functions is dependent only on other agent locations, or whether they also receive as input the values of non-agent locations. We will call the first type of agent locations *pure* agent locations. We shall refer to the latter as *Input* locations for obvious reasons: *Input* locations are the places where the state of the external environment exerts influence on the agent. Note that *Input* locations are still agent locations. There are of course also some non-agent locations whose transition function takes as input values of agent locations. These are the places where agent exert influence on the world. We will therefore call those agent locations whose value is used as input to the transition function of a non-agent location *Output* locations[3].

We postulate a difference between the nature of the transition functions for agent and non-agent locations. In the case of agent locations, we assume that the transition function is chaotic and hence non-predictable, whereas for non-agent locations it is entirely predictable. The reason for this distinction is that it allows us to give a more intuitive account of the notion of agent action. We define an *event* as a state change of a location. Clearly, it is then natural to assume that an *agent action* is a state change of an agent location. Now, if we assumed that the transition function for a agent location was a completely predictable function, then this would go counter to the free will intuition, the intuition that agents have control over their own actions. On the other hand, one would also like to avoid the possiblity of having to explain free will as involving some kind of "magical" process. We claim that by postulating that transition functions for agent locations are chaotic we avoid both horns of this dilemma. Agents's actions remain unpredictable and hence seem to involve some notion of voluntary control, while remaining completely deterministic, and hence non-magical[4].

There are a number of further aspects of our model that we would like to draw attention to. First, we can define different "flavours" of agent action. On the one hand, we have pure agent actions, events that take place in pure agent locations. On the other hand, there are non-pure agent actions, events in agent locations that are partly under the influence of non-agent locations. Events that take place in input locations are of course the primary example of non-pure agent actions. Also, we can distinguish between pure physical events, events in locations whose transition functions take as input only the values of other non-agent locations., and non-pure physical events, events that happen in non-agent locations whose transition function also takes as input values from agent locations.

Second, although agents can directly influence only a small subset of non-agent locations, namely only those whose transition functions receive tnput from output

---

[3]If we stick to the assumption that only neighbouring locations influence each other, then clearly the sets of output and input locations overlap. However, nothing in our model theory forces us to accept this assumption.

[4]Clearly, the use of chaotic functions is directly inspired by dynamics as well. Thus, we not only rely on dynamics for our main intuitions, it also provides some of the mathematical machinery that we can use in our model-theory.

locations, agents can of course indirectly influence other events as well. In particular, non-agent locations that are directly connected to an agent's output locations can in their turn influence other non-agent locations. It is through such chains that agents can influence locations whose transition functions are not directly influenced by the agent's output locations. In particular, such chains allow us to explain how an agent can influence another agent: in order for this to take place, there must some chain from the first agent's output locations to the second agent's input locations.

State Transition Functions and State Space

Although we could formulate the rest of our proposals in terms of transition functions for individual locations, it will be more straightforward to do so in terms of state transition functions, transition functions for the entire set of locations. It is of course a relatively trivial matter to construct state transition functions from the transition functions for individual locations: if $tf,...,tf'$ are the transition functions for the locations $l,...,l'$ in L, the transition function can simply be obtained by applying the local transition functions to each location. We will call such a transition function a *state transition functions*

The set of possible runs has been denoted R. Trajectories in R are generated by the iterated application of a transition function f, $s_{i+1} = f(s_i)$. The transition function f represents the changes brought about by the agent's actions, including doing nothing. Recall that the changes may be either internal or external to the agent.

We want our state transition functions to be nonlinear functions f which have attractive limit sets. In the case of a limit set which is just a single point, we have a fixed point s_{fixed}, such that $f(s_{fixed}) = s_{fixed}$. Here s_{fixed} is an attractor. For all attractors there is an open set, called the domain of attraction D, such that for all states $s \in D$ the iterated application of f eventually carries the state into s_{fixed}.

We distinguish global attractors from local attractors. The transition function may assign the same state to a subset of locations at different times. We call such a local fixed state a local attractor. By analogy to global domains of attractions we can define a local domain of attraction in a straightforward manner.

Hedonic Functions

In order to use the model theory for the interpretation of desires, we introduce hedonic functions. Intuitively, the hedonic function specifies the amount of pleasure or pain an agent experiences in some state. The hedonic function h^{π} of a process π maps states into hedonic data values. The domain of hedonic data values H is a partially ordered set, containing a distinguished element *neutral*, corresponding to a neutral hedonic data value. All other hedonic data values are either hedonically greater than or smaller than *neutral*. The hedonic relational operator will be denoted by $<_{Hed}$. We shall assume that the hedonic state of an agent depends only on the local state of the agent. As indicated below, we will interpret the hedonic state as a computational analogue of potential field, with a potential of zero corresponding to the hedonic *neutral*.

The hedonic state results from the superposition of attractive and repelling potentials at the point corresponding to the current state. These potentials are produced by the agent's value system and by the current goal. The contributions of individual

attractors and repellors can be separately computed as $h^{\pi}(s,s_{fixed_i})$, where s_{fixed_i} is the fixed point state corresponding to attractor i.

As is usual in utilitarian agent theories, we assume that an agent acts in order to maximise its hedonic state. The computation the agent executes to determine its action is therefore the optimisation of the hedonic function. Maxima of the hedonic function correspond to limit sets in the state space of the agent.

Computational Analogues of Force, Mass and Potential

We assume that for each point s in GS^L we can define the distance $d(s, s_{fixed})$ as the distance between the point s and the fixed point s_{fixed}. If we regard GS^L as an n-dimensional space, then this could be euclidean distance. If we interpret each function iteration of the transition function f as a unit of time then we can define the velocity at s as the distance $d(s, f(s))$, since this will be the distance travelled in unit time. The definition of a force vector $\boldsymbol{F}$ acting at the point s follows the mechanical analogy and is the product of acceleration and mass. It is natural to interpret mass in our computational domain as some measure of the size of the state s. For example, we can take the number of locations occupied by the agent process, n, as this measure. Then,

$$\boldsymbol{F}(s, s_{fixed}) = \mathrm{n} \cdot d(s, f(f(s))) - d(s, f(s)).$$

The analogy can then be even further extended by defining force, as in physics, as the gradient of a potential, $\boldsymbol{F} = \boldsymbol{grad}\, H$.

The joint effect of two or more fixed points at s can then be reflected by vector addition of the forces acting at s. Let us denote two such forces by $\boldsymbol{F}_i$ and $\boldsymbol{F}_j$ for two different fixed points. The joint effect is then

$$\boldsymbol{F} = \boldsymbol{F}_i \oplus \boldsymbol{F}_j$$

where $\oplus$ denotes vector addition.

We can now assess the relative strengths of two attractors by comparing the magnitudes of the two forces and say that $Greater(s, s_{fixed_i}, s_{fixed_j})$ if $|\boldsymbol{F}_i| > |\boldsymbol{F}_j|$.

The intuitive agent theoretic interpretation of these concepts is then as follows. As stated before, the potential is interpreted as the hedonic state. Components of the potential correspond to the values of the agent. The forces correspond to the intensity of liking. The concept of relative intensity, or preference, is based on the comparison of forces. We model the activity of the agent as following gradients in a potential field produced by the superposition of all the forces, i.e. values, acting on the agent. Gradient following corresponds to hedonic maximisation.

Conclusions

We have described some intuitions about the interpretation of axiological aspects of agent theory in terms of concepts from physical dynamics. The first steps have been

taken towards formalisation by sketching a model theory. This model theory can be used straightforwardly for the construction of a logical language in which to reason about an agent's hedonic state, likes, goals and values. We believe that the model theory can also be used for an integrated interpretation of axiological, epistemic and praxiological aspects of agent theory.

As indicated in Kiss (1991), such a model theory can also offer a link between concerns for formalisation and concerns for implementation strategies. As shown by Rosenschein's work on situated automata theory and the implementation language REX, there is a complementary relationship between a mathematical model and a physical phenomenon, both of which can be taken as alternative interpretations of a logic. When this is the case, the logic can be used for reasoning about a design, the mathematical model provides the semantics of that reasoning, while the physical phenomena (or their computational analogues) can be used for the implementation of the design. Thus, by adopting a model theory that is much closer to the physical world than is the case in for example a possible world model, we hope that the step for design (and a logical analysis of this design) to an actual implementation is considerably reduced.

References

Abraham, R. H. and Shaw, C. D. (1981). *Dynamics, the Geometry of Behavior, Parts 1-4* . Santa Cruz, CA: Aerial Press.

Barraquand, J. and Latombe, J.-L. (In press). Robot motion planning:a distributed representation approach. *International Journal of Robotics Research.*

Cohen, P. R. and Levesque, H. (1985). Speech acts and rationality. *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics.*

Cohen, P.R. and Levesque, H. (in press). Rational interaction as a basis for communication. In P. R. Cohen, J. Morgan, & M. E. Pollack (Ed.), *Intentions in Communication* Cambridge, Massachusetts: MIT Press.

Cvitanovic, P. (1984). *Universality in Chaos*. Bristol: Adam Hilger,

Devaney, R. L. (1986). *An Introduction to Chaotic Dynamical Systems* . Menlo Park, CA: Benjamin/Cummings.

Demazeau, Y. (1991) Coordination patterns in multi-agent worlds applications to computer vision and robotics. *IEE Colloquium on Intelligent Agents.*

Halpern, J. Y. and Moses, Y. (1985). A guide to the modal logics of knowledge and belief: preliminary draft. *Proceedings of the 9th IJCAI*. Los Altos, CA: Kaufmann.

Halpern, J. Y. and Moses, Y. (1990). Knowledge and common knowledge in a distributed environment. *Journal of the Association for Computing Machinery*, *37*(3), 549-587.

Kiss, G. R. (1988). *Some aspects of agent theory* Report HLD/OU/WP/GRK/24, HCRL, The Open University.

attractors and repellors can be separately computed as $h^{\pi}(s,s_{fixed_i})$, where s_{fixed_i} is the fixed point state corresponding to attractor i.

As is usual in utilitarian agent theories, we assume that an agent acts in order to maximise its hedonic state. The computation the agent executes to determine its action is therefore the optimisation of the hedonic function. Maxima of the hedonic function correspond to limit sets in the state space of the agent.

Computational Analogues of Force, Mass and Potential

We assume that for each point s in GS^L we can define the distance $d(s, s_{fixed})$ as the distance between the point s and the fixed point s_{fixed}. If we regard GS^L as an n-dimensional space, then this could be euclidean distance. If we interpret each function iteration of the transition function f as a unit of time then we can define the velocity at s as the distance $d(s, f(s))$, since this will be the distance travelled in unit time. The definition of a force vector $\boldsymbol{F}$ acting at the point s follows the mechanical analogy and is the product of acceleration and mass. It is natural to interpret mass in our computational domain as some measure of the size of the state s. For example, we can take the number of locations occupied by the agent process, n, as this measure. Then,

$$\boldsymbol{F}(s, s_{fixed}) = \mathrm{n} \bullet d(s, f(f(s))) - d(s, f(s)).$$

The analogy can then be even further extended by defining force, as in physics, as the gradient of a potential, $\boldsymbol{F} = \boldsymbol{grad}\, H$.

The joint effect of two or more fixed points at s can then be reflected by vector addition of the forces acting at s. Let us denote two such forces by $\boldsymbol{F}_i$ and $\boldsymbol{F}_j$ for two different fixed points. The joint effect is then

$$\boldsymbol{F} = \boldsymbol{F}_i \oplus \boldsymbol{F}_j$$

where $\oplus$ denotes vector addition.

We can now assess the relative strengths of two attractors by comparing the magnitudes of the two forces and say that $Greater(s, s_{fixed_i}, s_{fixed_j})$ if $|\boldsymbol{F}_i| > |\boldsymbol{F}_j|$.

The intuitive agent theoretic interpretation of these concepts is then as follows. As stated before, the potential is interpreted as the hedonic state. Components of the potential correspond to the values of the agent. The forces correspond to the intensity of liking. The concept of relative intensity, or preference, is based on the comparison of forces. We model the activity of the agent as following gradients in a potential field produced by the superposition of all the forces, i.e. values, acting on the agent. Gradient following corresponds to hedonic maximisation.

Conclusions

We have described some intuitions about the interpretation of axiological aspects of agent theory in terms of concepts from physical dynamics. The first steps have been

Kiss, G.R. (1990). Value mechanisms in a theory of agents. In J.R. Galliers (Ed), *Proceedings of the first belief representation and agent architectures workshop*. Technical Report 194, University of Cambridge, Computer Laboratory.

Kiss, G. R. (1991). Autonomous agents, AI and chaos theory. *First International Conference on Simulation of Adaptive Behavior: From Animals to Animats*. 1990. Cambridge: MIT Press.

Koditschek, D. E. (1989). Robot planning and control via potential functions. In O.Khatib, J. J. Craig, & T. Lozano-Perez (Ed.), *The Robotics Review 1*. Cambridge, Mass.: MIT Press.

Moore, R.C. (1985a). What about desires? Set of OHP films used in a seminar, personal communication.

Moore, R. C. (1985b). A formal theory of knowledge and action. In J. R. Hobbs, & R. C. Moore (Ed.), *Formal Theories of the Commonsense World*. Norwood, New Jersey: Ablex.

Rosenschein, S. (1986). Formal theories of knowledge in AI and robotics. *New Generation Computing, 3*, 345-357.

Rosenschein, S. and Kaelbling, L. (1986). The synthesis of digital machines with provable epistemic properties. *Proceedings of the 1986 Conference on Theoretical Aspects of Reasoning About Knowledge*. Los Altos, CA: Morgan Kaufmann.

Shoham, Y. (1989). Time for action. *Proceedings of the 11th IJCAI*. p. 954-959. Morgan Kaufmann.

Thompson, J. M. T. and Stewart, H. B. (1986). *Nonlinear Dynamics and Chaos* . Chichester: Wiley.

DEPENDENCE RELATIONS AMONG AUTONOMOUS AGENTS

Cristiano CASTELFRANCHI, Maria MICELI
Social Behavior Simulation Project
Institute of Psychology, CNR,
Viale Marx 15
I-00137 Rome
ITALY
e-mail: pscs@irmkant.bitnet

&

Amedeo CESTA
Dept. of Computer and System Sciences
University of Rome "La Sapienza"
Via Salaria 113
I-00198, Rome
ITALY
e-mail: amedeo@irmkant.bitnet

Abstract
The basic thesis of this work is that human interactions are neither unpredictable nor bounded, but they are undertaken autonomously on the grounds of a number of basic principles and conditions. Among these, a crucial role is played by the objective dependence relationships holding among agents. In this paper we report about a first step in providing a computational theory of dependence as a tool for interaction control. We define non social as well social dependence, and try to show how dependence relationships are organized into complex patterns (such as multiparty, multigoal, unilateral, and bilateral dependence). We then show how a given set of dependence relationships may produce new dependence relationships. Finally, we explore the relationship between dependence and influencing, describing how one's dependence on another is predictive of one's goal of influencing the other, as well as of the latter's power of influencing the former.

1. INTRODUCTION

Communication control has always been a crucial problem in distributed system. The limited capacity of communication channels represents a bottleneck for the performance of those systems. In Distributed Artificial Intelligence (DAI), the problem is amplified because of the continuous need for interaction among agents (e.g., for negotiation). The proposed solutions try either to communicate implicitly using a shared memory (blackboard systems [8]) or, if the single agent is more autonomous, to apply specialized control strategies (see for example [1]). Both approaches use solutions quite different from the behavior of human beings in similar situations. Cognitive agents generally find implausible to put a request message in a mailbox and wait for somebody to answer it; neither do they apply a standard protocol for the interaction. When an agent needs somebody else for achieving a goal, she reasons about knowledge of sociality and social relations. Such knowledge is used both in the stage of decision formation and in actual interaction.

We have been studying human behavior and developing models of it for several years (see [2, 6]). In this paper we attempt to describe the relations upon which context-dependent human interaction is based and try to devise some principles controlling it. Our main argument is that human interactions are neither unpredictable nor bounded but they are undertaken autonomously on the grounds of a number of basic principles. Those principles are formulated in a quasi-formal way, and seem to represent a necessary background for a computational model of context-related social interaction.

One of the fundamental notions of social interaction is the *dependence* relation among agents. In our opinion, the terminology for describing interaction in a multi-agent world is necessarily based on an analytic description of this relation. Starting from such a terminology, it is possible to devise a calculus to obtain predictions and make choices that simulate human behavior.

In this paper, we present our formalism for dependence relation and describe some deductions to be made on the grounds of this relation in order to obtain rational choices. In particular, we distinguish between resource dependence and social dependence, and show properties and special cases of both. We also give the basic axioms and show which types of actual interactions are strongly based on dependence. In particular, we try to show the relation between dependence relationships among agents and the action of influencing of one agent respect to another as probably the most relevant form of interaction in real social contexts.

The paper is structured as follows: in Section 2, we provide our basic definition of dependence and describe its properties; in Section 3, we describe some principles for deriving a dependence relationship from another; in Section 4, we describe the relation between dependence and influencing; in a concluding Section, we point out some aspect still lacking in our theory.

2. DEPENDENCE AND ITS PROPERTIES

As already argued in [5], dependence is undoubtedly the ground relation upon which the whole construction of sociality is based. In the following, we first analyze a non social, or pre-social, form of dependence, namely the one between an agent and a resource. Then we proceed to its social version, the dependence between two agents, and finally try to describe some types of social dependence relationships involving more than two agents.

In the paper we mainly refer to the formal apparatus used by Cohen and Levesque in [3]. In the following x and y denote agent variables with $x \neq y$ always implicitly stated, a denotes an action

variable, ***r*** a resource, and ***p*** a well formed formula representing a state of the world. The predicate ***(RESOURCE r a)*** means that ***r*** is needed in order to perform ***a***. ***(CANDO x a)*** means that agent ***x*** has the action ***a*** in his repertoire, that is he is able to do it by himself. We use the following definition similar to the one in [3]:
(DONE-BY x a) =def (DONE a) $\wedge$ ***(AGT x a)*** whose meaning is quite obvious.

2.1 Non social dependence

Dependence is not necessarily a social notion. A relation of dependence may be said to occur whenever: a) any object or event in the external world may increase, if used, the probability that a given state of the world be realized, and b) that world state is represented as a goal by at least one agent. In such a case, we say that agent to be *dependent on* the enabling object or event. The latter will then be called a *resource*. Resources enter the structures of the actions (see also [9]). An action can be modeled as a relation holding among agent(s), goal(s), and resource(s). A set of resources is required for any action to take place. In our notion, cubes, tables and hands are resources in the block world. In the social world, others may be used as resources (not only in exploitation but also in prosocial action: in help, in a quite abstract sense, the recipient is a resource of the action "give help").

Agents are usually dependent on the existence of resources. We call this type of dependence a *resource dependence* (described by the ***R-DEP*** predicate), to distinguish it from *social dependence*, ***S-DEP***, (see below):

D1. ***(R-DEP x r a p) =def*** ***(GOAL x p)***
$\wedge$ ***(RESOURCE r a)***
$\wedge$ ***((DONE-BY x a)*** $\supset$ ***(EVENTUALLY p))***

r is then a resource for ***x*** to achieve his goal that ***p***. Thus, for instance, ***x*** is resource dependent on a hammer for having a nail driven into a wall. *

2.2. Social dependence

Our basic definition of social dependence is as follows [6]:

D2. ***(S-DEP x y a p) =def*** ***(GOAL x p)***
$\wedge \neg$ ***(CANDO x a)***
$\wedge$ ***(CANDO y a)***
$\wedge$ ***((DONE-BY y a)*** $\supset$ ***(EVENTUALLY p))***

that is: ***x*** depends on ***y*** with regard to an act useful for realizing a state ***p*** when ***p*** is a goal of ***x***'s and ***x*** is unable to realize ***p*** while ***y*** is able to do so. In this context, ***y***'s action is a resource for ***x***'s achieving his goal.

---

* We want to point out the difference between the given definition of R-DEP and the following one where **w** denotes a plan for achieving **p** (or an <action-expression> following [3]):
D1b. (R-DEP x r a p) =def $\forall$w (GOAL x p) $\wedge$ ((ACHIEVE w p) $\supset$ (IN a w)) $\wedge$ (RESOURCE r a)
which means that x depends only from resources of actions *essential* to the achievement of his goal. Even if this definition is reasonable, the one we use in the paper is intended to stress the fact that agents may have a number of alternative ways to achieve their goals either acting by themselves or asking others to act. Their behavior is the result of a decision making process. This is true as well in the case of S-DEP.

It should be stressed that, unlike what most DAI work seems to take for granted, social dependence as well as resource dependence is *not fundamentally mental*. It is an *objective* relationship, in that it holds *independently* of the agents' awareness of it: x may depend on y even though they both ignore the fact. However, many relevant consequences may derive from x's and y's (either unilaterally or mutually) becoming aware of it: to mention just the most salient ones, x may try to *influence* y to pursue p, while y may choose whether to adopt x's goal or not (see [6]; see also later on in the text).

Moreover, not only a dependence relationship may be *known*; it may also be *wanted*, in that either x or y may actively "work" on maintaining or strengthening the relationship. And, not only a dependence relationship may be wanted once established. It may even be *created* by the agents, by producing those objective conditions that define a dependence relationship (a certain goal in x's mind; the lack of a certain power condition, etc.). So, for instance, y may *influence* x and induce him to have p as a goal of his own; since p cannot be achieved by x without y's help, y has created a dependence of x on her by means of an influencing strategy; otherwise, supposing that x already has p as a goal of his and is also endowed with the power conditions useful for achieving it, y may deprive x of some of them (say, by stripping him of a certain resource), thus making x *become* dependent on her relative to p.

2.3. Patterns of dependence relationships

Dependence relations set up a social network (that we call the DNP-net, to stress the fact that it is a baseline for the socalled contract net [7]) among agents, independent of, and often preceding, their awareness. Several special cases of net can be recognized:

a) OR-Dependence. Very often, there exist disjunctive compositions of dependence relations; that is, x may depend on y_1 OR on y_2 (or on y_3, etc.) for the same p, for at least two possible reasons:

-- the same action a useful for realizing p is performable by a number of agents (each independent of the other); so, it is sufficient for x to have a performed by one of them (say, the most available or willing):

$$(GOAL\ x\ p) \wedge \neg (CANDO\ x\ a)$$
$$\wedge\ (\bigwedge_{i=1,n} (CANDO\ y_i\ a)) \wedge ((\bigvee_{i=1,n} (DONE\ y_i\ a)) \supset (EVENTUALLY\ p))$$

-- alternative actions are useful for realizing p, and for each of them x is dependent on a different agent. In such a case, x's dependence with regard to p varies with the act, and then the agent, considered.

$$(GOAL\ x\ p) \wedge (\bigwedge_{i=1,n} \neg (CANDO\ x\ a_i))$$
$$\wedge\ (\bigwedge_{i=1,n} (CANDO\ y_i\ a_i)) \wedge ((\bigvee_{i=1,n} (DONE\ y_i\ a_i)) \supset (EVENTUALLY\ p))$$

b) AND dependence. Two cases may be distinguished in which there is a conjunction of dependence relations, namely the *multiparty* and the *multigoal dependence*:

-- we call *multiparty dependence* the case in which x depends on more than one agent for realizing p, this happens when more than one single act is needed for achieving one and the same goal, and for each act x depends on a different agent: $\bigwedge_{i=1,n} (S\text{-}DEP\ x\ y\ a_i\ p)$;

-- *multigoal dependence* happens when x depends on the same agent for realizing a number of unrelated goals: $\bigwedge_{i=1,n}$ *(S-DEP x y* a_i p_i*)*.

c) Bilateral dependence. So far, just cases of unilateral dependence (of ***x*** on ***y***) have been described. However, dependence may also be bilateral (of ***x*** on ***y*** and of ***y*** on ***x***). Bilaterality should not be confused with symmetry. The DEP predicate is in fact asymmetrical, in the sense that ***x***'s dependence on ***y*** relative to a certain action for a given goal does *not* imply ***y***'s dependence on ***x*** relative to the same action for the same goal, and vice versa. On the contrary, in bilateral dependence either the actions or both the actions and goals implied are not the same for ***x*** and ***y***. There are in fact two possible kinds of bilateral dependence:

--*mutual dependence*, which occurs when ***x*** and ***y*** depend on each other for realizing a *common goal* ***p***, which can be achieved by means of a plan including at least two *different* acts such that ***x*** depends on ***y***'s doing a_1, and ***y*** depends on ***x***'s doing a_2:

(S-DEP x y a_1 *p)* $\wedge$ *(S-DEP y x* a_2 *p)*

as observed in a previous work [6], *cooperation* is a function of mutual dependence: there is no cooperation without mutual dependence;

-- *reciprocal dependence*, which occurs when ***x*** and ***y*** depend on each other for realizing different goals, that is, when ***x*** depends on ***y*** for realizing ***x***'s goal that p_1, while ***y*** depends on ***x*** for realizing ***y***'s goal that p_2:

(S-DEP x y a_1 p_1*)* $\wedge$ *(S-DEP y x* a_2 p_2*)*

reciprocal dependence is to *social exchange* what mutual dependence is to cooperation.

3. SOME PRINCIPLES OF A THEORY OF DEPENDENCE

So far we have attempted to provide some definitions of various forms of dependence (resource dependence vs. social dependence) and of different patterns of dependence relationships. Now, a number of interesting consequences may be drawn from the above. We do not aim here at showing all the possible principles according to which a dependence relationship can be derived from another; we just outline some of the most common ones.

3.1. From resource dependence to social dependence

Resource dependence is likely to produce social dependence. In order to describe this property we introduce the notion of resource control. An agent ***x*** *controls* a resource ***r*** when he is able to do an action a_1 by which he allows any other agent to perform all the actions requiring the resource:

D3. *(CONTROL x r) =def* $\forall y \exists a_1 \forall a_2$ *(CANDO x* a_1*)* $\wedge \neg$*(CANDO y* a_1*)*
$\wedge$ *(RESOURCE r* a_2*)*
$\wedge$ *((DONE-BY x* a_1*)* $\supset$ *(CANDO y* a_2*))*

If agent ***x*** depends on resource ***r*** for a given ***p***, and agent ***y*** controls ***r***, then agent ***x*** depends on agent ***y*** for using ***r***. So, in this context social dependence (of ***x*** on ***y***) is the joint result of resource dependence (of ***x*** on ***r***) and resource control (of ***y*** over ***r***):

A1. $\exists a_1$ *((R-DEP x r a p)* $\wedge$ *(CONTROL y r))* $\supset$ *(S-DEP x y* a_1 *p)*

There exist, then, at least two sources of social dependence relationships:
a) ***x*** directly depends on some action of ***y***'s;
b) ***x*** depends on some resource which is controlled by ***y***; hence, he depends on ***y***.

However, (b) can be seen as a sub-case of (a), in that also in (b) ***x*** comes to depend on some action of ***y***'s: if ***r*** is controlled by ***y***, ***x*** depends on ***y***'s action of "letting ***x*** use ***r***".

By the way, ***CONTROL*** should be articulated into at least three possible sub-cases, each implying a particular action by ***y*** of "letting ***x*** use ***r***"; these cases might be unformally described as follows:
1) ***y*** *possesses* ***r***; hence, a condition for ***x***'s using ***r*** is ***y***'s *permission* to use it; ***y***'s act on which ***x*** depends is exactly ***y***'s permission;
2) ***y*** *is using* ***r*** at the same time when ***x*** would like to use it, and ***r*** is such a resource that cannot be used by different agents at the same time. So, ***x*** depends on ***y***'s act of *stopping* using ***r***;
3) ***r*** is *spatially available* to ***y***, while it is not available to ***x***, in the sense that ***r***'s location coincides with ***y***'s location, and ***x*** cannot use ***r*** unless ***y*** makes ***r***'s location change from hers to ***x***'s, that is, unless ***y*** *gives* ***r*** to ***x***.

Thus, it might be concluded that social dependence of ***x*** on ***y*** relative to ***p*** is always a dependence on ***y***'s actions of two sorts: either actions which cause ***r*** to be available to ***x*** for ***p*** (stopping using ***r***, giving ***r*** to ***x***, permitting ***x*** to use ***r***), or actions which directly produce ***p***[**] .

3.2. Dependence via influencing

Another interesting case of generation of DEP relations implies the mediating role of some agent's power of influencing another. Our basic definition of the power of influencing, ***INFL-POWER***, is the following:

D4. *(INFL-POWER x y a p) =def (CANDO x a)*
$\wedge$ *((DONE-BY x a)* $\supset$ *(EVENTUALLY (GOAL y p)))*

That is: ***x*** has the power of influencing ***y*** if he ***CANDO*** such an act that makes ***y*** have ***p*** as a goal of her own. As we shall see in Section 4, this action generally implies making ***y*** *believe* something which is somehow related to ***p***. For instance, an act of that sort might be a threat ("If you don't pursue ***p***, I will thwart you in ***q***"-- where ***q*** is some other goal of ***y***'s).

If ***x*** depends on ***y*** relative to ***a*** for realizing ***p***, and ***z*** has the power of influencing ***y*** to do ***a***, then ***x*** depends also on ***z*** for realizing ***p***:

A2. *((S-DEP x y* a_1 *p)* $\wedge$ *(INFL-POWER z y* a_2 *(DONE-BY y* a_1*)))* $\supset$
(S-DEP x z a_2 *p)*

[**] Moreover, a number of distinctions can be done modifying definition **D3**, which describes an agent as the *administrator* of a resource. An agent can be described as the only one able to do something by means of a given resource:
D3b. (CONTROL x r) =def $\forall$y $\forall$a (CANDO x a) $\wedge$ (RESOURCE r a) $\wedge$ $\neg$(CANDO y a).
A third and stronger definition describes a controller as the agent who *prevents* others from using the resource:
D3c. (CONTROL x r) =def $\forall$y $\exists a_1 \forall a_2$ (CANDO x a_1) $\wedge$ $\neg$(CANDO y a_1) $\wedge$ (RESOURCE r a2)
$\wedge$ ((DONE-BY x a_1) $\supset$ $\neg$(CANDO y a2))

We will discuss in more detail both the goal and the power of influencing and their relation with dependence in the next Section.

3.3. Generative power of multiparty dependence

Let us suppose a simple case of AND dependence, where x depends on more than one agent (say, on y and on z) for realizing p: $((S\text{-}DEP\ x\ y\ a_1\ p) \wedge (S\text{-}DEP\ x\ z\ a_2\ p))$. This AND dependence may generate a further dependence of y on z, *in case y is benevolent toward x.* In [2] and [6], we argue against the notion of benevolence, suggesting some further refinement of it; however, to our current purposes, it is sufficient to refer to a simpler definition, in line with Cohen & Levesque's [4]:

D5. $(BENEVOLENT\ y\ x\ p)$ **=def** $(BEL\ y\ (GOAL\ x\ p)) \supset (EVENTUALLY\ (GOAL\ y\ p))$

So, if y believes that x has the goal p then also y comes to have the same goal p.

Now, if y is benevolent toward x and believes that x has the goal that p, also y (besides x) comes to depend on z (provided y is unable to perform a_2), since z's action a_2 is necessary for realizing p:

A3. $((S\text{-}DEP\ x\ y\ a_1\ p) \wedge (S\text{-}DEP\ x\ z\ a_2\ p) \wedge (BENEVOLENT\ y\ x\ p)) \wedge (BEL\ y\ (GOAL\ x\ p)) \supset (S\text{-}DEP\ y\ z\ a_2\ p)$

Of course, if in turn also z is benevolent toward x, y and z will mutually depend on each other relative to p.

4. PREDICTIVE POWER OF DEPENDENCE RELATIONSHIPS: FROM DEPENDENCE TO INFLUENCING

One of the most interesting aspects of the DEP relations lays in their predictive power, that is, in the possibility to predict *other social relationships* and goals from dependence relationships.

4.1. From dependence to the goal of influencing

Among the social goals predictable from a dependence relationship, a crucial role is played by the *goal of influencing*. In our view, one's goal of influencing another is the goal of increasing the probabilities that the other pursues (or does not pursue) a certain goal that p [2]. However, here we can propose a simplified version of that definition: so, by x's *goal of influencing* y, $INFL\text{-}GOAL$, we mean x's goal that y has a certain goal p:

D6. $(INFL\text{-}GOAL\ x\ y\ p)$ *=def* $(GOAL\ x\ (GOAL\ y\ p))$

Let us start from our basic definition of social dependence (see **D2**). First of all we need that this objective social relationship between x and y is assumed by x. In fact, one of the ways in which new goals are acquired implies that people learn they are involved in certain relationships. Now, by assuming his dependence on y relative to a, x will also assume that he can achieve his goal that p by means of y's performing a: $(BEL\ x\ (S\text{-}DEP\ x\ y\ a\ p)) \supset (BEL\ x\ ((DONE\text{-}BY\ y\ a) \supset (EVENTUALLY\ p))$. Then, according to a condition-action rule formulated as follows:

A4. $((BEL\ x\ (q \supset p)) \wedge (GOAL\ x\ p)) \supset (GOAL\ x\ q)$

x will come to have the goal that $(DONE\text{-}BY\ y\ a)$, in order to achieve his goal that p:

T1. $(BEL\ x\ (S\text{-}DEP\ x\ y\ a\ p)) \supset (GOAL\ x\ (DONE\text{-}BY\ y\ a))$

So, if x believes he is dependent on y relative to a, then he will have the goal that y performs a. But given the postulate on rational agenthood, according to which, in order to perform an action, an agent must want that action, $(GOAL\ x\ (DONE\text{-}BY\ y\ a))$ is actually equivalent to $(GOAL\ x\ (GOAL\ y\ (DONE\text{-}BY\ y\ a)))$. Then x's dependence on y, when assumed, will also imply x's goal that y *has the goal to do* a:

$(BEL\ x\ (S\text{-}DEP\ x\ y\ a\ p)) \supset (GOAL\ x\ (GOAL\ y\ (DONE\text{-}BY\ y\ a)))$

Now, being $(GOAL\ x\ (GOAL\ y\ (DONE\text{-}BY\ y\ a)))$ nothing but a goal of influencing y relative to the goal that $(DONE\text{-}BY\ y\ a)$ (see **D6**), x's dependence on y relative to a certain a useful for p will imply, when assumed, x's goal of influencing y to perform a:

T2. $(BEL\ x\ (S\text{-}DEP\ x\ y\ a\ p)) \supset (INFL\text{-}GOAL\ x\ y\ (DONE\text{-}BY\ y\ a))$

So, if an agent assumes to be dependent on another relative to some goal, he will have the goal of influencing the other to perform the (set of) action(s) that allows him to achieve his goal. And, on the grounds of a given assumed DEP-net, a network is derivable of possible goals and actions of influencing (INFL-net).

4.2. From dependence to power of influencing

However, the goal of influencing is not sufficient for an agent to succeed in influencing another. Also the *power* of influencing is necessary, that is, the power of making someone do what we want. We have already provided a simplified definition of the power of influencing (see **D4**).

Many are the possible bases of one's power of influencing. What we are mainly interested in here is the power of influencing derivable from a dependence relationship. If x is (and assumes to be) dependent on y's performing a certain act in view of p, y is quite likely to have the power of influencing x relative to some other goal of x's. We will try to describe the main steps of this derivation, which, it should be stressed, is but a rough derivation, and would surely benefit from a number of refinements.

4.2.1. Dependence as a basis for the power of influencing

As we know from **T1**, if x assumes his dependence on y relative to a certain a (say, painting a wall) useful for p (having the wall painted), x will have the goal that y performs that action: $(GOAL\ x\ (DONE\text{-}BY\ y\ a))$, e.g., the goal that y paints the wall.

Now we need some other condition -- some sort of "persuasive" power of y over x about a means-ends relationship between some action on x's part and y's action a. Suppose y has an action a_1 such that x comes to believe that y will do a (painting the wall) if x performs some

other action a_2 (giving y some money). We can write something like:

$$(CANDO\ y\ a_1)$$
$$\wedge\ ((DONE\text{-}BY\ y\ a_1) \supset$$
$$(EVENTUALLY\ (BEL\ x\ ((DONE\text{-}BY\ x\ a_2) \supset (DONE\text{-}BY\ y\ a))))$$

Action a_1 can be either a communicative act of promise (as in this case) or even threat, or any "demonstrative" behavior useful for making x believe the means-end relation between x's action and y's.

Suppose also that $(DONE\text{-}BY\ y\ a_1)$ holds. Now, as we know from the condition-action rule, **A4**, if x believes that q implies p and has the goal that p, then he will also have the goal that q. Applying the condition-action rule to x's goal that $(DONE\text{-}BY\ y\ a)$, we obtain:

$$((BEL\ x\ ((DONE\text{-}BY\ x\ a_2) \supset (DONE\text{-}BY\ y\ a)))$$
$$\wedge\ (GOAL\ x\ (DONE\text{-}BY\ y\ a))) \supset (GOAL\ x\ (DONE\text{-}BY\ x\ a_2))$$

That is, x will come to have the goal of giving y the money $(GOAL\ x\ (DONE\text{-}BY\ x\ a_2))$, in order to obtain that $(DONE\text{-}BY\ y\ a)$ -- on condition that the value of $(DONE\text{-}BY\ y\ a)$ be greater than the cost of pursuing $(DONE\text{-}BY\ x\ a_2)$: only in this case, in fact, x would accept $(DONE\text{-}BY\ x\ a_2)$ as a goal of his own, in view of $(DONE\text{-}BY\ y\ a)$. So, y is in fact endowed with an action a_1 such that $(GOAL\ x\ (DONE\text{-}BY\ x\ a_2))$. This equals to saying that y, in virtue of both x's dependence on her and her ability to make x believe the implication $(DONE\text{-}BY\ x\ a_2) \supset (DONE\text{-}BY\ y\ a)$, has the power of influencing x to $(DONE\text{-}BY\ x\ a_2)$:

T3. $((BEL\ x\ (S\text{-}DEP\ x\ y\ a\ p)) \wedge (CANDO\ y\ a_1)$
$\wedge\ ((DONE\text{-}BY\ y\ a_1) \supset$
$(EVENTUALLY\ (BEL\ x\ ((DONE\text{-}BY\ x\ a_2) \supset (DONE\text{-}BY\ y\ a))))\ ) \supset$
$(INFL\text{-}POWER\ y\ x\ a_1\ (DONE\text{-}BY\ x\ a_2))$***

4.3. The act of influencing: goal plus power of influencing

So far, we have observed that: x (the dependent agent) has the *goal* of influencing y, while y has the *power* of influencing x. But, in this situation, x is the one who is the most interested in influencing y. And having a goal is a necessary, but not sufficient, condition for pursuing it, that is, for transforming that goal into an actual *intention*. A rational agent x, who is interested in influencing y relative to p, will pursue his goal of influencing y if he believes that goal to be

---

*** The expression (CANDO y a_1) $\wedge$ ((DONE y a_1) $\supset$ (EVENTUALLY (BEL x ((DONE x a_2) $\supset$ (DONE y a))))) can be seen as an instantiation of a more general case of a relationship between two agents, where the former is able to make the latter believe something. In other words a new predicate might be introduced, BEL-POWER, defined as follows:
D7. (BEL-POWER x y a p)= def (CANDO x a) $\wedge$ ((DONE-BY x a) $\supset$ (EVENTUALLY (BEL y p)))
that is: x has the power to make y believe p if he can do an action a (be it a simple communication of a fact p -- for instance, saying "It is raining" -- or a more indirect and subtle persuasive strategy -- for instance, taking an umbrella before going out) such that y comes to believe p (for instance, that it is raining). A particular instantiation of **D7** is the case when p corresponds to the implication between x's doing a_2 and y's doing a.
So, **T3** might be rephrased as follows:
T3b. ((BEL x (S-DEP x y a p)) $\wedge$ (BEL-POWER y x a_1 ((DONE x a_2) $\supset$ (DONE y a)))) $\supset$
(INFL-POWER y x a_1 (DONE x a_2))

achievable; in particular, he must believe he has the *power* of influencing **y**. Now, what can he do in order to have that power?

Of course, he can do a lot of things, among which just appealing to **y**'s benevolence. But, again, in the context of dependence relationships, the strategy of greatest interest would be *trying to find out some "dependence" of* **y** *on him*. In other words, **x** may try to derive his power of influencing **y** from **y**'s dependence on him relative to some goal. The goal in question might even be the same **p** relative to which he depends on **y**: in that case, **x** will try to persuade **y** that **p** is a *common* goal and that they (**x** and **y**) are related each other by a *mutual* DEP-link; as already observed, this kind of dependence is typical of *cooperation*. Otherwise, **x** will try to find out some other goal **q** relative to which **y** may depend on him, and persuade **y** of their *reciprocal* dependence, that is typical of social *exchange*.

5. CONCLUSIONS AND FURTHER DEVELOPMENTS

In this paper we have tried to move some steps toward a theory of dependence in decentralized intelligent systems. Our aim has been to clarify how to apply such a theory to the problem of communication control among agents. We have tried to show that dependence is the basis and the reason for social interaction. We have sketched how, starting from knowledge about dependence, it is possible for an agent to devise actions of influencing other agents that are "realistically" able to do what he needs. In our view, the realism stems from the agent's coming to believe that he is involved in a dependence relation, and reasons and chooses to act according to that belief.

The present stage of our study is strongly based on the analysis of human social behavior. Further efforts must be carried out in order to achieve a satisfactory theory of both formal and computational aspects. Moreover, a number of interesting aspects have been neglected here, and should be addressed in a further development of the model.

First of all, one would need an analysis of the criteria for inference control in this model. In fact, innumerable dependence relationships may stem from all the possible benefits/detriments any agent may cause to another agent: if **y**'s action repertoire includes an action whose effect avails or damages a goal of **x**'s, **x** depends on **y** relative to that goal. Moreover, dependence relationships are in principle transient: a dependence relationship between **x** and **y** may arise from **x**'s *temporary* lack of an enabling condition for pursuing a given goal. And the goal itself may be a very contingent one in **x**'s mind. Hence, the risk that dependence relationships proliferate without any possible control.

In order to be able to select the most *relevant* dependence relationships within a given social world, and to predict various forms of social interaction on the grounds of these relevant relations, some criteria for the relevance of dependence relationships should be postulated. Among them, a few criteria might be: the presence vs. absence of **x**'s (the dependent person's) power of influencing **y** to perform the action **a** needed for realizing **p**; the "importance" of the goals with regard to which dependence occurs; the frequency of those goals; their being "active", that is, their entering the agent's actual decision processes, versus "inactive"; the contingency versus permanence of the lack of power conditions producing dependence.

Some quantitative aspects of dependence relationships would also improve the predictive power of the model. In particular, dependence may vary in its *degree*. The degree of dependence of **x** on **y** relative to **p** might be defined as the ratio between the *strength* of a DEP-link, which is a function of the coefficient of value of the goal that **p**, and the *number of alternatives* (agents

other than y on which x may depend) available to x. The higher the degree of dependence the more relevant the dependence relationship in question.

Finally, possible predictions might be put forward about the communicative acts occurring between agents in a dependence relationship: in fact, various communicative acts (requests, commands, etc.) might be inferred on the grounds of, say, the type of resource x needs from y (information, physical action, etc.) and the particular goal x needs to influence y to pursue.

ACKNOWLEDGMENTS

We would like to thank Luigia Carlucci Aiello and Rosaria Conte for their precious comments and suggestions. The authors are the only ones responsible for any fault still in the paper.

REFERENCES

[1] Campbell, J.A., D'Inverno, M.P., Knowledge Interchange Protocols, in Y.Demazeau, J.P.Muller (Eds), *Decentralized A.I.*, North Holland, Amsterdam, The Netherlands, 1990.

[2] Castelfranchi, C., Social Power: A Point Missed in Multi-Agent, DAI and HCI, in Y.Demazeau, J.P.Muller (Eds), *Decentralized A.I.*, North Holland, Amsterdam, The Netherlands, 1990.

[3] Cohen, P.R., Levesque, H.J., Intention Is Choice with Commitment, *Artificial Intelligence*, 42: 213-261, 1990.

[4] Cohen, P.R., Levesque, H.J., Rational Interaction as a Basis of Communication, in P.R.Cohen, J.Morgan, M.E.Pollack (Eds.), *Intentions in Communication*, MIT Press, Cambridge, MA, 1990.

[5] Conte, R., Castelfranchi, C., Mind Is Not Enough. Pre-cognitive Bases of Social Interaction. Tech. Rep. TR-IP-PSCS-41, Institute of Psychology, CNR, November 1990.

[6] Conte, R., Miceli, M., Castelfranchi, C., Limits and Levels of Cooperation: Disentangling Various Types of Prosocial Interaction, in *Proc. of the 2nd Workshop on Modelling Autonomous Agents in a Multi-agent World*, Paris, France, August 1990.

[7] Davis, R., Smith, R.G., Negotiation as a Methaphor for Distributed Problem Solving, *Artificial Intelligence*, 20: 63-109, 1983.

[8] Erman, L.D., Hayes-Roth, F., Lesser, V.R., Reddy, D.Raj, The Hersay-II Speech Understanding System: Integrating Knowledge to Resolve Uncertainty, *ACM Computing Surveys*, 12: 213-253, 1980.

[9] Van Dyke Parunak, H., Distributed AI and Manifacturing Control: Some Issues and Insights, in Y.Demazeau, J.P.Muller (Eds), *Decentralized A.I.*, North Holland, Amsterdam, The Netherlands,1990.

A Game Theoretic Approach to Distributed Artificial Intelligence and the Pursuit Problem

Ran Levy
Jeffrey S. Rosenschein
Computer Science Department, Hebrew University
Givat Ram, Jerusalem, Israel
phone:(+972)-2-585-353
fax:(+972)-2-585-439
ranlevy@shum.huji.ac.il, jeff@cs.huji.ac.il

Abstract

Research in Distributed Artificial Intelligence (DAI) often considers the problem of how best to utilize multiple automated agents to accomplish a given task. A canonical problem, the Pursuit Problem [1], was suggested as a useful tool for evaluating alternative approaches to the distribution of knowledge and control among intelligent, cooperative problem-solvers. Work on the Pursuit Problem was carried out by several researchers; for example, Stephens and Merx [14] compare alternative approaches to solving the problem.

In [5], a game theoretic model was proposed for DAI in which each agent works for his own selfish goals. In this paper, we suggest a method for solving the Pursuit Problem using game theoretic techniques, by incorporating the global goal of a group of agents into their local interests. Although applied to the Pursuit Problem, the technique has wider applicability throughout DAI. We present the results of a simulation of the game theory model for the Pursuit Problem, and compare results to those of other models.

1 Introduction

Research in Distributed Artificial Intelligence (DAI) often considers the problem of how best to utilize multiple automated agents to accomplish a given task. Work in DAI has made use of several approaches to the distribution of knowledge and control among intelligent, cooperative problem-solvers. The contract-net model of Davis and Smith [3] uses bidding and contracting to achieve cooperation in the assignment of tasks to processors in a multiprocessor system. Malone *et al.* [10] refined the above framework by introducing a more explicit economic model to cooperation.

Another direction of research is presented in [1]. It is assumed that cooperation among agents is a function of their organizational structure and the way they communicate with one another. Benda *et al.* examined several organizational structures, consisting of three basic components (communicating agents, negotiating agents, and a controlling agent) in connection with a pursuit problem, first presented in that research.

The Pursuit Problem models a configuration of four intelligent agents, together performing a given task that requires some sort of coordination (explicit or implicit). Four blue agents, positioned on different locations over a grid, are attempting to surround a fifth red agent. See Figure 1. The red agent moves randomly in any possible direction (i.e., one not blocked by a blue agent). In addition to the global goal of the system, each agent may have some local goal of its own. Those local needs may contradict both the common goal and local goals of other agents. Some sort of negotiation (explicit or implicit) might be used to achieve a compromise. Benda *et al.* concluded that an organization with one controlling agent and three communicating agents was the best to solve the problem.

Similar conclusions are suggested by Stephens and Merx [14]. They compare the results of using different approaches for solving the Pursuit Problem (autonomous agents, communicating agents, controlling agent, and negotiating agents), and show that the controlling agent model is superior to the others. It guarantees capture of the red agent in all 6 test cases, in a relatively small number of moves. They also noted that the autonomous agents model was the least satisfactory approach, with the explanation that since each agent has a local goal, they did not work efficiently towards achieving the global goal, sometimes completely failing to achieve that global goal.

Gasser and Rouquette [4] developed a framework for representing orga-

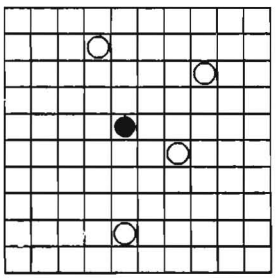


Figure 1: The Pursuit Problem

nizational knowledge, using the pursuit game, to investigate decentralized coordination mechanisms. Their analysis introduces a six-phase solution, where each phase involves a different organization of agents. Singh [13] applied his theory of "group intentions" to the pursuit problem to demonstrate that theory's general utility to DAI.

Rosenschein *et al.* [11, 5, 16] analyze various rationality assumptions on multi-agent system behavior, along with their implications. These results (such as the existence of dominance analysis, and iterated dominance analysis) suggest that Game Theory techniques, along with appropriate rationality assumptions, might allow greater flexibility in the autonomous agents model of coordination. One of the goals of this work is to suggest a method for incorporating global goals into the local interests of all agents through the use of Game Theory techniques.

2 Necessary Concepts from Game Theory

In order to clarify our discussion in the following sections, we here give a brief overview of concepts, definitions and theorems of Game Theory. The material covered here is mainly due to Von Neumann and Morgenstern, Nash, Maschler and Davis, and Shapley.

2.1 Non-Cooperative Games

Let N be a set of n players, $N = \{1, \ldots, n\}$. Let S^i be a finite set of strategies for player i, $i \in N$. We will denote by S the cartesian product $S^1 \times \cdots \times S^n$.

Definition 1 *A* Finite Non-Cooperative Game in Normal Form *is a system* $G = (S^1, \ldots, S^n; H^1, \ldots, H^n)$, *where* S^i *is a non-empty set of strategies of player* i, *and* H^i *is a payoff function,* $H^i : S \rightarrow R$ *for player* i, $i \in N$.

One of the interesting questions about non-cooperative games is whether a player can guarantee his minimal payoff. It is reasonable that a player might prefer to use a "solid" strategy that provides some guaranteed income (though minimal), over taking risks for the chance of making a larger profit, but also with a higher probability of losing. This leads us to the concept of equilibrium points. Definition 2 states the following about an equilibrium point: for each player, as long as the other players maintain their same strategies, his current strategy is the best he's got.

Definition 2 *An n-tuple of strategies* $s \in S$ *is a* Nash Equilibrium Point *if for each player* i, $i \in N$, *and for each strategy* $\widehat{s^i}$ *of player* i,

$$H^i(s|\widehat{s^i}) \leq H^i(s)$$

where the strategy $s|\widehat{s^i}$ *is derived from* s *by substituting* $\widehat{s^i}$ *for* s^i.

In pure strategies, such equilibrium points do not always exist, and even when they do exist, they are not necessarily the best solution for games (the well-known Prisoners' Dilemma is an example of this).

2.2 Cooperative Games

Games with side-payments are games in which the participants get immediate (and transferable) utility (that is, money). In a cooperative game (with side payments), some players may form a coalition and together design their strategies so they can increase their total income. This total income then has to be shared—in some manner—among all the coalition's members. The theory of cooperative games (with side-payments) assumes that a coalition achieves some payment, and deals with the different methods for sharing the profit.

Definition 3 *We define a* Cooperative Game in Coalitional Form with side-payments *as a pair* (N, v), *where* N *is the set of players, and* $v : 2^N \rightarrow R$ *is a payoff function, where* $v(S)$ *is the payment to coalition* S, *and* $v(\emptyset) = 0$.

A cooperative game is therefore stated in terms of the *coalitional function* v. Note that no strategies appear in the definition. Once a coalition is formed, it is of no concern how it gains its utility. A player should take into account the coalitions he may participate in and their incomes, and decide which coalition is the best for him. We will be using the Shapley Value [12] in order to determine the way in which the income is divided; this technique provides us with a single payoff vector specifying the income distribution.

Definition 4 *A* Payoff Vector *for N is a function $x : N \to R$.*

The i-th coordinate in the payoff vector x corresponds to the payment for player i. Let G^N be the set of all cooperative games over N.

Definition 5 *A* solution *to cooperative games is a function $\psi : G^N \to R^N$.*

The function ψ represents a method for sharing $v(N)$ among all players in N.

The Shapley value satisfies three properties (see [15]). The *symmetry* property guarantees that there is no preference by the solution of one player over another player. What is important to the solution concept is not the identity of a player, but the role it plays. *Efficiency* guarantees that players cannot get (according to the solution) more than N can afford. Finally, the *marginality principle* states that a player's "value" to the coalition is measured by his contribution to the coalition.

The Shapley Value may be calculated using the following formula:

$$\psi^i(v) = \sum_{S \subset N, i \notin S} \frac{(n - |S| - 1)!\ |S|!}{n!} [v(S \cup \{i\}) - v(S)]$$

3 A Model for the Pursuit Problem

From our game theoretic point of view, the Pursuit Problem has two aspects to be considered. First of all, the players should be given methods for solving the game, so that they have the ability to cope with problems they face. In addition, we must also set the payment policy according to which the players are being paid for their actions. The idea that local interests of the players should not interfere with the global goal of the group of players should be

incorporated into the payment policy of the game. We are *integrating* a global goal into agents' local goals, not replacing their local goals with a global goal. Another of our design aims for the system was to have the agents reach a solution with the minimum amount of communication.

3.1 Payment Policy

The payment policy we suggest incorporates two ideas. First, the convergence to the final state should be as close as possible to optimal (i.e., fast). Moreover, it should ultimately represent the global interest of the system (i.e., capture). Therefore, the payoff function is the sum of two distinct arguments.

For every move, a player is paid an amount of money proportional to the difference of his distance from the red agent before and after the move. Distance is measured using city block metrics. This amount of money may be positive or negative (or zero, if the player decides to remain in his position). This argument of the payoff function causes the agent to prefer moves that tend to bring the player closer to the red agent.

The second argument of the payoff function encourages agents to coordinate their actions and block the red agent.[1] Given a coordinated strategy of a coalition $S \subseteq N$, we thus use the following arguments:

- The sum of differences in players' distance from the red agent (using the city-block metrics). We denote this sum by $\sum_{i \in S} d^i$.
- The number of blocked escape directions of the red agent, using the strategy of S. Let k_S designate this number.

In order to emphasize the importance of cooperation among players, it is suggested that the second argument of the payoff function will be the k_S's power of some number (unless k_S is zero, in which case the second argument will also be zero). There is, however, one point which must be taken into account: the payoff function should make a distinction between a cooperative and a non-cooperative coalition. This distinction is essential to

[1] The red agent can move in one of four possible directions. We say that a coalition has "blocked" an escape direction if some member of the coalition could physically obstruct that direction were the red agent to choose it. The coalition is rewarded for the number of directions that are simultaneously blocked as a result of its actions.

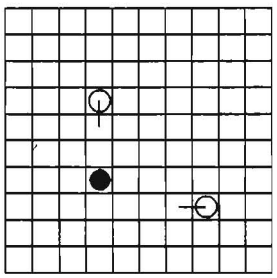


Figure 2: An Example With Two Players

characterizing equilibrium points, as discussed in Section 4.1. For example, consider the second argument to be 2^{k_S}. There is no way to distinguish between a coalition of two players who together block one escape-direction (the payoff to this coalition should be $2 + 2^1 = 4$), and four players moving separately, blocking no escape-direction (the non-cooperative "coalition" of four thus gets $4 + 0 = 4$). A similar argument holds for both 3^{k_S} and 4^{k_S}. As shown in Section 4.1, the proof of the characterization lemma is based on a base of 5 for k_S. Hence it was chosen for the payoff function.

Example. Consider Figure 2, in which only two players appear, for simplicity. N, therefore, is the set $\{1, 2\}$, and:

$$v(\emptyset) = 0$$

$$v(\{1\}) = \sum_{i \in \{1\}} d^i + 5^{k_{\{1\}}} = 1 + 5 = 6$$

$$v(\{2\}) = \sum_{i \in \{2\}} d^i + 5^{k_{\{2\}}} = 1 + 5 = 6$$

$$v(\{1,2\}) = \sum_{i \in \{1,2\}} d^i + 5^{k_{\{1,2\}}} = 2 + 5^2 = 27$$

4 Solution Algorithms

We derive a solution that mixes cooperative games with non-cooperative games. As each player has to make its own decision, without explicit coordination with the others, each player has to solve a four-player non-cooperative

game with side payments. This game will be denoted by G, and what follows is a description of how it should be constructed.

The set of agents (or players) is:

$$N = \{1, 2, 3, 4\}$$

Each player $i \in N$ has a set of pure strategies, given by:

$$S^i = \{\text{South, West, North, East, Stay}\}$$

Let S be defined by:

$$S = \prod_{i \in N} S^i$$

We use $H^i(s)$ in order to specify the payoff to player i for a strategy $s \in S$. Using the payment policy described in the previous section, notice that each move $s \in S$ actually defines a four-player cooperative game with side-payments in the following manner: let $T \subseteq N$ be a coalition of players. Then

$$v_s(T) = \begin{cases} \sum_{i \in T} d^i + 5^{k_T} & \text{if } k_T \neq 0 \\ 0 & \text{if } k_T = 0 \end{cases}$$

and

$$v_s(\emptyset) = 0$$

We refer to this cooperative game as the "local game" for that move. The amount of utility, represented by $v_s(N)$, will be shared among the four players in accordance with the Shapley value. Player i (for $i \in N$) will get $\psi^i(v_s)$, where ψ represents the Shapley value, and

$$\sum_{i \in N} \psi^i(v_s) = v_s(N)$$

Having solved the local game, a player should now return to the "global" non-cooperative game, and define the payoff function for player i ($i \in N$) for each move $s \in S$ by:

$$H^i(s) = \psi^i(v_s)$$

Hence, we complete the formalism of the non-cooperative game, and let G be:

$$G = (S^1, S^2, S^3, S^4;\ \ H^1, H^2, H^3, H^4)$$

The game G may be constructed, analyzed and solved by each player on its own. Because of the desirable properties of equilibrium points, we design our agents so that they choose to play a strategy that corresponds to an equilibrium point in G.

4.1 Equilibrium Points

In this section we inspect the relation of equilibrium points to what we shall call "optimal points." For our purposes, an optimal point represents moves that are best for the system as a whole. It turns out that the game G defined above has at least one equilibrium point, depending on the specific scenario.

Definition 6 *A coordinated strategy $x \in S$ is said to be an* optimal point *in G (defined above) if for all $y \in S$ the following condition holds:*

$$x(N) = \sum_{i \in N} h^i(x) \geq \sum_{i \in N} h^i(y) = y(N)$$

At an optimal point, the coalition N gets the maximum amount of utility it can get in G.

Theorem 1 *A point $x \in S$ is an equilibrium point in G if and only if x is an optimal point in G.*

For the proof of this theorem and subsequent ones, see [9]. A straightforward corollary of the theorem is the following:

Corollary 1 G does have *an equilibrium point, since surely it has an optimal point.*

There are situations in which there exist more than one equilibrium point. An example of such a situation appears in Example 4.1.

Example.

Consider Figure 3. Players 3 and 4 have two symmetric directions in which they can move (regarding distance from the red agent, and blocking escape-directions). Therefore, every two strategies of the form

$$(s^1, s^2, EAST, SOUTH)$$

$$(s^1, s^2, SOUTH, WEST)$$

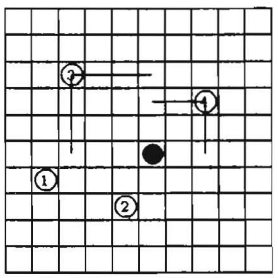


Figure 3: A Situation With Two Equilibrium Points

will be equal in worth to each other.

The payment to the group of all players, N, is the same for all optimal points (this follows immediately from the definition of optimal points). Can there be different situations in which N receives the same payment, and what characterizes such situations?

Lemma 1 *If at two points in G, N gets the same payment, then:*

1. *The number of blocked escape directions in the two points is the same.*
2. *The total difference in the distance of players from the red agent is the same in the two points.*

For the proof, see [9]. The proof uses the constant 5 as the base of the second argument in the payoff function as a result of the fact that there are 4 blue agents. Were there 5 blue agents, the base would thus need to be 6.

4.2 Scope and Limitations of the Algorithm

One of the main goals of our system design was to minimize the need for communication among agents. From our experiments we concluded that in our model, a general solution without any communication is impossible.[2]

Consider the left side of Figure 4 in which the two blue agents are at distance 3 from each other. Agent 1 will surely select $SOUTH$ as his next move. Agent 2 may choose $SOUTH$ or $WEST$ (ignore for the moment

[2]Obviously, another model, even one that continues to use Game Theory, might have different properties, and may well have different limitations as well.

Figure 4: Typical Problems, Cases 1 and 2

agents 3 and 4, and the exact equilibrium points). If agents 1 and 2 would not coordinate themselves via a negotiation process, agent 2 may move to the $WEST$ and block agent 1, who cannot recover from this afterwards. In the right side of Figure 4 (distance 2 between agents) coordination is needed to prevent a situation where both agents move to the same location (agent 1 moves to the $SOUTH$, agent 2 to the $WEST$).

Experimental results supported the hypothesis that if the distance between agents is 3 or less (using city block metrics), explicit negotiation is needed to synchronize them.

5 Experimental Results

A computer program to implement our model was written in Common Lisp. Two questions were to be tested and answered by the simulation:

- What are the initial states of the problem for which the model does provide suitable solutions?
- How does the existence of more than a single equilibrium point affect the need for communication?

The question of "convergent" initial states, i.e., those states for which we know the solution to converge, depends on the definition of the payoff function. Therefore, the following discussion is closely tied to our particular payoff function. Once a player identifies a *single* free escape direction, he does not give it up. If two (or more) players compete over that direction, the solution will probably not converge. We say "probably" because random

Figure 5: Step 8 of Case I and Step 7 of Case II

movements of the red agent may change one situation into the other one, in which the solution may converge. In other words, if the initial state is such that resources are *available* to players, and all they have to do is decide on the best method to share them, then success is guaranteed (subject to random behavior of the red agent). In order to cope with situations in which resources are not directly available, one should correct the payoff function in an appropriate way.

In order to answer the question of how multiple equilibrium points affect the need for communication, let us turn to an example (the middle of the game is seen on the left side of Figure 5). The original problem was solved in 11 moves, but from step 8 on, interference of an outside user (who simulates an inter-agent discussion to resolve symmetrical equilibrium points) is necessary. The answer of the outside user on step 3 and 4 is actually an arbitrary selection. However, from step 8 in test-case I and step 7 in test-case II (see Figure 5), a bit of thought must be given in order to prevent the game from reaching a dead end. This is necessary because in those steps, blue agents 1 and 2 get too close to each other. When agents sense that they are very close, they should coordinate via communication, so that they can prevent themselves from competing over the same resource.

In Section 4.1 we discuss the symmetry of multiple equilibrium points, and show that this symmetry indicates the existence of equivalent actions by one or more players. Test cases I and II show that as long as the blue agents who have symmetric actions are far from each other and from the red agent, there is no importance to the equilibrium point that is selected. However, as the blue agents get closer to the red agent (up to a distance of 3 or less from the red agent), the significance of intra-agent communication grows,

and sometimes becomes necessary. These phenomena may also be observed from other test cases. See [9] for full experimental test results.

One problem that needed to be addressed in the simulation was that of calculating equilibrium points in non-cooperative games. This subject is discussed in the literature (see, for example, the work of Chin *et al.* [2], Harsanyi [6] and Rosenmüller [7]). There are several mathematical algorithms to find and calculate equilibrium points in such games. In the actual implementation, however, we used a simple exhaustive search over all points in the non-cooperative game in order to find equilibrium points. In a more realistic system, simplicity would be abandoned in favor of efficiency.

6 Comparison and Conclusions

In [14], Stephens and Merx summarize the performance results of four methods of solution on six different scenarios. They specify three possible outcomes, *capture*, *stalemate*, and *escape*. We used the six scenarios as test cases for our simulation. The results are summarized in Appendix A. In three of the cases the game ended in the capture of the red agent, and in two others in stalemate. In one case (scenario 1 of [14]) the simulation did not even get close to the solution, with all the agents clustered far away from the red agent and satisfied to remain there. The payoff function was not designed for that kind of situation and failed to motivate the blue agents to capture the red agent. Table 1 extends Table 4 in [14] to include our results.

Interestingly, our model fails in scenario 6 where *all* other models succeed. This result demonstrates the nature of the underlying model. In the first few stages of the game, the agents appear to be proceeding correctly. However, at a particular point near the end of the game, the agents continue in their rush towards the original capture positions, even though the red agent has meanwhile scurried away. In Section 5 we pointed out that communication should be used to select one of several equilibrium points. This example shows that there are cases in which communication might also be used to prevent the system agents from taking a misguided path.

Although there are random elements involved (i.e., the movement of the red agent), we can still reach some preliminary conclusions regarding the effectiveness of our model. In those cases for whom our payoff function is known to be appropriate, the results show convergence to the solution,

sometimes with a shorter number of moves than in other methods. However, the definition of the payoff function has the disadvantage of not covering all cases, as seen by cases 1, 5 and 6. To its credit, however, agents in this model do not need to communicate with one another most of the time. Communication is needed only when agents get close to other agents.

See [9] for the application of these techniques to the real-world problem presented originally in [8].

A Test Results

Table 1: Comparison Of Performance Results

DAI System	Scenario 1		Scenario 2	
	Moves	Outcome	Moves	Outcome
Autonomous-Agent	10	Escape	10	Stalemate
Limited-Communication	10	Escape	10	Stalemate
Negotiating-Agent	11	Stalemate	11	Capture
Controlling-Agent	17	Capture	12	Capture
Game-Theory	None	Escape	8	Capture

DAI System	Scenario 3		Scenario 4	
	Moves	Outcome	Moves	Outcome
Autonomous-Agent	7	Stalemate	10	Stalemate
Limited-Communication	7	Stalemate	10	Stalemate
Negotiating-Agent	8	Capture	11	Capture
Controlling-Agent	10	Capture	12	Capture
Game-Theory	7	Capture	15	Capture

DAI System	Scenario 5		Scenario 6	
	Moves	Outcome	Moves	Outcome
Autonomous-Agent	7	Stalemate	7	Capture
Limited-Communication	7	Stalemate	7	Capture
Negotiating-Agent	8	Capture	8	Capture
Controlling-Agent	10	Capture	10	Capture
Game-Theory	6	Stalemate	6	Stalemate

References

[1] M. Benda, V. Jagannathan, and R. Dodhiawalla. On optimal cooperation of knowledge sources. Technical Report BCS-G2010-28, Boeing AI Center, Boeing Computer Services, Bellevue, Washington, August 1985. Cited in Les Gasser and Nicolas Rouquette, Representing and Using Organizational Knowledge in Distributed AI Systems. *Proceedings of the 1988 Workshop on Distributed Artificial Intelligence*, May 1988.

[2] H. H. Chin, T. Parthasarathy, and T. E. S. Rayhaven. Structure of equilibria in n-person non-cooperative games. *International Journal of Game Theory*, 3:1 - 19, 1974.

[3] Randall Davis and Reid G. Smith. Negotiation as a metaphor for distributed problem solving. *Artificial Intelligence*, 20(1):63–109, 1983.

[4] Les Gasser and Nicolas Rouquette. Representing and using organizational knowledge in dai systems. In Les Gasser, editor, *Proceedings of the 1988 Workshop on Distributed Artificial Intelligence*, Lake Arrowhead, California, May 1988.

[5] Michael R. Genesereth, Matthew L. Ginsberg, and Jeffrey S. Rosenschein. Cooperation without communication. In *Proceedings of The National Conference on Artificial Intelligence*, pages 51–57, Philadelphia, Pennsylvania, August 1986. The American Association for Artificial Intelligence.

[6] J. C. Harsanyi. Oddness of the number of equilibrium points: A new proof. *International Journal of Game Theory*, 2:235 - 250, 1973.

[7] J. Rosenmüller. On a generalization of the lemke-howson algorithm to noncooperative n-person games. *SIAM Journal on Applied Mathematics*, 21(1):73–79, 1971.

[8] Yan Jin and Takeo Koyama. Multiagent planning through expectation based negotiation. In *Proceedings of the 10th International Workshop on Distributed Artificial Intelligence*, Bandera, Texas, October 1990.

[9] Ran Levy. A game theoretic approach to distributed artificial intelligence and the pursuit problem. Master's thesis, Hebrew University, 1991.

[10] Thomas W. Malone, Richard E. Fikes, and M. T. Howard. Enterprise: A market-like task scheduler for distributed computing environments. In B. A. Huberman, editor, *The Ecology of Computation*, pages 177 – 205. North-Holland Publishing Company, Amsterdam, 1988.

[11] Jeffrey S. Rosenschein and Michael R. Genesereth. Deals among rational agents. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 91 – 99, Los Angeles, California, August 1985.

[12] Lloyd S. Shapley. A value for n-Person games. In Alvin E. Roth, editor, *The Shapley Value*, chapter 2, pages 31–40. Cambridge University Press, Cambridge, 1988.

[13] Munindar Singh. Group intentions. In *Proceedings of the 10th International Workshop on Distributed Artificial Intelligence*, Bandera, Texas, October 1990.

[14] L. Stephens and M. Merx. Agent organization as an effector of dai system performance. In Miroslav Benda, editor, *Proceedings of the 9th Workshop on Distributed Artificial Intelligence*, pages 263–292, Bellevue, Washington, September 1989.

[15] H. P. Young. Individual contribution and just compensation. In Alvin E. Roth, editor, *The Shapley Value*, chapter 17, pages 267–278. Cambridge University Press, Cambridge, 1988.

[16] Gilad Zlotkin and Jeffrey S. Rosenschein. Negotiation and task sharing among autonomous agents in cooperative domains. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 912–917, Detroit, Michigan, 1989.

Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

DFKI
-Bibliothek-
PF 2080
6750 Kaiserslautern
FRG

DFKI Publikationen

Die folgenden DFKI Veröffentlichungen oder die aktuelle Liste von erhältlichen Publikationen können bezogen werden von der oben angegebenen Adresse.
Die Berichte werden, wenn nicht anders gekennzeichnet, kostenlos abgegeben.

DFKI Publications

The following DFKI publications or the list of currently available publications can be ordered from the above address.
The reports are distributed free of charge except if otherwise indicated.

DFKI Research Reports

RR-90-01
Franz Baader: Terminological Cycles in KL-ONE-based Knowledge Representation Languages
33 pages

RR-90-02
Hans-Jürgen Bürckert: A Resolution Principle for Clauses with Constraints
25 pages

RR-90-03
Andreas Dengel, Nelson M. Mattos: Integration of Document Representation, Processing and Management
18 pages

RR-90-04
Bernhard Hollunder, Werner Nutt: Subsumption Algorithms for Concept Languages
34 pages

RR-90-05
Franz Baader: A Formal Definition for the Expressive Power of Knowledge Representation Languages
22 pages

RR-90-06
Bernhard Hollunder: Hybrid Inferences in KL-ONE-based Knowledge Representation Systems
21 pages

RR-90-07
Elisabeth André, Thomas Rist: Wissensbasierte Informationspräsentation:
Zwei Beiträge zum Fachgespräch Graphik und KI:
1. Ein planbasierter Ansatz zur Synthese illustrierter Dokumente
2. Wissensbasierte Perspektivenwahl für die automatische Erzeugung von 3D-Objektdarstellungen

24 pages

RR-90-08
Andreas Dengel: A Step Towards Understanding Paper Documents
25 pages

RR-90-09
Susanne Biundo: Plan Generation Using a Method of Deductive Program Synthesis
17 pages

RR-90-10
Franz Baader, Hans-Jürgen Bürckert, Bernhard Hollunder, Werner Nutt, Jörg H. Siekmann: Concept Logics
26 pages

RR-90-11
Elisabeth André, Thomas Rist: Towards a Plan-Based Synthesis of Illustrated Documents
14 pages

RR-90-12
Harold Boley: Declarative Operations on Nets
43 pages

RR-90-13
Franz Baader: Augmenting Concept Languages by Transitive Closure of Roles: An Alternative to Terminological Cycles
40 pages

RR-90-14
Franz Schmalhofer, Otto Kühn, Gabriele Schmidt: Integrated Knowledge Acquisition from Text, Previously Solved Cases, and Expert Memories
20 pages

RR-90-15
Harald Trost: The Application of Two-level Morphology to Non-concatenative German Morphology
13 pages

RR-90-16
Franz Baader, Werner Nutt: Adding Homomorphisms to Commutative/Monoidal Theories, or: How Algebra Can Help in Equational Unification
25 pages

RR-90-17
Stephan Busemann
Generalisierte Phasenstrukturgrammatiken und ihre Verwendung zur maschinellen Sprachverarbeitung
114 Seiten

RR-91-01
Franz Baader, Hans-Jürgen Bürckert, Bernhard Nebel, Werner Nutt, and Gert Smolka :
On the Expressivity of Feature Logics with Negation, Functional Uncertainty, and Sort Equations
20 pages

RR-91-02
Francesco Donini, Bernhard Hollunder, Maurizio Lenzerini, Alberto Marchetti Spaccamela, Daniele Nardi, Werner Nutt:
The Complexity of Existential Quantification in Concept Languages
22 pages

RR-91-03
B.Hollunder, Franz Baader: Qualifying Number Restrictions in Concept Languages
34 pages

RR-91-04
Harald Trost
X2MORF: A Morphological Component Based on Augmented Two-Level Morphology
19 pages

RR-91-05
Wolfgang Wahlster, Elisabeth André, Winfried Graf, Thomas Rist: Designing Illustrated Texts: How Language Production is Influenced by Graphics Generation.
17 pages

RR-91-06
Elisabeth André, Thomas Rist: Synthesizing Illustrated Documents
A Plan-Based Approach
11 pages

RR-91-07
Günter Neumann, Wolfgang Finkler: A Head-Driven Approach to Incremental and Parallel Generation of Syntactic Structures
13 pages

RR-91-08
Wolfgang Wahlster, Elisabeth André, Som Bandyopadhyay, Winfried Graf, Thomas Rist
WIP: The Coordinated Generation of Multimodal Presentations from a Common Representation
23 pages

RR-91-09
Hans-Jürgen Bürckert, Jürgen Müller, Achim Schupeta
RATMAN and its Relation to Other Multi-Agent Testbeds
31 pages

RR-91-10
Franz Baader, Philipp Hanschke
A Scheme for Integrating Concrete Domains into Concept Languages
31 pages

RR-91-11
Bernhard Nebel
Belief Revision and Default Reasoning: Syntax-Based Approaches
37 pages

RR-91-13
Gert Smolka
Residuation and Guarded Rules for Constraint Logic Programming
17 pages

RR-91-15
Bernhard Nebel, Gert Smolka
Attributive Description Formalisms ... and the Rest of the World
20 pages

RR-91-16
Stephan Busemann
Using Pattern-Action Rules for the Generation of GPSG Structures from Separate Semantic Representations
18 pages

DFKI Technical Memos

TM-89-01
Susan Holbach-Weber: Connectionist Models and Figurative Speech
27 pages

TM-90-01
Som Bandyopadhyay: Towards an Understanding of Coherence in Multimodal Discourse
18 pages

TM-90-02
Jay C. Weber: The Myth of Domain-Independent Persistence
18 pages

TM-90-03
Franz Baader, Bernhard Hollunder: KRIS: Knowledge Representation and Inference System -System Description-
15 pages

TM-90-04
Franz Baader, Hans-Jürgen Bürckert, Jochen Heinsohn, Bernhard Hollunder, Jürgen Müller, Bernhard Nebel, Werner Nutt, Hans-Jürgen Profitlich: Terminological Knowledge Representation: A Proposal for a Terminological Logic
7 pages

TM-91-01
Jana Köhler
Approaches to the Reuse of Plan Schemata in Planning Formalisms
52 pages

TM-91-02
Knut Hinkelmann
Bidirectional Reasoning of Horn Clause Programs: Transformation and Compilation
20 pages

TM-91-03
Otto Kühn, Marc Linster, Gabriele Schmidt
Clamping, COKAM, KADS, and OMOS: The Construction and Operationalization of a KADS Conceptual Model
20 pages

TM-91-04
Harold Boley
A sampler of Relational/Functional Definitions
12 pages

TM-91-05
Jay C. Weber, Andreas Dengel and Rainer Bleisinger
Theoretical Consideration of Goal Recognition Aspects for Understanding Information in Business Letters
10 pages

DFKI Documents

D-89-01
Michael H. Malburg, Rainer Bleisinger:
HYPERBIS: ein betriebliches Hypermedia-Informationssystem
43 Seiten

D-90-01
DFKI Wissenschaftlich-Technischer Jahresbericht 1989
45 pages

D-90-02
Georg Seul: Logisches Programmieren mit Feature -Typen
107 Seiten

D-90-03
Ansgar Bernardi, Christoph Klauck, Ralf Legleitner: Abschlußbericht des Arbeitspaketes PROD
36 Seiten

D-90-04
Ansgar Bernardi, Christoph Klauck, Ralf Legleitner: STEP: Überblick über eine zukünftige Schnittstelle zum Produktdatenaustausch
69 Seiten

D-90-05
Ansgar Bernardi, Christoph Klauck, Ralf Legleitner: Formalismus zur Repräsentation von Geo-metrie- und Technologieinformationen als Teil eines Wissensbasierten Produktmodells
66 Seiten

D-90-06
Andreas Becker: The Window Tool Kit
66 Seiten

D-91-01
Werner Stein , Michael Sintek
Relfun/X - An Experimental Prolog Implementation of Relfun
48 pages

D-91-03
Harold Boley, Klaus Elsbernd, Hans-Günther Hein, Thomas Krause
RFM Manual: Compiling RELFUN into the Relational/Functional Machine
43 pages

D-91-04
DFKI Wissenschaftlich-Technischer Jahresbericht 1990
93 Seiten

D-91-06
Gerd Kamp
Entwurf, vergleichende Beschreibung und Integration eines Arbeitsplanerstellungssystems für Drehteile
130 Seiten

D-91-07
Ansgar Bernardi, Christoph Klauck, Ralf Legleitner
TEC-REP: Repräsentation von Geometrie- und Technologieinformationen
70 Seiten

D-91-08
Thomas Krause
Globale Datenflußanalyse und horizontale Compilation der relational-funktionalen Sprache RELFUN
137 pages

D-91-09
David Powers and Lary Reeker (Eds)
Proceedings MLNLO'91 - Machine Learning of Natural Language and Ontology
211 pages
Note: This document is available only for a nominal charge of 25 DM (or 15 US-$).

D-91-10
Donald R. Steiner, Jürgen Müller (Eds.)
MAAMAW'91: Pre-Proceedings of the 3rd European Workshop on „Modeling Autonomous Agents and Multi-Agent Worlds"
246 pages

MAAMAW'91 Pre-Proceedings of the 3rd European Workshop on "Modeling Autonomous Agents and Multi-Agent Worlds"

D-91-10
Document

Donald D. Steiner, Jürgen Müller (Eds.)