The Knowledge Engineering Review

On distributed artificial intelligence

LUIS EDUARDO CASTILLO HERN
Department of Artificial Intelligence, University of Edinburgh

Abstract

Distributed Artificial Intelligence has been loosely defined in terms of computation by distributed, intelligent agents. Although a variety of projects employing widely ranging methodologies have been reported, work in the field has matured enough to reveal some consensus about its main characteristics and principles. A number of prominent projects are described in detail, and two general frameworks, the *system conceptual model* and the *agent conceptual model*, are used to compare the different approaches. The paper concludes by reviewing approaches to formalizing some of the more critical capabilities required by multi-agent interaction.

1 Introduction

This review describes current work in distributed Artificial Intelligence (DAI). This field has now matured enough to permit identification of the main characteristics of the field and some of the major issues.

The article is organized into five sections. Section two provides a brief introduction to the motivations within the field on the general view of DAI as the area dealing with cooperative solution of problems by decentralized, loosely coupled collections of intelligent agents. It defines some initial concepts and terminology to focus attention on the tradeoffs involved, and discusses advantages and disadvantages of distribution, and the relation between DAI and other AI areas and major research themes.

As the field of DAI matures there has been an increasing emphasis on the need for research into knowledge, belief, communciation and multi-agent interaction. Three main issues have emerged: global coherence, knowledge representation, and communication. Global coherence concerns how well the system as a whole, functions in a strategic sense. Knowledge representation in each agent can be specialized; this gives rise to issues in reasoning, representation and diversity of knowledge. Communication also has more than one role: as a way of achieving cooperation and as a way of obtaining information.

Section two also provides a conceptual framework consisting of a number of dimensions of the conceptual models of both the system and the agents. This provides a common language for describing DAI and a way of comparing and contrasting widely ranging methodologies. The system model moderates the interactions and actions of the agents leading to coherent cooperation. The agent model relates to the agents' structure, capabilities and awareness.

In section three the proposed framework is applied to four "problem solving architectures" of different grain size. In each of these examples there is a general description of the system, a review of both the conceptual model of the system and agents, and a general discussion of specific features of the example in terms of the above framework.

Section four describes formal approaches to the central issues of knowledge and belief representation, communication and conflict management, and other aspects of multi-agent interaction. Section five draws some general conclusions.

2 Distributed artificial intelligence

2.1 Initial concepts

Distributed artificial intelligence is concerned with the cooperative solution of problems by a *decentralized and loosely coupled* collection of agents (also known as actors, knowledge sources or node processors). Each agent can sometimes be embodied in a distinct processor node (but not necessarily), all are logically independent one from another. The processor nodes may be physically as well as logically distinct. Agents display sophistication in an AI sense with a capability for reasoning, planning and communicating.

The agents *must cooperate* as individually they may not have sufficient information to solve the entire problem; or be the most appropriate to do the task. Mutual sharing of information is necessary to allow the group as a whole to produce a solution. Agents can share tasks and/or results. Planning is an inherent capability in DAI systems. When more than one agent is acting they require some type of planning capability in order to cooperate and reduce conflict.

The term *decentralized* means that control (expertise), data and knowledge are logically and often geographically distributed. There is neither global control nor global data storage. *Loosely coupled* means that the individual agents spend more time in computation than in communication.

2.1.1 Why distribute?

Distribution is appropriate in problems with multiple active agents, natural geographic decomposition, natural functional decomposition, or when it is required to distribute control. Distributed problem solving may also be a source of execution speed, since it may be easier to have twenty machines cooperate than to build a machine twenty times faster. Flexibility is also an important feature that is enhanced in distributed systems. Motivation for work in DAI is addressed by Chandrasekaran (1981), in an introductory article; Rosenschein (1985b) and Huhns (1987) also address this question. Among the motivations are:

1 "Cooperation among multiple expert systems with different but complementary (possibly overlapping) expertise, can provide a solution for problems whose domains are not contained in one expert system."
2 Reusability, as a small independent expert system could be part of different distributed expert systems.
3 Efficiency due to parallelism, similar tasks can be executed by groups of specialized processors.
4 Decomposition of processing is a basic strategy for controlling the complexity of computation. DAI is the most appropriate solution when the problem itself is distributed, as in distributed sensor nets and distributed information retrieval.
5 Appropriate distributed computing increases the prospects for graceful degradation of response when there is degradation of input data or a failure of portions of the system. Distribution of control avoids "bottlenecks".
6 Distribution is a more "natural" approach to open systems. As the system grows and increases its complexity, a distributed mode is more adaptable to change, e.g. upward extensibility.
7 Potentially, DAI systems can solve problems that are too large for a centralized system.

2.1.2 Relationships between DAI and other areas

DAI imports most of its basic techniques and methodologies from AI: knowledge representation, theories of belief, and agent formalisms. These basic techniques and extensions are created through generalization for distributed or parallel domains. Other areas which relate to DAI are distributed systems (particularly formalisms or communication), parallel hardware and software, and formal specification in general.

In turn the general research interests of AI are stimulated by research in DAI—techniques for reasoning about knowledge, actions, deductions, and planning. Formal systems for knowledge representation, action and belief systems will be strongly influenced by research in DAI. This is due to the need for an agent to reason about other agents' intentions, plans, and beliefs to improve the understanding of these techniques. Agents need to know how to interweave execution and planning and when to ask other agents for information and when to compute it themselves. In execution and monitoring, techniques need to be developed to allow agents to check the execution of other agents in DAI systems.

2.1.3 Problems presented in DAI systems

DAI also brings with it a host of problems. Once we distribute control, it becomes less clear how we can ensure coherent, cooperative behaviour. Once we distribute responsibility for various parts of a problem, each agent has only incomplete knowledge about the world, and those incomplete views may become inconsistent.

Some difficulties in DAI systems are pointed out in Cammarata et al. (1983) and McArthur et al. (1982). These problems can be summarized into four groups: planning, cooperation, interweaving activities, and information gathering. In planning, agents must be able to decide when to solve a problem or perform their tasks and when to ask other agents to solve or perform. They must cooperate, i.e. be able to decide when to interrupt their work to satisfy information requests from other agents and when to accept other tasks. When gathering information because of the dynamic environment and because agents send information asynchronously, agents must decide how and when to update their own state. Updates can be made by computing or by communicating. Beside these activities, agents must have the capability of interweaving them in an effective way.

2.2 Research issues

The nature of DAI systems enhances agent perception of the environment and itself. Agents need to be aware of their own capabilities and need to reason about other agents' intentions, plans and beliefs. This fact focuses research onto these issues more than traditional AI.

Early work focused on large-grained DAI systems, such as the contract net system or blackboard based architectures. New research is refining these architectures and experimenting with control strategies; providing frameworks with which to integrate heterogeneous control strategies and fine grained distribution of activities. Also, there is an increasing emphasis on the formal basis for representing knowledge, belief, and interaction.

Several research problems have become salient to the distribution of control, responsibility and knowledge inherent in DAI systems: global coherence and cooperation, knowledge organization, and communication.

2.2.1 Global coherence

One general characteristic of DAI systems is their global coherence. The coherence of a system has to do with how well the system as a whole functions in a strategic sense. How readily the activities of the system as a whole may be assessed and traced to the actions of specific elements. The end result of the system's activity should have certain desired properties; these properties may differ depending on the system, but usually include such things as correctness, or agreement among individual agents as a "final answer". Underlying global coherence are two critical factors: control and task decomposition.

Task decomposition and coordination are strongly linked; once a task has been decomposed, coordination must be considered. Agents' actions must be coordinated to produce appropriate resources (information and partial results) at the correct time. A centralized control model is less

flexible and less capable of taking maximum advantage of distributed processing resources than a system in which control is also decentralized. Explicit decentralization is easier to implement (map) to a distributed architecture. Since a centralized control system is based on a global view of the problem the important interactions between agents can be predicted and synchronized. The extent to which control is distributed in a system obviously affects its coherence. A system with a single central control mechanism more readily displays overall strategic focus and is more understandable in the large. When local agents have control over their own actions, have their own goals and their own utility functions they will pursue their own interests. The question then arises of how these locally motivated agents can be designed so as to fulfil global goals and which principles of cooperation/competition should be used to insure such coherence.

Possible goals of cooperation as pointed out by Durfee and Lesser, (1987b), include:

- Improving performance (from an overall solution) by working in parallel.
- Increasing the variety of solutions by letting "robust" agents form local solutions without being overly influenced by other agents.
- To increase credibility in solutions by having agents rederive (possibly by different methods) each other's results.
- To increase the probability of success in finding a solution. Because it could be found despite agent failure by assigning important tasks to multiple agents.
- To minimize the time agents must wait for results from each other by coordinating activities.
- To reduce communication by being more selective about what messages are exchanged.
- To improve the use of computational resources by allowing agents to exchange tasks and hence provide a load balance mechanism.

Because agents cannot achieve all of these goals they cooperate differently in each particular case. Self interest in agents can lead to cooperation, though not always. Agents whose individual goals are compatible might unwittingly cooperate as they pursue these goals separately; or they may cooperate intentionally to better achieve their self interest. Three important approaches have been taken to improve coordination among cooperative agents: multi-agent planning, negotiation and the functionally effective cooperative approach.

The first approach, (multi-agent planning) typically has one planning agent who plans for the entire system. The planning agent forms a multi-agent plan that specifies the actions each agent should take, then the planning agent distributes the plan amongst the agents. An alternative form of multi-agent planning is to provide each agent with accurate models of the other agents (e.g. Georgeff process models, or Konolige belief systems, see section four for a detailed review), and allow the agents to cooperatively form a multi-agent plan. In the negotiation approach an agent will decompose a task into subtasks and will announce the tasks he wants done through a negotiating protocol (Davis and Smith, 1983). In the functionally accurate cooperative approach, agents cooperate by exchanging partial solutions and eventually converge on an overall system solution. To cooperate coherently the agents need to predict what partial solutions will be exchanged. These concepts will be covered in detail in the next section.

2.2.2 Knowledge representation

The "representation diversity" of knowledge refers to the extent to which each agent has its own specialized knowledge and localized representation of information. If system elements differ in their representation technique then the agents need to know how to reformulate information of other agents into their own local representation. In this respect the degree of specialization of the individual agent has a great impact. In a highly specialized problem solving agent with its own internal structure this problem of reformulation becomes severe. In systems with a large number of "primitive" nodes, reformulation is a lesser problem.

Knowledge is divided into domain specific Knowledge Sources (KSs) each of which contains

expertise of a particular domain. KSs are commonly formed in agreement with a data hierarchy for the problem to be solved. KSs are usually chosen to handle data at one level of abstraction or to bridge two levels (Erman et al., 1980). Indexing is used to connect the agents who have tasks to be executed with agents that are appropriate for doing those tasks. Smith (1979) recognizes two major indexing types: task oriented knowledge and knowledge source oriented knowledge. Task oriented knowledge is useful for finding agents capable of executing tasks and KS oriented indexing is useful for finding tasks to be executed by an agent.

The distribution of knowledge has another two inherent dimensions: Static knowledge, or how knowledge is initially loaded into the agents, and the dynamic distribution or how knowledge is transferred between agents as work on the global problem proceeds. The criteria for a good dynamic distribution of knowledge are minimization of message traffic and avoidance of critical function nodes that could reduce processing speed and create reliability problems.

2.2.3 Communication

There are two main issues related to communication: the tradeoff between communication and computation and, communication as a way of achieving cooperation and coherence in the system.

Communication vs Computation

At the heart of the tradeoff between communication and computation lies the fact that information can be gained in two ways: through querying an agent who already has it, or by performing the computations to answer the question itself. This second option may not always be possible because of the lack of information or knowledge to do it. By communicating intermediate results, agents may be able to save each other from duplicating computation. However if communication is expensive or undesirable it may be desirable to increase computing time. Experimentation in tradeoffs for search problems is reported in Kornfeld (1982).

Communication can be slower than computation, especially if the agents of the system are very loosely coupled. Attempting to interconnect a large number of high speed processors can easily lead to saturation of the bandwidth available. However, over short distances permanent hardwired links can be very effective. Davis and Smith (1983) point out several implications from these observations. For one thing it means that efficient communciation is needed so as not to saturate the available channels. With an efficient protocol, less time is used in communicating. This leads to the closely related issue of grain size of the messages that are passed amongst elements, with advantages for information sharing among elements being balanced against the difficulty of handling such complex messages. The grain size and the modularity of the problem decomposition have a great impact on the overheads involved in task distribution. Dependent tasks will require large communication overheads to maintain global coherence, so fine grained distribution of tasks may consume excessive resources through communication overheads.

Communication and Cooperation

In some instances communication becomes the main element responsible for cooperation. In organizational paradigms of negotiation, messages become complex structures. Protocols help determine the content of the information transmitted rather than simply providing a means to send bits from one node to another.

In distributed environments there are three main characteristics of the communication content that have an impact on coherence (Durfee, 1987a): relevance, timeliness and completeness. *Relevance* regarding the information consistent with the solution and the total information derived by the system. More irrelevant messages attain more distraction from the goal. *Timeliness* measures the impact of the message on the current activity of the receiving node. Message timeliness varies as node activity progresses. If you receive the answer to the task you were about to begin, then it has a high timeliness. Message *completeness* refers to the fraction of a complete solution that the message represents. This affects the coherence by reducing the number of partially or fully redundant

messages communicated amongst the nodes. Achieving completeness is important in minimizing communication.

These characteristics are interdependent, and developing communication policies to improve coherence will trade-off among them. The basic policy of the experiments by Durfee (1987), is a "locally complete" communication policy; i.e. an agent transmitting only those hypotheses which it cannot improve upon. This policy minimizes the number of transmissions and maximizes completeness. A drawback is that agents tend to keep hypotheses too long. A time out mechanism was introduced to handle timeliness.

2.3 Framework

DAI is a young field whose ground is reflected in the various efforts that have been made to classify work in the field. One of the first classifications published was presented in the workshop of DAI reported by Smith (1984). It consists of a graph that shows the classes of applications that were presented in the workshop. This is mainly a grouping of the presented papers into five subcategories: Application, Problem Characteristics, Methodology, Goal, and Design Level. Each area is subdivided several times and publications fitted into these slots. What is lacking is a common framework language to compare and contrast the research reported.

As the field has evolved, some consensus has grown about the characteristics of DAI systems. This makes it possible to define a framework to classify DAI. Rosenschein (1985b), in an appendix to his doctoral thesis presents an overview of DAI and introduces the different schools of thought, research issues and prototypes of systems in different domains of application. Jagannathan and Rajendra (1986) provide an annotated bibliography. This is organized by the institution in which the main author of the paper works. It does not provide any insights or other classification of the work reported. A related classification is made by Doran (1986), who distinguishes between structure and behaviour; emphasising behaviour as a function of structure. Huhns (1987), presents a framework composed of eight dimensions in which DAI systems can be classified. These dimensions are: system model, grain, system scale, agent dynamism, agent autonomy, agent resources, agent interaction and result formation.

The framework presented in this section considers all of these concepts as they are incorporated in conceptual models both of the system and the agent. This separation between system and agent is important because it is possible to analyze, compare and relate characteristics of the system model with characteristics of the agent model. The system model moderates the actions and interactions of the agents which lead to a coherent cooperation. The agent model dimensions relate to the structure and capabilities of the agents. This determines the potential behaviour and awareness of its environment. These dimensions are presented in Fig. 1. In this framework only dimensions of the conceptual model are considered. In an analysis of particular instantiations of these models, further dimensions relating to the implementation would need to be included.

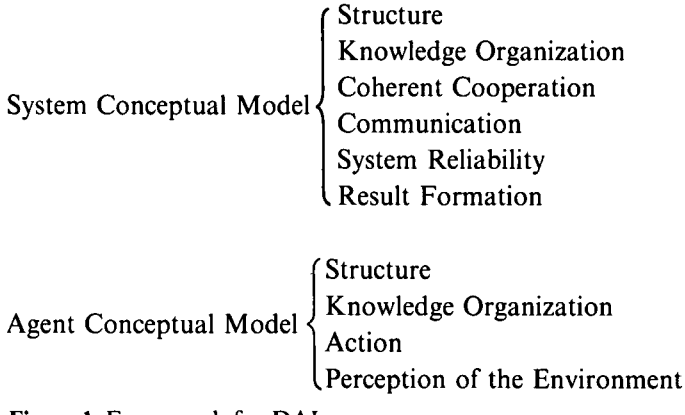

Figure 1 Framework for DAI.

2.3.1 System conceptual model

Structure refers to the agents' interdependencies and relationships, or how agents work together as a cluster for a given task. The *instantiations* can be dynamic in the sense that they can be modified in terms of components or roles; or static in the sense that the organization and roles are predefined and do not change during execution.

Knowledge organization within the system refers to the representation, distribution and retrieval of knowledge. How is the knowledge partitioned and how can it be accessed? Generally, in AI, knowledge is grouped according to domain specific knowledge sources (experts), each of which can be physically apart and can be accessed by more than one agent. *Static knowledge* refers to the initially loaded knowledge; *dynamic knowledge* refers to the acquisition of new knowledge by the agent (by assignment of new KSs, by communication with other agents or by computation). *Coherent cooperation* refers to the organizational policies used to achieve global coherence. The important issues in coherent cooperation are task decomposition, coordination policies, control and synchronization. *Communication* within the system refers to three main issues: (a) the grain size of the message, (b) the homogeneity of the communications protocol, (c) sharing resources as a form of communication. *System reliability* refers to the characteristics for enhancing reliability that are present in the model. *Result formation* can be promoted in two ways: by decomposition of tasks into subtasks, and by synthesis from agents' information.

2.3.2 Agent conceptual model

Structure refers to the agents architecture in terms of its components with a definition of its objectives and the relationships that it holds to form an agent. One needs to know whether this structure is dynamic, and whether it can vary from one agent to another. Knowledge organization refers to the representation of the knowledge, the ownership of the knowledge (whether it is managed by the agent), and the capability for static and dynamic knowledge acquisition. *Action* refers to the acting capacities of the agents, the types of actions they can perform (communication abilities). *Perception* of the environment, refers to how (if at all) the agent senses and models other agents and its environment in general.

3 DAI models

To analyze work in DAI the framework defined in the previous section has been used. The systems analyzed are: contract net, multi-agent computing environment, distributed vehicle monitoring testbed, and an architecture for control and communication. In each of these case studies there is a general description of the system, with the conceptual equivalences of the author's terminology and the concepts defined by the framework. Secondly, the conceptual model of the system is reviewed, thirdly the conceptual model of the agent is reviewed, and finally a general discussion is given on the particularities of the systems, and further references are given. Not all the dimensions in the framework can be applied to all the models analyzed. It is assumed that the reader has knowledge of the concepts presented in the framework (section 2.3).

3.1 Contract net

The contract net is defined by Smith (1978) as a collection of problem solvers (nodes) that we identify with the concept of agent in the framework (section 2.3). Contract net is a collection of agents that cooperate through negotiation to perform a given task. The negotiation is governed by a protocol: the contract net protocol, through which an agent broadcasts the availability of a task to other agents, examines "bids", and then awards the task to the winning bid.

An agent allotted part of a task distributes an announcement through the network to other agents

describing subparts (subtasks) of the overall problem, these agents if interested reply with a bid, indicating their capabilities for the tasks. The original agent evaluates all the "bids" and awards the task to one of the bidders.

3.1.1 System conceptual model of the contract net

Structure. The system structure is a collection of agents. There is no other relation between agents but through the role played in the execution of an individual task: either the manager or the contractor. The manger is responsible for monitoring the execution of a task and processing the results of the execution. The contractor is responsible for the actual execution of the task. Agents are not designated *a priori* as one or the other, as these are only roles and any agent can dynamically take them. During the course of problem solving, a particular agent takes on both roles (perhaps even simultaneously for different contracts).

Knowledge organization. Knowledge is divided into domain specific knowledge sources each of which contains expertise in a particular domain. Static knowledge is assigned at initialization of the system. It is composed of the KSs assigned to the agents. With regard to the dynamic distribution of knowledge, it can be affected in three ways:

(1) an agent can directly request the transfer of the required knowledge,
(2) an agent can broadcast a task announcement in which the task is a transfer of knowledge, and
(3) an agent can note in its bids for a task that it requires particular knowledge in order to execute the task.

Dynamic distribution enables effective use of available computational resources, it also facilitates the addition of new agents as they acquire the procedures and data necessary for their participation.

Coherence and cooperation. Cooperation is given through negotiation and it is governed by a high level communication protocol called the "contract net protocol"; this governs all interaction between agents. The contract net protocol is a problem solving protocol to assign problem solving interpretations to the bit streams—which is to control the process used to effect communication. Cooperation in the control net is based on the dynamic decomposition and distribution of subproblems. To allow the distribution of subproblems to take place there must be a way to find idle experts capable of performing the tasks, and the contract net supplies a mechanism to solve this problem.

Communication. Quoting from Davis and Smith (1983) "Message is the heart of the issue . . ."; messages indicate what kind(s) of things agents should say to one another. Each message is composed of a number of slots that specify the kind of information needed in the message. There are several defined message types, the most important are: task announcement, task bidding, task award, and task termination. Messages are commonly communicated by broadcasting. Focus addressing is a more direct communication used where the generality of a broadcast is not required. The Common Internode Language forms a common basis for communication amongst all agents in the system. This language is used to specify the information in the slots of a message.

System reliability. Reliability is enhanced by the distribution of control and data, together with shared responsibility for tasks (by managers and contractors). The failure of a contractor can be recovered by reannouncing the appropriate contract. The model does not consider any synchronization mechanism for resources because the KSs belong to an agent and other agents do not have access to them except through the communication protocol.

Result formation. Result formation is controlled by the agent acting as a manager. Each manager is responsible for the results of its contracted tasks. The solution is reached through decomposition of tasks and by task sharing of the agents. Each agent is responsible for solving the tasks it bid for.

3.1.2 Agents conceptual model

The contract net does not explicitly define the agent's model, but it assumes an agent has the

capabilities described in the following sections. A specific agent model is left for specific implementation. As an example, the implementation of the contract net agent using MACE (section 3.2) takes an organization of four MACE agents: a communication manager, a contract manager, a task processor and a knowledge base.

Structure. It has a communication component, a processing component, knowledge sources and knowledge of its own KSs and capabilities.

Knowledge organization. It can have knowledge sources in any knowledge formalism. It should support dynamic allocation of knowledge.

Action. It is able to communicate (to manage the contract net protocol), it must be able, if necessary, to require the information that it needs to perform a task, and must be able to select the agents that will execute the task it announces.

Perception of the environment. It can evaluate a task announcement message and decide whether a task is of sufficient interest for it. It can also broadcast bids to the network.

3.1.3 General discussion

The implementation of a contract net has two phases: initialization of the net and operation. Initiating the contract net framework to a specific domain is a considerable problem. The protocol provides a site for embedding a particular type of information (e.g. eligibility, specification, etc.), but does not specify exactly what this information is for any specific domain. In this sense the contract net framework supplies a problem solving structure.

A contract net is useful when it is not known in advance which expert to call, and it seems most useful when careful selection of KS is important. This framework seems well matched to problems that can be viewed in terms of a hierarchy of tasks, or levels of data abstraction.

The contract net protocol offers a way of distributing control, sharing responsibility for tasks, and offers a degree of reliability, but there are many issues unsolved in its implementation. If the central agent fails the entire system fails; there are no mechanisms for reassignment of tasks in the event of failure; problem decomposition (the network assumes an already decomposed problem) is not resolved; and subproblem interaction is not treated.

References. Smith (1978) produced one of the first publications, with emphasis given to knowledge organization and there is a general discussion of the protocol. Smith and Davis (1981) discuss problem resolution by task sharing (used by the contract net) and the alternative (possibly complementary) approach, result sharing. Davis and Smith (1983) present a more detailed classification of the contract net protocol of negotiation and present an implementation of the model in the context of a distributed sensing system. Van Dyke Paprvnak (1987) presents an implementation for manufacturing control.

3.2 MACE—multi agent computing environment

MACE is an instrumented testbed for building a wide range of DAI systems with different levels of granularity. It provides optional facilities for knowledge representation and reasoning capabilities. The MACE model is a collection of agents, each agent belongs to a class and is uniquely identified. All agents are assigned to processors. Agents can communicate with each other and have knowledge about the model of some other agents in their environment (its' acquaintances). Agents can be clustered into sub-units or coalitions (called organizations) which act in response to particular problems. The agent concept of the framework is identified with an executable MACE agent.

3.2.1 System conceptual model

Structure. Each agent is a self-contained object. Agents may be related through roles played with other agents in any organization. Any two agents that belong to an organization can be related through the following roles:

- org-member; one agent is a member of the organization which the other agent manages;
- my-org-manager; one agent is the top-level manager of the organization of which the other agent is a part;
- co-worker; one agent is a member of the organization of which the other agent is part.

An organization is a structure of expectations and commitments to behaviour; it exists only indirectly through the commitments and expectations of its members. The term organization refers to an abstraction which allows agents to treat a collection of activities as being part of a known concerted effort or a known role. A role here is a set of expectations. An organization is an abstract object about which agents can reason. MACE represents an organization as an object shell and a communication agent called manager. The communication manager is called after the organization and its primary function is to disseminate messages to the agents.

Each member of the organization is represented in the world model with an organization member role entry. Thus the organization abstraction is really the set of all acquaintances who are organization-members. The manager engine maps most incoming messages to the addresses of the agents which handle them. The basic work of the manager then, is allocation of work arriving from outside the organization.

Knowledge organization. The knowledge of the system resides in the agent's attributes. Each agent is provided with certain knowledge but it can dynamically improve it through communication via messages. In special cases an agent can inherit the skills of another agent.

Coherence and cooperation. MACE is an instrumented testbed for building different DAI systems at different levels of granularity. Many different cooperative strategies can be implemented. These depend on the particular application to be developed.

Communication. Agents communicate by sending messages. Messages can be addressed to individual agents, organizations, or all the agents of a particular class. Every agent has a unique address, known to itself and the agents who know about him (see acquaintance attribute in the agent conceptual model).

System reliability. There are no particular enhancements, but there are tools to easily implement different strategies to increase reliability.

Result formation. This is achieved through the abstraction "organization" which allows agents to treat a collection of activities as being part of a known concerted effort. The representation of an organization is by an abstract shell and a communication agent called manager; its basic work is task allocation.

3.2.2 Agent conceptual model

Structure. The structure of an agent is defined by the attributes defined in its class. There are some attributes present in any agent. The following four attributes are a sample of the common ones defined in an agent:

1 The *incoming-messages attribute*, which contains all the messages received by the agents.
2 The *status attribute*, which reflects the current status (running, inactive, waiting, stopped) of the agent.
3 The agent's *acquaintances attribute*, which contains the agent's model of itself and the other agents it knows about.
4 The agent's *processor attribute*, which defines the processor on which the agent will be loaded.

MACE agents take actions when the kernel evaluates a function called the engine of the agent. An agent's engine is the only executable part of the agent. The agent and its associated attributes (e.g. a set of production rules which it references) provide the procedural knowledge incorporated in an agent.

Knowledge organization. The knowledge an agent has resides in its "attributes". The attributes are not directly visible to any other agent, but its contents may be sent in messages to other agents.

Attributes provide a way of specializing the local knowledge and may have the full power of an associative deductive database, incorporating pattern directed retrieval and inferences. Any technique for representing knowledge can be used in the attributes.

Action. Agents can take three kinds of action: (1) they can change their internal state by manipulating their attributes, (2) they can communicate through messages to individual agents, to a group of agents, or to all the agents of a particular class, (3) they can send monitoring requests to the system kernel. In particular, some of the actions they can perform are:

- they can create or destroy other agents;
- they can interpret the contents of a message and act in response;
- they can deactivate themselves when they consider it adequate.

To carry out their actions, agents may need to contain locally defined functions. These functions are a type of procedural knowledge which may be called upon by the agent's engine as it runs. They may be sent in messages in order to acquire skills.

Perception of the agent's environment. The model of the agent's world resides in its acquaintances attribute. This attribute contains the explicit model of the agent itself and of the other agents in its world (this world is a partial view of the system, as an agent may not have the models of all the agents in the system). The acquaintances attribute is an associative database that models agents in terms of the following attributes:

> *name*: the name of the modelled agent (this is uniquely identified by the system)
> *class*: the name in a class an agent can be
> *address*: where to find the modelled agent
> *roles*: the relationships the modelled agent bears to the agents contained in the model, or the role it plays
> *skills*: the capabilities of the modelled agent, defined by skill descriptions (skill descriptions are pairs of goals the skill will achieve and alternative methods of achieving them)
> *plans*: these are an agent's view of the way the modelled agent will achieve its goals
> *goals*: this is what the agent that contains the model understands of the modelled agent's goals (what it wants to achieve)

When an agent is created (instantiated) it is given a model of itself, of it's creator (the agent or source responsible for its creation), and its acquaintances. The agent can dynamically alter its concept of the world by adding acquaintances.

3.2.3 General discussion

In the conceptual model description I have avoided implementation details to give better conceptual clarity. The MACE environment provides the following facilities:

- maps agents onto processors;
- handles inter-agent communication;
- language for describing languages;
- tracing and instrumentation;
- facility for remote demons;
- a collection of system agents which construct user agents from descriptions, monitor executions, handle errors, and interface with the user.

Agents can monitor events (events include sending or receiving specific messages), these events are monitored by demons which are mappings from event descriptions to messages. An event can be the effect of some agents visible actions or the result of kernel actions or of state changes in the agent doing the monitoring. In the second case events are monitored by "imps". Imps are predicates on internal attributes such as certain predefined kernel events. Imps are also used for synchronization,

they are evaluated after each execution cycle. The MACE engine shell also provides an imp for "pattern-trigger" which maps a message template to a procedure.

MACE does not consider a dynamic (run time) inheritance of attributes, each MACE agent is a self-contained object which inherits nothing from other agents at run time. Inheritance is important for knowledge representation and there are important research issues which center around how to implement inheritance in a distributed environment. The philosophy underlying MACE implementation views MACE as an attempt to provide the tools for prototyping experimental systems which run in real parallel environments, but not for production quality delivery systems. The emphasis is on experimentation and conceptual clarity, not on efficient execution of MACE agent communities.

MACE has been used to build distributed systems of different granularity: ranging from the large grain of a contract net, the medium grain of a distributed blackboard, to a small grained rule agent. The description language permits the description of agents of varied complexities. An example of how it works is presented in Table 1. It contains a partial example from Gasser et al. (1987a) of one contract net agent built in MACE.

Reasoning about the division of work can be done at a high level, this is because of the high level organizations permitting references to a group of agents with shared goals as reference to an individual.

New agents can be created by example (of existing agents) through the description language rather than by coding each one explicitly. MACE does not provide for the dynamic transfer of agents from one processor to another.

Table 1 Partial agent definition in MACE.

```
; A node manager of a contract net agent (organization with 3 members).
( ( name CNODE-1)
( import ENGINE from CNET-DEF)
( acquaintances
( CNODE-1
            [roles (SELF CNODE-MANAGER)]
            [goals(TASK-ANNOUNCEMENT 4)
                  (ANNOUNCE-TASK $)
                  (ANNOUNCED-AWARD $)
                  (DIRECTED-AWARD $)
                  (ACCEPTANCE $)
                  (REFUSAL $)
                  (FINAL-REPORT $)
                  (INFORMATION $) . . .)]
            [skills([DIRECTED-AWARD $spec)
                  (use (send-to-agent
                  (roles CONTRACT-MANAGER ORG-MANAGER)
                  '(DIRECTED-AWARD $spec))))]
                  [(ANNOUNCE-TASK $task-spec)
                  (ref (acquaintance with
                  [roles (CNODE-MANAGER POTENTIAL-NQ-CONTRACTOR)]
                  [goals (TASK-ANNOUNCEMENT $))])
                  (by sending '(TASK-ANNOUNCEMENT $task-spec)))]. . .)]. . .)
(CNODE-2
            [roles (CNODE-MANAGER POTENTIAL-NQ-CONTRACTOR)]
            [goals (. . .
            . . .
(CNODE-3
            . . .
```

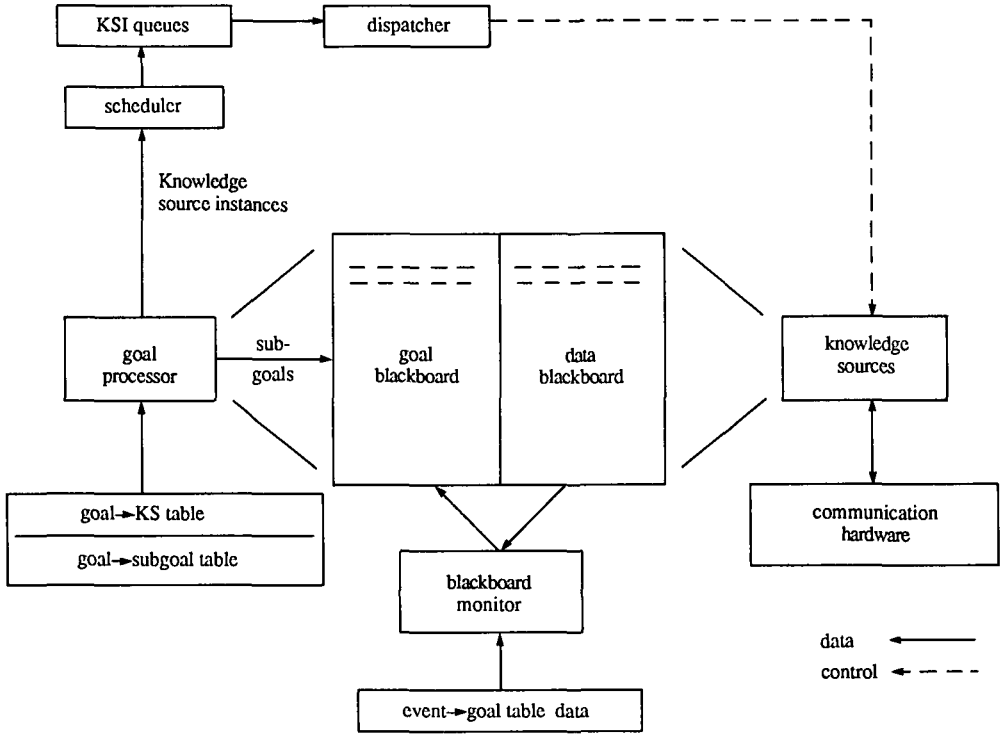

Figure 2 DVMT agent model. A KS forms hypothesis on the data blackboard which in turn trigger the formations of goals on the goal blackboard. The goal processor forms Knowledge Source Instantiations (KSIs) to satisfy the goals, and the KSIs are scheduled. The dispatcher then invokes the KS for the best pending KSI and the cycle repeats.

References. A detailed description of this system appears in Gasser et al. (1987a). An implementation of a production system is reported in Gasser and Tenorio (1986). An implementation of a blackboard system is reported in Gasser et al. (1987b).

3.3 Distributed vehicle monitoring testbed-DVMT

The DVMT is a testbed for distributed problem solving research in a distributed sensor net domain. The DVMT model is a collection of complex problem-solving agents each resembling a Hearsay-II blackboard architecture system (Erman et al., 1980), which cooperate to interpret signals from a sensor array. The agents communicate by sending messages to one another. The typical configuration is presented in Fig. 2. The system framework concept is identified with the network concept of the DVMT, the agent concept of the framework is identified with the node concept of the DVMT.

3.3.1 System conceptual model

Structure. The organizational structure of the system (network in DVMT terms) is obtained by providing each agent (node in DVMT terms) with a high level view (metalevel control) of problem solving in the system. This information resides in the "interest areas" component of the agent model. It specifies a general set of agent's responsibilities and agent interaction patterns. The sophisticated local control component of each agent is responsible for elaborating these relationships into precise activities to be performed by the agent.

Knowledge organization. Knowledge is deposited in the multilevel KSs of the agents. Agents are experts in a particular domain.

Coherent cooperation. Cooperation has evolved since the original DVMT. Figure 3 presents a more

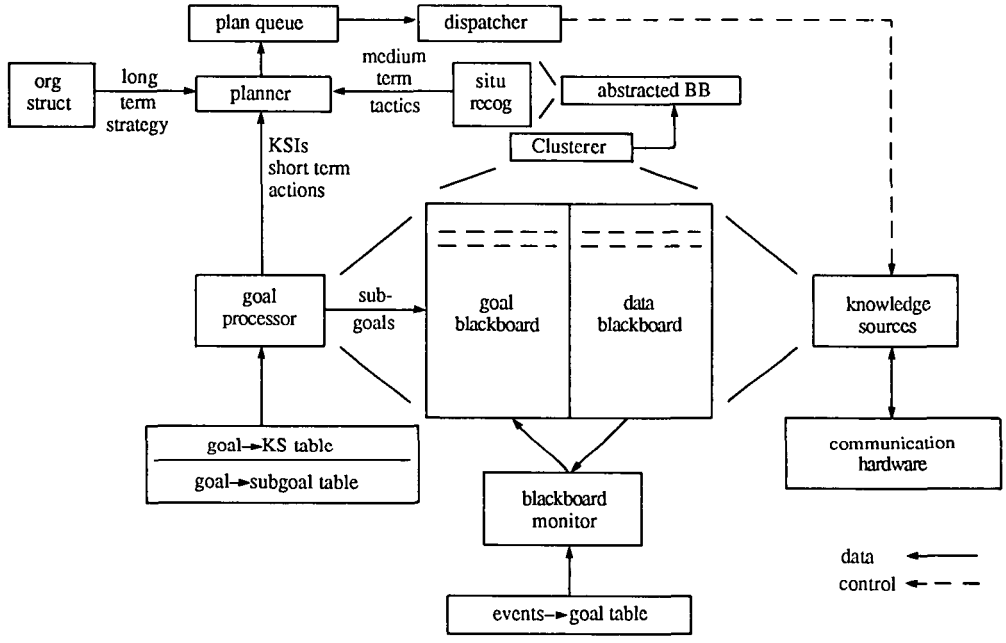

Figure 3 DVMT extended agent models. The planner integrates information about the KSIs, problem solving situations and organizational structure, to form plans. The dispatcher invokes the KS for the best KSI of the best plan. The cycle then repeats.

recent architecture of this system. Interest areas and authority specifications provide the interface between the activity decisions made by an agent, and organizational decisions. They can be used to control the amount of overlap and redundancy amongst agents, the problem solving roles of agents (such as "integrator", "specialist", etc.), the authority relationships between agents and the potential problem solving paths in the network. An agent's organizational responsibilities can be established and changed by simply modifying these data structures. The specification data structures themselves do not provide an explicit, high level representation of these organizational roles and responsibilities, but instead serve as a low level "job description" of those activities an agent should be performing and those activities an agent should be avoiding.

DVMT (using a language called EFIGE (Pattison et al., 1987) incorporates a way of describing the organizational communications structure amongst agents and the problem solving roles each agent will play. It provides a high level representation of the data structures contained in the "interest areas".

Agents have the capability to generate high-level goals and plans. System coherence may be improved allowing agents to communicate meta-level information based on their high level views and plans.

There are three additional components relevant to the coordination of the system:

1 A distributed task allocation component for deciding what dynamic information and processing goals should be transmitted amongst the agents. Given the organizational structure there are still decisions that balance the activities amongst agents based on the dynamics of the current problem solving situation.
2 A knowledge based fault-diagnosis component for detecting and locating inappropriate system behaviour; including inappropriate settings of the problem solving parameters that specify strategic and tactical system coordination.
3 An organizational self-design component for initially developing an organizational structure and for modifying that structure to reduce the effect of errors recognized by the fault-diagnosis component.

It is important to emphasize that the organization and coherence of the system rely on the intelligent

local control of the agents over deciding which local task to pursue in order to best contribute to coherent system performance.

Communication. As we can see from Figs. 2 and 3, DVMT has evolved and several communication schemes have been tried out. Communication is treated in particular in Durfee et al. (1987). Here a mechanism that allows agents to make intelligent use of communication resources is described. Simple communication policies are introduced into agents and communication decisions are improved. Most communication policies relate to goal states i.e. completeness of solution and timeliness of solution.

System reliability. N9 particular improvement.

Result formation. Results are formed by synthesis. DVMT obtains information which is interpreted by the agents; the system then interprets information from the agents for the global system result.

3.3.2 Agent conceptual model

Structure. The DVMT agent structure resembles a Hearsay-II blackboard system extended to include more sophisticated local control knowledge sources (KS) for communicating hypotheses and goals amongst agents and data structures called interest areas that specify the organizational role of a node. The agent has been provided with mechanisms to enable it to understand its current and likely future problem solving actions.

To understand the actual DVMT agent we will provide a historic review of its development and enhancement. The original Hearsay-II architecture (Erman et al., 1980) is an architecture that models multilevel cooperating knowledge sources. It was used in a speech understanding system. As systems with similar architectures were applied to different task domains, the rudimentary data-directed and instantaneous scheduling mechanisms provided were enhanced with sophisticated control capabilities (Nii and Feigenbaum, 1978; Erman et al., 1981). Corkill et al. (1982) in particular looked at the problem of extending the control component of Hearsay-II to develop and reason about the relationships between competing and cooperating KS activities (both past and potential) and amongst KS activities working on different aspects and levels of the problem. Corkill and Lesser (1983) developed an architecture that integrates data and goal-oriented control of KS activity via the generation of goals from blackboard events.

Further extensions to the architecture of goal-directed control were needed to provide an agent with the perception of the overall system organization in order to increase the coherence of the system. The approach taken (Corkill and Lesser, 1983) relies on each agent making sophisticated local decisions that permit it to plan sequences of activities and to adapt its plan based on its solving role in the system, on the status and roles of other agents in the network and self-awareness of its activities.

Yet further extensions were developed to allow an agent to refine its perception of the role it currently plays in the organization (Durfee et al., 1985, 1987). The agent was provided with the capability to generate and reason with a more complete view of its past, present and future activities. An abstracted blackboard will group suggested hypotheses and goals into a single structure that the report can use to formulate high level goals and to generate plans with which to achieve them.

Since each plan incorporates a sequence of actions, the pursuit of a specific plan allows the node to make reliable predictions about its actions in the near future. These predictions enable the agent to make medium-term dynamic refinements to how it views its role in the network organization and it may modify its local processing accordingly. If agents occasionally exchange meta-level information about the refined views of the organization roles they will be playing in the near future, each node will have a more global view of the system problem solving activity.

The following description of the latest agent architecture of the DVMT is presented in Durfee et al. (1987). It is included here for completeness.

> "Each problem solving node has a Hearsay-II, blackboard architecture, with knowledge sources and blackboard levels of abstraction appropriate for vehicle monitoring. A

knowledge source (KS) performs the basic problem solving tasks of extending and refining hypotheses (partial solutions), where each hypotheses tentatively indicates when a certain type of vehicle was at one or more discrete sensed times. The basic Hearsay-II architecture has been augmented to include more sophisticated local control and the capability of communicating hypotheses and goals amongst nodes. In particular a goal blackboard, a goal processing module, and a communication knowledge source have been added"

Knowledge organization. A knowledge source (KS) performs the basic problem solving task of extending and refining hypotheses (partial solutions). There are four blackboard levels (signal, group, vehicle and pattern), each of these blackboard levels is split into a blackboard level of location hypothesis and another for track hypothesis. In total, agents have eight blackboard levels with appropriate KSs for combining hypotheses at one level to generate more encompassing hypotheses on the same or on a higher level.

A knowledge source instantiation (KSI) represents the potential application of a particular KS to a specific hypothesis. Each agent maintains a queue of pending KSIs and reasons about the intentions or goals of the KSIs. Explicit representations of the agent's intentions to abstract and extend hypotheses and goals are stored in a separated goal blackboard and are given importance ratings. The scheduler ranks a KSI based both on estimated belief in the hypothesis and on the rating of the goals it is expected to satisfy.

Action. A DVMT agent can perform the following actions:

- it can communicate to other agents hypotheses, goals, and meta-level information;
- it can generate high level goals and plans adapting them to conform to the overall problem solving strategy;
- it can predict its actions in the near future.

Perception of the environment. The agent perceives its environment through the meta-level information of the system and the information gathered through communication. Meta-level information can be of two kinds (a) information about high level plans and goals of other agents; (b) information about the organizational structure of the system and the overall problem solving strategy.

Meta-level control via organizational structuring is introduced to the agent architecture through the use of a non-procedural and dynamically variable specification of the behaviour of each agent's planner, its scheduler, and its communication knowledge sources. These data structures are called interest areas. There are five sets of interest areas for each agent in the testbed.

3.3.3 General discussion

The development and enhancement of agents shows increasing concern about how to achieve global coherence. The original DVMT agents (1983) were sophisticated with effective local control, but lacked good perception of their environment. Although self-interested and totally independent, agents might cooperate unwittingly. A more reliable cooperation strategy was needed to achieve global coherence. The subsequent changes in the agent structure can be seen as refinements of the intention to provide an agent with a better perception of the system environment and of its own role in the system.

At IJCAI87 Durfee and Lesser have further extended the conceptual model presented previously by allowing it to support different styles of cooperation by using partial global plans (PGPs) with which to specify effective, coordinated actions for groups of agents. In this extended model, agents summarize information such as the plan's objective, the order of major plan steps, how long its extended tolerance is and details of actions (KSs) that have been taken or will be taken into the agent's plans, in their local plans, avoiding details about short-term planned action.

The agents selectively exchange their node plans to identify partial global goals (PGGs) by comparing local goals to determine whether they are part of a larger goal. For each PGG the

planner forms a partial global plan (PGP) that represents the concurrent activities and interactions of all the agents that are working in parallel on different parts of the same problem. The different styles of cooperation depend on how agents form, exchange, manipulate, and react to PGPs. For example in the multiagent planning style, where agents send all of their information to coordinating agents that generate and distribute multiagent plans, the PGP is the multiagent plan. In the contracting style (cooperation by negotiation) each contract is a PGP involving a pair of agents.

Corkill et al. (1987) are working with communication policies to optimize the knowledge that actual DVMT agents have of their past, present and future activities in order to achieve further coherence.

References. In the first publication, (Lesser and Corkill, 1983) the Hearsay-II system is decomposed to test distributed problem solving techniques. The issues in this distributed environment considered by Lesser relate to information on the blackboard, processing capabilities of the KSs and control. The second publication (Lesser, 1980) attempts to provide a methodology to measure the results of incremental opportunistic policy in distributed problem solving. The third publication (Corkill, 1982) contemplates the inclusion of goal-directed and data-directed blackboards and planning capabilities. In the fourth paper Corkill (1983) proposes the use of DVMT as a general tool to investigate distributed problem solving. Durfee (1987) reports research in communication policies intended to achieve global coherence in self-interested agents. Pattison et al. (1987) presents EFIGE, the language used to describe organizations. Durfee and Lesser (1987b) present the concept of partial global plans to support different styles of cooperation.

3.4 An architecture for control and communication in DAI systems

The Architecture for Control and Communication (ACC) is a general architecture for distributed artificial intelligence systems with particular focus on control and communication. ACC is dynamically self-organized and can improve its performance using the information obtained from the problem solving process (see Fig. 4). ACC is a collection of agents that can communicate and dynamically refine their perception of the capabilities of other agents. The ACC processing node is identified with the agent concept of the framework (section 2.3).

3.4.1 System conceptual model

Structure. The system is a collection of agents, each of them expert in a particular area. They are not related to each other except for requesting cooperation to solve a task.

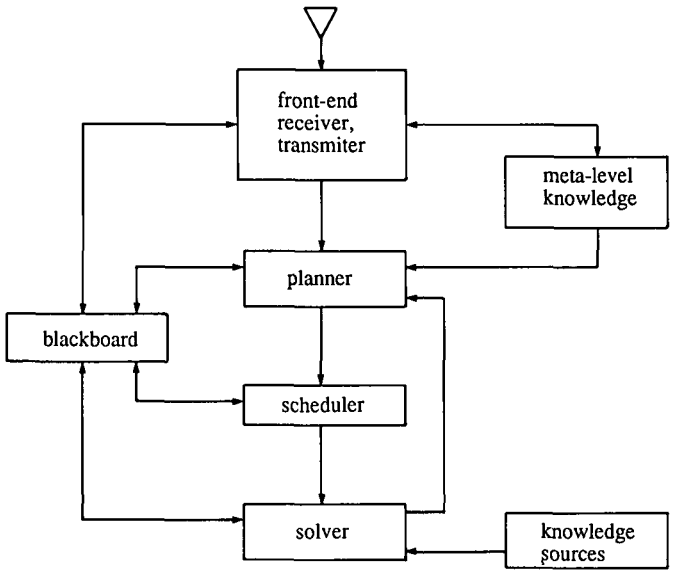

Figure 4 ACC: Architecture for Control and Communication.

Knowledge organization. There are three knowledge-based systems in the ACC:

1 A problem-dependent collection of knowledge sources.
2 A communication reasoning system in the form of meta-knowledge (meta-knowledge is defined as the knowledge about its own capabilities and other agents capabilities), about problem decomposition and result synthesis.
3 A mechanism for the accumulation of meta-knowledge.

Coherent cooperation. The coherence of the system is maintained by the goal-directed problem-decomposition attitude of the agent. Agents cooperate with each other through communication. An agent accepts a message if it is specifically addressed to it or if it is a system-wide broadcasted transmission. If the accepted message is a response (an answer type message) the associated data is directed to the corresponding task (i.e. placed into the corresponding frame). This data is also used to update the meta-knowledge that an agent has of the capabilities of other agents. In this way the system improves its communication and planning performance. If the data is no longer requested by any unresolved task it is discarded. If the message is a request (question type message) the agent decides (using its meta-knowledge component) whether the question is within its expertise areas. If it is not, the message is discarded without notifying the agent who originated it. But if the request falls within the areas of the agent's expertise, a task is generated to address the question.

When there are tasks whose solution need initial planning, the agent generates a problem-solving plan using its task planner component. If the task can be solved in more than one method, the focus of control mechanism makes a decision based on the current status of the system, the meta-knowledge accumulated and the expertise derived from past problem solving efforts. The subtasks belonging to the most promising path are called the current solution plan. This path exploits all parallel subtasks involved in the current problem solving. For unsolved subtasks that cannot be solved by the agent in the current solution plan the agent determines (through its front-end processor) its knowledge about other agents' capabilities and requests the task to be done.

Communication. The inter-agent communication language is problem dependent. The ACC provides a framework but the actual definition is left for a particular implementation. In the protocol of communication of the framework there are two types of messages communicated amongst agents: a question type and an answer type. In general, all requests are expressed as question type messages and all responses are expressed as answer type messages. An agent can directly communicate a message to other agents or use the system-wide broadcast transmission.

System reliability. Errors are reduced by a search process of deducible subplans and through the iterative behaviour of the problem solving.

Result formation. This is mainly achieved by the communication reasoning system in the form of metaknowledge about problem decomposition and result synthesis. A blackboard located in each agent is organized in a task-oriented frame-like structure. Each frame contains information about tasks, results, result synthesis, confidence values of results and a problem solving plan.

3.4.2 Agent conceptual model

Structure. All agents have an identical organization, formed from six components.

Meta-knowledge: contains the information about the solving capabilities of the other agents and itself.

Blackboard: is an active structure that allows information sharing amongst the agents by storing tasks, plans, and partial results, transmitting them at the appropriate time in the problem solving process within the agent.

Front-End Processor: enables the agent to share tasks and tentative results. It determines the transmission mechanism and agent destination of the agents' messages, according to the system protocol of cooperation.

Task Planner: is responsible for applying a suitable problem-decomposition mechanism and generating a problem solving plan. It provides all the paths of subtasks that can be followed to solve

the problem. All of these possible paths are evaluated and one of them is chosen as the current solution path.

Scheduler: manages the resources of the agent. It decides which tasks can access the resources and with what priority.

Solver: is in charge of activating the appropriate knowledge sources to solve the tasks paths (plan).

Knowledge Organization. The agent has knowledge of the kinds of problems it is interested in solving, as well as a corresponding problem decomposition and result-synthesis mechanism which is provided by its expert system.

Actions. An agent is capable of communicating with other agents through its blackboard components. An agent can engage itself in more than one problem solving plan. The blackboard component allows information sharing about them. Agents can share tasks and tentative results through the front-end processor component. The agents are capable of solving problems that fall within their expertise area. Their focus control mechanism selects the most promising solving plan and guarantees system coherence.

Perception of the environment. An agent can estimate the knowledge and capabilities of other agents as well as its own. This knowledge indicates how well an agent can solve a particular type of problem. This perception of itself and of other agents is concentrated in the meta-knowledge component of the agent architecture.

The agent dynamically refines its meta-knowledge as information is exchanged during the problem solving process. The meta-knowledge about other agents is organized by problem decomposition abilities—for each problem the following information is registered:

Problem: a brief description of the problem;
RS-Process: a result synthesis process or function;
Decomp: a decomposition mechanism;
Candidates: a list of processing nodes which might solve the problem (if it is not solvable by this agent);
Confidence: an actual or estimated measure of the quality of the solution.

3.4.3 General discussion

This architecture focuses on the control and communication aspects. Problem solving occurs as an iterative refinement of several mechanisms, including problem decomposition, kernel-subproblem solving and result synthesis, The system dynamically reconfigures itself, increasing its performance during operation. A major component of the agent is a database of meta-knowledge about the agents own expertise and those of other agents—this is dynamically updated. This architecture is suitable for DAI where the knowledge sources are distributed among the agents, and where the input data and the design constraints are available to all agents (as in VLSI design). Reductions, both in the size of the explored solution space at the task level and in the required communication bandwidths are achieved by a focus-control mechanism.

References. In Huhns et al. (1983) a methodology for communication between multiple agents by use of meta-level control primitives is presented. Ju-Yuan et al. (1985) presents a detailed architecture of the ACC (which is the main reference for this description).

4 Formalized approaches of multi-agent interaction

As the field of DAI has matured there has been an increasing emphasis on the need for research into the formalism necessary for one agent to reason about another agent's knowledge and beliefs, and for modelling multi-agent interaction.

Most of the work done in action and planning representation in AI assumes what Pednault (1987) calls the "classical framework". The classical framework is a state-based approach that sees

actions as relationships over states and supposes that the world does not change but for the effect of an action.

There are well-known difficulties for adapting the classical framework to parallel domains and there has been much recent interest in devising formalisms and specification techniques more suitable for describing parallel domains (typical of multi-agent interaction). This research has three main tendencies: the event-oriented (action-oriented) model (Lansky, 1987a, 1987b) where the world is represented in terms of a set of events; the state-interval model (Allen, 1984; McDermott, 1982, 1985), where actions are modelled as state sequences occurring over an interval of time, and the work on extending the classical approach so as to overcome its difficulties in expressing parallelism and dynamic worlds (Pednault, 1987).

Another important area of research in automatic plans has been the examination of the problem of synchronizing plans executed by different agents to avoid undesirable interactions between plans. Georgeff (1983) has made use of operating systems techniques to identify and protect critical regions in multiplans. In more recent work Stuart (1985, 1987) uses propositional temporal logic to synchronize plans. Rosenschein (1982) works with the synchronization of multi-agent plans where an intelligent agent constructs a plan to be executed by several other agents.

Most of the planning strategies assume the willingness of the agents to cooperate. At Stanford University Genesereth, Ginsberg and Rosenschein use game-theoretic and decision-theoretic approaches to model "the conflicting agent metaphor" (Genesereth et al. (1984a, 1984b)).

There are two major tendencies underlying the existing formal models of belief: the possible worlds approach (Hintikka, 1962, 1975); Appelt, 1980a, 1980b, 1981; Levesque, 1984) and the syntactic approach (Konolige, 1980).

There is work not directly connected with intelligent behaviour, on the formalisms of designing and specifying distributed concurrent systems. This work (Hoare, 1985; Milner, 1980) provides a mathematical foundation on which to base a framework for distributed intelligence. An example of this application can be found in (Georgeff, 1983) where Hoare's Theory of Communicating Sequential Processes is used for handling synchronization among agents. Formalisms more related to DAI are presented in Hewitt (1977), Hewitt and de Jong, (1983), and Halpern and Moses (1984).

In this section I will review the most relevant work mentioned, organizing it into the following areas: (1) action and planning; (2) the conflicting agent metaphor; (3) belief; (4) theories of communicating distributed processes.

4.1 Formalization of action and planning

4.1.1 On planning

As Pednault points out (Pednault, 1987), work in automatic planning has been concerned with ways of representing and solving what might be called the classical planning problem. In this type of problem, the world is regarded as being in one of a potentially infinite number of states. Performing an action causes the world to pass from one state to the next. The specification of a problem contains a set of goals, a set of allowable actions and a description of the world's initial state. The objective is then to find a sequence of actions that will transform the world from any state satisfying the initial state description to one that satisfies the goal description.

There are well-known difficulties with adapting this kind of framework to parallel domains (e.g. see Allen, 1981; McDermott, 1982; Georgeff, 1983; Pednault, 1987; Lansky, 1987a, 1987b), the foremost being caused by a tendency to require that interacting actions (those whose pre- and post-conditions interact) occur in isolation; they cannot occur simultaneously. Such an assumption is unsuitable for parallel domains, where simultaneity may even be a desired property. For example, one might wish to require that two events occur simultaneously. Similarly, one might wish to state explicitly the consequences of executing several events simultaneously. Another difficulty is linked to the specification of continuous processes (Pednault, 1987). In the classical planning framework the world does not change between the end of one action and the beginning of the next. This permits

a state to represent everything that is true of the world at a given point in time between one action and the next. In a dynamic simulation however, this is not sufficient, since the world may be continually changing of its own accord from one moment to the next. Hence what may be true at one point in time may not be true an instant later.

Yet another problem (Georgeff, 1983) that the classical planning framework presents is in the representation of the resources used during the performance of an action. This sort of information is critical in allocating resource usage and preventing conflicts.

In recent work Pednault (1987) presents an approach to formulating multi-agent, dynamic world planning problems as single-agent, static world problems and discusses ways in which to apply traditional models to these problems.

There has been much interest recently in devising formalism and specification techniques better suited to describing parallel domains. Lansky (1985, 1987a, 1987b) proposes an event-oriented (Lansky uses event and action as synonyms) model. In this approach the world is represented in terms of a set of interrelated events. The state of the world at any particular point in time is represented in terms of the set of events that have occurred up to that moment.

Allen and McDermott model events (actions) and/or predicates as state sequences occurring over an interval of time. Domain properties are described in terms of interval-ordering constants. This form of representation has the benefit of allowing the state of the world during an event to be modelled explicitly and be reasoned about. Event intervals can be temporally related in all possible ways. Most interval based models also explicitly utilize relationships between events. McDermott (1982), considers an action or event to be a set of sequences of states and describes a temporal logic for reasoning about such actions or events. Allen (1981) also considers an action to be a set of sequences of states and specifies an action by giving the relationships among the intervals over which the action's conditions and effects are assumed to hold. The notion of action in Georgeff's work (1983, 1985) is essentially the same as that of McDermott and Allen; he considers actions to be sets of sequences of world states. He puts emphasis on the need to consider various "mental entities" such as beliefs, goals, and intentions (Georgeff, 1987) and on the need to consider the resources used in order to synchronize activities (Georgeff, 1983).

Pednault. Pednault (1987) uses formulae of first order logic to express facts about states. He developed a new language called ADL (Action Description Language) to describe actions and their effects. The syntactic properties of ADL enable the frame problem of the situation calculus to be circumvented—at least to the extent that one only has to specify the changes that take place when an action is performed and not what remains the same.

ADL imposes certain constraints on the kinds of actions that can be described. The first constraint is that the precondition of the action must be representable as a well-formed formula. The second constraint is concerned with actions that create or destroy objects in the world. The effect of creating or destroying objects must be simulated by a mechanism that would require the domain of a state to include all objects that could exist. The third constraint is that actions must preserve the interpretations of free variables used as parameters in a plan, and consequently, a free variable that appears at more than one point in a plan should represent the same object at each of those points. The fourth constraint is that actions must be deterministic.

Pednault uses this formalism to show how some multi-agent, dynamic world problems may be formulated in the classical planning framework. The two main obstacles to formulating such problems in this way are the difficulties of representing simultaneous actions and representing continuous processes.

Continuous processes can be represented in the classical planning framework by interpreting

(1) a state as a chronicle of all that is true, was true and will be true of the world at every point in time;
(2) a state transition as the initiation of an action or course of action.

To model simultaneous actions as sequential ones, Pednault introduced the idea of a boundary condition. A boundary condition exists at a point in time and determines the future course of events

up to the next boundary condition. The effect of simultaneous actions is then modelled by modifying boundary conditions sequentially. The plan is solved and the resulting sequential plan is transformed into a parallel plan for coordinating their simultaneous execution.

Pednault supports the conjecture that it may be possible to transfer techniques for solving classical planning problems to frameworks that are better suited for the modelling of simultaneous actions and dynamic processes.

Lansky. Lansky works with a specification language designed for the description of domains with intrinsic parallelism. The specification language is based on the GEM (Group Element Model) concurrency model. GEM is an event-based model that focuses on domain events (both atomic and non-monotonic) and on the causal and temporal relationships among them. GEM semantics are based on first-order temporal logic. By the use of first-order temporal logic, constraints on history sequences, complex synchronization properties based on causality, temporal ordering and simultaneity can be expressed easily and naturally.

Based on the space model Lansky develops a planning architecture called GEMPLAN (Lansky, 1987b) to construct multi-agent plans. Quoting from Lansky and Fogelson (1987c),

> "GEM's underlying domain model is constructed in terms of events (while they are often considered distinct, we use the terms event, event instance and action interchangeably) that are localized in areas of activity. Event instances may be interrelated by three kinds of relations: the temporal order $\Rightarrow$, the causal relation $\sim>$ (modelling a direct causal relationship between event instances) and a simultaneity relation $\Leftrightarrow$ (modelling the necessary simultaneity of event instances). GEM utilizes two kinds of regions: element and groups. Elements are the most basic type of region. Every event must belong to some element and all events belonging to the same element must be temporally-totally-ordered—i.e. elements represent regions of temporal activity. Elements (and their constituent events) may then be clustered into groups. Each group represents a region of causally encapsulated activity. . . . GEM's domain model can be viewed as a two-tiered structure. The upper level consists of world plans. Every world plan consists of a set of events, elements, and groups and their interrelationships. Each event in a world plan models a unique event or action occurring in the world domain, each relation or ordering relationship models a relation between domain events and each element or group models a logical region of activity (we assume here that all events are atomic). World plans are meant to convey known information about a domain. This information may be incomplete in the sense that some potential relationships are left undetermined. . . . The lower tier of the GEM world model contains the set of potential behaviours or executions permitted by a world plan. These executions must conform to all the relationships established in the world plan—in essence they represent its possible 'completions'"

Given this underlying world model, domains are described by GEM specifications. Each specification consists of a set of constraints that limit the allowed executions or behaviours of that domain. A given world plan W is considered to satisfy a set of domain constraints if every one of its history sequences (i.e. executions) satisfies every constraint in the set. Just as elements and groups model the structural aspect of a domain they also serve as the structural components of the GEM specification language. Each specification is composed of a set of element and group declarations. Each element is associated with a set of event types (the types of events that may occur in the element). Each element group may also be associated with a set of first order linear temporal logic constraints. These constraints are localized, applying only to those events occurring within the element or group in which they are defined. Every specification also includes a set of default constraints imposed by the element/group structure of the domain itself.

All events belonging to the same element must be totally-ordered-temporally. Elements are often used to model limited resources which, by their nature, are constrained to support only one action at a time. Groups are used to represent regions whose boundaries limit inward causal effect. Group

boundaries impose limitations on the scope of causal effect and require simultaneity. Thus the effects of events may range freely unless they are explicitly blocked by a group boundary.

Group structure is a natural way of defining event independence in a general manner. An advantage of GEM's localization of constraints is that if a given region's constraints are known to be satisfied, the introduction of a new event at some other disjoint region can do nothing to violate that constraint.

4.1.2 On action

The formal modelling of action, faces what, in the literature, is called the frame problem. There are several difficult issues associated with the frame problem. The combinatorial problem is the difficulty of indicating all those things that do not change as actions are performed and time passes. The ramification problem is the difficulty of it being unreasonable to explicitly record all those things that do change as actions are performed and time passes. The qualification problem is that the number of preconditions for each action is immense. The overcommitment problem, is the need to assume that all domain actions are known.

One of the first attempts to solve the combinatorial and qualification problems is proposed by McCarthy. Quoting Vladimir Lifschitz (1987)

> "John McCarthy proposed approaching the qualification problem and the combinatorial problem using circumscription (McCarthy, 1980, 1986). With regard to the qualification problem, the idea is, instead of listing all the qualifications in the antecedent of one axiom, to condition the axiom on the fact that (a certain aspect of) the action is not 'abnormal' in the situation in which the action is performed. For each qualifying condition, a separate axiom postulates that the condition implies the abnormality of this aspect to the action. Circumscribing abnormality should ensure that the action has its expected effect whenever we are not in those exceptional situations which, according to the axioms, make the action abnormal. For the frame problem, in the blocks world context, an axiom which guarantees that the location of 'x' in the situation result (a,s) is the same as the location in the situation 's' unless a certain aspect of 'a' is abnormal. Then circumscribing abnormality should allow us to prove that 'x' does not change its position if the action consists of moving a block other than 'x', or in painting a block, etc., so that all 'frame axioms' should become provable."

Further analysis has shown that the result of minimizing abnormality relative to McCarthy's axiom set is not quite as strong as necessary because using circumscription we can only prove the common properties of all minimal models; minimizing abnormality will only give a disjunction.

According to Hanks and McDermott (1986), the problem is that circumscription knows nothing about time: ". . . the class of models we want our logic to select are not the 'minimal models' in the set-inclusion sense of circumscription, but the 'chronologically minimal' (a term due to Yaov Shoham). Those in which normality assumptions are made in chronological order, from earliest to latest, or equivalently, those in which abnormality occurs as late as possible. The paper by Hanks and McDermott was presented at the AAAI conference in 1986 and three more talks given at the same conference—Kautz, Lifschitz, and Shoham, met their challenge by formalizing, in somewhat different contexts the idea of chronological minimization. As could be expected, all of them had to use forms of logical minimization more general than circumscription in the sense of McCarthy.

Lifschitz (1987) argues that a way to identify the intended models from the unintended models of the axioms for actions, is identifying the intended model as the only minimal model whose changes in the values of fluents (A fluent in the Situational Calculus of McCarthy (1986) is a function defined on situations. A situation is the complete state of the universe at an instant of time.) are caused by actions.

In this section I will review the work of Ginsberg and Smith who have suggested a formalism based on the notion of possible worlds that presents an effective method for solving the

combinatorial, ramification (Ginsberg and Smith, 1987a) and the qualification problem (Ginsberg and Smith, 1987b). I will also review the mentioned work of Lifschitz (1987, and Georgeff (1985b).

Ginsberg and Smith. Ginsberg has been working with David Smith on automatic reasoning about actions. They have suggested a formalism based on the notion of possible worlds (Ginsberg, 1986), with which to reason about actions and their effects on the world. They presented a computationally effective method for solving the frame and ramification problems (Ginsberg and Smith, 1987a) and later they extended the method to deal with the qualification problem (Ginsberg and Smith, 1987b). In this later publication they present an inferential approach that makes it possible to determine inferentially which domain facts potentially qualify the action in question. This inferential approach differs from the most common "exhaustive" approach to qualification (Lifschitz, 1987; McCarthy, 1986; Shoham, 1986), that explicitly indicates under what circumstances the action is qualified; if none of these circumstances can be proven to have arisen, the action is assumed to be unqualified.

Ginsberg and Smith (1976) present a comparison of the inferential and the exhaustive approaches to qualification in terms of the computational resources needed by the methods in order to both describe the qualification on an action, and to determine whether or not any particular action is in fact qualified. Their results show that the inferential approach to qualification does not suffer from an exponential deterioration in performance as the domain becomes increasingly complex, whereas the exhaustive approach does.

Lifshitz. Vladimir Lifschitz (1987) applies circumscription to formalizing reasoning about the effect of actions in the framework of the Situation Calculus of McCarthy and Hayes (1986). The axiomatic description of causal connections between actions and changes allows them to solve the qualification and combinatorial problem using simple forms of circumscription whose result can be determined by syntactic methods (Lifschitz, 1987).

An important characteristic of the formulation presented in (Lifschitz, 1987) is that the predicates which he circumscribes are not fluents, they do not have situation arguments and so the conflict between minimizing in earlier or later instants of time (a problem presented in McCarthy, 1986) simply cannot arise.

Georgeff. Georgeff (1985b) highlights the role of "the process" in reasoning about the behaviour of agents in dynamic worlds. A model of events is constructed that provides for simultaneous action. A model-based law of persistence is introduced to describe how events affect the world. No frame axioms or syntactic frame rules are involved in the specification of any given event. An algebra of processes can be employed to ascertain critical properties of multi-agent systems such as freedom from deadlock and how systems of processes can be reasoned about, given specifications of the component processes. Most of the difficulties associated with the frame problem are avoided using the law of persistence together with notions of causality and derived predication.

4.1.3 On plan synchronization

In a case where several agents are acting together, plans must take into account the interaction between agents. An important part of the planning process is conflict resolution (Stuart, 1987), in which plans are reformed to prevent undesirable interaction between component parts. For multiple concurrent agents this often corresponds to synchronizing the plans.

There exist two major paradigms for synchronization of multi-agent plans (Rosenschein, 1982). The first paradigm is in the context of planning for multiple agents, where a single intelligent agent constructs a plan to be carried out by a group of agents and then hands out the pieces of the plan to the relevant individuals. In this context, the planner cannot simply tell each agent what action to perform; explicit mechanisms must exist for maintaining a specific sequence of actions. The second paradigm is distributed problem solving, where a group of intelligent agents together construct and possibly execute the final plan.

Rosenschein examines the problem of achieving synchronicity among a group of agents who will be carrying out a centrally-produced plan. Georgeff (1983) describes a method for synthesizing

multi-agent plans from single agent plans. Stuart (1985, 1987) uses propositional temporal logic to synchronize plans.

Rosenschein. Rosenschein (1986) has developed a formalism to use in the synchronization of multi-agent plans. He first extended Kurt Konolige's formalism for representing knowledge about other agents' beliefs and goals (see Konolige in section 4.3), to allow representation of their capabilities (he assumes that not all agents have identical capabilities). He also presents a theory of planning communication acts. This theory allows selective acceptance of goals and facts, and introduces an explicit means of inducing an agent to perform an act. Finally he presents an ordering mechanism which consists of sequencing operators and a planning heuristic.

Georgeff. Georgeff has developed a formal model to combine separate plans from independent agents. The primary concern is to avoid destructive interference caused by simultaneous access to shared resources. The model used assumes that the agents have separate goals, but that these goals do not directly oppose one another. He models actions as sequences of states and examines only the pairwise combinations of states so as to handle the synchronization and avoid potential conflicts. The type of communication act involved is particularly simple; it provides no information other than synchronization advice. He is not concerned with some of the more problematic issues in multi-agent planning, such as questions of belief or persuasion.

The multi-agent plan synchronizer described in Georgeff (1983) has been used to solve a number of tasks involving both cooperation and interaction avoidance. These problems include two arms working cooperatively to bolt subassemblies together, some typical world problems requiring "non-linear" solutions, and various "readers and writers" problems.

Stuart. Stuart (1985) describes a program that augments plans with synchronizing primitives to ensure appopriate conflict avoidance and cooperation. The theory of action underlying the program is presented in detail in (Stuart, 1987). In this theory, the environment in which actions are executed consists of a world state and a set of actions currently being executed.

Stuart uses propositional temporal logic to synchronize plans. The technique presented in (Stuart, 1987) describes the possible safe executions of a plan using propositional temporal logic and uses a theorem prover to generate a structure corresponding to possible safe executions. This structure guides the plan unification.

The environment defines a set of symbols called operators, and guides each operator on interpretations as an action. These operators are the means of interaction between an agent and the environment and are exchanged as messages. The execution of a plan corresponds to the sequence of messages between the environment and the agent. The interpretation of a plan is given as a non-deterministic finite automaton (agent) which exchanges messages with an environment for the commencement and the conclusion of primitive actions which take place over a period of time.

The problem addressed in (Stuart, 1985) is to ensure that a plan does not deadlock or allow any event to fail. Stuart developed a program which takes a plan and a description of an environment and generates a revised synchronized plan. His synthesized plan allows any and all execution sequences of the original plan which will generate correct interaction. The algorithm used to synchronize plans was originally proposed by Manna and Wolper (1981) and uses propositional temporal logic (PTL) formulae to express constraints on execution sequences of a plan. PTL is a logic for reasoning about sequences of states.

4.2 Conflicting agent metaphor

The research of Rosenschein, Genesereth and Ginsberg has been oriented towards defining a formal framework for rational interaction, i.e. interaction that would occur among rational agents, though they have also reported some work on cooperative agents (Rosenschein and Genesereth, 1984). Their efforts have focused on the "conflicting agent" metaphor; a framework that allows agents to have conflicting goals and permits a study of when these self interested agents cooperate and compromise. They developed a formalism based on game theory involving agents with possible distinct goals and no communication allowed (Genesereth et al., 1984; Genesereth, 1984). This work

treats a case where agents with potentially conflicting goals have to interact and no communication is possible, although there is sufficient sensory information available for the agents to deduce at least partial information about each other's goals and rationality.

This formalism borrowed the payoff matrix construct of game theory to model the interaction of agents. Game theory addresses the issue of what move a rational agent will make, given that other agents are also rational. They remove the *a priori* assumption that other agents will be rational.

The payoff matrix is used to designate the utility to the players of a particular joint move. This is better understood through an example. Consider the following matrix:

	c	d
a	3/1	2
b	2/5	0/1

The movements available to the first player are "a" and "b", and for the second player, "c" and "d". Suppose that in a joint movement the first player picks row "b" and the second player picks column "c". The first player receives a payoff of 2 and the second player, a payoff of 5.

The payoff function for a game (p) captures the information contained in the payoff matrix. It is a function whose value for a given player (say i) and a given joint move (m) is the payoff for player "i" if move "m" is made. In the example above:

$$p(1,(b,c)) = 2 \text{ and } p(2,(b,c)) = 5.$$

Based on this framework, a hierarchy of rationality assumptions was developed and various types of probable behaviour were shown to follow from each level of rationality.

The communication work presented in Genesereth et al. (1984) presupposes a variety of strong assumptions:

1 The agents are assumed to have common knowledge of the interaction matrix, including choices of action and their outcomes.
2 There is no incompleteness in the matrix.
3 The interaction is viewed in isolation (no consideration is given to future interactions).
4 There must be effective simultaneity in the agent's actions (otherwise other considerations have to be taken, like who goes first).

There has been research at Stanford University aiming to remove some of the most restrictive assumptions. There is work on the question of incomplete matrices, so that the conflict analysis can be applied to interactions with incomplete information (Rosenschein, 1986). Future work will focus on issues arising from multiple encounters such as retaliation and future compensation for present loss.

Rosenschein and Genesereth (1985) generalize the work on cooperation without communication. The model presented in this paper allows agents to make binding promises to one another and allows communication among the agents in the interaction. This formalism is an extension of the formalism used in Genesereth et al. (1984). As an example of the advantages gained when communication and promises were added to the interaction model, the formalism is applied to solve the multiple best plan case, where there are several outcomes recognized as best by all players.

4.3 Formalisms of belief

There are two main tendencies underlying the major existing formal models of belief; the first tendency bases its semantics on the possible world formalizations begun by Hintikka (1962, 1975). I shall call this tendency, the possible world approach. The second tendency takes a syntactic approach. In this approach the beliefs of an agent are identified with a set of sentences.

These two approaches take two extreme views of the granularity of the semantics; in the possible world approach, the beliefs of an agent are closed under logical consequence (what Hintikka called

logical omniscience). It is built into the logic of the possible world formalism that if "p" is believed and "p" logically implies "q", then "q" is believed as well. In the syntactic approach any two different sets of sentences are considered as distinct semantic entities.

There are deficiencies in both tendencies, when dealing with what the agent explicitly believes, that suggest (Levesque, 1984) the need of a model whose semantics would lie somewhere between the syntactic and the possible world approach, so that different sentences can represent the same beliefs without requiring that all logically equivalent sets do so. The logic of implicit and explicit belief presented in Levesque (1984), represents a step towards this direction.

The reason that the possible world approach to belief or knowledge leads to logical omniscience is that beliefs are characterized completely by a set of possible worlds (namely those that are accessible from a given possible world). Intuitively, these possible worlds are to be thought of as the full range of what the agent thinks the world might be like. If he only believes that "p" is true the set of possible worlds will be all those where "p" is true. However, because sentences which are tautologies will also be true in all these possible worlds, the agents is thought of as believing them just as if they were among his active beliefs. In terms of possible worlds there is no way of distinguishing "p" from these tautologies.

Logical omniscience presents three serious drawbacks (Levesque, 1984):

1 Every valid sentence must be believed.
2 If two sentences are logically equivalent, then one must be believed if the other is.
3 If a sentence and its negation are both believed, then every sentence must be believed.

The problem with the syntactic approach is that it considers any two sets of sentences as distinct semantic entities and consequently different belief sets. To see why this is a problem, consider that the conjunction of "p" and "q" belongs to the set of the agent's beliefs. Given a syntactic understanding of the set of beliefs there is no reason to assume that the agent believes the conjunction ("q and p") since ("p and q") may be in the belief set while ("q and p") may not.

I will now review the work of two researchers whose work on belief is related to DAI. These are Konolige (1984, 1985) who originally worked with Nils Nilsson in developing a multi-agent planner (Konolige, 1980) and takes a syntactic approach to belief, and Appelt (1980a, 1980b, 1981) who uses the possible worlds approach in the study of the relation between DAI and natural language understanding. I will also review the work of Levesque (1984), as he presents an approach to the representation of beliefs that overcomes some of the shortcomings of the possible worlds and syntactic approaches.

Konolige. Konolige takes a syntactic approach identifying an agent's beliefs with formulas of a first order language called object language; statements that an agent "knows" or "wants" something are modelled by introducing new "know" and "want" relations between an agent and a formula in the object language. Since it is necessary to be able to make statements about an agent's beliefs without being able to enumerate all those beliefs, a meta-language, whose terms refer to the object language, is used.

He does not use an axiom to the effect that agents believe the logical consequences of their beliefs, because he wants to admit the possibility that different agents use different procedures for making inferences. In this way he avoids logical omniscience.

Appelt. As Rosenschein (1985) reports, Appelt addresses the same issues as those addressed by Konolige. He uses Moore's first order axiomatization of the possible world semantics for the modal logic of belief, knowledge and action (Moore, 1985). He is mainly concerned with the relationship between DAI and natural language understanding and generation (Appelt, 1980a, 1980b). His thesis (Appelt, 1981) centered on a planner, known as KAMP (Knowledge and Modelling Planner) that generated utterances designed to influence the mental state (beliefs and goals) of a second agent; it was also capable of the formalization of cooperative multi-agent plans.

Levesque. Levesque (1984) suggests a new interpretation of the possible world characterization of belief. Instead of taking logical omniscience as an idealization in the modelling of the belief of an agent, it can be identified with what the world would be like if what he believed were true.

He argues the need for a logical language that would include two operators, B and A. B(p) will be true when "p" is explicitly believed, i.e. when it is actively held to be true by an agent. While A(p) will be true when "p" is implicit, i.e. when it follows from what is believed.

The possible world semantic is appropriate for dealing with the implicit beliefs of an agent but fails to represent properly the explicit belief of an agent because of the logical omniscience.

The problem with logical omniscence is that there is no way to distinguish between what the agent explicitly beliefs and the tautologies that are included among his active beliefs. One way to avoid all the tautologies as part of the agent's beliefs is by replacing the possible worlds with a different semantic entity, named situation. A situation may support the truth of some sentences and the falsity of others, but may fail to deal with other sentences at all, in particular sentences not relevant to what an agent actually believes (including some tautologies).

The idea underlying the logic of belief presented by Levesque is to identify explicit belief with a set of situations rather than possible worlds.

4.4 Theories of communicating distributed processes

This area is not directly related to DAI, but it can be used as a framework to construct Multiple Agents Systems. There is extensive work in this area; a brief review follows.

4.4.1 Hoare

Hoare (1985) developed CSP (Communicating Sequential Processes); a mathematical theory described by a systematic collection of algebraic laws. It provides a secure mathematical foundation for avoidance of errors such as divergence, deadlock, and non-termination and for achievement of provable correctness in the design and implementation of distributed computer systems.

The mathematical theory gives a rigorous definition of the concept of process and the operators, in terms of which processes are constructed. These definitions are a basis for the algebraic laws, the implementations and the proof rules. Recursion may be used to describe processes that last a long time or for ever. The behaviour of a process is recorded as a trace of the sequence of actions (events) in which it engages. Processes can be assembled together into systems introducing concurrency. Non-determinism is allowed and proved to be a valuable technique for achieving abstraction. It provides a complete mathematical definition of the concept of a non-deterministic process.

Communication is described as a special case of a synchronized interaction between two processes, one of which outputs a message at the same time as another inputs it and if buffering is required on a channel, this is achieved by interposing a buffer process between these two processes.

Hoare also designed a programming language "OCCAM" which incorporates most of the theory of CSP. OCCAM has been used for research in parallel algorithms and concurrent systems in transputers. To get a flavour of how CSP works I will comment on an example given in Hoare (1985) pp 74, about the dining philosophers problem (Dijkstra, 1971).

The Dining philosophers problem

There are several versions of this problem but basically it consists of five eminent philosophers in a monastery, each of which has a room in which to engage in his trade, (thinking). There is also a common dining room with a circular table which has five places. Each place has a chair and to the left, a fork. In the center of the table there is a large bowl of spaghetti which is constantly replenished. And so the story goes, a philosopher spends most of his time thinking, but when he feels hungry, he goes to the dining room, sits down in his chair, picks up his fork and because of the nature of spaghetti needs to pick up a second fork to serve himself from the bowl and eat. When he is finished he puts down both forks, gets up from his chair and continues thinking in his room.

An obvious problem is that a fork can only be used by one philosopher at a time. If another philosopher wants it he has to wait until the fork is available. To start, we need to define the set of

names of events (the alphabet) which are considered relevant, to describe the potential behaviour of the philosophers and the forks (the processes of concern).

α**PHIL(i)** = {i.sits down, i.gets up,
i.picks up fork.i, i.picks up fork.(i⊕1),
i.puts down fork.i, i.puts down fork.(i⊕1)}
α**FORK(i)** = {i.picks up fork.8, (i⊖1).picks up fork.i,
i.puts down fork.i, (i⊖1).puts down fork.i}

where ⊕ is addition and ⊖, subtraction, modulo 5, this result identifying the right hand neighbour of the philosopher. Each event except sitting down and getting up requires participation of exactly two adjacent actors, a philosopher and a fork (see Fig. 5).

Defining the behaviour; the life of a philosopher is described as repetition of a cycle of six events and the role of a fork is to be picked up and put down by one of its adjacent philosophers. Note the recursive definition of the processes.

PHIL(i) = (i.sits down → i.picks up fork.i → i.picks up fork.(i⊕1) →
i.puts down fork.i → i.puts down fork.(i⊕1) →
i.gets up → PHIL(i))

FORK(i) = (i.picks up fork.i → i.puts down fork.i → FORK(i)
|(i⊖1).picks up fork.i → (i⊖1).puts down fork.i → FORK(i))

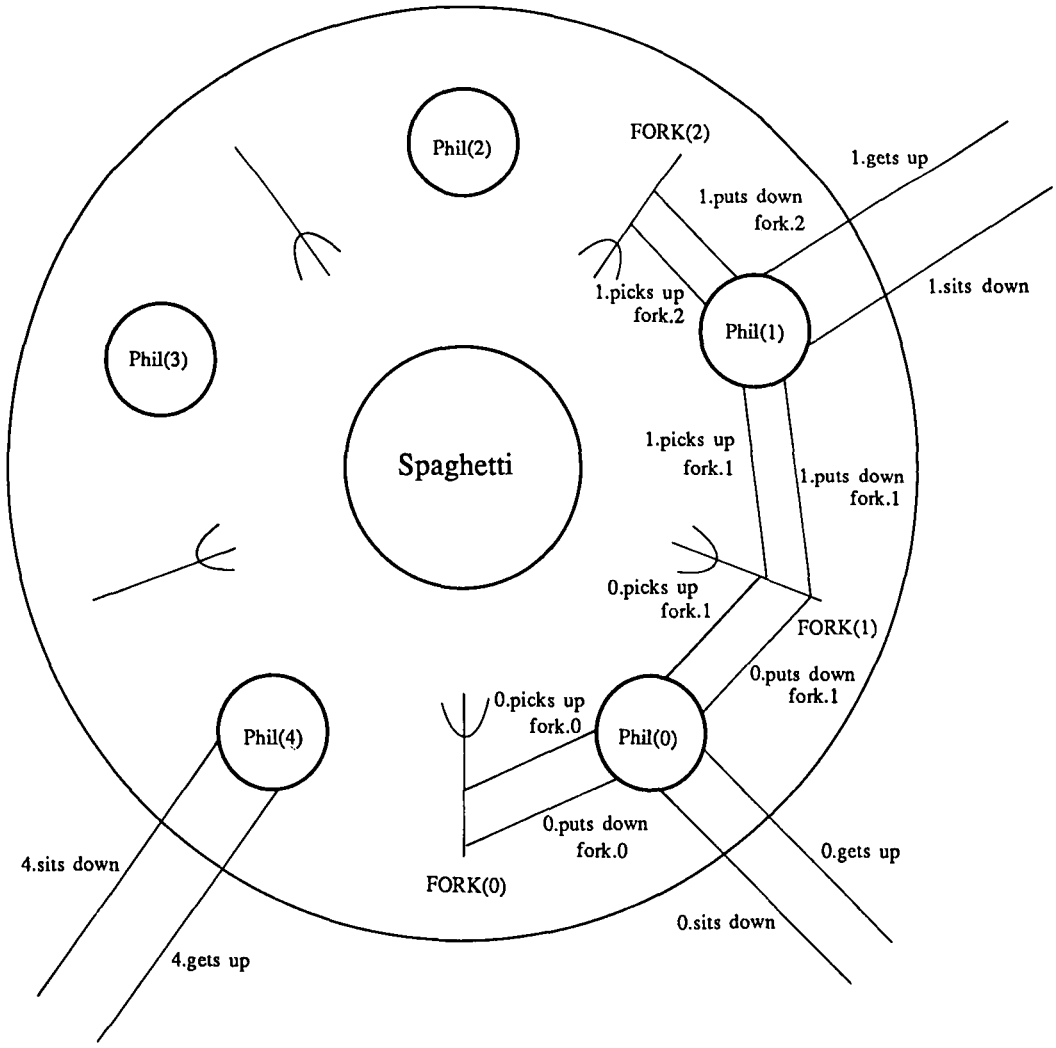

Figure 5 The seating arrangements in the dining philosophers problem.

The behaviour of the monastery is the concurrent combination of the behaviour of its elements.

PHILOS = (PHIL(0)||PHIL(1)||PHIL(2)||PHIL(3)||PHIL(4),
FORKS = (FORK(0)||FORK(1)||FORK(2)||FORK(3)||FORK(4),
MONASTERY = (PHILOS||FORKS)

The concurrent combinator P||Q denotes a process which behaves like the system composed of processes P and Q interacting with each other. This participation is regarded as events that require simultaneous participation of both processes involved, so in the monastery a philosopher picks up fork i when the fork i is lifted by him.

Of course there is the problem of deadlock, suppose all the philosophers decide to eat at the same time. They all go to the dining room, sit down, pick up a fork and then try and pick up the second fork—which is not there. So none of them can eat, and all of them starve. There are several solutions to this problem, one given by Hoare (1985) is having a "footman" who can help each philosopher to his chair. The footman's alphabet is defined as:

$\bigcup_{i=1}^{4}$ {i.sits down, i.gets up}

This does not allow more than four philosophers to sit simultaneously. This behaviour is defined as:

Let $U = \bigcup_{i=0}^{4}$ {i.gets up} $D = \bigcup_{i=0}^{4}$ {i.sits down}

FOOT (j) defines the behaviour of the footman with j philosophers seated.

FOOT(0) = (x:D → Foot(1))
FOOT(j) = (x:D → Foot(j + 1)|y:U → FOOT(j − 1)) for j ∈ {1,2,3}
FOOT(4) = (y:U → Foot(3))

A monastery free of deadlock is defined as

NEWMONASTERY = (MONASTERY||FOOT(0))

At the beginning, the events the process can be engaged in are in D. When there are four philosophers eating, no other philosopher can sit (the events the process can be engaged in are in U) preventing the deadlock. In Hoare (1985) a proof of the absence of deadlock is presented.

4.4.2 Milner

The Calculus of Communicating Systems (CCS) developed by Robin Milner (1980), provides a clear mathematical treatment of the general theory of concurrency and synchronization. The main objective of CCS is to provide a framework of constructing and comparing different models at different levels of abstraction. CCS is intended to serve as a framework for a family of models. As noted in Milner (1986a) the Calculus is founded on two central ideas: observation and synchronized communication.

"The first [idea] is observation; we aim to describe a concurrent system fully enough to determine exactly what behaviour will be seen or experienced by an external observer. Thus the approach is thoroughly extentional; two systems are indistinguishable if we cannot tell them apart without pulling them apart"

Secondly;

"Every interesting concurrent system is built from independent agents which communicate, and synchronize a communication is our second central idea. We regard communication between two component agents as an indivisible action of the composite system . . ."

Given that the nature of a process as a mathematical object is not universally agreed and cannot be precisely defined to serve as the basis of the calculus, the theory is based on the notion of observational equality: Processes are equal if and only if, they are indistinguishable in any

experiment based upon observation. A process is observed communicating with it. This equality relation defines a congruence relation over the terms of a language whose constructions reflect simple operational ideas. A process is understood to be a congruence class of terms.

There are many different approaches to the algebra of processes. The differences are in the constructions employed and in the notation of equivalence or congruences between processes. In Milner (1986b) a comparison of different sets of constructors and different equivalences in algebras of processes (including CSP) is made.

In Milner (1988) a historical review of CCS is presented. Also a comparison of the approaches taken by Tony Hoare (reviewed in the previous section), Mathew Hennessy, Jan Bergstra, W. Klop, Gerard Berry and their colleges is outlined and the main sources referred to.

4.4.3 Halpern and Moses

Halpern and Moses (1984), give a "knowledge perspective" to distributed protocols. They propose that communication in a distributed system can be viewed as the act of transforming the system's state of knowledge.

They represent knowledge as a hierarchy of "group knowledge", that is, states of knowledge of a fact p that the members of a group have. The weakest state is implicit knowledge, in which a group is said to know a fact p if, by pooling their knowledge, the members of the group could come to know p. The strongest state of knowledge is common knowledge; which corresponds to all members of the group knowing p, they all know that they all know p and this is essential for reaching agreements.

A distributed system is a finite collection of processors that are connected by a communication network. Communication is achieved by sending messages along the network. The processors are machines with clocks.

A processor has some initial information and it acquires additional information through message exchange. Processor i knows p (where p is a formula that talks about the real world and the processors' knowledge of it) exactly if it can deduce p from its information using a set of rules. All processors are honest, so that a processor sends a message stating p only if it knows p.

A protocol is a (possibly non-deterministic) distributed algorithm. A protocol essentially determines what messages can be sent by the processor as a function of their internal state.

Communication in a distributed environment can be regarded as the act of "improving" the state of knowledge of certain facts with respect to the hierarchy introduced. In practical distributed systems, common knowledge cannot be attained as communication cannot be guaranteed. Various relaxations of common knowledge are often attainable and suffice for many applications. These knowledge perspectives of communication can lead to a better understanding of issues in distributed environments.

The formalism of Halpern and Moses can be useful in providing lower bounds on the communication required of any protocol that guarantees to achieve certain goals. Further research by this team has focused on modal logics of knowledge and belief (Halpern and Moses, 1985) and logics of belief, awareness and local reasoning for multiple agent systems (Fagin and Halpern, 1988).

4.4.4 Hewitt

Hewitt (1977) developed an actor theory that describes important aspects of parallelism and serialization from the viewpoint of message passing objects. An actor system is composed of abstract objects called actors. Actors are defined by their behaviour when they accept communications. Actors interact in a purely local way by sending messages to one another. When a communication is accepted an actor can perform the following forms of actions concurrently: make simple decisions, create new actors, transmit more communications to its own acquaintances as well as to the acquaintances of the communication accepted, change its behaviour for the next message accepted (i.e. change its local state) subject to the constraints of certain laws.

The meaning of a message is the effect it has on the subsequent behaviour of the system i.e. the meaning of a message is determined by how it affects the recipients. The actor model is designed to aid in conceptual modelling of shared objects in a highly parallel open system.

The actor theory provides a mathematical framework to unify the conceptual basis of the Lambda Calculus and the Object-Oriented school of programming. This use of the theory is emphasized in (Hewitt and Jong, 1983), where actor theory is used to analyze the relationships between the roles of descriptions (supported by description languages that are designed to express properties but are incapable of taking action), and of actions (supported by procedural languages that have been designed to efficiently take action but suffer from a lack of descriptive capabilities) in large scale, open ended concurrent systems.

5 Concluding note

This article reviews current research on both organizational paradigms and theoretical underpinning of DAI. Research issues are presented and discussed.

A framework to describe the conceptual model of DAI systems has been presented. This framework presents the general principles and issues involved in current DAI research. Emphasis is given to conceptual clarity avoiding implementation detail. The framework consists of a number of dimensions relating to the conceptual model of both the system and the agent. The system model moderates the interactions and actions of the agents leading to coherent cooperation. The agent model relates to the agent's structure, its capabilities and awareness.

The framework presented is applied to problem solving architectures of different grain size and proves to be useful for comparing diverse systems. Using the framework dimensions it is possible to analyze systems with a common language. In particular two of the architectures analyzed were situated at different ends of the spectrum of conceptual clarity, the DVMT framework whose development shows an emphasis on empirical experiments and the MACE framework whose emphasis is on the conceptual clarity of the model proposed to provide a suitable testbed for experimental DAI systems. The emergent intelligence of a system is a general characteristic of DAI systems. This characteristic is related to the concept of global coherence. It refers to the fact that a DAI system will have a certain type and amount of overall problem solving ability as a function of the interaction of its components. This can be viewed from two perspectives: one regarding emergent intelligence in a DAI system as a network of individual intelligent systems designed to cooperate in some way, i.e. as a team of highly sophisticated agents. The alternative perspective shows a DAI system as a network composed of a large number of elements each of which is capable of a very limited amount of problem solving and in which the intelligence of the overall system is a result of the pattern of interaction among these "primitive" agents. From this perspective the intelligence of a system is more finely distributed among its elements. The connectionist model loosely represents this latter perspective (Feldman and Ballard, 1982; Toureltzky and Hinton, 1985; Rumehart and McClelland, 1986).

The area of emergent intelligence is an important research issue. Few authors actually mention this area explicitly, although it is intimately related to global coherence and organizational paradigms.

Agents need to be aware of their own capabilities and need to reason about other agents intentions, plans and beliefs. This fact focuses research into these issues more than traditional AI. There has been an increasing emphasis on the need for research into formalisms for agents to reason about another agent's knowledge and belief, and for modelling multi-agent interaction. Formalisms and specification techniques more suitable for parallel domains have been reviewed, particularly in the area of Global Coherence (planning, action, synchronization, belief, communication). An area not covered explicitly in this article is formal theories of knowledge representation. This topic is particularly well reviewed in Rosenschein, 1985a; Seel, 1988; and Halpern, 1986.

The principal sources of information are referred to in the course of the review. The views of DAI

expressed in this report can be seen as an overview of the current opinions of DAI researchers. The sources that guided the survey were the reports of the annual workshops on Distributed Artificial Intelligence, IJCAI proceedings, AAAI proceedings, AI Journal, AI magazine, and SIGART Newsletter.

Acknowledgements

This research has been sponsored by Consejo Nacional de Ciencia y Tecnologia de Mexico. I would also like to thank Gloria Quintanilla, Rajiv Trehan and Emilio Agustin for valuable discussion, and Jim Doran, (Essex University) and Paul Wilk (Edinburgh University) for their advice.

Reference Summary

- General DAI:
  Davis, 1980; Chandrasekaran, 1981; McArthur et al., 1982; Davis, 1982; Cammarata et al., 1983; Fehling, 1983; Smith, 1984; Doran, 1984; Doran, 1985; Rosenschein, 1985b; Doran, 1986; Halpern, 1986; Jagannathan and Dodhiawala, 1986; Huhns, 1987.
- Global Coherence:
  Smith, 1979; Corkill, 1979; Corkill, 1982; Corkill, 1983; Davis and Smith, 1983; Huhns et al., 1983; Durfee et al., 1985; Durfee et al., 1985b; Durfee et al., 1987b.
- Knowledge Representation:
  Nii and Feigenbaum, 1978; Smith, 1979; Erman et al., 1980; Gasser and Tenorio, 1986; Helpern, 1986.
- Communication:
  Kornfeld, 1982; Davis and Smith, 1983; Durfee et al., 1987a.
- DAI Architectures:
  —*Contract Net*: Smith, 1978; Smith, 1979; Smith and Davis, 1981; Davis and Smith, 1983; Van Dyke, 1987.
  —*MACE*: Gasser and Tenorio, 1986; Gasser et al., 1987a; Gasser et al., 1987b.
  —*DVMT*: Erman et al., 1980; Erman et al., 1981; Corkill and Hudlicka, 1982; Corkill and Lesser, 1983; Durfee et al., 1985a; Durfee et al., 1985b; Pattison et al., 1987; Durfee et al., 1987a; Durfee and Lesser, 1987b.
  —*ACC*: Huhns et al., 1983; Ju-Yuan et al., 1985.
- Formal Approaches:
  —*Planning*: Allen, 1981; McDermott, 1982; Georgeff, 1983; Allen, 1984; McDermott, 1985; Lansky, 1985; Georgeff, 1985a; Georgeff, 1985b; Georgeff, 1987; Pednault, 1987; Lansky, 1987a; Lansky, 1987b; Lansky and Fogelson, 1987.
  —*Action*: McCarthy, 1980; Georgeff, 1985b; Moore, 1985; Lifshitz, 1986; McCarthy, 1986; Shoham, 1986; Hanks and McDermott, 1986; Lifshitz, 1987; Ginsberg and Smith, 1987a; Ginsberg and Smith, 1987b.
  —*Synchronization*: Manna and Wolper, 1981; Rosenschein, 1982; Georgeff, 1983; Stuart, 1985; Rosenschein, 1986; Stuart, 1987.
  —*Decision Theory*: Genesereth, 1984; Genesereth et al., 1984; Rosenschein and Genesereth, 1984; Rosenschein and Genesereth, 1985; Rosenschein, 1986.
  —*Belief*: Hintikka, 1962; Hintikka, 1975; Appelt, 1980a; Appelt, 1980b; Konolige, 1980; Appelt, 1981; Levesque, 1984; Konolige, 1984; Konolige, 1985; Halpern and Moses, 1985; Fagin and Halpern, 1988.
  —*Communication*: Hewitt, 1977; Milner, 1980; Hewitt and de Jong, 1983; Halpern and Moses, 1984; Hoare, 1985; Milner, 1986a; Milner, 1986b; Walker, 1987; Milner, 1988.

References

Allen, J F, 1981. "A general model of action and time", *Computer Sc. Report TR 97*, University of Rochester N.Y., USA.

Allen, J, 1984. "Towards a general theory of action and time", *Artificial Intelligence* **23** pp. 123–154.

Appelt, Doug, 1980a. "A planner for reasoning about knowledge and action". In: *Proceedings of the First Annual Conference of Artificial Intelligence*, pp. 131–133.

Appelt, Doug, 1980b. "Planning natural language utterances". In: *Proceedings of the Eighteenth Annual Meeting of the Association for Computational Linguistics.*

Appelt, Doug, 1981. *Planning Natural Language Utterances to Satisfy Multiple Goals*, Ph. D. Thesis, Stanford University, Stanford Cal. USA.

Ballard, Dana, 1987. "Modular learning in neural networks", *Proceedings of the American Association of Artificial Intellignece*, pp. 279–284.

Cammarata, Stephanie, McArthur, David and David, Randall, 1983. "Strategies of cooperation in distributed problem solving", *Proceedings Eighth International Joint Conference on Artificial Intelligence*, 8–12 August Karlsruhe, West Germany, pp. 767–770.

Chandrasekaran, B, 1981. "Natural and social systems metaphors for distributed problem solving: introduction to the issue", *IEEE Transactions on Systems, Man, and Cybernetics* **SMC-11** (1) pp. 1–5.

Corkill, Daniel, 1979. "Hierarchical planning in a distributed environment", *Proceedings Sixth International Joint Conference on Artificial Intelligence*, Tokyo, Japan, pp. 168–175.

Corkill, Daniel, 1982. *A Framework for Organizational Self-Design in Distributing Problem Solving Networks*, Ph.D. Thesis, University of Massachusetts, Amherst, Mass. USA.

Corkill, D, Lesser, V and Hudlicka, E, 1982. "Unifying data-directed and goal-directed control: An example and experiments". In: *Proceedings of Second National Conference on Artificial Intelligence*, pp. 143–147.

Corkill, Daniel, 1983. "The distributed vehicle monitoring testbed: A tool for investigating distributed problem solving networks", *AI Magazine* **4**(3), pp. 15–33.

Corkill, Daniel and Lesser, Victor, 1983. "The use of meta-level control for coordination in a distributed problem solving network", *Proceedings IJCAI 1983, Eighth International Joint Conference on Artificial Intelligence*, 8–12 August 1983, Karlsruhe, West Germany, pp. 748–756.

Davis, Randall, 1980. *Report on the Second Workshop on Distributed Artificial Intelligence*, SIGART Newsletter No. 73.

Davis, Randall, 1982. *Report on the First Workshop on Distributed Artificial Intelligence*, SIGART Newsletter No. 80.

Davis, Randall and Smith, R G, 1983. "Negotiation as a metaphor for distributed problem solving", *Artifical Intelligence*, **20** pp. 63–109.

Dijkstra, E W, 1971. "Hierarchical Ordering of Sequential Processes", *Acta Informatica* **1** pp. 115–138.

Doran, J E, 1984. *First Thoughts About Designing Teams of Actors*, Proceedings of Alvey IKBS Planning Systems Work Shop No 2, Systems Designers Ltd. Fleet, Available for ICC, PO Box 26, Hitchen, Herts SG5 1SA, England.

Doran, J E, 1985. "The computational approach to knowledge, communication, and structure in multi-actor systems", *Artificial Intelligence and Sociology*, Ed. G N Gilbert and C Heath (Eds.), Gower: London pp. 160–171.

Doran, J E, 1986. *Distributed Artificial Intelligence and the Sociocultural Systems*, Internal Memo CSM-87, Sept Computer Sc. Dept. University of Essex, Colchester, England.

Durfee, Edmund, Lesser, Victor and Corkill, Daniel, 1985a. "Increasing coherence in a distributed problem solving network", *Proceedings Ninth International Joint Conference on Artificial Intelligence*, 18–23 August, Los Angeles Cal. USA pp. 1025–1030.

Durfee, Edmund, Lesser, Victor and Corkill, Daniel, 1985b. "Coherent cooperation among communicating problem solvers", *Proceedings of the 1985 Workshop on Distributed Artificial Intelligence*, Cal. USA.

Durfee, Edmund, Lesser, Victor and Corkill, Daniel, 1987a. "Cooperation through communication in a distributed problem solving network", In: *Distributed Artificial Intelligence*, Michael Huhns (Ed.) pp. 29–58.

Durfee, Edmund and Lesser, Victor, 1987b. "Using partial global plans to coordinate distributed problem solvers", *Proceedings of the International Joint Conference on Artificial Intelligence.*

Erman, Lee, London, P and Fickas, S, 1981. "The design and an example use of HEARSAY-III, In: *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, pp. 409–415. Morgan Kaufman, Los Altos, Cal. USA.

Erman, Lee D, Hayes-Roth, Frederick, Lesser, Victor and Reddy, D Raj, 1980. "The Hearsay-II speech understanding system: Integrating knowledge to resolve uncertainty", *Computing Surveys* **12**(2) pp. 213–253.

Fagin, Ronald and Halpern, Joseph, 1988. "Belief, awareness, and limited reasoning", *Artificial Intelligence* **34** pp. 39–76.

Feldman, J A and Ballard, D H, 1982. "Connectionist models and their properties", *Cognitive Science* **6** pp. 204–254.

Fehling, Michael, 1983. *Report on the Third Annual Workshop on Distributed Artificial Intelligence*, SIGART newsletter 84, spring, pp. 3–12.

Gasser, L and Tenorio, M F, 1986. *Rule-Agents: A Distributed Object Oriented Approach to Production Systems Using MACE*, DAI Group Research note 4, DAI Group, Dept of Computer Sc, USC, San Diego, Cal. USA.

Gasser, Les, Broganza, Carl and Herman, Nava, 1987a. "MACE: A flexible testbed for distributed AI

research", In: *Distributed Artificial Intelligence*, pp. 119–152, Michael Huhns (Ed.) M Kaufman Pub. Inc., Los Altos, Cal. USA.

Gasser, L, Bragaza, C and Herman, N, 1987b. "Implementing distributed AI systems using MACE. *Proceedings of the IEEE Third Conference on AI Applications.*

Genesereth, Michael R, Ginsberg, Matthew L and Rosenschein, Jeffrey S, 1984. "Cooperation without communication", *Proceedings of the American Association of Artificial Intellignece*, pp. 51–57. Revised version of the Report HPP-84-36, Heuristic Programming Project, Stanford University, September 1984.

Genesereth, Michael, 1984. *Solving the Prisoner's Dilemma.* Report No. HPP-84-41, Stanford University, Stanford, Cal. USA.

Georgeff, Michael, 1983. "Communication and interaction in multiagent planning", *Proceedings American Association for Artificial Intelligence*, Washington DC. USA pp. 125–129.

Georgeff, Michael, 1984. "A theory of action for multiagent planning", *Proceedings American Association for Artificial Intelligence Conference 1984*, pp. 121–125.

Georgeff, Michael, 1985a. "A procedural logic", *Proceedings of the International Joint Conference of Artificial Intelligence*, pp. 516–523.

Georgeff, Michael, 1985b. "A theory of process", *Proceedings of the 1985 Workshop on Distributed Artificial Intelligence*, Cal. USA.

Georgeff, Michael, 1987. "Reactive reasoning and planning", *Proceedings of the American Association of Artificial Intelligence*, pp. 677–682.

Ginsberg, M L, 1986. "Counterfactuals", *Artificial Intelligence* **30** pp. 35–80.

Ginsberg, M L and Smith, D E, 1987a. "Reasoning about action: A possible worlds approach", *Proceedings of the 1987 Workshop on Logical Solutions to the Frame Problem*, Laurance Kansas, USA.

Ginsberg, Matthew L and Smith, David E, 1987b. "Possible worlds and the qualification problem", *Proceedings of the American Association of Artificial Intelligence*, pp. 212–217.

Halpern, Joseph Y, 1986. "Reasoning about knowledge: An overview", *Proceedings of the Conference on Theoretical Aspects of Reasoning About Knowledge*, Morgan Kaufman Pub, Los Altos, Cal. USA.

Halpern, Joseph Y and Moses, Y, 1984. "Knowledge and common knowledge in a distributed environment", *Proceedings of the 3rd ACM Conference on Principles of Distributed Computing*, revised version pp. 1–28.

Halpern, Joseph Y and Moses, Y, 1985. "A guide to the modal logics of knowledge and belief: Preliminary draft", *International Joint Conference of Artificial Intelligence*, Los Angeles, Cal. USA.

Hanks, S and McDermott, D, 1986. "Default reasoning, nonmonotonic logics, and the frame problem", *Proceedings of the American Association of Artificial Intelligence*, pp. 328–333.

Hewitt, Carl, 1977. "Viewing control structures as pattern of passing messages", *Artificial Intelligence journal* **8** pp. 323–364.

Hewitt, Carl and de Jong, P, 1983. "Analyzing the roles of descriptions and actions in open systems. *Proceedings of the American Association of Artificial Intelligence*, pp. 162–167.

Hintikka, J, 1962. *Knowledge and Belief: An Introduction to the Logic of the Two Notions*, Cornell University Press, Cornell, NY, USA.

Hintikka, J, 1975. "Impossible Possible Worlds Vindicated", *Journal of Philosophical Logic* **4**, pp. 475–484.

Hoare, Charles A, 1985. *Communicating Sequential Processes.* Prentice-Hall International Series in Computer Science.

Huhns, M N, Stephens, L and Bonnell, R, 1983. "Control and cooperation in distributed expert systems", *IEEE*, 241–245.

Huhns, Michael, (Ed.), 1987. *Distributed Artificial Intelligence*, Morgan Kaufman Pub. Inc, Los Altos, Cal. USA.

Jagannathan, V and Rajendra Dodhiawala, 1986. *Distributed Artificial Intelligence: An Annotated Bibliography*, SIGART Newsletter, Number 95 pp. 44–56.

Ju-Yuan, David, Huhns, Michael N and Stephens, Larry M, 1985. "An architecture for control and communication in distributed artificial intelligence systems", *IEEE Transactions on Systems Man And Cybernetics*, 316–326 **SMC-15**, (3) pp. 316–326.

Konolige, Kurt, 1980. "Multiple agent planning systems", *Proceedings American Association for Artificial Intelligence Conference*, Stanford Cal. USA pp. 138–142.

Konolige, Kurt, 1984. *A Deduction Model of Belief and its Logics*, Ph.D. Thesis, Stanford University, Stanford, Cal. USA.

Konolige, Kurt, 1985. "A Computational theory of belief introspection", *International Joint Conference of Artificial Intelligence*, pp. 502–508.

Kornfeld, William A, 1982. "Combinatorial implosive algorithms", *Communications of the ACM* October **25**, (10) pp. 734–738.

Lansky, Amy, 1985. *Behavioral Specification and Planning for Multiagent Domains.* Technical Note 360, SRI International, Menlo Park, California, USA.

Lansky, Amy L, 1987a. "A representation of parallel activity based on events, structure, and causality",

Reasoning About Actions and Plans, M P Georgeff and A M Lansky (Eds.), Morgan Kauffman Pub. Inc, Los Altos, Cal. USA.

Lansky, Amy, 1987b. *Localized Event-Based Reasoning for Multiagent Domains*, SRI-AI TN 423, SRI International, Menlo Park, Cal. USA.

Lansky, Amy and Fogelson, David S, 1987. "Localized representation and planning methods for parallel domains", *Proceedings of the American Association of Artificial Intelligence* pp. 240–245.

Lesser, V and Corkill, D, 1983. "The distributed vehicle monitoring testbed: A tool for investigation distributed problem solving networks", *AI Magazine* **4** pp. 15–33.

Lesser, Victor R and Corkill, Daniel D, 1981. "Functionally accurate, cooperative distributed systems. *Transactions on Systems, Man, and Cybernetics* **SMC-11**, (1).

Levesque, Hector, 1984. "A logic of implicit and explicit belief", *Proceedings of the American Association of Artificial Intelligence*, pp. 198–202.

Lifschitz, Vladimir, 1986. "Pointwise circumscription: Preliminary report", *Proceedings of the American Association of Artificial Intelligence*, pp. 406–410.

Lifschitz, V, 1987. "Formal theories of action", *Proceedings of the Workshop on Logical Solutions to the Frame Problem*, Laurence Kansas, USA.

Manna, Z and Wolper, P, 1981. *Synthesis of Communicating Processes from Temporal Logic Specifications.* Report STAN-CS-81-872, Dept of Computer Sc., Stanford University, Stanford Cal. USA.

McArthur, Dave, Steeb, Randy and Cammarata, Stephanie, 1982. "A framework for distributed problem solving", *National Conference of Artificial Intelligence*, Carnegie-Mellon Univ, Pittsburgh, Penn. USA, AAAI, pp. 181–184.

McCarthy, J, 1980. "A form of nonmonotonic reasoning", *Artificial Intelligence Journal* **13** pp. 27–39.

McCarthy, J, 1986. "Applications of circumscription to formalizing common sense knowledge", *Artificial Intelligence* **28** pp. 89–116.

McCarthy, J and Hayes, P, 1986. "Some philosophical problems from the standpoint of artificial intelligence. In: *Machine Intelligence 4*, Edinburgh University Press, Meltzer B and Michie D (Eds.), pp. 463–502.

McClelland, James, Rumelhart, David and the PDP Research Group 1986. *Parallel Distributed Computing, Vol 2.* The MIT Press, Cambridge, Mass. USA.

McDermott, D, 1982. *A Temporal Logic for Reasoning About Plans and Processes.* Computer Sc. Research Report 196, Yale University, USA. Also *Cognitive Sc*, **6**, pp. 101–155, 1982.

McDermott, D, 1985. "Reasoning about plans. In: *Formal Theories of the Commonsense World*, Hobbs J and R Moore (Eds.), pp. 269–317, Ablex Publishing, Norwood, NJ, USA.

Milner, Robin, 1980. *A Calculus of Communicating Systems*, Lecture Notes in Computer Sc. 92, Springer Verlag, New York, USA.

Milner, Robin, 1986a. *A Calculus of Communicating Systems*, LCFCS Report ECS-LCFCS-86-7, Dept of Computer Sciences, University of Edinburgh, Edinburgh, Scotland.

Milner, Robin, 1986b. "Process constructors and interpretations", *Proceedings of the IFIP 86 Conference*, Dublin, pp. 507–514.

Milner, Robin, 1988. *Operational and Algebraic Semantics of Concurrent Processes*, LCFCS Report ECS-LCFCS-88-46, Dept of Computer Sciences, University of Edinburgh, Edinburgh, Scotland.

Moore, Robert, 1985. "A formal theory of knowledge and action", In: Formal Theories of Commonsense World, J R Hobbs and R C Moore, (Eds.), Ablex Publishing Co.

Nii, H and Feigenbaum, A, 1978. "Rule-based understanding of signals", In: *Pattern-Directed Inference Systems*, D A Waterman and R Hayes-Roth (Eds.), pp. 483–501, Academic Press, New York, USA.

Pattison, Eduard, Corkill, Daniel and Lesser, Victor, 1987. "Instantiating descriptions of organizational structures", In: *Distributed Artificial Intelligence*, M Huhns (Ed.), pp. 59–96, Morgan Kaufman, Los Altos Cal. USA.

Pednault, Edwin P D, 1987. "Formulating multiagent, dynamic-world problems in the classical planning framework", *Reasoning About Actions and Plans*, M P Georgeoff and A L Lansky, (Ed.), pp. 47–82, Morgan Kaufman Pub. Inc, Los Altos, Cal. USA.

Rosenschein, Jeffrey S, 1982. "Synchronization of multi-agent plans", *Proceedings of the National Conference of Artificial Intelligence*, American Association of Artificial Intelligence, pp. 115–118.

Rosenschein, Jeffrey S, 1985a. *Formal Theories of Knowledge in AI and Robotics*, Technical Note 362, SRI International, Menlo Park, Cal. USA.

Rosenschein, Jeffrey S, 1985b. *Rational Interaction: Cooperation Among Intelligent Agents*, Ph.D. Thesis, University of Stanford, Stanford Cal. USA. Also Published as STAN-CS-85-1081, Dept of Computer Sc., Stanford U. 1985.

Rosenschein, Jeffrey S, 1986. *Cooperation in the Presence of Incomplete Information*, Technical Report, Knowledge Systems Lab., Computer Sc. Dept, U. of Stanford.

Rosenschein, Jeffrey and Genesereth, Michael R, 1984. *Communication and Cooperation*, Report No HPP-84-5, Stanford Heuristic Programming Project, Stanford University, Stanford, Cal, USA.

Rosenschein, Jeffrey and Genesereth, Michael R, 1985. "Deals among rational Agents". *Proceedings of the International Joint Conference of Artificial Intelligence*, pp. 91–99.

Rumelhart, David, McClelland, James and the PDP Research Group, 1986. *Parallel Distributed Processing*, The MIT Press, Cambridge, Mass. USA.

Seel, N R, 1988. *Logics for Agent Design*, Technical Report 303-431, STC Technology Ltd, London Road, Harlow, Essex, England.

Shoham, Y, 1986. "Chronological ignorance", *Proceedings of the Fifth National Conference on Artificial Intelligence*, pp. 389–393.

Smith, Reid, 1978. *A Framework for Problem Solving in a Distributed Environment*, Ph.D. Dissertation, STAN-CS-78-700, Dept of Computer Sc., University of Stanford, Stanford, Cal. USA.

Smith, Reid, 1979. "A framework for distributed problem solving", *Proceedings of the International Joint Conference of Artificial Intellignece*, pp. 836–841.

Smith, Reid, 1984. "Report on the 1984 distributed AI workshop", *The AI Magazine*, Fall issue, pp. 234–243.

Smith, Reid G and Davis, Randall, 1981. "Frameworks for cooperation in distributed problem solving". *IEEE Transactions on Systems, Man, and Cybernetics* **SMC-11**, (3), pp. 61–69.

Stuart, Christopher, 1985. "An implementation of a multi-agent plan synchronizer", *Proceedings of the International Joint Conference of Artificial Intelligence*, pp. 1031–1033.

Stuart, Christopher, 1987. "Branching regular expressions and multiagent plans", *Reasoning About Actions and Plans*, M P Georgeoff and A M Lansky, (Eds.), pp. 161–187, Morgan Kaufman Pub. Inc., Los Altos, Cal. USA.

Touretzky, David and Hinton, Geoffrey, 1985. "Symbols among the neurons: Details of a connectionist inference architecture", *Proceedings of the International Joint Conference of Artificial Intelligence*, pp. 238–243.

Van Dyke, Paprvnak H, 1987. "Manufacturing experience with the contract net. In: *Distributed Artificial Intelligence*, Michael Huhns (Ed.), pp. 285–310.

Walker, David, 1987. *Introduction to a Calculus of Communicating Systems*, Report ECS-LFCS-87-22, University of Edinburgh, Scotland.