A SURVEY OF ARCHITECTURES FOR DISTRIBUTED ARTIFICIAL INTELLIGENCE

Technical Note 424

June 1988

By: Todd R. Davies, Computer Scientist
Representation and Reasoning Program
Artificial Intelligence Center

**APPROVED FOR PUBLIC RELEASE:
DISTRIBUTION UNLIMITED**

This paper was written during the first three months of 1987, and was included in a study of the feasibility of constructing distributed artificial intelligence testbeds, funded by an IR&D authorization to Tom Garvey. Its purpose was to list various activities undertaken through early 1987 in the area of DAI systems. Because of interest expressed by persons outside of that project, the report is released here as a technical note with very little revision. Because of the time delay it does not for the most part include research done during the last year and a half.

This research was also supported in part by the Office of Naval Research under Contract No. N00014-85-C-0251, SRI Project 8342.

333 Ravenswood Ave. • Menlo Park, CA 94025
(415) 326-6200 • TWX: 910-373-2046 • Telex: 334-486

Contents

Abstract

This report surveys literature and research in the field of distributed artificial intelligence (DAI) and provides an overview of computer architectures particularly suited to such research. It concentrates on work to date that has involved the construction of testbeds and development tools for DAI. It tries to draw some lessons from these efforts and suggests ways in which testbeds, which heretofor have been used primarily for experimentation, might be used in the full course of system development.

1 Introduction to Distributed Artificial Intelligence

This report surveys literature and research in the field of distributed artificial intelligence (DAI) and provides an overview of computer architectures particularly suited to such research. It concentrates on work to date that has involved the construction of testbeds and development tools for DAI. It tries to draw some lessons from these efforts and suggests ways in which testbeds, which heretofor have been used primarily for experimentation, might be used in the full course of system development.

1.1 Motivations and Distinction from Parallelism

The most general thing that can be said about research to which the term DAI is applied is that it all involves the attempt to understand and program intelligent behavior in some distributed context. For the most part, the nodes of the distributed system are processors or computers that are viewed as "agents." The agents interact with each other, and, in some cases, respond to a centralized controlling agent, but the "intelligence" in the system may be displayed by the whole system, by the individual agents, or at both levels. There seems to be an implicit distinction in the literature between work on *parallel processing* and work on *distributed* processing and problem solving. Use of the term 'parallel' to describe an approach to a problem indicates that the emphasis is on *performance* goals, i.e. eliminating the bottlenecks of serial processing. When an approach is instead described as "distributed," one can usually infer that the motivation is an increase in *functionality* of some type. For instance, a distributed system for solving a problem may be more *reliable* than a centralized system, because in the distributed case the effects of a failure are to a greater extent limited to just the node at which it occurs. A distributed design may be better in other ways for solving a particular problem, e.g. by providing a *better fit* to a partitioned problem, by being more *extensible* should the problem change or grow in scale, or by being more *modular* and therefore easier to design and debug [102,111,33,19,21]. Improved efficiency is often one of the goals as well—but the motivations usually go beyond efficiency alone when one speaks of "distributed," as opposed to "parallel," processing.

Much of the motivation for studying DAI comes from a desire to build intelligence into systems that are already distributed. Distributed databases, networks of workstations, and the like, already exist, because localized computers provide guaranteed response times and dedicated power [102,69]. Getting these systems to communicate intelligently is a DAI problem. So in some cases, a DAI approach is a necessity rather than a choice if one is to apply AI to a problem at all.

1.2 Varieties of DAI

The various problems being worked on under the rubric of DAI can be confusing to the uninitiated. Work reported at the seven DAI workshops held through 1986 [19,21,33,111,41,112][1] has ranged in its focus from neural networks to small numbers of workstations. Sometimes, the distributed nature arises from decomposing a single problem into multiple subproblems, and sometimes the distributedness is the main problem that arises from multiple agents or processes with different, possibly conflicting goals. Sometimes, the theory or techniques proposed focus on how an autonomous agent should behave in an environment with other intelligent agents, and sometimes they focus on how the "society" of agents should be organized to achieve globally desirable results. The agents can be of relatively equal status, or subservient to a central controller to whom they report. They can be specialized or general problem solvers, simple or complex, numerous or few, benevolent or competitive, tightly coupled to others or decoupled, communicative or silent, active or passive—in short, the space of possibilities is very large. What is common to all the approaches is only that they involve multiple processes.

Nonetheless, the field can be decomposed into a manageable number of subareas. The following list of nine such subareas of DAI is an attempt to distinguish between the types of problems being worked on under headings that have emerged in the literature. The subfields are given in rough order (from most to least) of how representative they are of work that has been called "DAI."

1. *Distributed Problem Solving.* Much of the work in DAI has involved building systems in which two or more agents work together to solve a single problem. In this subfield, the main problem is therefore how to achieve cooperation in the service of a common goal. The agents usually have complex capabilities of their own, so that much of the processing is done locally. Typical issues include when and from whom agents should seek help, how the group or its "director(s)" (if any) decide that the problem is solved, how different agents can work on separate parts of the problem semi-independently, when an agent should interrupt its own problem solving activities and answer a request, when it should volunteer information, who should have control over what, and so forth. Examples of systems that have been built for (and theories of) the cooperative solution of problems include [20,22,11,38,34,81,24].

2. *Distributed Processing.* Randy Davis distinguishes the distributed problem-solving paradigm discussed above from that of distributed *processing.* Davis writes, "Typically, for instance, most of the processing in such [distributed processing] systems is done at a central site and remote processors are limited to basic data collection. Control is not distributed and the processors do not cooperate in a substantial manner."[19] The goal for such systems is often to achieve fault tolerance in information gathering.

[1]The 1983 conference, held at the University of Massachusetts, was not reported.

In addition to issues such as how to combine evidence and otherwise integrate data from multiple "knowledge sources" (esp. sensors) that can have different capabilities and varying reliability, there has been a growing interest in this subfield in doing more of the interpretation processing at the site of the data collection [102]. This makes for more of a DAI-flavored problem. Still, the distinction from distributed problem solving is worth noting: The knowledge sources work independently of one another and report their results to a central agent who tries to make use of these results, or to a central representation like a blackboard. Examples of this work, most of which has been in the domain of perception, include [36,37].

3. *Multiagent Planning (Distributed).* Much of the work in this subfield could also be called distributed problem solving. However, whereas distributed problem solving may include tasks like monitoring in which agents are passive observers of the environment and other agents, in distributed *planning* problems the agents can take actions that affect the world and thereby change it for other agents. Another distinction from the distributed problem solving paradigm described above is that the problems of multiagent planning exist also in contexts in which the agents are not necessarily cooperating to solve a common goal. The focus is on an individual agent's goals and how they can be achieved when other agents and processes are around. The traditional problems of planning are therefore of interest, including how to compute the effects (and, importantly, the noneffects) of one's actions, how to know when an action is possible, and how to plan to achieve a goal when there are other agents to factor in and to make use of. Examples of research is this subfield include [50,70,20].

4. *Planning for Multiple Agents (Centralized).* In this paradigm, the planning process itself is not distributed. The issues rather focus on how an agent—either a controller or the designer—can coordinate multiple agents to suit particular needs. Typical problems include: how to decompose a problem, assign tasks, allocate resources, and schedule, and also who should be allowed to communicate with whom, and how much. Examples of work addressing these issues include [45,46,75,76]. This differs from distributed processing (see above) because the agents perform actions rather than just gathering information and interpreting it.

5. *Multiagent Domain Modeling.* Just as there is a subfield of distributed systems that deals with formalisms and specification languages for concurrent processing (e.g. Petri nets, data flow graphs) [88], one of the problems to be addressed in DAI is how to describe temporal and causal relationships between actions and events in the multiagent context. This area complements work in multiagent planning by addressing problems faced by the designer of a distributed planning system (or the scheduler in the centralized case), given reasoning and planning capabilities on the part of the agents. Typical issues include how to avoid interference between concurrent actions, what constraints should hold across agents for particular actions, how to represent these constraints and detect interference, and how the global behavior of the system of

agents can be predicted from facts about local behavior at different nodes. Examples of this work include [77,48,92,23,116].

6. *Intelligent Communication Between Agents.* Work in this subfield addresses the planning and interpretation of communication acts that take place between agents. In this subfield especially, one of the agents might be a person who is either the generator of an utterance to be interpreted by a computer, or the recipient of a communication initiated by the computer. The "communication" subfield therefore encompasses most research in computational linguistics, though the problems involved when all of the parties in a communication are artificial makes it especially pertinent to DAI. Typical issues include what message to send to convey an intention, desire, or belief, how to understand a received message in such terms, and how to plan speech acts and reason about the effects of these on the receiver. Some examples of this research are [12,13,1,95,96].

7. *Reasoning About Other Agents.* This subfield overlaps substantially with that of intelligent communication between agents, but also includes reasoning that is not based on communication from an agent (e.g. reasoning based on observed behavior or on beliefs about the agent and the situation) as well as reasoning about the effects on other agents of actions other than communicative ones—for example, believing that another agent has an intention and knowing that a particular action will interfere with that intention. Both this and the subfield above apply especially, but not exclusively, to distributed planning (they could also be problems in distributed interpretation or monitoring tasks, for instance). Some typical issues are how to infer things about the plans, beliefs, and goals of other agents, how to compute the effects of actions on these, and how beliefs about other agents and their plans should affect the agent's own reasoning. Examples of work in the area include [72,95,96].

8. *Rational Interaction.* This subfield gets its name from Jeff Rosenschein's Stanford Ph.D. thesis [102]. Unlike distributed problem solving, this work assumes that agents do not have a common goal or problem to solve, but rather must find ways to resolve conflicts between their goals by negotiation and compromise. Some issues that arise are what is rational (e.g., in a game-theoretic sense) for an agent to agree to, and what mechanisms for agreement and deal enforcement should be used. Some papers on this topic include [102,44,42,43,100,101].

9. *Massively Parallel Processing.* Some work on systems with thousands of nodes, e.g. relaxation labeling, neural networks, and semantic networks, has been deemed to be a part of DAI. In general, the "agents" in such systems are extemely simple homunculi, and the intelligence is intended to arise from the ways in which they are connected and changed at each iteration. For this reason, the philosophy behind this approach is often called "connectionism." Typical issues involve how the network topology should be set up, how each node should respond to its inputs, how connections should adapt to the data over time, and how to get convergence to an optimal state.

The literature in this area is very large, and growing. The work that has appeared in the DAI workshops includes [3,94,117,106].

Most of the papers that have appeared in DAI over the last decade and a half can best be categorized as being in one of these nine subareas of the field. But projects and systems often involve tackling more than one of them at a time. In the next two subsections, specific projects are surveyed that have given rise to important ideas in each of the subfields.

1.3 Survey of Approaches to DAI Problems

The literature of DAI is much too large to summarize in a way that mentions all of the projects that have been undertaken. Instead, this subsection are lists what might be considered the major *results* in the field—programs, techniques, and findings that have had an impact beyond the projects from which they arose. The presentation is chronological, but it omits all work on, and findings from, testbeds and development environments for DAI (covered in greater detail in section 2) as well as the work that has been done at SRI (see subsection 1.4). Other summaries of DAI include [64] and [102].

The first paper in AI to address distributed systems of agents appears to have been that of V. L. Stefanuk [114], which discussed theoretical aspects of systems whose main characteristics are "the antagonism of aims of different elements of the system and comparatively small information concerning the actions of other elements." However, work that both embodied the spirit of DAI and had important subsequent influence did not appear until two years later, in 1973 [26,59]. This initial research at Carnegie Mellon and at MIT concerned viewing systems implemented in single computers as collections of multiple agents, with the clear intention of making the abstract description reflect concrete implementations at a later point. There followed a period of developing more general techniques and designs for multiagent systems, using multiple processors, and, later, frameworks for experimenting with and reasoning about these techniques and designs, as well as principles for building distributed AI systems. There is still much work to be done. What follows is a listing and summary of the major developments (excluding testbeds and work done at SRI) in DAI.

1. *HEARSAY and Its Successors (CMU,Rand, UMass, WPI).* HEARSAY was a system for speech understanding originally implemented as a small number of coroutines realized as separate jobs in a PDP-10 [26,97]. The different "processes" communicated with each other on a shared data structure, which in HEARSAY-II became known as the "blackboard" [27,28]. The idea of the blackboard was that its contents represented the system of knowledge sources' (KSs') hypothesis about the received signal at any given moment. The hypotheses existed at different levels, so for instance a "phoneme

hypothesizer" knowledge source would post on the blackboard alternative pronunciations of a word hypothesized at the lexical level but not yet supported by hypotheses at the level of surface phonemes. The blackboard has become an important communication device for distributed processing systems, allowing multiple "expert" agents to work in parallel on a given problem and to work from each others' hypotheses. The approach of blackboards has been extended by others (e.g. [57]) to active, planning systems, and has been employed in the expert-system-building tool HEARSAY-III [29].

2. *Actors and Sprites (MIT).* "Actors" are Carl Hewitt's metaphor for programming objects, emphasizing "the inseparability of control and data flow" [59]. He and his colleagues have argued that all behavior can be analyzed in terms of the fundamental action of one object (actor) sending a message to another. Act2 [60] is an attempt to combine the descriptive character of declarative languages with the active character of procedural ones. Hewitt believes that logic, with its "truth-theoretic semantics," is an insufficient language for building what he calls "open systems" [61], which appears to be a synonym for distributed systems. As an alternative he advocates "due process reasoning" and "message passing semantics," in which actors construct arguments and build cases for hypotheses to be presented to an "arbiter," and the meaning of a message is the set of effects it has on other actors. Hewitt and W. A. Kornfield's "scientific community metaphor" [73] advocates a form of distributed problem solving that is modeled on the scientific community, with proposals and sponsors, and in which the KSs are called "sprites." Although Hewitt's terminology is often confusing, his views find expression in the work of many others in DAI and connectionism, and he has been advocating them for a long time. A problem that remains, even if he is right about how to go about solving DAI problems, is that of actually solving them.

3. *Beings and Agendas (Stanford).* The antecedent to Doug Lenat's AM program (described below) was one called PUP6, which consisted of "a community of interacting modules," (with more complex structure than Hewitt's actors) called "beings." The beings had a uniform architecture but were each designed "to simulate a particular expert in some small domain" [79]. PUP6 tried to use modules, who were each "expert" in some aspect of programming, to synthesize Lisp programs. In the subsequent Automated Mathematician (AM) program [80], Lenat introduced the additional notion of "agendas" as a global data structure in the same vein as blackboards. Using agendas as a communication scheme for distributed systems allows each agent or module to "share evidence about the merits of performing various tasks," although "no module need have any knowledge of how the other modules work or what knowledge they contain" [99].

4. *Contract Nets (Stanford,DREA,MIT).* Contract nets were proposed by Reid Smith [107] as an organizing strategy for distributed problem solving. The idea is that the problems to be solved are announced to all the agents, who bid to solve them, an idea later echoed in the "scientific community metaphor" of Kornfield and Hewitt.

Some problems that have been or might be pointed out for this approach are the lack of fault tolerance if the central dispensor becomes disabled, and the lack of redundancy in the resulting assignment [102], the communication overhead [64], and the lack of automatic dispensation when such dispensation is appropriate. Jeff Rosenschein has pointed out that all of these problems are solvable if the formalism is modified [102].

5. *Anarchy versus Hierarchy (Rand).* An interesting, nonobvious result from experiments with human subjects was shown by Wesson, Hayes-Roth, Burge, Stasz, and Sunshine [121]. The study was described by V. Jagannathan and Rajendra Dodhiawala in their annotated bibliography of DAI, as follows: "Experiments were conducted first involving human subjects trying to solve a dynamic crossword-like puzzle task. They note that the 'anarchic' human committee organization performed much better than the hierarchical organization. The yard stick was an omniscient single expert. The machine version of the experiments involved a blackboard to model the anarchic committee. In order to minimize communication, they note that the individual KS (expert) should have knowledge about what other experts know" [64]. Clearly, this one result does not establish the nonefficacy of hierarchies, but it provokes thought, and further experiments in this mode might help to debug our intuitions further about what works in human groups. Such research may be of use to DAI if the communication and control factors leading to the results can be elucidated and synthesized into principles that would apply to computer systems.

6. *Principles of Cooperation (MIT).* Jeff Rosenschein summarizes this work: "Another area examined by scientists at MIT has been 'principles of cooperation,' the notion that there are heuristic guidelines to be followed by computer systems that will be interacting with one another. The desire is to develop plans that are 'robust' (insensitive to other agents' actions) and 'cautious' (minimizing the chances for other agents to interfere). Towards these ends they have identified principles such as behaving predictably, generating simple plans, and using task domain knowledge to predict other agents' (future) plans [[20,109]]. A planning system was to be built using these principles, but it is not known whether such construction took place" [102].

7. *Neural Networks (Rochester, UCSD, CMU, Toronto, Caltech, AT&T, BostonU).* A great deal of work in neural networks has been done over the last five years, and anything close to a summary would not be possible here. For distributed AI, the ideas of central importance seem to be the organizing principles shared by these adaptive systems. Locally, at each processing element ("neuron") in such a system, the important unifying characteristic is that the state of the element is determined at each instant by a nonlinear function of a weighted summation of the states of the elements connected to it as inputs, as well as of (sometimes) its own previous state [105]. For the whole system, which may consist of millions of these elements, the central results show that global summarizing functions for the states and connection weights have accessible local minima, starting from initial states and weights [62,105]. A near-optimal

solution can be reached if simulated annealing is employed [66,58]. The weights may be set adaptively to minimize a summarizing function of error relative the patterns of states that arise or are specified as acceptable by an oracle, and the networks can very quickly classify input patterns to the desired output, with varying accuracy, when equilibrium is reached. Weighted voting networks may be useful for reasoning systems by making probabilistic and evidential inference more tractable [52] and by allowing a designer merely to specify what factors are relevant to believing, desiring, or intending something, without requiring the designer to specify probabilistic relations between propositions. The numerical relations can then be induced adaptively by the system. So the local model of a neuron used most often in these networks also seems useful for reasoning and representation of a more symbolic nature. This description does not do justice either to the capabilities or to the limitations of this approach. A good reference for the latter is the "Epilogue" of the recently expanded edition of *Perceptrons* by Marvin Minsky and Seymour Papert [91].

8. *Reasoning about Knowledge (IBM).* The work of Joe Halpern and his colleagues at IBM's Almaden Research Center has yielded important results in the theory of distributed systems. In particular, the logical modeling of distributed systems eases the design of efficient, fault-tolerant protocols that solve the problem of Byzantine Agreement [74] as well as it can be solved, i.e. that guarantee the earliest performance of simultaneous actions possible assuming some maximum possible number of failures [93,56]. What is important to note about this work for DAI is that these researchers have found success in using possible worlds models for multiple agents to analyze what each agent "knows" in each cycle or block of time about the states of the other processes, and that various models, depending on the application, can be useful for describing and reasoning about the knowledge, ignorance, and "beliefs" of agents [55, 31,32]. This work connects quite intimately with that of researchers at SRI concerning the relation between knowledge about other agents, communication, and planning.

1.4 DAI Research at SRI

Over the years, the Artificial Intelligence Center at SRI has produced a good deal of work in distributed artificial intelligence, particularly at the theoretical end.

In the area of natural language and communication, Doug Appelt wrote a planner called KAMP whose actions were utterances. The input to the program was a goal for changing the beliefs and desires of another agent [1]. More recently Phil Cohen has teamed with Hector Levesque of the University of Toronto in developing a formal theory of speech acts, beliefs, desires, and intentions, that can be used to reason about "rational" agents [12,13]. The interest in underlying principles of rationality, both for planning and for interpreting the actions and utterances of others, is an important characteristic of much of the DAI work at SRI. In particular, these principles are often thought to include design principles for agents with limited resources who must cope with a changing environment in real time. Martha Pollack's work [95,96] has concerned how agents can infer the plans of other agents from their utterances combined with beliefs about the other agent's beliefs and goals.

Much research has concerned planning in multiagent domains. In the early years of this research, Kurt Konolige and Nils Nilsson attempted to extend STRIPS-like planners for single agents to planning in the multiagent case [70]. a recent summary, "Over time, however, it was discovered that the underlying models of action for single agent planning were not sufficiently rich to capture the complicated synchronization properties of multiagent worlds. Subsequent research then began to explore richer action representations, and in particular tried to make use of work that had been done in computer science concurrency theory" [47]. has been done by Michael Georgeff and Amy Lansky. Georgeff has developed a means for handling the "frame problem" for planning agents (i.e. the problem of knowing what is and is not affected by an action) that takes advantage of the fact that there are other agents in the environment, where the environment itself can be viewed as an agent or process [50]. Both Georgeff and Lansky have worked on coordination, plan specification, and how to avoid interference between the actions of different agents [45,46,75,76]. Subsequent work has involved building representations of the concurrent actions and events in multiagent domains [77,48]. Most recently, Lansky and David Fogelsong have applied her Group Element Model (GEM) for concurrency and domain representation in an actual multiagent planning system known as GEMPLAN, which uses locality constraints in order to insulate agents planning for independent subregions of the planning space from having to consider changes that apply only to the tasks or regions assigned to other agents [78].

Reasoning about other agents in multiagent domains is the topic of current work by Kurt Konolige, Martha Pollack, and Todd Davies. Konolige's past work on how an agent can reason about the cognitive states of other agents using "experimental robot psychology" [72] and on how it can reason introspectively about its own beliefs [71], Pollack's work on plan inference from communication acts [95,96], and Davies' work on reasoning by analogy [16,17] may be able to be combined to form a general methodology for how an agent can reason about the intentions of other agents, their beliefs and their goals, in part by analogy to its own processes [49].

Part of the focus for the DAI research at SRI is on problems arising from the interaction of autonomous, mobile robots, both with other robots and with people. A group headed by Stan Rosenschein, which also includes Stan Reifel, Leslie Kaelbling, Nathan Wilson, Stuart Shieber, and Sandy Wells, has built one robot (named FLAKEY) that has sonar sensors, tactile sensing bumpers, and a camera for perception, a speech synthesizer and wheels for effectors, and a small on-board keyboard with display for receiving typed instructions and displaying data. Further features, such as an arm, may be added, as well as additional robots with whom FLAKEY can interact. Rosenschein, Kaelbling, and Fernando Pereira have been working on the theory behind the design of interacting automata embedded in an environment [103,104,65] and on modeling concurrent interacting automata [92].

In the area of perception, Marty Fischler and Oscar Firschein have advocated a "parallel guessing" approach to distributed processing for image understanding. In this approach, they "envision a parallel set of computers, each of which carries out a computation on a data set using some random or guessing process, and communicates the 'goodness' of its result to its co-workers through a 'blackboard' mechanism" [36].

In the area of massive or small-grained parallelism, Todd Davies has been exploring the ways in which neural network architectures could be constructed that embody knowledge that is described in a combinatorial language at the metalevel. Some preliminary results are reported in [18].

In short, research in DAI at SRI covers the full spectrum of varieties. Much has been developed in the form of techniques and theories for making distributed planning, communication, and processing computationally feasible. At this point, there is an especially strong interest in testing many of these techniques on realistic problems involving multiple robots and computers.

1.5 Traffic Monitoring as the Prototype Domain

Many of the programs written by DAI researchers, particularly the testbeds described in section 2, have as their application one form or another of traffic monitoring or control, largely because this domain has many of the characteristics of military problems, such as battle management, automated guidance systems, and automated assistants for pilots. In addition, there is potentially a more direct application of these systems to the Federal Aviation Administration's Advanced Automation Program (AAP) [40], which is expected to lead to "highly automated air traffic control by the turn of the century" [63]. The major goals of this program are to develop a system that will "be very reliable, be operational 24 hours a day, and be capable of beng upgraded without impeding the control of the air traffic" [63]. Obviously, the testing the software developed for this system (which is viewed as being functionally decomposed and partitioned into a distributed architecture [63]) in a very realistic simulation is of paramount importance [9,2]. The future for traffic control automation is not confined to aircraft; architectures for railroads and space vehicles are also being researched, developed, and manufactured [119].

1.6 Summary of Main Issues and Problems

The concerns of DAI researchers may be divided into "issues," that is, questions of the form "What approach *should* be taken along a particular dimension in building a system?," and "problems," i.e. questions of the form "How *can* a particular desideratum be achieved?" The listing of issues and problems in this subsection owes a great deal to previous lists provided in [102] and [112].

1.6.1 Issues:

The main issues appear to be the following.

1. *Control.* Agents may take all of their orders from a single directing agent, in which case control is centralized. At the opposite end of the spectrum, each agent can be completely autonomous, so that control is distributed throughout the entire system. There are various intermediate possibilities, including hierarchical control (like a business) and partitioned control (like an industry).

2. *Cooperativeness.* If agents always let the goals of others take priority, they might be called "benevolent" [44], or, to use a technical term, "nice." Alternatively, they may be negotiating agents, or they may be competitive or even hostile. Clearly, as is the case with most of these issues, the amount of cooperativeness will depend on the application—there is no single right level that covers all cases.

3. *Complexity.* The capabilities and architectures of each agent may range from simple to very elaborate, the advantages of the former being the possibility of having greater numbers of agents and ease of design, while for the latter the advantages are greater processing power at each node and more opportunity for specialization among the agents.

4. *Communicativeness.* How much agents say to those with whom they communicate can vary greatly from system to system. Agents can be completely silent, or they can communicate profusely. The cost of communication and the bandwidth impinge on this issue.

5. *Specialization.* Whether the agents should be homogeneous or very heterogeneous depends in large part on the task decomposition strategy. This is an important issue in the design of any system, since there is a trade-off between the ease of coordinating agents who share the same architecture, and the efficiency of specialization when the problem can be decomposed functionally.

6. *Redundancy.* In designing a system, one needs to evaluate how much the solution to a problem should depend on any one agent. Building-in redundancy makes the system fault-tolerant, but it can also result in less efficient use of a community of agents.

7. *Information Sharing.* There is a variety of possibilities for *whom* agents should share information with, as opposed to *how much* they should share (see Communicativeness). At one extreme, agents may share nothing. They may share information only with their nearest neighbors, or with the next agent up in the hierarchy, or only with a centralized controlling agent, or with every other agent in the system.

8. *Number of Agents.* Obviously, the number of agents in a multiagent system can range from 2 or 3 to many. The important consequences are for communication and overall system complexity. The number of connections required to maintain a constant communication time between arbitrary processors grows exponentially in the number of processors, but the spatial region defined by the processors, assuming no increase in the distance between nearest neighbors, grows at most only polynomially—as the square or the cube of the number of processors, depending on whether the agents live in a two- or a three-dimensional world. So communication between arbitrary agents in the system becomes harder as the number is increased. This makes the arrangement of agents important, so that they are close, for communication purposes, to the agents they need to talk to.[2]

9. *Decomposition.* Agents may inhabit a world or solve a problem based on function, object (as in man-to-man coverage in football), or space allotments. Another possibility is to assign tasks based on what agents are available by virtue of having completed their more recent assignments.

10. *Independence.* Similar but not identical to the issue of information sharing is the issue of how much of an agent's actions should be determined independently from those of other agents. A high degree of interdependence can magnify the effects of one agent's failure or of wrong information percolating through the system.

11. *Coherence.* A more abstract issue is the degree to which the system should appear to behave as a unified whole. In some systems a high degree of coherence is the aim, but in others the agents may be viewed as rather like a team, committee, or society.

1.6.2 Problems:

The main problems appear to be the following.

1. *Coordination.* Agents must respond to each other's actions at the right times and in the right way if a distributed system is to function properly. The focus for this problem is the set of mechanisms used by the individual agents to detect actions on the part of other agents and to control their own.

[2]This was pointed out to me by Stan Rosenschein.

2. *Conflict Resolution.* Two agents may have opposing goals. It is important, if conflicts are possible, that there be a means for detecting this, replanning in light of the conflict, and resolving known conficts in future intentions.

3. *Noninterference.* Even when agents share some goal, their intended actions may interfere with each other. This can be a problem both for the individual agents and for a central planner.

4. *Dealing with Incomplete Knowledge.* Because the cognitive states of other agents are underspecified by their actions, and because each agent has only a limited perceptual view of the world, an agent must infer what it does not know.

5. *Obeying Causal and Temporal Constraints.* In addition to avoiding interference, the actions of agents must be such that constraints of the form "A must occur before B occurs" and "C and D must occur simultaneously to cause E" are met by their collective plans or plan.

6. *Reliability.* Some systems require high levels of reliability, and others require less. To the extent that it is required, factors that affect it, such as failures at individual processor sites, and interdependence between components of the system, must be minimized.

7. *Communicating with Limited Bandwidth.* When agents are restricted to anything less than core-dumps broadcast to the whole system at every tick, they must communicate only partial information about their beliefs, goals, and intentions. This leads to a concern with how the important information can be selected.

8. *Allocating and Using Limited Resources.* Apart from communication channels, there are often other resources that are in short supply in a distributed system, such as tools, shared memory, and space. This is obviously related to the noninterference problem, but differs in that it emphasizes how priorities should be set given that a possible interference condition has been identified.

9. *Reacting to a Dynamic Environment.* Agents in a DAI system must be prepared for the possibility that their original plans could go awry, either because of unanticipated changes in the environment or because of unanticipated actions on the part of other agents; they must react in reasonable time and change their plans if necessary.

10. *Evaluating a Group Solution.* In distributed problem solving, an important question to be answered is when the agents should stop working on a given subgoal and conclude that it is fulfilled. This is difficult without some objective source of information, such as an oracle or an identifiable criterion for success about which the agents can agree.

11. *Extensibility.* In a system with a substantial lifetime, there is bound to be a need to add more agents to the system, more capabilities to the agents, or more

problems for the system to solve. If these possibilities are foreseen, the initial design should make the upgrading as painless as possible.

12. *Reducing Communication Uncertainty.* Well-known problems in distributed systems theory arise from uncertainties about whether, or at what time, a message from one processor is received by another. The protocols for communication should minimize the probability that an agent will act with unjustified confidence that a message has gotten through.

2 Testbeds and Development Tools for DAI

In the last section, a number of issues and problems of DAI were reviewed, as well as some specific approaches and findings relating to these issues and problems. In this section the part of research in DAI that has involved constructing "testbed" systems and DAI software development environments is explored. Testbed research involves the empirical testing and evaluation of different approaches, and development environments ease the task of building systems with multiple agents, under a potentially wide variety of approaches.

2.1 Survey of Testbeds

Most testbeds for DAI can be described as instances of the following scheme. An environment module is either simulated or actually created. The environment is often highly dynamic, and can be varied in, for example, the difficulty and complexity it poses, by adjusting parameters (in the simulated case) or by changing the setup (in the actual case). The testbed provides for a number of agents to "populate" this environment, i.e. to receive input from it. The number of agents may be fixed, or it may be a parameter, as may other aspects of the agents such as their architecture, what kinds of messages they send, what their policies for response are, and whom they are allowed to communicate with. The agents may be set up to work on a specific problem posed by the environment, like controlling traffic, or they may each be capable of being given different goals. The problem itself, in a distributed-problem-solving testbed, may have parameters to describe what agents should try to accomplish in a particular testbed run. Other parameters for the testbed may be how the problem is decomposed and assigned to the agents, and what the policies of the scheduler are, if one is present.[3] A search of the literature has revealed nine known testbed projects that have been initiated to date. Each of these is reviewed below. Of the eight, the first

[3]The description of this general framework comes largely from Tom Garvey (personal communication).

two projects—those at the University of Massachusetts and at the Rand Corporation—have been the most extensively reported, and are thus given fuller treatment in this survey.

2.1.1 University of Massachusetts' "Distributed Vehicle Monitoring Testbed"

The construction, use, and expansion of the Distributed Vehicle Monitoring Testbed ("DVMT") has been documented over the last eight years in a series of papers, all of them written in part by the testbed's chief architects, Victor Lesser and Dan Corkill [82,83,84,85,15,86,24,25,87].

Corkill and Lesser's primary aim has been to explore means for achieving and maintaining global coherence in a distributed-problem-solving network, in which the nodes share a common overall goal. Aspects of design that minimize the total time required for a network to solve a problem have been tested in many variations. The focus has therefore been on performance, which, as was mentioned earlier, is not usually considered by others to be the main problem in building *distributed* AI sytems. More will be said about this below. One of the goals in building the testbed was to discover principles for deciding what organization schemes are most efficient for particular situations, with the ultimate aim of programming-in criteria the system could use for changing its own organization to suit particular capabilities of the nodes and characteristics of the environment and problem. Some other questions to which experimentation was addressed included the following: What is the best balance between reliability and speed of the knowledge sources? How do the conditions of high and low error or noise in sensing affect what control regimes are most effective? To the extent that communication costs and reliability considerations preclude having a central controller, how can local decision making and communication achieve adequate levels of coordination between nodes? How can mechanisms for coordination be made simple enough so that their cost does not outweigh their benefits, and how does one determine this? What defines adequate coverage of the problem space by the network? How much connectivity is necessary? How much capability should each node have? What is the right mix between organizational and local control of a node, i.e. a medium between being a "company node" and a "lone wolf"? And finally, how should one balance the overhead required for cooperation between nodes against requirements on local processing? All of these questions may be best asked relative to a particular problem and environment.

The testbed has been in existence since the beginning of this decade. The number of problem-solving nodes has risen from between 2 and 4 to between 10 and 13, with about the same number of sensors. The task common to all of the simulations has been distributed traffic monitoring. Spacially separated processor nodes receive input from multiple, nearby sensors. The processors must cooperate to identify, locate, and track patterns of vehicles moving around and between them in a two-dimensional field. The vehicles generate acoustic signals that are picked up by the sensors. Each processor node was originally designed as an architecturally complete HEARSAY-II system [81] with capabilities for vehicle monitoring, and the processors have since been given planning capabilities. The processors transmit to other processors requests for the creation of hypotheses with specified attributes, as well as their own hypotheses. Hypotheses include specifying a time-location trajectory for an identified vehicle, the identity of the vehicle, and a degree of confidence. The goals to make certain hypotheses are also given importance ratings. There is an oracle that computes the optimal solution for each run ahead of time, and that is used for comparing the system's performance to the best possible performance. The earlier experiments were all done on four- and five-node systems. In the four-node version, the processors all participated in solving a problem, whereas in the five-node version, four of the processors sent their hypotheses to a fifth node that constructed the answer. The nodes in each version generate hypotheses consistent with their roles in the organization. Indeed, the hypotheses upon wich each processor is to work are a way of specifying the organization, but Corkill and Lesser make it clear that in the general architecture the node's role is just a guideline rather than an absolute—the node may be more or less internally directed depending on the setting of parameters. In later experiments a node has sometimes been given the capability to communicate "metalevel" information to other nodes; that is, information about what hypotheses it is working on rather than the hypotheses themselves.

Most of the experiments with the DVMT have been tests of the performance of a given parameter setting for the network and its nodes given a particular parameter setting for the environment. The testbed allows many parameters to be varied in a simulation run. The system allows arbitrary node-node and node-sensor connectivity, as well as arbitrary distributions of task processing capability. The accuracy of individual knowledge sources and sensors can be varied. Control policy is varied according to the level of priority each node gives to its own (internal) goals and hypotheses relative to those from the outside. A variety of communication policies can be invoked, such as transmitting hypotheses only, goals only, both, or neither. Within hypothesis communication a node may be set to send all of its hypotheses, only its completed ones, only those judged to be relevant to send, only its first and last, and so on. Vehicle characteristics can be varied, as can those of the communication channels. Placements of the sensors, their range, the number and size of the vehicles and the complexity of their signals, the hypothesis-forming power and type of the knowledge sources, consistency constraints on the number of legal hypotheses, and the amount of noise in the environment can all be varied independently. Transmissions of hypotheses and goals can be exclusively voluntary, exclusively in response to requests from other nodes, or mixed. At each node, one can set parameters for the weight given to local processing versus communication, weights for specifying the relative importance of each type of hypothesis and goal for that node to send and receive, threshold ratings for transmission and acceptance of hypotheses and goals, where to transmit to, weights for the use of received hypotheses and goals in processing, and for the priority of internal versus external goals. (This is not a complete list.)

Experiments have been run with the DVMT in many different configurations. The results are not easily summarized. One observed phenomenon was that when a node is externally directed, it can be distracted from productive problem solving by giving priority to a received goal or hypothesis when it should not. This led to degraded performance in some of the early simulations. Corkill and Lesser advocate "node skepticism" for robustness, that is, not giving full weight to received hypotheses and goals, but there do not appear to be any hard-and-fast rules for how much skepticism is the right amount. The idea is that even strong organizational responsibilities can and should be ignored when a node possesses strong information to the contrary, but this creates the problem of how a node should judge the relative value of its own information (or the importance of its own goals for solving the problem) versus that obtained and generated by other nodes [15]. Another general finding was that performance is improved if only higher-level information is exchanged between nodes, because of the heavy cost of communicating every low-level hypothesis. Corkill and Lesser found that, at least in the type of problem solving they investigated (finding the solution and then stopping), the overhead required for cooperation causes diminishing improvement in performance for each added node, although they appear to feel that improvement could be linear for continuous problem-solving systems. Other improvements in performance came from adding planners to the nodes and allowing metalevel communication so that nodes can coordinate with each other which problems they work on. The latter finding was more ambiguous in its scope. Corkill and Lesser found significant performance improvements with metalevel communication among nodes with overlapping interest areas and capabilities, but only if the performance without it was very suboptimal and if the bandwidth was sufficient to support the extra communication. No improvement was found to result from metalevel communication for nonoverlapping nodes. In general, Corkill and Lesser found that overlap turned out to be a very good thing, allowing flexibility among the nodes and improving reliability, and their mechanisms for improving performance (planning, metalevel communication, and so forth) worked better with it [24]. But of course for some types of tasks, more specialization and less overlap seem desirable. Blanket principles are very hard to extract from this work.

It seems worthwhile to list some of the problems that have arisen in the course of this work as well as some possible limitations of the methodology. A problem that was known from HEARSAY-II and that persisted in the earlier three- to five-node networks was that self-directed nodes led to wasted effort, global incoherence, misallocation of tasks (resulting in inaccuracy and untimely solutions), and lost processing due to waiting, nodes working at cross-purposes, and duplication of effort. In an effort to correct this, various coordination schemes were tried to make nodes more responsive to the rest of the network, but this resulted in a concommitant set of problems, including distraction and inefficiency due to overhead. Finding the right balance in this trade-off became, apparently, a delicate enterprise. The dominant theme in the findings seems to be "It all depends." Metalevel communication is useful sometimes, but not always. It may improve performance for overlapping nodes, but not if low bandwidth makes it too expensive, unless its absence causes a good deal of waste, in which case it might help—it all depends on the particular set-up, and the designer must have good intuitions about what can go wrong. These intuitions are surely aided by a careful reading of this work, but general principles are far less common in Corkill and Lesser than statements like "An organization that is specialized for one short-term situation may be inappropriate for another," and "No single static communication policy appears appropriate for all problem solving situations" [24].

One problem that, as these researchers point out, has not yet been addressed concerns when a node or network should decide that it has solved a particular problem. DVMT has an oracle, and problem solving continues until a solution is reached, but of course in real-world problem solving there is no oracle. Ideally, one would like it to be a fact about the system that its proposed solution of a problem in some fixed amount of time is guaranteed to be correct. This is, itself, a problem that seems well suited to a realistic testbed, since correctness can be empirically tested for a particular design. Corkill and Lesser's findings also do not apply to what they call "continuous" problem solving without a natural terminus, which seems more likely to be the kind required of real systems [24].

Corkill and Lesser have worked tirelessly and admirably on this experimental approach to DAI, and their papers contain a wealth of insights gained from their years of experience. They do not offer a prescription for building DAI systems given particular characteristics of the problem, environment, and possible networks, although it may be possible to come up with a fairly clear set of guidelines. They may well be planning to attempt this. In the meantime, the main benefit of this work seems to be that it has illustrated some of the many possibilities, trade-offs, and complexities involved in building DAI systems. The idea of building a testbed is also highly suggestive, not so much for its power in doing experimental AI but for its potential use as a development tool for real systems.

As was mentioned before, Corkill and Lesser's main objective has been to find the effects of design changes on *performance* in a DAI system. Their results are of the form, "In configuration n, with this particular pattern of vehicles and ghost data, agents set up to do x rather than y achieved solution in only 14 cycles instead of 19, where optimal is 10." An alternative approach that could build on this work would be to forget about efficiency, for the moment, and concentrate on giving the individual agents the knowledge and inferential power necessary to make intelligent decisions about what to do under a reasonable but uniform set of constraints on communication.

Later, one could concentrate on making the inference at each node efficient through parallel processing and fast algorithms. The reason one might be optimistic about this approach is that it conforms more to our experience as human problem solvers. We are not able to transmit all of our hypotheses and goals to everyone else, and receive them all, so it should not be seen as a problem that computers cannot either. However, we can make very fast and informed local decisions, understand our place in the scheme of things, decide when to seek help, and so forth. Perhaps this goal—focusing on the individual agent's processing capabilities in a way that takes account of other agents—would be a good way to proceed.

2.1.2 Rand Corporation's Framework for Distributed Pvroblem Solving

A group at the Rand Corporation, including Robert Wesson, Randy Steeb, Frederick Hayes-Roth, Perry Thorndyke, David McArthur, Stephanie Cammarata, Sanjai Narain, and William Giarla, reported work on an experimental environment for distributed air traffic control from 1980 until 1985 [120,118,89,90,11,113]. The questions that interested them were similar to the ones asked by Corkill and Lesser. They sought experimentally to find functional relationships between architecture structures and system performance measures [120]. In particular, they also assumed a single goal for the system and therefore sought globally coherent behavior arising from local computation.

The task for the system was slightly different from the one for DVMT. Multiple problem-solving nodes guided airplanes through a simulated three-dimensional airspace. The problem was to develop flight plans for each of the vehicles to ensure a separation of at least three miles horizontally and 1000 feet vertically between aircraft, while conserving fuel and allowing the airplanes to reach their destinations if possible. The researchers attempted to evaluate the effectiveness of cooperation strategies *holding constant the degree of expertise available at each node.* As was mentioned in the section on DVMT, the reverse approach may be fruitful for future testbeds. In this testbed, each agent knew its own airplane's flight plan, and had beliefs about the flight plans of the airplanes around it. When agents perceived a potential conflict (violation of the separation rules), they had to find some way, with or without communication, to change their plans [120,118,89,90]. Agents in the testbed had a constant bandwidth for communication (no more than five messages every 15 seconds), but the distribution of planning tasks was variable. Performance indices measured included the communication load, processing time, and "task effectiveness" (separation rule violations and, less importantly, fuel use) [11].

The parameters that were varied in the simulation changed over the course of the research. Originally, there were plans to experiment with various ways of assigning processors to airplanes, including space-centered (covering a region), function-centered (according to mission function), object-centered (one agent per aircraft), and plancentered (specialization by subgoal) [120]. In the reported research up through 1985, however, only the object-centered approach was ever used—aircraft were essentially identified with a processor. Initially, two general coordination regimes were tested—autonomous planning without cooperation (like cars on a freeway), and a cooperative strategy. The density of aircraft was varied for each of these strategies. The results were that, with low traffic, autonomous control produced conflict-free plans as often as cooperative control did, but in the high-density case the cooperative regime performed significantly better [118].

The experimenters then tested various ways to distribute the planning burden in the cooperative case. In one approach, the task was shared by the agents involved in a potential conflict. In three other approaches, a single agent was selected to plan the paths for both agents. Selection criteria for the designated agent varied and included selection of the least spatially constrained agent and of the most knowledgeable, least-committed agent. Results indicated that the latter selection criterion performed best among the task-centralized approaches. The authors made some effort to point out results that surprised them [11]. This seemed to make a stronger case for the utility of the testbed than is found in the writings of Corkill and Lesser, especially since there was more of an attempt to integrate the results into an understandable theory in the case of the Rand researchers. Unfortunately, they did not report anywhere near the volume of results that the DVMT researchers reported, so the robustness of the phenomena they discovered is not easy to assess. By 1984, the researchers reported working more on strategies involving individual decisionmaking [113], with the strategies written in ROSS, an object-oriented language for simulation originally developed by Phil Klahr and William Faught [67] and used later in the construction of the SWIRL system for simulating military air battles [68].

2.1.3 DREA's CNET

CNET was a system built by Reid Smith at the Defence Research Establishment Atlantic (DREA) in Dartmouth, Nova Scotia. It was conceived of as a "medium for writing programs for distributed problem solvers," but had many of the features of a testbed. Its domain was sensor data integration. Its name derived from the protocol for communication among the nodes of the system—the contract net. The questions to which it was addressed were similar in form to those of the DVMT and Rand projects, in particular "what task specific information to communicate, how to combine conflicting results from different nodes and coordinate node activities to maintain global coherence, and how to manage the tradeoff between communication and computation" [107]. One of the comparisons to which it was geared was that between task-sharing and result-sharing, a distinction introduced by Smith and Randy Davis [109] to capture the fact that cooperation may involve a distribution either of tasks or of results. Although the project was apparently abandoned due to Smith's departure from DREA, CNET was interesting in that it had a more powerful language for knowledge representation, based on the Unit Package [115,108], than most experimental DAI systems have had [110].

2.1.4 WPI's Distributed Sensor Networks Testbed

Peter Green of the Worcester Polytechnic Institute in Worcester, Massachusetts, reported in 1982 on a testbed for systems to acoustically track low-flying aircraft [53]. This grew out of the HEARSAY-II and Distributed Sensor Nets projects at CMU [81,37]. The DSN testbed was "a distributed AI system of six geographically distributed nodes with each node containing three processors that were used for data interpretation and message communications" [54]. The architecture was distributed HEARSAY-II, with separate blackboards at each node. The main conclusions were that the distributed HEARSAY architecture lacked the desired features of modularity of the nodes and focused ordering of KS invokation. What is meant by "modularity" is that the modules should be able to be developed by separate teams, "should be separately testable and obey good software engineering practices in terms of data hiding and freedom from interactions." The ordering problem arose because "KSs are triggered by the order in which the blackboard is searched rather than the KS execution being ordered by a consideration of the current problem state and goals." An attempt to solve these problems was the development tool AF, described in the next subsection [54].

2.1.5 GTE's Experimental Testbed

W. Frawley, Ralph Worrest, and Henrik Sandell of GTE Laboratories have been developing an experimental environment for exploring "real-time cooperative planning and action" in multiagent domains [39]. Only a few results have been reported. One early conclusion has been that cooperation and communication need to be designed into the system from the start to allow better use of it, that is, the agents should not be constructed first as isolated processors and then given the capability to interact. The comment by the authors that "creating a useful experimental environment for DAI research is a major undertaking" [122] is one worth noting, and goes very well with the Rand researchers' remark that "Distributed problem solving is an enigma" [11]. The thought is that in-principle arguments for the usefulness of cooperation and multiple agents are hard to make real; the practical issues are very complex. A promising idea of the GTE researchers is to construct taxonomies to characterize different dimensions and styles of behavior, and to try to reason out how they relate in addition to testing human intuitions on a system [122].

2.1.6 Stanford's Formal Framework for Rational Interaction

Some very interesting "theoretical testbedding" has been done in the Knowledge Systems Laboratory at Stanford by Mike Genesereth, Matt Ginsberg, and Jeff Rosenschein. Their point of departure from prior distributed problem-solving research is what they call the "benevolent agent assumption," viz: "All agents are assumed to have identical or nonconflicting goals and to freely help each other (with limited exceptions)" [44]. They reject this assumption and concentrate on how cooperation and compromise can be achieved by agents with potentially conflicting goals. As they point out, this "conflicting agent" metaphor is more realistic for many situations in which DAI might be applied. It also seems more general, since techniques derived under the assumption of benevolence appear less likely to scale when the assumption is removed than do techniques developed assuming potential conflict when that assumption is removed. Genesereth and his colleagues developed a formal framework for modeling and investigating what agents ought to do to resolve conflicts.

The following chronology of this work is taken from the group's summary for the 1984 Distributed Artificial Intelligence Workshop [44]:

> *Agents with identical goals, communication strategies.* Early work focused on the case of communicating agents that have identical goals, but potentially distinct (and conflicting) information about the world [[100]]. We have isolated strategies of information transfer that will cause agents to converge to identical plans, and identified strategies that will not cause convergence. We have also examined what role "lying" (i.e., the transfer of locally inconsistent information) might play in such interactions, and its limited utility.
>
> *Agents with distinct goals, no communication allowed.* Later work [[42, 43]] focused on the case where agents with potentially conflicting goals had to interact, and no communication was possible (i.e., after initial joint recognition of the interaction). A hierarchy of rationality assumptions was developed, and various types of behavior were shown to provably follow from each level of rationality. The notion of "common rationality" (where agents use identical decision procedures in choosing their actions) was introduced, and certain types of positive behavior were shown to result from this assumption.
>
> *Agents with distinct goals, promises allowed.* Our latest work [[101]] has introduced into the "conflicting agents" model the notion of binding promises and deal making. Agents with distinct goals are assumed to be capable of promising particular actions, contingent on certain conditions being met. Our research has answered the question of what kinds of promises are rational, given assumptions (once again) about the rationality of other

agents. It has also shown the power of communication: using a deal-making mechanism, agents are able to coordinate and cooperate more easily than in the communication-free model. In fact, there are certain types of interactions where communication allows mutually beneficial activity that is otherwise impossible to coordinate.

This work is a good example of the mileage that can be gotten from pad-and-pencil testbedding, which might be a beneficial first stage in the design of any DAI system. Because of its normative character, much of this work, at least in these early stages, failed to address the issue of resource limitations on how rational an agent can be. This has been a later topic of investigation in studying models of rational interaction and rational behavior, and is obviously crucial to investigate for practical uses of rational principles.

2.1.7 Intellicorp's Distributed Expert System Simulation Testbed

Robert Filman of Intellicorp has announced a plan to build a testbed to be populated with "collections of expert subsystems obeying different heuristic architectures." He is apparently interested in performance when expert systems of varying architectures attempt to cooperate and compete with one another. Examples of the architectures he has in mind include contract nets, distributed blackboards systems, the scientific community metaphor, "systems based on market economics, centralized control, classical operating system priority mechanisms, neural mechanisms, and hybrids of the above" [35].

2.1.8 Clarkson University's Testbed for Monitoring and Controlling Large Communications Systems

Susan Conry of Clarkson University recently reported a project that has been undertaken there to build a testbed for investigating cooperative problem solving in communications networks. The task is distributed over geographically separate agents, who must cooperate to assess the impact of various disturbances on the communication system, diagnose the sources of outages, and plan the reconfiguration of the network in the event of an outage [14].

2.1.9 Boeing's Experiments on Optimal Cooperation of Knowledge Sources

Miroslav Benda, V. Jagannathan, and Rajendra Dodhiawala of the Boeing Advanced Technology Center have done experiments to measure how the organization of knowledge sources in a blackboard framework influences system performance. The task for the community of agents was to corner an adversary. This ensured that the agents had to cooperate to solve the problem. They experimented with nine organizations and found performance differences among them [4,64,5].

2.2 Survey of Development Tools

All of the systems described above as "testbeds" embody a specific environment and family of problems for a proposed architecture to solve, and the role of the system is to generate experimental results to show what properties of an architecture are best suited to the particular version of the problem being run. In this subsection, environments of a more exploratory and general nature are considered. These development tools for DAI, which are sometimes also referred to as "testbeds" by the people who build them, generally allow a rather complete definition of the agents and their communication paths, as well as the problem to be solved and the environmental inputs to the system. Seven such systems are reviewed below.

2.2.1 AGORA (CMU)

AGORA is an environment in use at Carnegie Mellon "that has been designed specifically to build efficient problem solving architectures on a number of different multiprocessor computer systems" [7]. The driving problem for building this environment is the desire to speed the design of speech recognition systems, as part of a CMU effort to develop real-time, continuous-speech recognition without previous knowledge of the identity of the speaker. AGORA hides from the user all the details of the implementation in a multiprocessor or single-processor system, as well as shielding the user "from the complexity of interfacing programs written in different languages (C and Common Lisp are currently supported, and eventually Ada will also be supported) and executed in different operating systems (Berkeley Unix and Spice/Accent are currently supported)" [7]. The system provides facilities for defining knowledge sources, sets of hypotheses, and "knowledge clusters," or collections of knowledge sources that interact, communicate, or share some property in a way specified by the user. The system also provides blackboards for the knowledge sources to access, and schedulers for each cluster that coordinate the execution of its knowledge sources [7]. Roberto Bisiani claims that "AGORA has reduced the time to assemble a complex parallel system and run it on a multiprocessor from more than six man-months to about one man-month. The main reason for this lies," according to Bisiani, "in the fact that the details of communication and control have been taken care of by AGORA." He notes that even shorter times would be necessary for efficient development, e.g. running and completing a system in a single day [8].

2.2.2 MACE (USC)

A summary of MACE recently appeared in *Digital Design* [6]:

> One tool implemented on the iPSC system[4] with CCLisp is MACE from the University of Southern California. Developed on the IBM PC/AT, it was ported to other Common-Lisp standard environments–first to the Texas Instruments Explorer workstations and then to the iPSC. Because it is a flexible test-bed for building distributed AI systems, MACE may show how future AI systems will work. MACE programs are built from units called agents, which are objects encapsulating both data and program code. Rule-agents may represent a single rule or a rule-based system. Rule-based systems, a class of expert systems, capture knowledge as rules specified by a knowledge engineer. MACE allows testing and exploration of the parallelism of multiple-agent systems. It provides a complete program environment in which to define and use agents. It provides automatic load balancing by distributing agents across the nodes of a large-scale concurrent computer. Another MACE feature is instrumentation—meters and measuring devices that tell the programmer how the user application is distributed across the hardware system, and how efficient the execution is. This information is valuable not only to the researcher gathering data on experiments with parallel AI, but to the programmer debugging complex applications on many nodes. Most current work with MACE focuses on proving it is a description language for a variety of distributed AI systems.

A fuller account of MACE is given in [10].

2.2.3 Concurrent GESBT (SAIC)

The *Digital Design* article continues [6]:

> Another tool, which has been used to develop an end-user application [on the iPSC], is Concurrent GESBT. Concurrent GESBT helps to distribute an expert system across an array of iPSC nodes. Originally written in Common Lisp on the IBM PC, it includes additional message constructs. These constructs use the underlying CCLisp message streams to allow the expert systems on each node to communicate with expert systems on each of the other nodes. The resulting collection of cooperating expert systems offers a simple tool for distributed expert system applications.

[4]The iPSC AI computer from Intel Scientific Computers is enhanced by Gold Hill Computer's Concurrent Common Lisp (CCLisp). Implementing a concurrent configuration, the AI version of the iPSC supports up to 64 processing nodes. [Footnote in orginal.]

Concurrent GESBT is simple because it offers the programmer no help in distributing the application. The programmer determines how much knowledge to store in the knowledge base on each node, and how many nodes to use for the application. Most significantly, the programmer is responsible for tuning the application: optimal performance depends on a balance of computation per node (work for the individual expert) and communication of data between experts (the cooperation requirements of the problem). This is a difficult problem of load balancing, which is at the core of parallel processing research.

The first application for Concurrent GESBT is a demonstration battle management simulation developed at Science Applications International Corp. (McLean, VA), creators of GESBT. A community of cooperating expert systems uses captured knowledge to conduct air defenses for an aircraft carrier task group. Each expert system contains the defensive strategy for part of the total area about the aircraft carrier. The area around the aircraft carrier is divided into six sectors, each representing 60 degrees of the horizon.

Six sector commanders, located in an airborne radar command plane (E2C), are each assigned an expert system on a CCLisp node. A seventh expert system is assigned to the aircraft carrier commander, and an eighth expert system tracks total resources. The application can address more than 250 hostile airplanes as they converge on the carrier group from all directions. Each of the six expert systems uses its knowledge about defensive strategy to respond to the incoming airplanes. All six cooperate with neighboring expert systems to borrow or loan resources, to warn of threats entering new sectors and to report success or failure.

The Concurrent GESBT application uses only eight nodes of a 16-node iPSC system. The design is such, however, that many instances of the eight-node system can execute as each community of experts uses a different defensive strategy against the same threats. With concurrent execution of many scenarios, another expert system could choose among all of the simulation runs for the best defensive strategy. The final defensive strategy could then be presented to human operators, with justification and reasoning. Such a system would execute far faster than the human planning-staff equivalent, and of course the simulations could be run again and again as the problem data changed.

2.2.4 ABE (Teknowledge)

In 1985, a group at Teknowledge led by former HEARSAY researcher Lee Erman began work on ABE, which was described as "a multilevel architecture for developing intelligent systems" [30]. At the heart of the architecture is a computing model referred to as "module-oriented programming," which is presented as a generalization of object-oriented programming models such as Smalltalk and Actors. The extensions are that messages are broadcast to other modules who are attuned to receive that type of message, and the module definitions must encapsulate how they manage a set of processor resources (processors, memory, cycles) to accomplish pending module actions. The modules are constructed hierarchically. The goals of this project are to allow reuse of previously contstructed modules, integration of modules constructed in different languages, and large-scale development of application systems [30].

2.2.5 AF (WPI)

Peter Green of Worcester Polytechnic Institute has reported work on a "framework for real-time distributed cooperative problem solving" called AF (for "Activation Framework"). This architecture attempts to remedy some of the problems he encountered in the Distributed Sensor Networks Testbed (see previous subsection), namely the lack of modularity and control focus in the distributed HEARSAY architecture. The basic approach is to make what were HEARSAY nodes into what are here called "activation framework objects (AFOs)," which generate hypotheses at varying levels of activation (translated into immediacy for processsing by other nodes) and with varying "weighted evidence levels" to indicate strength of support for other hypotheses. In other words, it is a connectionist system, except that the machinery at each node is more complicated than that of model neurons because there are expert knowledge and procedures for hypothesis generation in each AFO. The architecture has been used to develop a navigator for an autonomous vehicle and a system for a "smart conveyer belt robot" [54].

2.2.6 Advanced Architectures Project (Stanford)

The Advanced Architectures Project of Stanford's Knowledge Systems Laboratory encompasses four subprojects, each of which is to contribute to the goal of achieving "2-3 orders of magnitude in overall speedup for expert sytems applicatons through the exploitation of parallelism" [98]. The subprojects are (1) *The CARE Simulator*, a program to simulate machines with "up to 256 processors" that is reported to run very slowly ("2-3 orders of magnitude slower than an equivalent program running serially"); (2) *Lamina*, "an object oriented programming extension to Lisp which is designed to operate in a distributed memory multiprocessor environment"; (3) *CAGE*, or Concurrent AGE, a version of the program AGE "extended with constructs for the exploitation of concurrency which is targeted at shared memory and hardware"; and

(4) *Poligon*, "a high level language and system for the implementation of blackboard-like AI applications on distributed memory machines" [98]. This project is funded by DARPA.

2.2.7 Connectionist Simulator (Rochestor)

The following is a description of the Rochester Connectionist Simulator for neural net simulation, which appeared with the announcement of its general release in February of 1987:

> The Rochester simulator allows construction and simulation of arbitrary networks with arbitrary unit functions. It is designed to be easily extensible both in unit structures and user interface, and includes a facility for interactive debugging during network construction. The simulator is written in C and currently runs here on Suns, Vaxen, and the BBN Butterfly multiprocessor (and should run on any UNIX machine). There is a graphics interface package which runs on a Sun under Sun tools, and is primarily designed for interactive display of the flow of activation during network simulation . The simulator is easy to use for novices, and highly flexible for those with expertise. [51]

2.3 Conclusions About Testbeds and Environments

The terms 'testbed' and 'development tool' have herein been distinguished from each other primarily along the dimension of purpose. Testbeds, in addition to providing a more complete specification of the problem than do environments, have as their primary purpose the carrying out of experiments to determine which of several candidate architectures is the best for a particular problem. Development tools encompass environments and languages for building distributed AI systems to solve a more general class of problems. An interesting possibility is that the two notions (testbed and tool) could be combined to form a tool for the development and testing of actual systems. The benefits of such a merger are more easily seen when one looks at the limitations of either approach in isolation. Development tools that lack a strong simulation of the environment and problem to be worked on offer little in the way of testing for the systems built with them, so while the definitional facilities may make things very nice for the programmer, it may be harder to test the system gradually during development. Testbeds put the emphasis on being able to test a variety of architectures quickly, which limits the specification of each architecture to filling-in parameters; the language for development is not rich enough, nor is the scope of possible architectures large enough, to support software development in a full sense. There are certainly places in the example systems described above in which the limits have been stretched. A system that is a true hybrid of testbed and tool could be of great use for the development of real systems, by eliminating the limitations inherent in either approach individually.

This suggests perhaps a different motivation for testbeds than has led to their construction in the past. To the extent that testbeds are designed merely to generate experimental results, it seems that their value, though nonnegligible even in this context, is somewhat diminished by the fact that much of what one learns qualitatively from performing such experiments could have been predicted beforehand, and that in other cases the causal factors that go into a result are so complicated as to make generalization impossible. For learning general principles about DAI, careful thought experiments may often be both more effective and less expensive, but at the point where these ideas are to be applied, it seems that a testbed is indispensable in the verification of well-thought-through, but potentially bug-ridden, designs. One might conclude, then, that building testbeds is a good idea, but in a different style and for different reasons from those that have been built in the past. Namely, the testbed should provide a rich development environment for exploring architectures pertinent to the driving problem, rather than just providing some parameters to be varied, and it should be used to find a good design tailored to the actual problem to be solved, rather than to find general rules for how to tackle a class of problems that may or may not be of equal importance.

References

[1] Appelt, D. E. *Planning Natural-Language Utterances to Satisfy Multiple Goals.* Ph.D. thesis, Stanford University, 1982. Issued as Technical Note 259, Artificial Intelligence Center, SRI International, Menlo Park, California, 1982.

[2] Avizienis, A. On the Achievement of a Highly Dependable and Fault-tolerant Air Traffic Control System. *Computer*, (February):84-90, 1987.

[3] Ballard, D. Summary of Rochester Group Effort. *SIGART Newsletter*, (73):49, 1980.

[4] Benda, M., Jagannathan, V., Dodhiawala, R. On Optimal Cooperation of Knowledge Sources. Boeing Artificial Intelligence Center, Boeing Computer Services, 1985.

[5] Benda, M., Jagannathan, V., Dodhiawala, R. T. On Optimal Cooperation of Knowledge Sources: An Experimental Investigation. In Sridharan, N. S., editor, Report on the 1986 Workshop on Distributed Artificial Intelligence. *AI Magazine*, 8(3):75-87, 1987.

[6] Billstrom, D. and Teeter, J. AI Meets Parallel Processing. *Digital Design*, (December):38-41, 1986.

[7] Bisiani, R. AGORA: An Environment for Building Problem Solvers on Distributed Computer Systems. In *Distributed Artificial Intelligence Workshop*, Sea Ranch, California, 1985.

[8] Bisiani, R. Agora. In Sridharan, N. S., editor, Report on the 1986 Workshop on Distributed Artificial Intelligence. *AI Magazine*, 8(3):75-87, 1987.

[9] Bleistein, S., Goettge, R., Petroski, F., and Wiseman, R. Capacity Management of Air Traffic Control Computer Systems. *Computer*, (February):73-82, 1987.

[10] Braganza, C. and Gasser, L. *MACE Multi-agent Computing Environment, Version 6.0, Release Note 1.0.* Technical Report CRI 87-16, Computer Research Institute, University of Southern California, Los Angeles, California, 1987.

[11] Cammarata, S., McArthur, D., and Steeb, R. Strategies of Cooperation in Distributed Problem Solving. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, Karlsruhe, West Germany, 1983.

[12] Cohen, P. R. and Levesque, H. J. Speech Acts and Rationality. In *Distributed Artificial Intelligence Workshop*, Sea Ranch, California, 1985.

[13] Cohen, R. R. and Levesque, H. J. Persistence, Intention, and Commitment. In *Reasoning About Actions and Plans: Proceedings of the 1986 Workshop*, Morgan Kaufmann, Los Altos, California, 1987.

[14] Conry, Susan E. DAI at Clarkson University. In Sridharan, N. S., editor, Report on the 1986 Workshop on Distributed Artificial Intelligence. *AI Magazine*, 8(3):75-87, 1987.

[15] Corkill, D. D. and Lesser, V. R. The Use of Meta-level Control for Coordination in a Distributed Problem Solving Network. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, Karlsruhe, West Germany, 1983.

[16] Davies, T. *Analogy.* Undergraduate Honors Thesis, Stanford University, 1985. Issued as Informal Note IN-CSLI-85-4, Center for the Study of Language and Information, Stanford University, 1985.

[17] Davies, T. R. and Russell, S. J. A Logical Approach to Reasoning by Analogy. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence.* Milan, Italy, 1987.

[18] Davies, T. R. Some Notes on the Probabilistic Semantics of Logistic Function Parameters in Neural Networks. To Appear In *Neural Networks (Special Issue): Proceedings of the First Annual Meeting of the International Neural Network Society*, Boston, Massachusetts, 1988.

[19] Davis, R. Report on the Workshop on Distributed AI. *SIGART Newsletter*, (73):42-52, 1980.

[20] Davis, R. *A Model for Planning in a Multi-agent Environment: Steps Toward Principles of Teamwork.* Working Paper 217, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1981.

[21] Davis, R. Report on the Second Workshop on Distributed AI. *SIGART Newsletter*, (80):13-23, 1982.

[22] Davis, R. and Smith, R. G. Negotiation as a Metaphor for Distributed Problem Solving. *Artificial Intelligence*, 20:63-109, 1983.

[23] Drummond, M. E. Refining and Extending the Procedural Net. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, California, 1985.

[24] Durfee, E. H., Lesser, V. R., and Corkill, D. D. Coherent Cooperation Among Communicating Problem Solvers. In *Distributed Artificial Intelligence Workshop*, Sea Ranch, California, 1985.

[25] Durfee, E. H. and Lesser, V. R. Incremental Planning to Control a Blackboard-based Problem Solver. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, Philadelphia, Pennsylvania, 1986.

[26] Erman, L. D., Fennell, R. D., Lesser, V. R., and Reddy, D. R. System Organizations for Speech Understanding: Implications of Network and Multiprocessor Computer Architectures for AI. *Third International Joint Conference on Artificial Intelligence: Advance Papers of the Conference*, Stanford, California, 1973.

[27] Erman, L. D. and Lesser, V. R. A Multi-level Organization for Problem Solving Using Many, Diverse, Cooperating Sources of Knowledge. In *Advance Papers of the Fourth International Joint Conference on Artificial Intelligence*, Tbilisi, Georgia, USSR, 1975.

[28] Erman, L. D., Hayes-Roth, F., Lesser, V. R., and Reddy, D. R. The Hearsay-II Speech-understanding System: Integrating Knowledge to Resolve Uncertainty. *Computing Surveys*, 12:213-253, 1980

[29] Erman, L. D., London, P. E., and Fickas, S. F. The Design and an Example Use of Hearsay-III. In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, Vancouver, B. C., 1981

[30] Erman, L., Fehling, M., Forrest, S., and Lark, J. S. ABE: Architectural Overview. In *Distributed Artificial Intelligence Workshop*, Sea Ranch, California, 1985.

[31] Fagin, R. and Halpern, J. Y. Belief, Awareness, and Limited Reasoning: A Preliminary Report. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, California, 1985.

[32] Fagin, R., Halpern, J. Y., and Vardi, M. Y. What Can Machines Know? On the Epistemic Properties of Machines. In *Proceedings of the National Conference on Artificial Intelligence*, Philadelphia, Pennsylvania, 1986.

[33] Fehling, M. and Erman, L. Report on the Third Annual Workshop on Distributed Artifical Intelligence. *SIGART Newsletter*, (84):3-12, 1983.

[34] Fikes, R. Automating the Problem Solving in Procedural Office Work. In *Proceedings of the AFIPS Office Automation Conference*, Houston, Texas, 1981.

[35] Filman, R. E. Architectures for Distributed Problem Solving Systems. In *Distributed Artificial Intelligence Workshop*, Sea Ranch, California, 1985.

[36] Fischler, M. A. and Firschein, O. *Parallel Guessing: A Strategy for High-Speed Computation.* Technical Note No. 338, Artificial Intelligence Center, SRI International, Menlo Park, California, 1984.

[37] Fox, M. Distributed Artificial Intelligence at Carnegie Mellon University: Distributed Sensor Nets. *SIGART Newsletter*, (73):44, 1980.

[38] Fox, M. S. The Intelligent Management System: An Overview. In Sol, H. G., editor, *Processes and Tools for Decision Support*, North-Holland Publishing Company, 1983. Also Issued as Technical Report CMU-RI-TR-81-4, Robotics Institute, Carnegie Mellon University, 1981.

[39] Frawley, W. and Worrest, R. Distributed Artificial Intelligence at GTE Laboratories. *AI Magazine*, 6:237, 1985.

[40] Garot, J. M., Weathers, D., and Hawker, T. Evaluating Proposed Architectures for the FAA's Advanced Automation System. *Computer*, (February):33-45, 1987.

[41] Gasser, L. The 1985 Workshop on Distributed Artificial Intelligence. *AI Magazine*, 8(2):91-97, 1987. Issued as Technical Report No. USC-86-222, University of Southern California, 1986.

[42] Genesereth, M. R., Ginsberg, M. L., and Rosenschein, J. S. Cooperation Without Communication. Report No. HPP-84-36, Heuristic Programming Project, Stanford University, 1984.

[43] Genesereth, M. R., Ginsberg, M. L., and Rosenschein, J. S. Solving the Prisoner's Dilemma. Report No. HPP-84-31, Heuristic Programming Project, Stanford University, 1984.

[44] Genesereth, M. R., Ginsberg, M. L., and Rosenschein, J. S. A Formal Framework for Rational Interaction. *AI Magazine*, 6:237-238, 1985.

[45] Georgeff, M. P. Communication and Interaction in Multi-Agent Planning. In *Proceedings of the National Conference on Artificial Intelligence*, Washington, D.C., 1983.

[46] Georgeff, M. P. A Theory of Multiagent Planning. In *Proceedings of the National Conference on Artificial Intelligence*, Austin, Texas, 1984.

[47] Georgeff, M. P. and Konolige, K. ONR Research on Distributed Reasoning and Planning. Research Summary, Project 8342, Artificial Intelligence Center, SRI International, Menlo Park, California, 1986.

[48] Georgeff, M. P. The Representation of Events in Multiagent Domains. In *Proceedings of the National Conference on Artificial Intelligence*, Philadelphia, Pennsylvania, 1986.

[49] Georgeff, M. P. and Konolige, K. G. Distributed Reasoning and Planning: Part One – Technical Proposal. Prepared by the Artificial Intelligence Center, SRI International, for the Office of Naval Research, 1986.

[50] Georgeff, M. P. *Many Agents Are Better than One.* Technical Note 417, Artificial Intelligence Center, SRI International, Menlo Park, California, 1987.

[51] Goddard, N. Rochester Connectionist Simulator Release in April. Electronic Message Posted to Neuron Digest, 17-FEB-1987 16:40.

[52] Goldman, A. I. *Epistemology and Cognition*, Harvard University Press, Cambridge, Massachusetts, 1986.

[53] Green, P. E. DSN Test-bed Tour and Demonstration. In *Proceedings of the Distributed Sensor Networks Workshop*, MIT Lincoln Laboratory, Lexington, Massachusetts, 1982.

[54] Green, P. E. AF: A Framework for Real-time Distributed Cooperative Problem Solving. In *Distributed Artificial Intelligence Workshop*, Sea Ranch, California, 1985.

[55] Halpern, J. Y. and Moses, Y. *Towards a Theory of Knowledge and Ignorance: Preliminary Report.* Research Report RJ 4448 (48136), IBM Research Division, San Jose, California, 1984.

[56] Halpern, J. Y. and Moses, Y. Knowledge and Common Knowledge in a Distributed Environment. 1987. Unpublished manuscript.

[57] Hayes-Roth, B. A Blackboard Model of Control. Report No. HPP-83-30, Heuristic Programming Project, Stanford University, 1984.

[58] Hecht-Nielsen, R. Performance Limits of Optical, Electro-optical, and Electronic Neurocomputers. Unpublished manuscript.

[59] Hewitt, C., Bishop, P., and Steiger, R. A Universal Modular ACTOR Formalism for Artificial Intelligence. In *Third International Joint Conference on Artificial Intelligence: Advance Papers of the Conference*, Stanford, California, 1973.

[60] Hewitt, C. and de Jong, P. Analyzing the Roles of Descriptions and Actions in Open Systems. In *Proceedings of the National Conference on Artificial Intelligence*, Washington, D. C., 1983.

[61] Hewitt, C. The Challenge of Open Systems. *Byte*, (April):223-242, 1985.

[62] Hopfield, J. J. Neural Networks and Physical Systems with Emergent Collective Computational Abilities. *Proceedings of the National Academy of Sciences USA*, 79:2554-2559, 1982.

[63] Hunt, V. R. and Zellweger, A. Strategies for Future Air Traffic Control Systems. *Computer*, (February):19-32, 1987.

[64] Jagannathan, V. and Dodhiawala, R. Distributed Artificial Intelligence: An Annotated Bibliography. In *Distributed Artificial Intelligence Workshop*, Sea Ranch, California, 1985.

[65] Kaelbling, L. P. An Architecture for Intelligent Reactive Systems. In *Reasoning About Actions and Plans: Proceedings of the 1986 Workshop*, Morgan Kaufmann, Los Altos, California, 1987.

[66] Kirkpatrick, S., Gelatt, C. D. Jr., and Vecchi, M. P. Optimization by Simulated Annealing. *Science*, 220:671-680, 1983.

[67] Klahr, P. and Faught, W. S. Knowledge-based Simulation. In *Proceedings of the First Annual National Conference on Artificial Intelligence*, Stanford, California, 1980.

[68] Klahr, P., McArthur, D., and Narain, S. SWIRL: An Object-oriented Air Battle Simulator. In *Proceedings of the National Conference on Artificial Intelligence*, Pittsburgh, Pennsylvania, 1982.

[69] Kleinrock, L. Distributed Systems. *Computer*, (November):90-103, 1985.

[70] Konolige, K. and Nilsson, N. J. Multiple-Agent Planning Systems. *Proceedings of the First National Conference on Artificial Intelligence*, Stanford, California, 1980.

[71] Konolige, K. A Computational Theory of Belief Introspection. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, California, 1985.

[72] Konolige, K. *Experimental Robot Psychology.* Technical Note 363, Artificial Intelligence Center, SRI International, Menlo Park, California, 1985.

[73] Kornfield, W. A. and Hewitt, C. E. The Scientific Community Metaphor. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-11:24-33, 1981.

[74] Lamport, L. and Fischer, M. J. *Byzantine Generals and Transaction Commit Protocols.* Technical Report Op.62, Computer Science Laboratory, SRI International, Menlo Park, California, 1982.

[75] Lansky, A. L. Behavioral Specification and Planning for Multiagent Domains. Technical Note 360, Artificial Intelligence Center, SRI International, Menlo Park, California, 1985.

[76] Lansky, A. L. Specification and Analysis of Concurrency. In *Distributed Artificial Intelligence Workshop*, Sea Ranch, California, 1985.

[77] Lansky, A. L. A Representation of Parallel Activity Based on Events, Structure, and Causality. In *Reasoning About Actions and Plans: Proceedings of the 1986 Workshop*, Morgan Kaufmann, Los Altos, California, 1987.

[78] Lansky, A. L. Localized Event-based Reasoning for Multiagent Domains. Technical Note 423, Artificial Intelligence Center, SRI International, Menlo Park, California, 1988.

[79] Lenat, D. B. Beings: Knowledge as Interacting Experts. In *Advance Papers of the Fourth International Joint Conference on Artificial Intelligence*, Tbilisi, Georgia, USSR, 1975.

[80] Lenat, D. B. AM: An Artificial Intelligence Approach to Discovery in Mathematics as Heuristic Search. In Davis, R. and Lenat, D. B., editors, *Knowledge-based Systems in Artificial Intelligence*, McGraw-Hill, New York, 1982.

[81] Lesser, V. R. and Erman, L. D. Distributed Interpretation: A Model and an Experiment. *IEEE Transactions on Computers*, C-29:1144-1162, 1980.

[82] Lesser, V. and Corkill, D. Cooperative Distributed Problem-solving and Organizational Self-design. *SIGART Newsletter*, (73):47, 1980.

[83] Lesser, V. R. and Corkill, D. D. Distributed Interpretation Testbed. *SIGART Newsletter*, (80):16-17, 1982.

[84] Lesser, V., Corkill, D., Pavlin, J., Lefkowitz, L., Hudlicka, E., Brooks, R., and Reed, S. A High-level Simulation Testbed for Cooperative Distributed Problem Solving. In *Proceedings of the Third International Conference on Distributed Computer Systems*, 1982.

[85] Lesser, V. and Corkill, D. Coordination in Distributed Problem Solving. SIGART Newsletter, (84):8-9, 1983.

[86] Lesser, V. R. and Corkill, D. D. Coherence in Distributed Problem Solving. *AI Magazine*, 6:235-236, 1985.

[87] Lesser, V. R., Durfee, E. H., and Corkill, D. D. Current DAI Research at the University of Massachusetts. In Sridharan, N. S., editor, Report on the 1986 Workshop on Distributed Artificial Intelligence. *AI Magazine*, 8(3):75-87, 1987.

[88] MacQueen, D. B. Models for Distributed Computing. Report No. 351, Laboratoire de Recherche en Informatique et Automatique, Domaine de Voluceau Rocquencourt, Le Chesnay, France, 1979.

[89] McArthur, D., Steeb, R., and Cammarata, S. A Framework for Distributed Problem Solving. In *Proceedings of the National Conference on Artificial Intelligence*, Pittsburgh, Pennsylvania, 1982.

[90] McArthur, D., Steeb, R., and Cammarata, S. Distributed Intelligence for Air Fleet Control. *SIGART Newsletter*, (84):5-6, 1983.

[91] Minsky, M. L. and Papert, S. A. *Perceptrons: An Introduction to Computational Geometry (Expanded Edition).* Cambridge, MA: The MIT Press, 1988.

[92] Monteiro, L. F. and Pereira, F. C. N. Outline of a Sheaf-Theoretic Approach to Concurrency. In First Annual Symposium on Logic in Computer Science, Cambridge, Massachusetts, 1986.

[93] Moses, Y. and Tuttle, M. R. Programming Simultaneous Actions Using Common Knowledge. In *Proceedings of the 27th Annual IEEE Symposium on Foundations of Computer Science*, Toronto, Ontario, 1986.

[94] Parunak, H. V. D. and Kindrick, J. A Connectionist Model for Material Handling. In Sridharan, N. S., editor, Report on the 1986 Workshop on Distributed Artificial Intelligence. *AI Magazine*, 8(3):75-87, 1987.

[95] Pollack, M. E. *Inferring Domain Plans in Question Answering.* Ph.D. thesis, University of Pennsylvania, 1986. Issued as Technical Note 403, Artificial Intelligence Center, SRI International, Menlo Park, California, 1986.

[96] Pollack, M. E. A Model of Plan Inference that Distinguishes between the Beliefs of Actors and Observers. In *Reasoning About Actions and Plans: Proceedings of the 1986 Workshop*, Morgan Kaufmann, Los Altos, California, 1987.

[97] Reddy, D. R., Erman, L. D., Fennell, R. D., and Neely, R. B. The Hearsay Speech Understanding System: An Example of the Recognition Process. In *Third International Joint Conference on Artificial Intelligence: Advance Papers of the Conference*, Stanford, California, 1973.

[98] Rice, J. P. Advanced Architectures Project. In Sridharan, N. S., editor, Report on the 1986 Workshop on Distributed Artificial Intelligence. *AI Magazine*, 8(3):75-87, 1987.

[99] Rich, E. *Artificial Intelligence*, McGraw-Hill, Inc., New York, 1983.

[100] Rosenschein, J. S. and Genesereth, M. R. Communication and Cooperation. Report No. HPP-84-5, Heuristic Programming Project, Stanford University, 1984.

[101] Rosenschein, J. S. and Genesereth, M. R. Deals Among Rational Agents. Report No. HPP-84-45, Heuristic Programming Project, Stanford University, 1984.

[102] Rosenschein, J. S. *Rational Interaction: Cooperation Among Intelligent Agents.* Ph.D. thesis, Stanford University, 1985.

[103] Rosenschein, S. J. Formal Theories of Knowledge in AI and Robotics. *New Generation Computing*, 3:345-357, 1985.

[104] Rosenschein, S. J. and Kaelbling, L.P. The Synthesis of Digital Machines with Provable Epistemic Properties. In *Theoretical Aspects of Reasoning About Knowledge: Proceedings of 1986 Conference*, Morgan Kaufmann, Los Altos, California, 1986.

[105] Rumelhart, D. E., McClelland, J. L., and the PDP Research Group. *Parallel Distributed Processing: Explorations in the Microstructures of Cognition, Volume I: Foundations*, MIT Press, Cambridge, Massachusetts, 1986.

[106] Shastri, L. Knowledge Representation and Inference in a Parallel Evidential Framework. *AI Magazine*, 6:238, 1985.

[107] Smith, R. G. *A Framework for Problem Solving in a Distributed Problem Solving Environment.* Ph.D. thesis, Stanford University, 1978. Issued as Technical Report No. STAN-CS-78-700 (HPP-78-28), Department of Computer Science, Stanford University, 1978.

[108] Smith, R. G. and Friedland, P. E. *Unit Package User's Guide.* Report No. DREA TM 80/L, Defence Research Establishment Atlantic, Dartmouth, Nova Scotia, 1980. Also Issued as Report No. HPP-80-28, Heuristic Programming Project, Stanford University, 1980.

[109] Smith, R. G. and Davis R. Frameworks for Cooperation in Distributed Problem Solving. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-11:61-70, 1981.

[110] Smith, R. G. CNET: Simulation of Distributed Problem Solving. *SIGART Newsletter*, (80):16, 1982.

[111] Smith, R. G. Report on the 1984 Distributed Artificial Intelligence Workshop. *AI Magazine*, 5:234-243, 1985.

[112] Sridharan, N. S. Report on the 1986 Workshop on Distributed Artificial Intelligence. *AI Magazine*, 8(3):75-87, 1987.

[113] Steeb, R., Narain, S., Cammarata, S., and Giarla, W. Cooperative Intelligent Systems. *AI Magazine*, 6:225, 1985.

[114] Stefanuk, V. L. Collective Behaviour of Automata and the Problems of Stable Local Control of a Large-scale System. In *Second International Joint Conference on Artificial Intelligence: Advance Papers of the Conference*, London, England, 1971.

[115] Stefik, M. J. *Planning with Constraints.* Technical Report No. STAN-CS-80-784 (HPP-80-2), Department of Computer Science, Stanford University, 1980.

[116] Stuart, C. J. A New View of Parallel Activity for Conflict Resolution. In *Reasoning About Actions and Plans: Proceedings of the 1986 Workshop*, Morgan Kaufmann, Los Altos, California, 1987.

[117] Sutton, R. S. Learning Distributed, Searchable, Internal Models. In *Distributed Artificial Intelligence Workshop*, Sea Ranch, California, 1985.

[118] Thorndyke, P., McArthur, D., Cammarata, S., and Steeb, R. Distributed Problem Solving in Air Traffic Control. *SIGART Newsletter*, (80):17-18, 1982.

[119] Turner, D. B., Burns, R. D., and Hecht, H. Designing Micro-based Systems for Fail-safe Travel. *IEEE Spectrum*, (February):58-63, 1987.

[120] Wesson, R., Steeb, R., and Hayes-Roth, F. Distributed Intelligence for Air Fleet Control. *SIGART Newsletter*, (73):47-48, 1980.

[121] Wesson, R., Hayes-Roth, F., Burge, J. W., Stasz, C., and Sunshine, C. A. Network Structures for Distributed Situation Assessment. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-11:5-23, 1981.

[122] Worrest, R. W. and Sandell, H. S. K. DAI Research at GTE Labs: Cooperative, Time-constrained Problem Solving. In *Distributed Artificial Intelligence Workshop*, Sea Ranch, California, 1985.