A Survey

of

Distributed Artificial Intelligence

Alan H. Bond

Concurrent Computation Program
California Institute of Technology
Pasadena, California 91125

and

Les Gasser

Distributed Artificial Intelligence Group
Computer Science Department
University of Southern California
Los Angeles, CA 90089-0782

August 1988

1 Introduction

1.1 What is DAI?

Most Artificial Intelligence research investigates how a single agent can exhibit intelligent behavior - solving problems using heuristic or knowledge-based methods, planning, understanding and generating natural language, and so on. Several recent developments together have provoked interest in concurrency and distribution in AI: the development of powerful concurrent computers, the proliferation of multi-node computer networks, and the recognition that much human problem solving and activity involves groups of people. *Distributed Artificial Intelligence* (DAI) is the subfield of AI concerned with coordinated, concurrent action and problem solving.

On the basis of the historical interests of researchers in DAI [Davis80, Davis82, Fehling83, Smith85a, Gasser87b, Sridharan87] we will divide the world of DAI into two primary arenas. Research in *Distributed Problem-Solving* (DPS) considers how the work of solving a particular problem can be divided among a number of modules or "nodes" which cooperate at the level of dividing and sharing knowledge about the problem and the developing solution [Lesser87, Smith81b]. In a second arena, which we will call *Multi-Agent* (MA) systems, research is concerned with coordinating intelligent behavior among a collection of (possibly pre-existing) autonomous intelligent "agents:[1]" how they can coordinate their knowledge, goals, skills, and plans jointly to take action or solve problems. The agents in a multi-agent system may be working toward a single global goal, or they may be working toward separate individual goals which interact. Like modules in a DPS system, agents in a multi-agent system must share knowledge about problems and solutions. But they must also reason about the *processes of coordination among the agents.* In multi-agent systems the task of coordination can be quite difficult, for there may be situations (in so-called *open sytems*) where there is no possibility for global control, globally consistent knowledge, globally shared goals or success criteria, or even a global representation of a system [Hewitt85, Hewitt86].

Research in a third arena, related to DAI, which we will call *Parallel AI* (PAI), is concerned with developing parallel computer architectures, languages, and algorithms for AI. These are primarily directed toward solving the performance problems of AI systems, and not with conceptual advances in understanding the nature of reasoning and intelligent behavior among multiple agents. Earlier, PAI was not highly differentiated as a subdiscipline separate from DAI (cf. [Davis80, Davis82, Fehling83]). However, for the most part, we will be only minimally concerned with Parallel AI in this survey (and in the book as a whole), consistent with more recent specializations in the field. We will focus on approaches to the problems of distributing and coordinating knowledge and action in distributed problem-solving and multi-agent systems, and this is what we shall term "DAI." It is important to note, however, that developments in concurrent languages and architectures may have profound impacts on DAI system architectures, reliability, knowledge representation, etc.

Another dimension for differentiating systems with different types of concurrency is how much of

[1] For most of this survey we will rely upon a simple and intuitive notion of an *agent* as a computational process with a single locus of control and/or "intention". But this view is actually very problematic: Agents may be implemented using concurrent subprocesses [Agha86a, Gasser87d], they may have multiple and conflicting goals, and the nature and reality of the concepts "goal" and "intention" are unclear (cf. [Cohen87a, Suchman87]). The process of defining the boundaries of what comprises an agent interacting with a world is also fraught with difficulties, from the standpoint of epistemology, social psychology, and, we think, from the standpoint of computer science. See, for example, [Bentley54, Dewey16, Mead34]

the reasoning about problem solving and coordination is done by system developers, and how much is done by the system itself; how adaptive can the system itself be to changes in the problem or context? Parallel AI systems may be able to adapt to temporal uncertainty (e.g. indeterminate computation speeds) but not to alternative solution paths or the loss of problem-solving knowledge. A DPS system may be adaptive to uncertainty in problem solving knowledge [Lesser81] but not to alternative problem contexts or changing problem-solving roles for modules. A multi-agent system may be able to form and restructure coordination frameworks based on emerging contexts and changing problem-solving roles without the intervention of a programmer.

Research in Distributed AI promises to have wide ranging impacts in cognitive science (e.g., mental models, social cognition) distributed systems (reasoning about knowledge and actions in distributed systems, architectural and language support for Distributed AI), human-computer interaction (task allocation, intelligent interfaces, dialogue coherence, speech acts), basic AI research (problem representations, epistemology, joint concept formation, collaborative reasoning and problem-solving), and the engineering of AI systems ("cooperating expert systems", distributed sensing and data fusion, cooperating robots, collaborative design problem-solving, etc.). But research into DAI is also attractive for more fundamental reasons: in order to coordinate their actions, intelligent agents need to represent and reason about the knowledge, actions, and plans of others. Distributed AI research can help us improve our techniques for representing and using knowledge about beliefs, action, plans, goals, etc. [Davis80]. Moreover, others have suggested that DAI may draw from and contribute to other disciplines, both absorbing and providing theoretical and methodological foundations [Chandrasekn81, Lesser83, Wesson81]. "Fields of study heretofore ignored by AI - organization theory, sociology, and economics, to name a few - can contribute to the study of DAI. Presumably, DAI will advance these fields as well by providing a modeling technology suitable for precise specification and implementation of theories of organizational behavior" [Wesson81, pg. 18].

1.1.1 A Methodological Note

From a methodological perspective, virtually all extant research in DAI has focused on how a collection of agents can interact to solve a single common "global" problem, such as designing a VLSI chip, devising a globally consistent interpretation of geographically distributed sensor data, or constructing a globally coherent plan for several agents. The range of problems and the objects of study appropriate to DAI research is actually much wider than this, and a larger conceptual frame should be brought to bear in analyzing the problems and possibilities of DAI. There are three classes of objects of study for DAI, all of which comprise coordinated aggregates of intelligent agents. A *natural systems* approach to Distributed AI would study the strategies and representations that people use to coordinate their activities, in much the same way that cognitive scientists investigate individual cognition in people. This approach would, of course, include computer modeling and simulation of the coordination activities of people. An *engineering-science* perspective on Distributed AI would investigate how to build functioning, automated, coordinated problem-solvers for specific applications. In many cases, the problems of coordination would be simplified (over those in multi-person aggregates) by using standard communication protocols and relying upon global viewpoints. A *person-machine coordination* approach would be useful in analyzing and developing collections of people and machines working together in coordinated ways. Within this approach, research in DAI is already seeding research and practice in the areas of computer supported cooperative work and office automation [Barber80, Barber82, Barber83, de Jong88, Croft88, Fikes82, Hewitt86, Nirenberg86].

In each case, it may be tempting to carve away the study and understanding of people interacting. But this can be difficult. In analyzing person machine coordination, for example, it is clearly necessary to account for the person to person interactions in the workplace to understand how people work one-on-one with machines [Gasser84, Gasser86, Suchman87]. Complete analysis of engineered DAI systems may also demand consideration of which conflicts are to be settled by people "outside" the system and which are handled "autonomously." The nature of the boundaries dividing "the system" from "the environment" are analytically problematic.

This survey is organized around a set of basic questions for DAI applicable to all these approaches. Nonetheless, the primary focus of this survey and book is distributed *artificially* intelligent systems, examining what is currently known and what is not known about answers for these questions. In this way, we hope to allow scientists and engineers to appreciate the open problems and the currently accepted wisdom in the field.

1.1.2 Historical Antecedents of Distributed AI

Distribution and parallelism have historically been important themes in artificial intelligence. The conceptual basis for concurrent processes underlying artificial intelligence has been well established and motivated by many, from the inception of AI in the early fifties, up to and including Minsky's vision of the mind as a society of cooperating agents [Minsky79, Minsky80, Minsky86] and Arbib's conception of the brain's information processing as a collection of concurrent "schemas" [Arbib85, Arbib88]. During the fifties, there were two main approaches to AI - heuristic search using list processing methods on the one hand and neural net modeling on the other. Neural models reached conceptualized forms in Selfridge's Pandemonium [Selfridge59] and Rosenblatt's Perceptron [Rosenblatt62]. The computational structures used were layers of "neurons", the processes they modeled were usually pattern recognition tasks, and the main programming method was synaptic learning. There were some theoretical results in learning theorems. Hybrid analog and digital nets were explored in probability state variable (PSV) systems [Lee63]. Feedback in neural nets was investigated by Aleksander [Aleksander75], who also generalized the systems to networks of RAM memory elements. Unfortunately, the lack of sufficiently powerful programming or learning methods, theoretical intractability, but principally implementation difficulties, made such work difficult. It has been taken up anew in the present implementationally more fortunate era, as *Parallel Distributed Processing* [McClelland86] and *Connectionism* [Feldman82].

Over time, serial list processing problem solving methods were generalized into quasi-parallel symbol processing models, which began with the pattern-directed control of Planner [Hewitt71], and with the production system approach of Newell and Simon [Newell72] in the late sixties. The pattern directed approach of Hewitt allowed separate processes, triggered associatively off a global database of symbolic structures. The parallelism was motivated by the advantages of separating knowledge from inference, and of modularizing knowledge into identifiable and meaningful parts. Each part had a specified problem solving method and also a specification (in the form of a symbolic pattern) of the problems to which it was applicable.

Processes could not communicate easily or retain failure information in the Planner model, and it was this observation that led to its generalization in Conniver [Sussman72]. In Conniver, processes could suspend themselves and be reactivated. Each concurrent process was seen as an expert on some small area of activity. Conniver was not widely used or developed (like many experimental

systems in AI), but it influenced later systems.

The production system approach of Newell and Simon took a different tack. Here, the units of processing - production rules - did not correspond so well to units of knowledge. Rather they corresponded to primitive symbol manipulation processes from which active data structures or higher-level processes could be built. The left hand side of a production rule was a conjunction of primitive tests, and the right hand side was a sequence of primitive actions. The rules operated "in parallel" in the sense in that they associatively monitored a common database and fired when a successful match became possible. Production systems were intended to facilitate the discovery of new representational structures for psychological modeling. Unfortunately, in spite of the development of several different basic production rule architectures by Newell, and the representation of many existing AI systems and the development of novel AI systems in production system form, the approach has not directly yielded the intended insights, although it has been influential in many other ways.

Blackboard systems evolved from the production system approach. In a Blackboard system, a set of processes typically called *knowledge sources* (KS's) share a common database or *Blackboard* of symbolic structures often called (and indeed denoting) *hypotheses*. Each KS is an expert in some area, and may find a hypothesis it can work on, solving it, creating new hypotheses and modifying other existing hypotheses. The set of processes thus cooperates by sharing the common blackboard, rather like a group of human experts, each endowed with expertise and a piece of chalk, and using a common chalkboard. Conventional blackboard systems, exemplified by Hearsay-II and its descendents [Erman80, Hayes-Roth85, Nii86b, Nii86a], were not truly parallel, since they ran on serial machines, and since they incorporated schedulers which assured sequential invocation of knowledge sources to maintain blackboard and knowledge source consistency. However they can be called quasi-parallel, since the best conceptual model of their operation is quite parallel, and since they rely upon semantically distributed conceptions of knowledge.

These three architectures are precursors but not exemplars of contemporary DAI. In post-Hearsay research, Lesser and Erman suggested dividing the interpretation problem among *several* Hearsay-like blackboard systems, and they studied several possible task decomposition regimes [Lesser80a]. By the late 1970's, work on Actors and the PUP6, Contract Net, DVMT, and Rand ATC systems had begun, and the first phase of research into DAI as we now know it was in full swing [Corkill79, Davis80, Hewitt77a, Lenat75, Smith79, Lesser81, Steeb81].

In the current AI research which is explicitly concerned with concurrent models of intelligent behavior, there are two main currents. *Distributed Artificial Intelligence* research addresses the problems of designing and analyzing large-grained coordinated intelligent systems. *Connectionism* research is devoted to explaining higher mental functions and higher-level reasoning processes by reference to highly parallel collections of processes made up of very simple computing elements. This paper will not cover research in fine-grained Distributed AI or Connectionism. For an introduction to the latter, see [Feldman82, McClelland86].

1.2 Exemplary Systems

In this section we give a brief overview of several exemplary DAI systems, so that the reader may gain some basic and intuitive familiarity with standard DAI concepts and experience. We will draw

on the ideas embodied in these systems to illustrate many of the problems and solution strategies of DAI, and to set the stage for more complete definitions of some concepts later. We illustrate three approaches here - a collection of *blackboard-based* problem solvers called the *Distributed Vehicle Monitoring Testbed (DVMT)*, a system called the *Contract Net (CNET)* which exploits flexible distributed task allocation, and an approach to multi-agent computation and reasoning in *open systems* based on the *actor model*. These differ along dimensions of flexibility, adaptability, and degree of assumed prior organization among computing agents.

1.2.1 The Distributed Vehicle Monitoring Testbed (DVMT)

The DVMT is a major ongoing project at University of Massachusetts at Amherst [Lesser81, Corkill82, Lesser83, Durfee87a, Durfee87d, Durfee87e]. This system provides an example application in which both perception and knowledge are distributed. The DVMT uses collection of identical, blackboard-based systems (see Section 1.1.2) to solve problems of monitoring and interpreting data from a set of sensors at spatially distributed but fixed locations. Data (which may be noisy or redundant) from some subset of the sensors covering the entire region is routed to each problem-solving node. The nodes then cooperate to construct a global picture of vehicle traffic across the whole sensor net.

In the DVMT experiments, problem-solving performance or *coherence* has been calculated in terms of efficiency of the system measured by simulated clock time to solution. For each experiment, an optimal solution time is calculated, based on each node having complete global data at all times and always making the correct choice. When sensor regions overlap, or when signals are noisy, the nodes may have incomplete or inconsistent data, and performance can degrade due to redundant or misguided work. The research goal is to achieve performance as close to the the optimal level as possible, with incomplete or noisy data. To gain efficiency, UMASS researchers have found that they need to enhance both *meta-level control* among nodes and *local control* within nodes. That is, individual nodes should have knowledge about their own problem solving behavior and about the behavior of other nodes and the system as a whole.

Three kinds of meta-level control have been explored:

- *Exchanging meta-level information* - one node explains its actions to another by exchanging information about how or why it has acted, abstractions of its state, or high-level solutions it has generated.
- *Planning* - a node creates a plan for its activity, which can be communicated to others. Ultimately, the nodes build and exchange *partial global plans* which are shared, partially complete views of joint problem-solving activities, giving each node greated insight into the roles and intentions of its collaborators.
- *Organization* - Some organization can be impressed upon the system in the form of *problem-solving roles* for each node. In the DVMT, a role is a restriction on the capability or focus of a node, so that the node does *not* perform certain actions or look at certain areas of data. This yields a division of responsibility and hence a division of labor among the nodes.

For different problems or different states of problem-solving, different node organizations may be needed. Hence, the DVMT experiments have begun to address the problem of *organization self-design* [Corkill82]. For each new problem, the system needs ways of recognizing and creating the appropriate set of roles. The DVMT project has begun (essentially experimental) research into *organizational knowledge*, namely how to associate a particular collection of roles with a particular type of problem and level of performance.

Making a transition from one role organization to another in the course of problem-solving requires recognizing and characterizing problem-solving situations, and instantiating new organizational forms in a decentralized manner. As a part of organizational self-design, a group of problem-solving nodes must be able to diagnose and observe its own behavior, in order to tell how well it is working, so that it can know when to adapt and how to adapt, and so it can recognize when certain sensors or problem solvers are disabled. Toward this end, the DVMT experiments have investigated how to model and diagnose system behavior, using expected problem-solving trajectories as a basis for reasoning.

The project's principal contributions to the DAI field include:

- The study of interaction among complex individual problem solvers, each of which is a powerful Hearsay-like system.
- Investigations of *functionally accurate, cooperative* (FA/C) systems, which converge on solutions despite incomplete and possibly inconsistent data, as distinct from systems which maintain complete global consistency. The DVMT system is designed to converge on the best acceptable answer, even though it consists of a set of asynchronous evocations of possibly inconsistent knowledge sources.
- Enhancements to the sophistication of local control in blackboard systems (e.g. integrated goal directed and data directed control).
- Mechanisms for *Meta-level control* through communication, individual and joint planning, and organizational roles.
- Approaches to *organization self-design.*
- Preliminary studies of how to abstract, model and diagnose distributed system behavior.
- Careful, thoughtful experimental methodologies and controlled experiments to test the performance of control enhancements.

1.2.2 The Contract Net (CNET)

The Contract Net was a distributed problem solving system designed by R. Davis and R. G. Smith [Davis83, Smith80, Smith81a, Smith81b]. The primary goal of this system was opportunistic, adaptive task allocation among a collection of problem-solvers, using a framework called "negotiation", based on task announcements, bids, and awarded "contracts". The basic architecture comprised nodes with *manager* and *worker* roles; any node could be at any one time both a manager and a worker for different tasks. A manager for a task:

- Decomposed its task into subtasks. Task decomposition was given, i.e. the task was described in a way that made it already decomposable.
- Constructed task announcements for each subtask and distributed them either by broadcasting to all nodes or by focused addressing to a selected subset of nodes.
- Selected the most appropriate bid and allocated the subtask to that bidder.
- Monitored progress on the contract, possibly requesting information, interim reports etc. and was free to reallocate the subtask if the contractor failed to complete it in a timely manner.
- Integrated the partial results produced by its contractors into a complete solution at its level of decomposition.

Worker processes which received task announcements evaluated their own eligibility and returned bids to do the subtask if interested. They then performed subtasks allocated to them, provided interim reports on progress, partial results, etc., and reported results produced from performing subtasks to the manager of the subtask.

The Contract Net Protocol (CNP) was designed to solve the problem of *opportunistic connection* between managers and workers. Nodes could exchange procedures and data to augment their capabilities during problem-solving, to provide incrementally better fit of workers to tasks. However, the CNET included no general *problem solving protocol*; issues such as meta-level control or measures of *global coherence*[2] were outside the scope of the protocol. What coherence existed was a product of eligibility and mutual selection criteria, which were purely local. Because of this, the Contract Net could deal reasonably well with the kind of problems which are easily decomposable into independent subproblems, and where the mix of subproblems matched the availability of worker nodes. If decomposition was difficult, or there was interaction between subtasks, it did not work well. There was no provision for lateral communication between subtasks or for global optimization via changes in roles. and little mechanism for explicit reasoning about the global impacts of local decisions or possible harmful interactions among nodes. In the work of Davis and Smith, the CNET was applied to the N-Queens problem (treated as distributed search) and to a distributed sensor net task [Davis83]. Others have extended the CNET framework to real-time task scheduling and manufacturing control [Parunak87, Ramamritham85].

Some major innovations of the Contract Net project were:

- The study of distributed task allocation, by a process of mutual selection and agreement. This scheme is different than a manager-centered invocation (e.g. procedure calls) or worker-centered invocation (e.g. data-driven computations) in that it relies on mutual selection by both manager and worker processes.
- A differentiation between task sharing and result sharing, where task sharing is sharing the labor on tasks to be done, and result sharing is combining partial results from several agents into an aggregated solution.
- Control based upon mutual commitments or "contracts" among agents.
- The design of a communication protocol for one style of interaction termed "negotiation".

[2]Coherence is discussed in Section 4 of this survey.

1.2.3 Actors and Open Systems

The study of Actors and Open Systems has been an important sustained research project at MIT since 1973, lead principally by Carl Hewitt [Agha86a, Agha86b, Clinger81, Hewitt77a, Hewitt77b, Hewitt84a, Hewitt86]. Hewitt raised a broad research question in his early Actors work [Hewitt77a]: What should be the communication mechanisms and conventions of civilized discourse for effective problem-solving by a society of experts? Newell had pointed out that in conventional single-agent problem-solving paradigms a problem-solving agent "appears to be wandering over a goal net much as an explorer wanders over the countryside, having a single context and taking it with him wherever he goes". The single-agent view leads researchers to focus on the internal processes of search or reasoning processes with a single locus of control and focus of attention. This, too, has led to preoccupations with organizational mechanisms such as goal stacks and agendas for making control decisions and changing contexts.

Hewitt's basic insight was that, rather than viewing a control structure as a sequence of choices made by a single decision maker in a web of choice points, it could instead be seen as a pattern of messages passing among a collection of computational agents he called "actors". Using this model, the structure of control immediately became parallel, making the actors concept a convenient opportunity to exploit the growing interest in parallel computer architectures.

Kornfeld and Hewitt also suggested the scientific community as a model of problem-solving. Problem-solving could then mimic how scientists build, communicate, test, and revise theories. (Lenat had by this time already called for problem-solving via a "committee of experts" and constructed his prototype PUP-6 system [Lenat75].) Recognizing that the actual work of scientists has much in common with other human work (cf. [Latour87]) Hewitt has since expanded his view to draw upon and develop theories of problem solving work in many human organizations [Hewitt86].

An actor system comprises two parts: the actors, which are the only computational entities, and an operating system which creates and destroys actors, and passes messages (actually other actors) among them using the universal communication mechanism called *actor transmission*.

The definition of each actor comprises two parts, as well: a *script* which defines the behaviors of an actor upon receipt of a message, and a finite set of *acquaintances* which are the other actors known to the actor. Events in the actor model consist solely of the arrival of actor messages; the internal state of an actor is invisible outside it. When a message arrives, an actor may take three types of actions: it may change its local state (if it does this it is called a *serialized actor*, because its state depends on the order in which messages are received), it may create new actors, or it may send messages to other actors.

Actor models face, (as do other DAI approaches) the issues of coherence - the actor community must not degenerate into a society of bureaucrats which sends many messages but makes little progress. Issues of actor design, decisions on granularity of data and control representations, composition of actors into larger communities, constraints on actor behaviors, achieving higher-level performance goals with only local knowledge, etc. which are characteristic of other DAI approaches still appear in actor systems and have yet to be addressed comprehensively. In addition, realistic actor communities depend upon a comprehensive *message delivery (addressing) layer*, which presents difficulties in a large, dynamic, possibly open system.

Hewitt and his associates are also interested in actors as an underlying model for *due process reasoning* in *open systems* [Gerson86, Hewitt84a, Hewitt85, Hewitt86]. Hewitt notes that many real-world distributed systems (e.g. office work, inter-bank transactions, large distributed databases such as a group of libraries, etc.) exhibit characteristics of 1) mutual inconsistency in knowledge or belief, 2) bounded influence or "arms-length relationships" among components, 3) asynchronous operation, 4) concurrency, and 5) decentralized control. He calls these systems *open systems*. Reasoning and coordinated action in open systems depend upon using debate and negotiation to mediate among "microtheories" - internally consistent but globally inconsistent sets of facts or beliefs.

The main features of Actor systems are:

- The ability for a society of experts or actors to achieve intelligent behavior;
- A theory of computation which supports fine-grained and naturally concurrent computation;
- Control structures as patterns of messages rather than as sequential choice among alternatives;
- Non-serialized actors, as opposed to serialized actors, having a local state which persists on reinvocation.

Open Systems are systems which:

- are composed of independently developed parts in continuous evolution;
- are concurrent, asynchronous, and have decentralized control based on debate and negotiation;
- exhibit many local inconsistencies;
- consist of agents with bounded knowledge and bounded influence;
- have no fixed global boundaries visible to the agents comprising the system.

1.3 Rationales for Distributed AI

Elements of an intelligent system are distributed if there is some *distance*[3] between them, and where some significant cost and/or some intermediary process is entailed in connecting them. There are many reasons for distributing intelligence. These include classical reasons for distributing any program or system, and reasons particular to Distributed AI systems. In some domains where AI is being applied, (e.g., distributed sensing, medical diagnosis, air traffic control) knowledge or activity is inherently distributed. The distribution can arise because of geographic distribution coupled with processing or data bandwidth limitations, because of natural functional distribution in a problem, or because of a desire to distribute control, e.g., for fail-soft degradation, modular knowledge acquisition, etc. In other domains, we may expect to simplify the development and evolution of intelligent systems by building them as collections of separate but interacting parts. Typical rationales for distributing an AI system include:

[3]We mean a *conceptual distance*, with respect to some conceptual frame such as time, space, semantics, etc. - see below.

- **Adaptability:** Logical, semantic, temporal, and spatial distribution allows a DAI system to provide alternative perspectives on emerging situations, and potentially greater adaptive power.
- **Cost:** A distributed system may promise to be cost-effective, since it could involve a large number of simple computer systems of low unit cost. If communication costs are high, centralized intelligence with distributed perception or sensing may be more expensive than distributed intelligence.
- **Development and Management:** If an intelligent system can be built in a distributed way, each part could be developed separately by a specialist in a particular type of knowledge or domain. A distributed intelligent system may be extensible [Rosenschein85a].
- **Efficiency or Speed:** Concurrency may increase the speed of computation and reasoning (but concurrency must be traded off against problem dependent coordination overhead).
- **History:** One might already historically have a basis for a distributed system. This could be a collection of existing resources, such as a network of control computers on an aircraft, or a collection of workstations and people in an office or industrial environment. It could also be an existing set of facilities or software processes, (e.g., expert systems) which one would like to integrate.
- **Isolation/Autonomy:** For protection and local control, parts of a system may be separated and isolated from each other; sometimes termed "arms-length relationships."
- **Naturalness:** Some problems are better described as collections of separate agents; there is a "better fit" to a problem or domain because elements are "naturally" distributed along some of the axes of distribution discussed above [Rosenschein85a].
- **Reliability:** Distributed AI systems may be more reliable than centralized systems by providing redundancy, cross-checking, and triangulation of results [Mason88].
- **Resource Limitations:** Individual computational agents have bounded rationality, bounded resources for problem solving, and possibly bounded influence, necessitating cooperation and coordination to solve large problems.
- **Specialization:** Knowledge or action may be collected in specialized, bounded contexts, for purposes of control, extensibility, comprehensibility, etc.

Distributed AI systems present quite difficult problems of analysis and development, in part because the following aspects can be independently distributed:

- **Action-in-the-World -** Simultaneous actions may take place at different physical locales, or actions may occur at different points in time.
- **Focus of Attention** - Different agents may focus on different aspects of the world.
- **Authority** - Different agents may have different levels of authority or responsibility for different aspects of a situation.

- **Credibility** - The support for some conclusion may be distributed, or different agents may be differentially trustworthy or credible.
- **Interpretations** - Events and objects may mean different things to different agents.
- **Knowledge** - Different agents may have different knowledge, or no agent may hold a complete representation of some important collection of knowledge, a situation, or a problem.
- **Perception** - Different agents may perceive different events or objects, giving them partial or incommensurate views.
- **Reliability** - Different agents may be differentially reliable, and overall system reliability may be contingent on some collection of subsystems.
- **Resources** - Different agents may have different resources or levels of resources.
- **Value** - Different agents may place different values upon resources and the outcomes of actions.
- **Work** - Different agents may perform different parts of a task.

We may define the conceptual "distances" which define distribution (and which describe *how* the above aspects can be distributed) in several ways:

- **Computation Cost:** The cost of using knowledge or drawing on a specialized skill (measured in time, space, or other resources) may vary depending upon its location, and so cost of use is a way of defining a distribution.
- **Spatial Distance:** The spatial location of processes, knowledge, sensor input and action effectors defines an axis of distribution. Both physical and human information processing depends upon sensing, input, transduction, etc. of data. This occurs at spatially distributed locations, at different times, and on different and therefore distributed devices. There is a cost, often a large cost, involved in moving this input data to a single central point in the system for processing. Similarly, the *action* taken by an intelligent system typically is taken upon distributed systems, distributed knowledge-bases and spatially distributed actuators.
- **Temporal Distance:** Events and knowledge may be temporally distributed in that they appear or become inaccessible through time. Some knowledge may not be available at a given time, because it has not yet been produced or derived. Whether knowledge can be pre-compiled and stored or must be deduced is a matter affecting temporal distribution and the utility of the knowledge; storage is one intermediary which makes knowledge accessible through time. Deduced knowledge is "less local" temporally and in terms of costs and resources required. Thus the form and content of existing knowledge bases affects choices about distribution. Especially in real-time systems, there may be costs for waiting or rushing for temporal coordination.
- **Logical or Deductive Distance:** The degree of logical dependency between items of knowledge (i.e., the need to rely upon intermediary deductive processes or supporting facts to make some new knowledge accessible) is a basis for distribution or aggregation. Logical dependency is often exploited in parallelizing production systems, for example.

- **Semantic Distance:** People cluster knowledge into specializations pertaining to the conventionalized and practical uses of knowledge. Thus there may be a knowledge cluster for homeopathic medicine and another for scientific medicine, or one for plumbing and another for carpentry, etc., which comprise different and incompatible knowledge and representations. Knowledge is sometimes sparse between conventional knowledge clusters; *translation processes* provide intermediaries. The clustering of knowledge items and their access and use within a set or language of knowledge description expressions may be a cause for distribution.

1.4 Basic Problems of Distributed AI

The basic problems of Distributed AI are:

- How to formulate, describe, decompose, and allocate problems and synthesize results among a group of intelligent agents;
- How to enable agents to communicate and interact: what communication languages or protocols to use, what and when to communicate, etc.
- How to assure that agents act coherently in making decisions or taking action, accommodating the global impacts of local decisions and avoiding harmful interactions.
- How to enable individual agents to represent and reason about the actions, plans, and knowledge of other agents in order to coordinate with them; How to reason about the state of the coordinated process (e.g. initiation and termination).
- How to recognize and reconcile disparate viewpoints and conflicting intentions among a collection of agents trying to coordinate; How to synthesize views and results.

Each of these problems appears in some form in all DAI application domains, though some of them (e.g., some problems of disparate reference frames) can be "designed out" in some engineered DAI systems. The solutions to these problems partly depend on features of particular application domains, and part of the task of DAI research involves determining what are the domain-dependent features and how they affect the answers to these questions. Ultimately, we would like to understand how a DAI system can reason about solving many of these problems itself, because one goal of a multi-agent system is adaptive self-configuration.

In the remainder of this survey we consider each of these problems in greater detail, and discuss tools and languages for constructing DAI systems.

2 Task Description, Decomposition, Distribution and Allocation

2.1 Introduction

Chandrasekaran [Chandrasekn81] observes that "Social organizations from honeybee colonies to a modern corporation, from bureaucracies to medical communities, from committees to representative democracies, are living examples of distributed information processing embodying a variety of strategies of decomposition and coordination." When work is to be done by a collection of agents in some coordinated fashion, it is important to answer the question of division of labor and organization: *which agent does what task, when?*. A distribution of tasks among agents requires that the tasks be *formulated* and *described* in ways which account for their distribution or allow them to be distributed. Tasks which require more resources or knowledge than is possessed by one agent must be *decomposed* so that they can be accomplished. Tasks must be *allocated* - assigned to particular agents which will actually perform them. All of these activities are deeply interdependent. The languages and concepts used for task description and formulation will affect how tasks can be decomposed into parts, and what dependencies explicitly exist among tasks; the same task described from different perspectives may require different partitioning and different skills. Choices of decomposition affect how tasks can be allocated, because the skills and knowledge of agents who will perform tasks must eventually match the task requirements [Lesser80a]. Task decomposition is affected by dependence among tasks, exchange of messages, adjacency requirements and dynamically changing patterns of dependency. Decomposition decisions may need to be revised with contextual changes. Resources available for allocation may affect distribution and aggregation choices. Finally, the work of making these decisions is itself a task which autonomous (collections of) agents must address; in the extreme, they will need to reason about their representations and formulations of problems and the decomposition/allocation processes they use. In multi-agent systems in particular, agents may need to negotiate over the appropriate formulation of problems and appropriate responsibility for the description, decomposition and allocation decisions, in order to form and maintain problem-solving alliances. In such cases (i.e. without a global view), problems may be *constructed, not decomposed*. Agents need to make decisions about problem descriptions, the appropriate boundaries for problem *elaborations*, and *aggregation and combination of tasks into clusters*. A complete treatment of these problems for DPS and MA systems would need to treat each of these activities as potentially dynamic tasks which themselves had to be allocated and settled by collections of agents.

There has been remarkably little work in DAI which addresses automated problem formulation and decomposition. Correspondingly greater effort has been put toward flexible task allocation mechanisms *after* a problem has been described and decomposed into subproblems. The Contract Net and its successors all have addressed flexible opportunistic *allocation* of tasks; decompositions were provided by programmers. Similarly in the DVMT the descriptions, partitioned knowledge for data interpretation, and regional responsibilities (input data) have been generated by designers; the system itself makes semi-autonomous opportunistic allocation decisions on which nodes perform which particular aggregation tasks. Actor systems, too, have thus far treated allocation decisions as dynamic, but not description or decomposition decisions (e.g. [Kornfeld81b]).

2.2 Description

Decomposition choices are critically dependent on how a problem is described, because it is the collection of attributes and descriptive categories for stating the problem which provide a language for expressing inter-problem and inter-agent dependencies. In real settings, formulation of problems requires some *representation* for the problem, as well as decisions on the boundaries of the problem and what is known and unknown [Gasser88a]. In a conventional DPS system many of these activities are carried out by designers; in MA systems, they may well be done by the agents themselves. This may require the effort of elaborating a problem to cover all aspects; all problem description knowledge may not be available to one agent, and, especially in a dynamic setting, a problem description might be collaboratively (and even dynamically) constructed. While there is very little extant research in this area in DAI (but see [Hinke88]), some relevant research has occurred in knowledge acquisition systems [Lefkowitz87]. We regard this as a very fruitful area for further research.

2.3 Decomposition

The problem of task decomposition can be seen from several perspectives. In typical decomposition processes, a single "supertask" is decomposed into smaller subtasks, each of which requires less knowledge, or fewer resources. In general, there should be a choice among alternative task decompositions, depending upon the ability of agents to actually perform tasks. Alternative decompositions are sometimes conventionally obtained by alternative problem reduction operators, corresponding to an OR branch in a goal graph. Further types of decomposition can be obtained by transformation of problem representation [Amarel68, Simon69]. There is however more to the problem of task decomposition than this. Most decomposition processes flow directly from the descriptions of available operators for problem-solving, indicating that they actually depend upon designers' forethought in operator construction and description. Key research questions include how we construct or select a set of operators for the task set to be produced, and how we construct and decompose problems so as to minimize the costs of knowledge distribution and resource distribution?

The decomposition problem for multiple agents is more complex because of the need to match resources and capabilities of different agents with appropriate tasks; there must be sufficiency of knowledge, resources, and control in the overall system to bring about an effective solution [Lesser80a]. This means that any decomposition regime will need to account for the capabilities and resources of agents, and hence to make decisions about alternative types and granularity of decompositions. Subtask aggregation - coalescing several tasks into one larger-grained unit - is one method of varying the granularity of tasks distributed to agents. It also becomes important in MA systems wherein multiple agents working without centralized control must aggregate their efforts for greater capability [Agha86a, Gasser87d], rather than decomposing a global problem.

Difficult problems of decomposition arise because of dependencies among subproblems and among the decisions and actions of separate agents. Conflicts over incompatible actions and shared resources may place ordering constraints on activities restricting decomposition choices and forcing a need to consider decomposition in a *temporal* (or other resource) dimension as well as the more usual dimensions of knowledge, location or abstraction dimensions (see below). When plans are

generated independently by several agents, there may be problems in reconciling subplans produced by each agent to reduce dependencies and conflicts [Cammarata83, Durfee87a, Durfee87c, Georgeff83, Steeb81]. In some circumstances, the issue of *redundancy* enters the decomposition problem. Choices about redundancy are related to the tradeoffs between efficiency and reliability; redundancy should be eliminated to improve efficiency, but may be necessary for reliability.

In summary, intelligent approaches to task decomposition need to consider the representation of tasks, several dimensions of decomposition, available operators that can be applied to perform subtasks, available resources, and dependencies among tasks.

2.3.1 Decision Bases

Several researchers have given their own views of the basis for problem decomposition. In their study of distributed interpretation problem solving, Lesser and Erman [Lesser80a] argued for three dimensions of problem decomposition. Problems could be decomposed along lines of location (e.g. spatial, temporal, logical), information level or degree of abstraction, and what they called "interest areas," which included the given partitioning of skills among knowledge sources. This work was the distribution basis for later experiments using the DVMT [Lesser83]. Rand researchers investigating air traffic control (ATC) (a domain in which geographic distribution is inherent) [Steeb81] proposed several bases for reasoning about how to decompose problems. These included solving the total problem from several different perspectives at once, solving geographically local problems locally, decomposing problems by spatial clustering, decomposing a problem by functional clustering (e.g. stage of flight), and decomposing a problem hierarchically, with one agent producing an abstract plan, and several low level agents doing local plan refinement. But what general basis do we have for making decisions on how to decompose tasks? The basis of all task decomposition approaches is to find find dependencies and logical groupings in problem tasks and knowledge. There are several bases for cleaving dependencies and decomposing tasks which have commonly appeared in the literature. These include:

Abstraction Level: Numerous researchers have used abstraction levels as a basis for decomposition [Sacerdoti77, Erman75, Erman80, Lesser83, Wesson81]. Decomposition of the problem into abstraction levels provides a simple task decomposition basis: level-specific tasks are generated. Abstraction levels become a way of defining particular functional groups. Gomez and Chandrasekaran [Gomez81] studied distributed medical problem solving, and suggested that specialists be hierarchically organized, by reference to abstractions of medical knowledge. Finally, Wesson [Wesson81], in an empirical study, found that hierarchical organization worked better than freely ("anarchic") communicating committees, in message puzzle tasks, and Benda et al. found that hierarchically controlled organizations worked more efficiently than fully negotiated ones for certain problem-solving tasks, though they did not study the effects of uncertainty or dynamism in the problem domain [Benda85]. From a task aggregation perspective, levels in hierarchies should be formed when global coordination can be improved using data abstracted for a lower level. However, extracting this information in the case of a particular problem is difficult. While it is very tempting to think we may be able to rely upon decomposition by abstraction level as a potential for automated problem decomposition, there remains the great difficulty of how to define, recognize, or construct these different abstraction levels in a problem.

Availability of Coordination: A problem can be decomposed when appropriate control mechanisms for that decomposition can be provided - decompositions should make successful control possible and control choice explicit or easy. By distributing control decisions, some control over the processing may lost, depending upon degree of communication, certainty, relative authority, and reliability of processes. It may be difficult to detect failure (cf. [Halpern87]), and it is usually impossible to detect deadlock without a global picture.

Control Dependencies: Tasks can be decomposed to reduce control dependencies. Along with bounded resources, individual agents may may have limited control (power or authority) over each other's actions. Hewitt terms this "bounded influence" [Hewitt84a]. In the DVMT the degree of interagent control can be set by varying how important a node rates internally versus externally generated tasks [Lesser83].

Data Dependencies: Relative data dependencies among tasks along axes of semantics, logical dependencies (cf. [deKleer86, Doyle79]), or temporal dependence can serve as a basis for decomposition choices. Tasks related by strong or numerous knowledge production and consumption relationships may be grouped together.

Data Partitioning: Tasks can be decomposed by taking account of natural or dependency-related partitions in input data. For example, in distributed sensor networks, spatial distribution of input data provides a natural basis for decomposition of tasks. Lesser and Erman [Lesser80a] describe the relation between data and time or space as a "location" dimension. It is also related to resource minimization and to product/function dimensions.

Functional/Product Division: Malone suggests (by analogy to human organizations) decomposing problems along either functional or product dimensions. *Functional decomposition* refers to grouping the classes of generic tasks needed to produce any product into individual clusters by task type. *Product decomposition* means grouping all tasks related to one product into one cluster, and dividing along multiple product lines [Malone87a, Malone88a]. Functional and product decompositions respectively correspond to the Rand functional and spatial dimensions for ATC problem-solving, and to abstraction level or interest area versus location dimensions of Lesser and Erman. Malone also provides a structural analysis of proposed theoretical benefits of different decompositions.

Interaction Levels: Structurally, greater numbers of subtasks and greater degrees of interaction among subtasks may require more intermediate control tasks. This is the classic *span of control problem* (cf. [Fox81, Malone87a, Malone88a]). For example, Wesson et al. suggest that higher interaction among sub-problems will increase the need for authority (and hierarchy) [Wesson81] because as the need for a global picture increases, so does the need for an integrator node to be able to assert conclusions or policies to a subtask node. The span of control is related to uncertainty, resources and bounded rationality of the agents.

Need for Redundancy: If redundancy is needed for reliability or uncertainty reduction, tasks can be decomposed into redundant subtasks. For example, Wesson et al. [Wesson81] suggest that if redundancy of control is necessary (e.g. for reinforcement or reduction of control uncertainty) additional control tasks should be generated and some subtasks should be assigned to more than one controlling task. However, redundant tasks require methods for discriminating redundant results [Durfee87d]. Subtasks may also be generated which treat the same problem from different

prespectives, and this too requires discrimination and integration of results [Steeb81].

Resource Minimization: Tasks can be decomposed by reviewing their resource requirements and finding ways to minimize resource use, for example by exploiting locality in parts of problems. Rand researchers, for example, suggest that local problems be solved locally to minimize uncertainty, and communication and coordination costs [Steeb84, Steeb86].

Uncertainty: Greater uncertainty in a task or its components may require spawning redundant subtasks which address a problem from different perspectives [Kornfeld81b, Lesser81]. Moreover, additional articulation work [Bendif87, Gasser86] in the form of control or coordination tasks (e.g. communication) may be necessary [Lesser83, Durfee87d].

Uniqueness Avoidance: Davis and Smith [Davis83] pointed out the problem of *unique nodes*. Any agent which has unique data or process produces a form of centralization of control and resources, which can create potential bottlenecks. They suggested dividing up a unique node functionally, or distributing it redundantly.

2.3.2 Methods for Solving Decomposition Problems

General classes of solutions have been proposed for the problem of task decomposition, though there appear to be few principles, methods, or experimentally validated techniques for automatically decomposing tasks by a DAI system. There are some theoretical approaches - decomposition to achieve deadlock avoidance etc., and heuristic approaches - using models of agents, reliability, etc. Some classes of solution are as follows:

Pick tasks which are inherently decomposable: That is, the given representation of the task contains its decomposition. The description of the states, the space of states and the operators leads to a natural decomposition, using the selector operations of the data structure. An example is the N-queens problem used in the Contract Net work. Here, the state is given by a data structure, which is essentially a tree of subproblem descriptions.

Decomposition by a programmer: This typical programming approach is used for example in the DVMT and in Hearsay-II. It is useful because there seem to be few known principles or methods for automatically decomposing tasks. The problem is intimately tied up with the representation of the task; in most systems, decompositions possibilities are already built into programmer generated action descriptions.

Hierarchical planning: A hierarchical planner does genuine task decomposition. It generates tasks which are goals to work on. However, this provides only a partial solution, because the decomposition depends upon the prior descriptions of the available execution operators, which are usually static. This produces limits to reconfiguration. See for example Corkill's distributed hierarchical planner [Corkill79].

Load balancing: Load balancing gives good examples of choices of decompositions, and some guidance. Load balancing has been mainly explored in the case of large numerical computations on parallel computer architectures [Fox88]. A very large set of numerical tasks, numbering in the tens of thousands, is partitioned into subsets, by an allocation to processors which

optimizes communication and computation load. Currently, optimization methods such as simulated annealing are used. Common dynamic load balancing techniques such as bidding schemes [Malone88b, Parunak87, Ramamritham85] typically rely on predefined decompositions and tasks of fixed grain size, and provide less insight into task decomposition methods.

Minimally connected subgraphs: If a problem can be described as a collection of interdependent elements, using a graph structure, then a decomposition algorithm can identify minimally connected subgraphs. Again, decomposition is built into the problem description. The difficulty is identifying the graph from the problem. For example, Levin and Moore [Levin77] observed that problems are solved by interaction with other agents. Describing the required pattern of interaction can give a dynamic pattern of decomposition in terms of interaction requirements. Davis and Smith regard problem decomposition as the creation of the map of subproblems, and believe this should be part of a DPS system's task [Davis83]. Once this map is created, the locality of action and information represented in the subproblem map allow for decomposition. Similarly, Pavlin, Hudlicka, Lesser use dynamic problem-solution graphs to represent problem-solving activity and propose that information so gained be used as a basis for organization self-design [Pavlin84, Hudlicka86b, Hudlicka86a, Hudlicka87].

Subtask aggregation: Tasks can be decomposed also by *aggregating* collections of known operators to fit the requirements of subparts of a larger task. This requires manipulation of subsets of tasks and operators, and the ability to structurally aggregate a collection of operators. Structural aggregation or composition of operators has been addressed in systems of actors and in the MACE testbed [Agha86a, de Jong88, Gasser87d].

2.4 Task and Resource Allocation

The problem of allocating particular tasks to particular agents is the problem of assigning responsibility for a particular activity. Task allocation is a metaproblem which may be addressed statically by a designer, e.g. by establishing a collection of specialized agents with pre-determined roles. If we conceive of task allocation as a kind of *articulation work* [Bendif87, Gasser86] to be done dynamically by a collection of agents themselves, using techniques such as contracting [Malone83, Malone88a, Smith81b, Smith81a, Parunak87] and organization self design [Corkill82, Lesser83], it is just another task, and can like other tasks, also be allocated recursively [Bond87]. Decisions on when and how to reallocate tasks - when to make organizational transitions or redesign the organization - depend upon knowledge linking performance to task allocations and on how available are slack resources for redesign [Gasser88b, Gasser88a, Pasquale88, Rosenschein83, Wesson81].

There are several choices of what aspects of a problem or task to provide in order to allocate responsibility for accomplishing that task to a particular agent. A task to be done can be completely or partially constructed by the agent itself, it can be assimilated via interaction with a "controlling" agent, or it can be given by a designer and embedded in the structure of the agent. For example, a task can be allocated by sending to an agent an entire problem description, a solution method, and a control trigger, which the agent simply enacts. Alternatively, an agent can be provided with data to which it can apply methods it already has, or it can receive methods to apply to data it already has. (These last two options, along with a task allocation message which serves as a control trigger, are provided in the Contract Net.) Finally, an individual agent may receive only a control

or responsibility "trigger," while gathering for itself its problems, its data and knowledge, and its solution methods. This range of choices indicates we need some basis for deciding what level of responsibility and choice to give to agents. Dynamic task allocation of any type requires reliable communication, coordination overhead, or redundancy. Durfee, Lesser, and Corkill [Durfee87d] noted that dynamic task passing (as in the Contract Net) may be a problem where there are error-prone channels, because of the mutual agreement required. Research in "reasoning about knowledge" [Halpern86a, Halpern86b, Halpern87] and common knowledge protocols promise to help address this problem.

2.4.1 Bases for Making Task Allocation Decisions

In the literature, there are several bases for choosing which agent should get which tasks of a collection. These include:

Bottleneck Avoidance: Allocation decision should avoid bottlenecks by overloading particular unique or critical resources [Davis83].

Fit to Specification: Tasks should be allocated to those available agents which provide the best fit to the task specification [Davis83, Smith81a].

Knowledge Dependency: Task coordination and precedence constraints affect allocation decisions when a node cannot work until another has finished its task[Durfee87d]. Task precedences are one type of task interdependencies. With adequate interaction capability, tasks addressing matters of consistency, definition, or direction, should be left to a node with the most global view [Cammarata83, Wesson81].

Overlap in Roles: Durfee and his colleagues [Durfee87d] noted that nodes should have overlapping responsibility to achieve flexibility and coherence. The Rand group [Steeb81] also noted that overlap would alleviate hand-off problems between clusters of problem solvers. Mechanisms of specifying "interest areas" - a node's responsibilities in a partial solution space - can be used to control overlap and redundancy in nodes [Lesser80a, Durfee87d].

Reliability, and Uncertainty Avoidance: Tasks whose results or completion are uncertain should be allocated redundantly, to reduce the uncertainty and improve reliability [Lesser83]

Resource Consumption: From one perspective, tasks should be allocated so that their use of critical resources is lowest. Pasquale, for example, suggests allocating tasks so that times to complete are minimized (when completion time is the critical success criterion) [Pasquale88]. Similarly, highly interdependent tasks should be allocated to agents in similar regions (e.g. spatial or semantic regions) to minimize critical communication and synchronization costs [Lesser80a, Steeb81, Steeb84, Steeb86]. From another perspective, resource slack can be a useful indicator for allocation decisions. In the Rand air traffic control work [Steeb81], a strategy for exploiting slack resources was developed by choosing the agent with greatest resource slack as a manager for centralized planning and organization tasks.

Urgency: Redundant allocation of urgent tasks may occur dynamically, and may be related to the breakdown of authority under load. For example, Wesson et al. suggest a strategy of *constructive*

usurpation: in a busy system, when an agent with authority for a task will not do it due to load anyone who can perform the task should, because results are more important than protocols [Wesson81]. Wesson et al. also suggest that allocation decisions should be decentralized when busy, and centralized when resources are slack, noting that allocation is itself work which consumes scarce resources. Anticipatory self-allocation may also be useful; if an agent hasn't received a specific request to do something, but knows it will be useful, it may anticipate requests if the system is not busy [Wesson81]. This requires global knowledge and coordination to avoid redundancy.

2.4.2 Mechanisms for Making Task Allocation Decisions

Task allocation can be approached with a number of mechanisms which range from planned allocation to opportunistic, market-like strategies. The following are some typical mechanisms for making allocation decisions:

Market Mechanisms: Tasks can also be allocated without explicit planning using various market mechanisms wherein available tasks are matched with available agents by generalized agreement and possibly mutual selection [Fox79, Smith80]. Underlying mechanisms for matching and decision making include "pricing" tasks according to their criticality or urgency [Malone83, Gasser87a, Kurose85, Malone87a]. Agents may use self-selection from a global or regional agenda of tasks, based on fit and task prices [Gasser87a].

Markets need policies and protocols through which agents communicate their intentions for doing particular tasks [Davis83, Smith80, Smith81a, Malone88b], and reach agreement using negotiation and conflict resolution when there is contention for tasks or when certain important tasks are overlooked [Durfee87d, Durfee87e]. Mutual agreement is reached when agents make a decision based on the best fit between the task and the agent. This is the method used in the Contract Net system [Davis83]. Fit can be assessed via pattern match (cf. Planner [Hewitt71]), possibilities lists (cf. Conniver [Sussman72]), metarules and metalevel control (cf. [Hayes-Roth85, Erman80], or transfer by discussion [Lenat75].

Multi-Agent Planning A planner or collection of planners can combine the work of task decomposition and task allocation by treating agents as specialized resources and objects which interact and depend upon each other. Multi-agent planning can be done using a single centralized planner with global plan synchronization and conflict elimination [Georgeff83, Georgeff84, Georgeff87c, Pednault87, Stuart85], or with distributed planners which make joint multi-agent plans [Corkill79, Durfee86, Durfee87a, Durfee87c]. Conflicts in task allocations can be resolved by allowing agents with related interests to exchange and elaborate proposed activities, using techniques termed *partial global planning* [Durfee86, Durfee87a, Durfee87c] and *negotiation* [Benda85, Conry86].

Organizational Roles: Organizational roles are predetermined or slowly changing policies for assigning task responsibilities to agents. An agent cannot accept tasks which do not meet its role in the organization, and perhaps cannot be presented with such tasks from others, assuming role knowledge is disseminated.

Recursive Allocation: Recursive allocation is linked to subproblem decomposition, by letting agents which are handling problems do the work of allocating their subproblems. This mechanism

was employed in the Contract Net [Davis83], where allocation work itself is decentralized. Policies for allocation (organizational design) can also be accomplished by recursive global reassignment. With a view of its subordinates' load and tasks, an agent can reassign responsibilities; this can be done recursively[Wesson81].

Voting: At least one project has explored voting as a task allocation mechanism. Rand Air Traffic Control agents allocated centralized planning tasks to one central agent chosen opportunistically by the set of agents by voting using common evaluative conventions. [Steeb81]

2.4.3 Allocating Resources

Resources are the products which are consumed to accomplish problem solving work. Resource allocation is a sub-problem of task allocation in a DPS or MA system. Allocating resources to tasks is a way of prioritizing the tasks, because tasks without resources cannot run [Kornfeld81b]. Some typical resources which are allocatable in DAI systems include processing time and communication bandwidth. For example, in some air traffic control problems it is necessary to restrict some communication to allow important collaborators to interact [Steeb81]. Resource bounded reasoning is important in any real system, and recent research has begun to address tradeoffs in resource allocation and real-time performance, using reasoned approaches to reducing search complexity termed "approximate processing" (e.g. [Lesser88]). Temporal resource allocation is the problem of doing the most pressing or critical activities first. For example, Wesson et al. suggested pursuing less important activities in less dynamic situations (e.g. look for global obstacles in a robot control system when the system is not busy). "Less important" work may be work which an agent knows another may do - here redundancy reduction provides a means for temporal resource allocation [Wesson81].

Representation has impacts on reasoning about resource allocations. Agents using logic-based representations of action and belief based on possible worlds semantics have had theoretical difficulty with resource allocation and resource-bounded reasoning partially due to the problem of *logical omniscience*. In short, it is difficult to avoid concluding that an agent believes the deductive closure of its explicitly declared knowledge (which may be infinite), yet this is impossible in any resource-bounded system. Konolige has presented a logic based approach to resource bounded reasoning in [Konolige86].

Several important approaches to resource allocation are resource allocation using specialist "sponsor" agents [Kornfeld81b, de Jong88] which allocate fixed portions of resources in a manner analogous to research sponsors, resource allocation based on the criticality of tasks [Lesser88] and allocation via resource pricing [Chandrasekn81, Kurose85]. Distributed resource allocation can be done by recursive allocation of resource allocation tasks to subagents, and by decentralized bidding schemes where tasks bid for resources, thereby changing prices. There is much more room for interaction between researchers in DAI and in economics, on the subject of resource allocation mechanisms and policies.

3 Interaction, Language and Communication

Interaction is important as a basic concept in DAI because it is the processes of interaction which make it possible for several intelligent agents to combine their efforts. Problems of different agents may exhibit mutual dependency, so agents must interact to solve them. We will take *interaction* to mean some type of collective action in a MA or DPS system wherein one agent takes an action or makes a decision which has been influenced by the presence or knowledge of another agent. While knowledge, perception, goals, actions, etc. may or may not be distributed, interaction is inherently distributed, and interaction is inherently dependent upon the coordinated action of at least two agents. We are ultimately interested in two kinds of interaction: *non-routine interactions* in which the actual terms and conditions of interaction are uncertain and under development, and *routine interaction* wherein individual acts (which are part of interactions) are carried out under particular expectations of future actions of other agents actions and belief states. In natural systems, there is no "pure" interaction of either type; there is a constant motion between the problem-solving and routine, with one predominating in a given situation. Both types of interaction critically depend on agents' *models of each other*, e.g. to formulate expectations, etc.

Interactions may occur through through explicit linguistic actions and through other actions in the world. They involve reactions of one agent to some representation of another agent it holds (part of the interpretive context) [Mead34]. Ideally, each interaction in a DAI system would cause revisions in each agent's model of the other - it seems that this is a requirement of adaptive communication, but this is rarely the case in existing DAI systems.

Rational actions are actions which an agent takes in response to goals it has, in the belief that the action will satisfy the goal. A communication exists only when the *intention to communicate* exists, because knowing this intention to communicate provides additional information for both the sender and the receiver. One model of communication is that *A communicates with B* when *A acts upon B* with linguistic actions (manifesting the intention to communicate) and *B reacts to A* (accomplishing the communication). Agents may be able to communicate meaningfully in order to structure their coordination, but coordinated interaction does not necessarily require communication. Agents may coordinate based on models they have of each other, without interaction (except that prior interaction or design necessary to set up the models). For example, Axelrod, Rosenschein and Genesereth, and Tenney and Sandell have studied how cooperation can occur without explicit communication [Axelrod84, Rosenschein86, Tenney81a].

3.1 Interaction

Several dimensions of multi-agent interaction are important for viewing organized aggregates. These include *among whom* the interactions take place; *when* interactions occur (This matters because of the possible temporal/causal relationships among new knowledge and beliefs across agents; *what is the content* of an interaction or communication e.g. results to be shared or tasks to be assigned); *how the interaction is accomplished* (e.g. what processes are involved or what resources are utilized); *why* the action occurs? (what goals have driven the action); and *what is the basis of commonality* (Is there a common language, shared interpretive context, or reference?).

Since action in the systems we consider is generally goal directed, many interactions are derived from

goals, and thus is traceable to some outcome or performance criteria (i.e., "did the (inter)action achieve the goal?"). For analysis, individual actions and patterns of action, communication, and interaction, can be linked to questions such as *What are the varieties of possible interactions? What are the interactions which obtain in a given situation? Why these interactions and not others? What is the result of this pattern of interactions? What patterns of interaction can exist?* These questions become more central as we try to conceptualize and describe frameworks for coordination, such as organizations. One perspective on describing an organization is its *pattern of activity* (e.g. patterns of interaction). Since organized interactions are at some level routine and predictable, they form patterns which provide basis for expectations of the behaviors of other agents. (Organized actions may of course be subject to revision or reinforcement.)

The routinization, conventionalization and codification of interaction patterns (whether done by designers or the agents themselves) results in a language system (cf. the discussion of the origins of the PUP6 communication protocols in [Lenat75, Lenat87]). Typically, tokens from a conventionalized dictionary are composed into messages, which are exchanged by agents. Automated language systems usually have grammatical constraints and internal semantic relations encoded into programs for generating or interpreting communications. The relation of the message to the communication environment is usually termed the *pragmatics* of the language utterances.

Considering communication in the context of cooperating agents leads to further issues. Communication is often an extended series of exchanges with a common purpose - a dialogue, which may have conventions of form and mechanism. Another issue is the sharing of knowledge to form the commonality upon which communication and cooperation may proceed. We can enquire as to what language and communication mechanisms maintain this commonality. For example, how can the alignments of meaning be maintained among agents in dynamic worlds where each has incomplete knowledge (cf [Winograd86])? This makes us concerned with the effect and role of knowledge of other agents in communication and language. Conceiving communication and interaction in a framework of goals for utterances, knowledge of participants, and planned actions to change that knowledge, we may synthesize a unified description of action and communication where communications are treated as actions, usually called *speech acts* [Austin62, Cohen79].

3.1.1 Reasoning about Languages

Working with distributed agents, we need to design and understand the language used for interaction, communication and organization. This means we need to know what knowledge to represent for communicating, and how to represent it in an interaction language. For example, agents may need to communicate about their mutual beliefs, their knowledge of each other, their current goals, etc. Communicating agents in general will have disparate knowledge and so the language system may have to allow for differences in knowledge, for communication and cooperation to succeed in spite of disparate knowledge, for communicating about disparity, and so on. One topic which has attracted the attention of linguists has been the problem of disparate references, that is, two agents may refer to the same object using different terms and from different viewpoints [Walker78].

Typical DAI systems have employed inflexible, pre-designed communication languages. For more adaptive DAI systems, we need more flexible approaches based on linguistic knowledge. Some of these may come from research in dialogue structures and speech act theory.

3.1.2 Context and Speech Acts

In terms of simply making others aware of data such as observations, results etc., different kinds of information may be appropriate for different agents. Also, different levels of abstraction may be appropriate. Criteria are thus needed for deciding what types of information to represent and how to represent them. The communication of goals is often needed for cooperation [Durfee87d]. Awareness of others' goals is needed, and along with the ability to allocate a goal from one agent to another (sometimes called *goal induction*) [Cohen79, Rosenschein82, Wilensky84].

It may be necessary to keep a track of other agents' states in order to know who to communicate with. This leads to issues of how accurate models of other agents need to be to allow effective communication, and of course to cost trade-offs in whether to communicate more abstractly and to spend resources on inferring meanings, or whether to spend resources maintaining more precise models of other agents and their communication requirements [Lesser83, Durfee87d, Durfee87e, Wesson81]. In a completely decentralized system, bounded influence (the limitations of control over other agents' actions and knowledge) is a problem [Hewitt85, Hewitt86].

Since awareness of other agents is crucial to cooperation, information must be continually communicated and updated. However this may impose a formidable burden on communication resources, and so communication, both in terms of destination and content, must be selective. In the DVMT, for example, communicating the wrong or redundant information can lead to harmfully *distracting* an agent from useful tasks. Selecting the proper information to communicate is difficult. To combat specificity problems such as these, manager nodes in the Contract Net can employ focussed addressing to limit interaction to agents known to be relevant, reducing communication loads. Conversely, where there are similar agents working on related problems the optimal communication level may involve some duplication of effort by the agents.

To capture the structure of larger interaction patterns, we need knowledge representations for the description and realization of dialogues [Grosz85, Grosz86, Grosz87]. We also need to identify and represent the speech acts involved in interaction, which will include indirect actions such as inferring the plans of other agents. Intentional, language-based interactions include the communication of beliefs and goals. Some interactional goals include understanding and influencing the beliefs and actions of other agents.

3.1.3 Using Fixed Interaction Languages

Standard conventions about the representations of knowledge held by other agents can directly lead to interaction languages. For example, the PUP6 system used standard message types related to the standard structure of agents. Since each agent had identical parts, queries were restricted to the values of those parts. [Lenat75, Lenat87]. Similarly, the Contract Net Protocol [Smith80] introduced a common internode language, which consisted of structured message types such as task announcements, bid messages, award messages, etc. The Contract Net used a framelike structure for messages, with different frames for each type of message. The message types were specialized to the types of information needed to solve the contract net's primary problem, market-like task allocation. There were no messages, for example, which could directly handle control, reorganization, or focusing information. Virtually all DAI systems have used standard interaction languages put in

place by designers. The DVMT employs some levels of reasoning about communication activities, primarily concerning what information to communicate (based on knowledge of interest areas of other nodes) and when to communicate it. Communication is handled by separate send and receive knowledge sources in each node, whose activities can be planned and prioritized like those of any other knowledge source.

Different choices of what to communicate to another agent have been studied in the DVMT project. This has been done manually for each experiment, and there have been no experiments with automatic or programmed criteria. Agents could communicate sensor data for the distributed sensor sector allocated to them, which is usually relevant to agents allocated to spatially adjacent sectors; interpreted data at more abstract and more condensed levels; goals, communicating current goals and allocating goals to each other by mutual agreement, and plans for their own activity to improve network awareness.

The Rand Air Traffic Control work [Steeb81] worked with the notion of *information distribution policies*. These were characterized along 4 binary dimensions: 1) selective or broadcast communication; 2) unsolicited or on-demand communication; 3) acknowledged or unacknowledged communication, and 4) single transmission or repeated transmission communication. The Contract Net used targeted broadcasting for announcing contracts. This answered the question of whom to communicate with by keeping relevance criteria which were applied to each task description to determine the "mailing list" for broadcasting. The DVMT project studied the interesting and important case where there are similar agents working on related problems, with duplication of effort.

3.1.4 Interaction Protocols and Dialogues

The individual message types used in the Contract Net combined with their expected responses yield a protocol for interaction which spans more than one interaction. A contract here is an agreement relating to any task to be allocated, so this protocol is of quite general applicability. This interaction regime elegantly provides for two way transfer of information with the potential for complex information transferred in both directions, local evaluation in the context of individual knowledge sources, and symmetric mutual selection of control points.

Levin and Moore [Levin77] introduced the notion of a *dialogue game*, which gives a way of initiating an interaction of a specified type, and of controlling it with a partially ordered set of subgoals. The participants in the dialogue game take roles and attempt to achieve the goals in the given (partial) temporal order. Levin and Moore are thus distinguishing between the function and the topic of a dialogue. Interaction types observed in naturally occurring dialogues, in which one person interacts with another for some purpose include Helping, Action-seeking, Information-seeking, Information-probing, Instructing, and Griping, for example. The speaker and hearer in such a dialogue ordinarily hold multiple goals, which are related in highly constrained ways. Persons cooperate in dialogues only if both hold goals that are subserved by the cooperation. In the dialogue games model, interaction proceeds at two levels. At one level, interaction decides whether to *enter* a given dialogue game, which roles are to be taken by the participants, and when to terminate. At the second level, the dialogue game is entered and the interaction is controlled by the speaker and hearer taking their agreed roles to solve the set of goals defined by the game, and instantiated by the current context. Dialogue games shed light on comprehension issues in natural language. A goal-oriented view of language limits the explosion of inferences by giving a focus, favoring

inferences which look for goals. The dialogue games model also provides a theory of the indirect uses of language. Other multi-sentential knowledge representations for the description of naturally occurring dialogue include "rules" [Labov74] and "sequences" [Sacks74].

3.1.5 Planning Communications

Communication between problem-solving agents may originate as part of a procedurally constructed dialogue such as the Contract Net Protocol [Smith80], or they may be treated as planned activities, akin to *speech acts*. Austin introduced speech acts as requests, assertions, suggestions, warnings etc. [Austin62]. Searle tried to specify necessary and sufficient conditions for the successful performance of speech acts, involving the intentions of the participants. To satisfy their own goals, people often *plan* their speech acts to affect their listeners beliefs, goals and emotional states. Cohen and Perrault [Cohen79] have developed a *plan-based theory of speech acts*, which has been further extended into a deep theory of the relationships between belief, intention, and rational communication by Cohen and Levesque [Cohen87a, Cohen87b].

Speech acts can be modeled as operators in a planning system. Operators for requesting and informing can be defined so that *compositional adequacy* is achieved i.e. requests to inform, and requests to request etc. can be used consistently. Rosenschein used Cohen and Perrault's [Cohen79] operators as the basis of a multi-agent plan synchronization regime [Rosenschein82], but neither Rosenschein nor Cohen and Perrault were able to adequately handle goal induction. Both used the predicate cause-to-want as a basis, assuming that when one agent received a request from another, a cause-to-want condition would be conditionally generated, and it would immediately imply cooperation and the successful induction of a goal of doing the requested action. The DVMT uses specialized communication knowledge sources for sending and receiving messages from other problem solvers [Lesser83]. In this way, the same control regimes can be applied to communications reasoning as to other problem solving activities [Durfee87e].

4 Coherence and Coordination

Coherence and coordination are concepts widely used in analyzing and describing DAI systems, but they are seldom clearly defined or differentiated. We will use the term *coherence* in discussing properties of the total DAI system (or of some region within it) and the term *coordination* in discussing particular patterns of activity and interaction among agents. Coherence will mean how well the system behaves as a unit, along some dimension of evaluation. Coherence may be evaluated by examining several dimensions of a system's behavior:

Solution Quality: The system's ability to reach satisfactory solutions, and the quality of the solutions it produces.

Efficiency: The system's overall efficiency in achieving some end.

Clarity: The conceptual clarity of its actions, and the usefulness of its representation. Can the system's behavior be described and represented so that an outside system observer can understand it? An understandable and describable system can also use descriptions of itself so that individual agents can understand it, and also diagnose faults, analyze performance etc.

Graceful Degradation: How gracefully it degrades in the presence of failure or uncertainty; what is its behavior at the limits of its environment, specification or self description?

Coordination, on the other hand, is a property of interaction among some set of agents performing some collective activity. The degree of coordination exhibited among a collection of agents refers to the extent they can avoid "extraneous" activity (e.g. the indirect activity of synchronizing and aligning their tasks) in achieving their primary ends. We call this extraneous activity *articulation work*, following [Bendif87, Gasser84, Gasser86, Gerson86, Strauss85]. Effective coordination implies some degree of mutual predictability and lack of conflict; the more unexpected conflict must be ironed out, the less well coordinated the agents. We don't explicitly consider *cooperation* as a separate concept here, because we (and others) see it as a special case of coordination among non-antagonistic actors [Axelrod84, Rosenschein85b]. Coordination does not necessarily imply cooperation, since antagonists can be coordinated, for example in a legal proceedings. Coordination and coherence are partially related: better coordination may lead to greater efficiency coherence, by reducing articulation work. However, good local decisions do not necessarily add up to good global behavior because good local decisions may have unfortunate global impacts.

Most DAI researchers have been interested in a system's coherence as measured by its efficiency. From this perspective, incoherence can result from conflict over resources, from one agent unwittingly (or maliciously) undoing the results of another, and from duplicate actions being carried out redundantly [Davis83, Lesser83]. For example, duplication of task announcements is possible in the Contract Net, where it can be limited by using *focussed addressing* (sending announcements only to nodes with known interests), but this requires models of other agents and more certainty about the world.

Inordinate problem or interaction *complexity* may also reduce coherence. Fox defined complexity as excessive demands on rationality [Fox81]. He identified information, task and coordination complexity. When task requirements exceed current resource capacity bounds the need to make resource allocation choices adds additional work and uncertainty, which may lead to incoherence

or lack of coordination.

The primary difficulty in establishing coherence and coordination is the attempt to achieve them *without centralized control or viewpoints.* Achieving global or regional coherence with only local control is probably the primary problem to which DAI researchers have addressed themselves to date. This problem is also important in distributed computing systems of all types (see, e.g., [Huberman88a, Pasquale88]). From a DAI perspective the problem is rich because agents must reason about the intentions and knowledge states of other agents and, in DPS systems, about the overall goals of problem solving, generally under conditions of disparate knowledge and/or representations [Lesser83, Hewitt86].

Lesser and Corkill [Lesser87] have suggested that solution coherent behavior in a DPS system requires achieving three conditions: *Coverage:* Each necessary portion of the overall problem must be included in the activities of at least one node. *Connectivity:* Nodes must interact in a manner that permits the covering activities to be developed and integrated into an overall solution. *Capability:* Coverage and connectivity must be achievable within the communication and computation resource limitations of the network (cf. [Lesser80a]). Other forms of coherence may be more difficult to link to specific system attributes for design purposes. Lesser and Erman introduced the concept of *distraction*, meaning the degree to which a node's focus can be shifted based on interactions with other nodes. Positive distraction shifts a node to tasks more globally useful, while negative distraction introduces redundant or diversionary effort. Much more research is needed to understand how to achieve coherence, especially with only local control, but several approaches have been advanced in the current literature.

4.1 Organization

An *organization* can provide a framework of constraints and expectations about the behavior of agents (e.g. a set of "roles") which focuses the decisionmaking and action of particular agents. Corkill, Durfee, and Lesser have suggested that organization can be seen as a long term, strategic load-balancing technique, which can improve efficiency coherence [Corkill82, Durfee87d]. The organization of a collection of agents can be revised to accomplish by default what focused addressing or direct load balancing would by direct action [Durfee87d]. There is general agreement that no organization structure is appropriate in all situations; this has been pointed out long ago by contingency theorists of human organizations (e.g., [Galbraith73, Lawrence69, Perrow67, Thompson67]). Fox [Fox79, Fox81] developed a taxonomy of organizational types which evolve as an organization grows and becomes more efficient and flexible. According to Fox, as organizations grow, they produce multiple products. Since agents compete for resources in multiple product organizations, they are often reorganized into a multidivisional hierarchy, comprising a division for each product, with some strategic control level over them. With further development a collective organization emerges, where the hierarchy is split into separate organizations which cooperate to achieve a shared goal - rather like shared long term contracts. The next development in the reduction of control and information flow in organizations is a competitive organizational system. We lack concrete theories of organizational performance, but Malone in particular has begun to study the comparative information processing performance of rigid organization structures [Malone87a, Malone88a]. However, there is still meager insight linking attributes of situations and problems to organizational design. What little DAI-related research exists primarily focuses upon structural models of organi-

zation (cf. [Pattison87, Malone88a, Malone88b]), a view considered problematic by some (see e.g. [Gasser88a, Strauss78]).

Centralized and Hierarchical Organization: Typical hierarchical organizations associate greater control with a more global viewpoint. Nodes with more global information guide nodes with less global information as decisionmaking data flows "upward" in progressively more abstracted forms and control flows "downward." Chandrasekaran [Chandrasekn81] notes that the simplest hierarchy is two levels wherein one agent at the top level has complete information and all authority to control problem-solving agents at a second level. In such a "centralized" organization, coherence is achieved by limiting decisionmaking and by centralizing all decisions in one agent, at the possible expense of solution quality in a dynamic environment [Steeb81]. For example, coherence was generally assumed by the Rand team to be difficult to achieve in the air traffic control problem, so they opted for the conservative solution of complete decisionmaking centralization [Steeb81].

Organization as Authority Structure: Authority structures are one way of reducing coordination work. When one node has authority over another, the latter must accept a goal or result. Solution coherence may be increased or decreased depending on the accuracy of the global view provided by the authoritative node and on the need for global coordination for higher quality [Wesson81]. Lowering the coupling and interdependencies among agents can reduce the need for authority structures among nodes by reducing conflict among nodes and hence reducing the need to resolve it with authority [Wesson81]. Authority structuring can improve coherence if it is coupled with information about which nodes have the most accurate views of a situation [Durfee87d]. The most accurate nodes should be allowed to guide the less-informed ones.

Market-Like Organization: Numerous authors have proposed that distributed computer systems be organized as markets, and have studied their performance under various (usually closed world) market assumptions (e.g. [Kurose85, Malone88b, Ramamritham85, Sproull78]). Competitive, market-like organizations have been proposed to organize DAI systems as well [Davis83, Fox81, Kornfeld81b, Smith80, Parunak87, Stefik88]. For example, the Contract Net architecture supports a market in both tasks and processing resources, through competitive bidding, while Kornfeld and Hewitt and later Hewitt alone have suggested competitive schemes for generating and settling problems, rather than task allocations [Kornfeld81b, Hewitt85, Hewitt86]. Most research has considered mechanisms for implementing market-like organizations, while relatively little DAI research has gone into understanding local decisionmaking policies and market rules which produce coherent system performance. The emergence and reconfiguration of market structures has received virtually no attention in DAI, and merits more research.

Organization as a Community with Rules of Behavior: An organization may be construed as a set of locally interpreted *rules of behavior*, rather than as an externally defined structure. For example, in Lenat's PUP6 system, knowledge was organized as a community of interacting agents ("Beings"), in contrast to routinized organizational authority structures [Lenat75]. A Beings system was a "flat" organization of specialists, including a "chooser" specialist for conflict resolutions. The entire community had an imposed constraint that each agent must be represented using the same standard structure. The rules of behavior were derived from the common structure of agents. Agents could only interact by making queries about particular aspects of other agents' predefined structure. Hewitt later urged the search for "rules of civilized discourse" among agents [Hewitt77a] in his quest for concepts underlying decentralized control. But as has been pointed out for both

human and automated systems, local interpretation renders global conceptions of rules problematic (cf. [Gasser88a, Gerson86, Lesser83, Manning77, Winograd86]).

4.2 Increased Localization

Two extreme coordination regimes are: 1) Predefine a fully partitioned set of actions for each node with full synchronization, making problem-solving completely routine, and 2) Broadcast all state changes so every agent has complete awareness of the network conditions, allowing them to make fully informed local decisions [Durfee87d]. In most DAI systems, and in particular in open systems, neither of these extremes apply. Both are only useful if bandwidth is available, if the situation doesn't change faster than the synchronization, communication, or decisionmaking works, if there is no uncertainty in the task structure or dynamism in the world, and if the system scale is kept within resource limits. Several mechanisms have been suggested for improving localization:

Specialization: Specialization can improve performance by reducing and focusing the responsibilities of a node, and hence reducing its local decision making overhead [Durfee87d].

Reducing Dependencies: Coordination and coherence can be increased if local dependency among nodes can be reduced so that there is less possibility for harmful interaction and correspondingly lower computation or communication overhead. This could be achieved by good task decomposition, but it is not obvious how to connect problem characteristics with decision dependencies. Interagent dependencies can also be reduced by supplying slack resources [Fox81], because agents will have less contention over the resources which are available.

4.3 Increased Local Capability

Increased coordination and coherence should result from making each agent more capable locally (e.g. giving it better problem-solving knowledge, better internal control, or greater resources), and by making it evaluate each of its potential actions in the light of how it will affect network problem solving [Durfee87d]. Nodes can get closer to optimality overall by improving their local decisions to make them less redundant and/or less disruptive of the work of other nodes. *Local planning* is one way to improve local capability. Plans (sequences or conditional partial orderings of activities) can give a node the ability to reason about what it is and will be doing, and thus to know whether or not to send or accept certain temporally-constrained data [Durfee87d]. Local planning gives a node greater control over its own processing, reducing local redundancy and extraneous work. This occurs by 1) packaging common sequences of activities into macro-operators, and 2) using goals to focus on specific highly-rated tasks and planning methods for achieving them [Corkill83].

Increased local capability must be balanced against local overhead costs and alternative strategies such as increased communication. Experimental results on coherence [Durfee87d] indicate that as a node becomes less certain about what and where to communicate the computational overhead for planning and meta-level communication becomes more acceptable. This is more true when planning becomes relatively cheap compared to problem-solving activities.

4.4 Planning

Greater coordination can be achieved by aligning behavior of agents toward common goals, with explicit divisions of labor. Techniques such as centralized planning for multiple agents, plan reconciliation, distributed planning, organizational analysis, and appropriate control transfers (e.g. focus of attention strategies based on reasoning about the state of local problem solving [Lesser83]) are ways of helping to aligning the activities of agents by assigning tasks after reasoning through the consequences of doing them in particular orders. To align activities of several agents using planning, plan interactions must be controlled. Plan interactions may involve incompatible states, incompatible orders of steps, or incompatible use of resources. Plan synchronization can be performed at several points. It can be done during problem decomposition [Corkill79]. It can be done during plan construction, by building smoothly interacting plans hierarchically [Corkill79], by aligning partial plans incrementally [Durfee87c], or by reasoning about interactions and dependencies as a part of planning [Rosenschein82]. It can also be done after plan construction [Georgeff83]. The control of plan interactions can be done using *multiagent planning* wherein a single agent generates plans for multiple agents [Georgeff83, Steeb81], or using *distributed planning* and dividing up the planning activities as well [Corkill79, Durfee87c, Hayes-Roth79a, Hayes-Roth85].

Multiagent Planning: Generating multiagent plans requires reasoning about how actions of different agents may interfere, and thus requres explicit representations for parallel actions. Structures and state based representations of parallel activity based upon logical formalisms have been proposed by [Allen84, Backstrom88a, Dean87, Georgeff84, Georgeff86a, Georgeff87b, Georgeff87a]. The frame problem is of paramount importance. Techniques under investigation for handling it include localized, *event based* (rather than *state based*) action representations [Lansky85, Lansky87a, Lansky87b], the use of indexical logic operators such as fluents [Georgeff87b, Georgeff87a], and treating multi-agent plans using single-agent frameworks [Pednault87]. One approach to multiagent planning, due to Georgeff, is to insert communication acts into single agent plans, so they can avoid harmful interactions. In Georgeff's work, actions are microsequences of states. Plan reconciliation is achieved in three stages by a single global agent. First, an interaction analysis is performed, to determine where plans interact. Next, a safety analysis determines where plans have potential conflicts. Finally, unsafe state sequences states are coalesced into critical regions, and conventional synchronization mechanisms based on communication acts are applied. The communication acts concern properties of actions, resources and physical constraints at the time of execution of the plan [Georgeff83].

Distributed Planning: In distributed planning, a single plan is produced by the cooperation of several agents. Each agent produces a subplan, but there may be conflicts among subplans which need to be reconciled. One hierarchical approach, due to Corkill, is to synchronize levels of planning in all the agents, communicating shared variables between goals and resolving conflicts at each level before refining plans to lower levels [Corkill79].

Mutual plan construction is not well understood. It is confounded by disparity of goals and intentions, as well as in world knowledge. All the problems of multiagent planning exist, along with the problems of inconsistent world views due to distribution. Durfee, and Durfee and Lesser have originated the idea of *partial global planning* as a mechanism to enable communicating problem solvers to incrementally construct mutually coherent plans [Durfee87a, Durfee87c]. Added benefits may accrue by using multiple perspectives in the planning process at different levels of abstraction,

as Hayes-Roth has suggested in work on the OPM system (cf. [Hayes-Roth85]).

4.5 Increasing Contextual Awareness

Coherence and coordination can be improved by giving nodes greater knowledge about the context in which they are making decisions - greater knowledge of the goals, plans, and activities of other agents, deeper knowledge of the problem domain (which is the context for individual domain dependent decisions), and greater temporal context (e.g. greater lookahead, history, etc.). Tenney and Sandell have done extensive characterization and theoretical analysis of coordination possibilities of agents with various levels of contextual awareness, ranging from no communication and purely local knowledge, to abstracted global or regional network models [Tenney81a]. Several mechanisms for increasing contextual awareness include:

Network Views: Each node can be given a general view of network responsibilities to guide its problem solving and communication decisions, i.e. its place in an organization. Note that it is not enough to *establish* roles - they must be communicated and widely known so that they affect local decisions. Three representation options are to encode responsibilities in network centered models [Pattison87], task centered models [Durfee87c] or agent centered models [Gasser87d].

Extrapolation, Prediction, and Modeling of Others: Making predictiions and generating expectations of other agents' behavior are powerful approaches to coordination. For example, Ginsburg suggests using assumptions of common rationality as a basis for structures which allow rationally coordinated decisions [Ginsberg87]. Wesson et al. suggest communicating information about how data will change, along with the data itself [Wesson81]. Durfee, Lesser, and Corkill suggest exchanging plans to allow nodes greater access to the anticipated future behaviors of others. In speech act related research, Cohen and Perrault [Cohen79] develop belief models of other agents (useful information about the belief context of other agents' actions) as a foundation for coordinated communication activities. Rosenschein [Rosenschein82] and Morgenstern use belief and knowledge models to generate workable multiagent plans. (See also Section 5 of this survey.)

Bruce and Newman[Bruce78] distinguish between single actor plans and *interacting plans* in their work on understanding stories about coordinated actions. The motivation for their research was the analysis and representation of narrative text. Analysis of narratives should explicate interactions among plans, and is important to DAI from the standpoint of representing and reasoning about the plans and intentions of other agents. In their framework, an understander must reason about how agents represent the plans and beliefs of others. Beliefs about plans determine actions. Actions include the perceptions of each others plans, the example used - the Hansel and Gretel story - illustrates these points. Analysis of stories proceeds in terms of actions connected through goals, effects and enabling conditions. An actor can interpret the actions of another in terms of goals, and change his actions accordingly. Deception and differing beliefs are a common feature of stories. Representation of interacting plans involves showing how the beliefs and plans of one character are embedded in the beliefs and plans of another. Their scheme provides for the representation of intentions and the effects of actions. They show that a single actor plan can be modified by the needs of cooperative interaction with others, and how cooperative interactive episodes can be transformed and used deceptively by one party in achieving his or her own covert goals.

4.6 Managing Communication

Agents can improve both coordination and coherence by managing what, how, and when the communicate with each other. Communication may provide agents with the knowledge to place their actions in the context of what others are doing, and may help synchronize actions [Georgeff83, Rosenschein82, Stuart85]. It may also provide information generated elsewhere which will improve the quality of a node's local decisionmaking. Wesson et al. derived two basic motivators for communication: sharing time-varying information about the external world they are sensing, and sharing time-varying information about their own internal states of processing [Wesson81]. Knowing when something has changed sufficiently to notify another agent is tricky; if the world did not change, or was in a steady state, no messages would need to be exchanged (but no action could take place). Wesson et al. suggest that if a widely held tentative conclusion is found to be incorrect, others should be immediately notified of the fact (to save redundant computation in other agents) [Wesson81].

Three characteristics of domain-level communicated information relate communication and coherence: relevance, timeliness, and completeness [Durfee87d]. *Relevance* is the amount of information in the message consistent with the global solution. *Timeliness* measures the extent to which a message will influence the current activity of the receiving node. The timeliness of a message varies with the state of a node. A message which will have no effect should not be sent. *Completeness* is the fraction of a complete solution the message represents. More complete messages reduce redundancy - they are more discriminatory in Wesson et al.'s [Wesson81] sense, because more complete messages can be combined with in fewer ways. With greater awareness of the global situation, and with classification of messages on these dimensions, a node can make better decisions about what to send and what not to send to increase coherence (i.e reduce negative distraction and increase positive distraction). Nodes must also know what to listen to and what to ignore. Knowing what *not* to communicate is also important for coordination and coherence. Explicit communication is not always necessary for synchronization, as it may be accomplished by using agent models and reasoning about the activities of others, e.g., under assumptions of common knowledge and rationality [Rosenschein86, Tenney81a]. There is always a tradeoff over the need for reliable and timely information and the costs of communication and computation. But equally important, communicating obsolete or uncertain information may negatively distract an agent, if it is worse than what they already have [Lesser83].

For communication which aids coherence and improves coordination, agents should exchange only highly *diagnostic* data: data which is consistent with the smallest number of current hypotheses or decision alternatives, and so will have the greatest discriminating effect. Exchanging one piece of this data can discriminate multiple hypotheses, killing many birds with one stone [Wesson81]. Durfee et al. believe that messages should contain less detailed information about domain specific actions, and more detailed information about plans for problem solving activities. Messages should contain meta level information, rather than domain level data such as partial solutions [Durfee87d]. In addition, meta level information specifically intended to enhance coherence can enable nodes to make more informed communication decisions [Durfee87e].

4.7 Managing Resource Use

Regulating the use of consumable resources in a DAI system is a method of encouraging coherence and coordination. Allocating resources is one mechanism for focusing the attention of a community of agents, either through market mechanisms or through explicit control policies (cf. [Kornfeld81b]). Coherence and coordination can be improved if the agents with the most accurate, global, or discriminating viewpoints are allocated adequate resources [Steeb81, Steeb84]. Allowing slack resources at critical coordination points can also improve coordination and coherence by reducing the overhead of decisionmaking in contentions over scarce resources.

Knowledge resources such as domain problem knowledge or capability, and decisionmaking resources such as the degree of freedom of choice can also be a factor in achieving coherence. For example, allowing greater (or lesser) freedom of choice for each participant (effectively making each participant less (or more) certain of the future commitments of others) is one way to make coordination easier (or more tightly constrained) by controlling "slack" in a decisionmaking process [Chandrasekn81, Fox81]. For actions which are mutually constraining, agents can improve coordination by allowing others maximum freedom of choice, for example by using *least commitment* strategies or relaxing quality requirements. Chandrasekaran [Chandrasekn81] suggests a benevolent *principle of least commitment* as a coherence device; each agent makes only conservative changes to allow other agents greatest information and freedom for their own decisions. Some commitment is unavoidable at each choice, but proper choices can provide useful islands of certainty rather than overly constraining obstacles. In a related strategy for coherence called *local relaxation* [Chandrasekn81], each agent's results or commitments may be relaxed to become consistent with those of its local neighbors (cf. [Goldstein75, Sycara85a].

4.8 Managing Uncertainty

Both Galbraith's and Perrow's contingency theories of (human) organization note that organizational structure is related to the uncertainty and diversity of the task environment of the organization [Galbraith73, Perrow67]. Uncertainty, in this context, can be defined as the difference between the information available and the information necessary to make the best decision. Fox [Fox81] discusses several types of uncertainty not exploited in contingency theory, including: *information uncertainty* - uncertainty of the correctness of data, *decision algorithm uncertainty* - uncertainty due to lack of knowledge of outcomes of decision, *environmental uncertainty* - poor estimation of changing environment, and *behavioral uncertainty* - individual agents may not deliver on their contracts.

The quality of knowledge among nodes in a network affects the amount of uncertainty and error (in data and communications) that can be accommodated [Lesser83] for a given level of solution quality. Solution coherence is thus related to knowledge and data uncertainties. Formal progress is being made (see for example [Halpern86a, Halpern87]), with respect to the problems of how common knowledge can be guaranteed in the face of particular communication failures.

Conversely, degrees of coherence and coordination can be controlled by managing the degree of uncertainty willingly tolerated. As agents become less certain about the place of their actions in a global picture, they have a harder time making local decisions with better global effects.

Manipulating the quality and certainty of interim solutions may be one way of achieving real time performance [Lesser88].

4.9 Pluralism

Kornfeld and Hewitt [Kornfeld81b] base their "Scientific Community Metaphor" on the idea that the success (i.e. quality and robustness, or solution coherence) of scientific research depends critically upon the concurrency and diversity of the scientific community. They feel that parallelism is fundamental to the design and implementation of expert systems, independently of technological pressures for parallelism, for this reason. The language Ether was designed by Kornfeld for creating highly parallel problem solving systems. (The correspondence of Ether to the scientific community is not direct, but seen only at a high level of abstraction.) More recently, Hewitt has called for reasoning based upon "due processes" which incorporate multiple conflicting points of view for problem solving [Hewitt86].

Coordination has been conceptualized as a problem of aligning activity from multiple perspectives in [Bond87, Gasser84, Gasser86]. Misalignment results in "disarticulation" in a lattice of activity, and serves as an indicator of how and where perspectives must be integrated to achieve coordination. Others have suggested using uncorrelated perspectives to help stabilize control through redundancy. Fox, among others, has also noted that function or product decomposition in an organization will give a beneficial mesh effect [Fox81]. Redundancy of decisions from disparate perspectives has been explored as a stabilizing organizational principle in the Rand air traffic control work and in the DVMT [Durfee87d]. In particular, the Functionally Accurate, Cooperative (FA/C) framework of Lesser and Corkill [Lesser81] relies strongly upon the reconciliation of multiple competing hypothetical explanations for phenomena, and upon combining evidence from multiple points of view.

4.10 Abstraction and Meta-Levels

Abstraction is a powerful device for enhancing coherence and coordination in DAI system design. Nodes are most likely to need communicate with other nodes which are semantically closer, with respect to their mutual problem. Data abstraction can lead to a relaxation of this rule, as data abstraction and globality go hand-in-hand. That is, abstraction increases the semantic scope of data. Abstracted data is potentially more applicable to a wider variety of cases, hence semantically more global. In the distributed sensor net examples of the DVMT and of Wesson's group, there is a strong relationship between knowledge which is problem specific (in a semantic sense) and geographic location. This allows these researchers to assert that nodes are more likely to need to communicate with others which are working with geographically similar information [Lesser83, Wesson81]. Thus communication channels and decisions (resources) can be structured along two dimensions: problem locality, and data abstraction. This gives guidelines for resources allocation and organizational structure which lead to more coherent systems. Abstraction also is used centrally in contract net. Task abstraction enables a node to compare and select tasks within a common framework, and bid abstraction enables a manager to compare and select bids.

Meta-level Information should allow nodes to reason about the past activity of a node (what it

has done) the current activity (what is going on now) and the future activity [Durfee87d]. For example, abstracted blackboard information can help in making high-level focusing and control decisions which depend upon results from past activities. There is some evidence to indicate that nodes which work with redundant data perform better when they exchange metalevel information about the goals and plans of other nodes, because it allows them to avoid redundant work. This information enables agents to better assess the effects of communications on the activities of other nodes [Durfee87d]. The abstractions afforded in meta-level communication serve two useful purposes - fewer messages need to be exchanged, (communication reduction) and computation to interpret all the messages and predict future actions in other nodes is reduced (increasing computational coherence). With no overlap, meta-level communication is unnecessary [Durfee87d]. With moderate overlap it is sufficient to exchange enough information to avoid immediate redundancy, and with high overlap, enough meta-level information must be exchanged to prevent future redundancy. Metalevel information can nearly halve the time necessary to generate a solution in highly-overlapping environments [Durfee87d]. It indicates that to achieve reliability via redundancy, metalevel reasoning may be important for focusing. Of course, if global performance is at a premium, meta-level communication will help but will increase communication load. Moreover, obsolete meta-level information can be dramatically counterproductive where there is high overlap.

4.11 Adaptation

Finally, adaptation can improve coherence and coordination over time, by allowing agents to reconfigure their styles of processing and interaction to best conform to the requirements of changing problem situaitons. Locally, when a node's activities are controlled by local planning, replanning in response to new contingencies can be an important adaptive strategy. It requires interleaving of planning and execution. An approach to locally reactive replanning in a DPS system has been presented in [Durfee86].

Organizations of agents can also be adapted, by changing the roles, knowledge, and activities of agents to conform to new problem situations. Fox [Fox81] points out that *organizational adaptability* is very important for efficiency because programmed organizational responses will not be appropriate in environments where uncertainty is high. Organizational adaptation is critically dependent on identifying dysfunctional organizational aspects, projecting new organizational forms with better performance, and having mechanisms for making transitions among organizations. These, in turn, depend heavily upon representations of organization and organized activity, about which we have few insights (see Section 5 of this survey). Several DAI systems have explicitly addressed explicit organizational reconfiguration. At one extreme, the Contract Net of Davis and Smith performed dynamic organizational reconfiguration, to suit the opportunistically generated requirements of an unfolding problem, but there was little persistence of organizational form [Davis83]. Rand ATC researchers examined several strategies for replacing a centralized controller using voting and shared convention schemes [Cammarata83, Steeb81, Steeb84]. Adaptation or *organization self design* has been a longstanding goal of the DVMT project. Corkill suggested underlying representational mechanisms for reconfigurable organizations of problem solvers in the DVMT [Corkill82], and Pattison et al. developed a high-level language for specifying organization, while Corkill, Hudlicka, Lesser and Pavlin developed schemes for reasoning about the adequacy of problem solving behavior in the DVMT [Hudlicka86a, Hudlicka87, Pavlin83, Pavlin84]. However, to date these have not been combined into a complete organization self design system. Last, Gasser and his colleagues believe

that adaptability requires complex reasoning about organization. They have begun to study representations for organized activity which allow for flexible reconfiguration, based on conceptualizing "organization" as locally held collections of expectations about the knowledge and action of other agents, rather than as role restrictions or fixed interaction structures [Gasser88a].

Unfortunately, there has been very little research on learning and adaptation in DAI systems (but see [Huhns87b]). Even with the adaptive regimes already known, there is virtually no opportunity for incorporating performance feedback to tune organizational forms for greater coherence. Collaborative and distributed learning approaches provide a very fruitful area for further research.

5 Modeling Other Agents and Organized Activity

In early DAI research, Wesson et al. reported that "One of the most important principles we devised involved the use of models to simulate and predict other nodes' activities" ([Wesson81] p. 14). Meaningful interaction between two agents requires them to have at least implicit knowledge of each other, such as the knowledge encoded in a communication protocol or language. Meaningful communication through language is impossible without some agreement on the intended effects of an utterance. One agent must know what reaction to expect upon receipt of a message it sends, to intelligently plan communication. From this perspective, even a typewriter has encoded a model of its user - the shape and structure of its keyboard.

Coordination, which is important for avoiding harmful interactions and because local decisions have global impacts, is only possible when some agent has some expectation about the character of the interaction (this may be a designer). This expectation may be implicit, but it may also may require reasoning. If one aim of DPS and MA systems is decentralized control, then to make local decisions in varying circumstances, individual agents must be able to reason about their impact on other agents, and about possible undesirable impediments. Durfee, Lesser, and Corkill citeDurfeeetal87a,Durfeeetal87b use the term *network awareness* to describe the knowledge that individual nodes use for coordinating with others. One example is the need for adaptation. To allocate tasks adaptively, for example, even a central controller (manager) needs to know (or learn) what potential task-performing agents can do.

Finally, an agent must be able to reason about its own activities for reasons of control and coordination with others. In this respect, an agent must *objectify* itself (incompletely) to see where it fits in a coordinated process, or what the outcomes of its own actions are. Indeed, sociologists and social psychologists in the *symbolic interactionist* tradition have proposed that the primary mechanism for creating organized societies of individuals in the ability of the individual to generate and use internal models of others and of him/herself to reflect upon actions and their effects [Abraham82, Blumer69, Cooley64, Mead34].

Restrictions in agents' roles or interactions, using, for example, organizational roles or fixed communication protocols, can limit what agents need to explicitly know about others. For example, the Contract Net protocol [Smith80] limits what an agent needs to know to the eligibilities of other agents for a proposed subtask, the addresses of other agents for communication, what contracts have been assigned, and (sometimes) what is the state of processing, of other agents (e.g. finished or not).

5.1 Rationales for Agent Modeling

There are several general reasons for modeling other agents. First, models are useful for predicting the requirements and effects of events not directly sensible (e.g. because they are in the future or elsewhere in a system). For example, Wesson et al. believe that agents in a Distributed Sensor Network should represent and predict changes to the belief systems of their relevant neighbors. The benefit is that frequent data reporting can be replaced by a system which reports only data which it knows is relevant and/or needed for predicted future actions. This improves coherence (less communication work, less filtering of unneeded data) at the expense of maintaining and using

the belief model [Wesson81]. This predictive approach has also been employed in the DVMT. The implications of increasing predictive power are that each node moves more toward being a universal node, with a more complete global simulation of the system in which it resides (including the other agents) which it builds over time. There is of course a tradeoff here, based on resource constraints. Two problems come to mind: 1) how to generalize situations and expectations into routine patterns and thus develop general skills for prediction (and to reduce individual processing load for individual cases), and 2) which prior predictive information to delete or overlook (as new predictions enter the picture).

Prediction can be useful for reducing communications because only necessary data need be communicated [Wesson81, Cohen79, Cohen87b, Cohen87a]. Wesson et al. [Wesson81] provide a limited example of how modeling the world and other agents using their PAN (Process Assembly Network) concept would reduce communications significantly, by allowing nodes to autonomously maintain predictions about upcoming events. These predictions reduce the necessity of sharing conforming information, and leave communication channels free for sharing surprising, unpredictable information. Moreover, it is possible to coordinate without communication (save the communication necessary to set up the models!) by using explicit models of values and possible choices of other agents, as Rosenschein et al. have investigated [Rosenschein85a, Rosenschein86]

Agent models may also prove useful for evaluating the credibility, usefulness, reliability, or timeliness of data. Durfee et al. suggest that to deal with new information which is potentially distracting, nodes must have knowledge about others' activities, to decide, for example, whether what they send you is believable or timely enough to be acted upon [Durfee87d].

Agent models may improve efficiency by focusing activity or directing search. As indicated earlier, knowledge of the data and resource requirements of other agents may prevent unnecessary communication, and engender early communication of important data. Agent models also aid in thinking about where to get particular information and how available it is [Huhns87b, Davis83]. Overhead and bidding delays can be reduced using *focused addressing*, which can be improved using meta-level communication, organization structuring and plans [Durfee87d, Durfee87e]. In the Contract Net, for example, nodes could use focussed addressing to announce tasks to agents they knew had appropriate capabilities [Davis83, Smith80].

5.2 What Knowledge of Others Must be Represented?

Existing DAI research provides some guidance about what kinds of knowledge about other agents can be useful. As we have reiterated throughout, much of this knowledge is needed in *any* DAI system, though it is often implicit in the system, and may be known explicitly only to the designers.

Knowledge of Agent Capabilities: Capability knowledge is necessary for allocation decisions, performance assessment, feasibility analysis, and to know what knowledge they may be able to provide (cf. [Davis83, Smith81a, Durfee87d, Durfee87e, Durfee87a, Lenat75]).

Knowledge of Agent Resources and Demands: Real time constraints (or other resource constraints) may require that nodes model the temporal or resource behavior of other nodes to know what to send them and when to expect answers [Durfee87d, Durfee87e, Wesson81]. For example, Wesson et al. suggest the following communication heuristics:

If a node is known to be very busy, (assuming single priority messages) don't send it a message. It won't process the message before the world has updated itself, and the message will just add to communication overhead [Wesson81].

If a node is known to have inadequate resources send it aggregated information which will enable it to focus on types of tasks which involve making important conclusions and exploiting its important local data, rather than on global and less important tasks (such as building a global obstacle map, which someone else could do) [Wesson81].

A variant of this second heuristic has been examined in the DVMT in experiments designed to test the performance of the system, in response to varying levels of abstraction in exchanged information.

Knowledge of Responsibilities: Assigned responsibilities are a way of reducing task allocation overhead, and have been exploited in Rand experiments [Cammarata83, Steeb86] and as *organization structuring* in the DVMT. Responsibility assignments must be communicated and known, to affect adaptive decentralized task allocation.

Knowledge of Solution Progress: Knowledge of an agent's processing state is important to detect deadlock and liveness, but also to predict whether it will be useful to exchange any information with it. If an agent is known to be running behind, then newly generated information may be out of date before it is useful to the agent.

Knowledge for Communicating: Communication knowledge falls into several classes: Knowledge of channels, languages, protocols, etc., and knowledge about what will be useful to communicate. For example, the ACTORS systems at MIT have explicit models of other agents which are typically limited to the location or name of each agent. This requires an underlying operating system which keeps track of the locations of all agents, and takes responsibility for delivering messages. While any two automated agents typically use common language and protocol knowledge for communication, the extent to which this knowledge must be (or is actually) "common" or "shared" among some (e.g. human) agents is not settled [Winograd86].

Knowledge of Beliefs, Goals, Plans, Actions: Coordination requires at a minimum some knowledge of the *actions* of other agents, and the ability to reason about the effects of those actions [Cohen79, Georgeff83, Georgeff84, Georgeff87c, Fikes71, Schank77, Wilensky84]. Being able to anticipate or explain actions requires further reference to knowledge of the plans, goals, and beliefs of agents which have led to actions, and causal relations among these, as well as knowledge about how these will evolve. For Bruce and Newman [Bruce78], representing a social episode includes a representation of each character's *role*. A role is the set of actions the character expects to perform, and the intentions that can reasonably be inferred from these actions, given the assumption that the characters are cooperating. In the Rand ATC work [Steeb81], each agent needs to know plans of other planes as well as speed, heading, fuel, destination and emergency status. Bruce and Newman' representation scheme [Bruce78] allows representation of causal relationships among states, enabling actions etc. Changing the actions of others may also require providing justifications, and meaningful justifications can only be generated using knowledge of the other agent's beliefs, goals, etc. (or the *assumption* of common beliefs). For reasoned communication and for synchronization of plans, detailed knowledge of the beliefs and goals of other agents is necessary and has been discussed in the framework of several domains [Bruce78, Cohen79, Cohen87a, Cohen87b, Levin77, Rosenschein82].

The completeness of models of other agents is a difficult issue. More complete models of the beliefs of other agents may be ineffective because they may require that an agent duplicates the processing of another node; This is a matter of computation vs. communication and problem-solving speed requirements and efficiencies [Durfee87d, Durfee87e]. Moreover, some belief models may require large amounts of processing. Virtually all working DAI systems use abstracted and heuristic models of other agents rather than precise propositional models of belief such as those described by Konolige [Konolige86], Cohen and Perrault [Cohen79], Cohen and Levesque [Cohen87a, Cohen87b, Cohen87a], or Halpern [Halpern86a, Halpern86b], which are largely used for theoretical and analytical purposes. With better understanding and better algorithms, these more formal approaches may become more widely used for implementation.

5.3 How Should Knowledge of Others be Organized?

Knowledge of other agents can be organized relative to tasks, actions and processes which are occurring, relative to the agents being modeled, or using multiple perspective representations. For example, in the Contract Net, knowledge of task assignments and processing state is organized relative to tasks. The DVMT organizes knowledge of task assignments relative to goals of the processing being done, so it is easy to retrieve information about how goals are being accomplished. In the MACE DAI Testbed, each agent has an associative database of knowledge of other agents from which knowledge can be retrieved relative to tasks, goals, plans, skills, organizational roles, or agents, allowing for flexible and multidimensional modeling [Gasser87d, Gasser87c].

5.4 Problem Solving Process Knowledge

Process knowledge includes knowledge of problem-solving roles, roles in the multi-agent process, (e.g. constraints on an individual's potential activities as in a DVMT "organization"), knowledge of states and interactions (e.g. termination), and knowledge of behavior: how the world and beliefs will change. Process knowledge is important for and related to the *predictive knowledge* mentioned above. Wesson et al. in fact have suggested that the important aspects to model about other agents are those which will *change*, and that these can be recognized by modeling the world and looking for things which will force changes, such as obstacles, or the arrival of new relevant information [Wesson81]. To do this they proposed using "active hypotheses" which could update themselves along with the expected trajectories of change in the world, and periodically could check their conditions with respect to conditions in the real world. Thus messages should contain not just static data, but also dynamic procedures for changing that data over time, allowing agents to reduce communication. These dynamic procedures are in effect transportable process models of aspects of other agents or of the world.

Scripts are well known AI techniques for representing processes of activity involving several agents [Schank77]. Scripts encode knowledge of the agents involved, their respective roles, and temporal precedence of actions. Strangely, there has been little application of script-like structures in DAI research.

5.4.1 Representing control knowledge

By control knowledge, we mean knowledge which informs decisions about what activity to do next. In general, control knowledge makes reference to some aspect of the process state, and thus the state of the problem-solving or mutual activity must be represented (possibly in abstracted form) to reason about control. Control knowledge and process representation are deeply intertwined. With this definition, it is also clear that the distinction between control knowledge and other types of knowledge is sometimes difficult to recognize; domain specific problem data may influence control decisions. Lesser and Corkill mention two kinds of control knowledge in a DPS system: control within an agent (local control), and control of interactions among agents for coherence and coordination (network control). They note that there is an intimate relation between local control and network control if we hope to achieve truly distributed control; the global network performance must be derived from local control decisions. Internal control knowledge in the DVMT is represented in the form of "interest areas" which specify the type of tasks a node considers relevant to its interests.

Some representation of processes can encode control knowledge. If processes are represented as sequences of state changes, with control points and correctness criteria, the control points allow synchronization and the ability to express dependencies among agents (or between an agent and its environment) [Georgeff83]. Some explicit representations of control knowledge are found in global-memory blackboard systems in the form of scheduling tables and explicit *control blackboards* [Erman80, Lesser83, Hayes-Roth79a, Hayes-Roth79b, Hayes-Roth85, Nii86b, Nii86a]. In these systems, control is opportunistically linked to the system state as represented on the control blackboard or in scheduling rules.

5.5 Organizational knowledge

By organizational knowledge, we mean one of four types of knowledge: general principles of organization (to be used by self-designing and organizationally adaptive systems), knowledge of the organizational method used in the system, a set of default expectations about actions and beliefs of others, or a structural view of relationships among nodes, such as communication structure, authority, knowledge, etc.

5.5.1 Representing Organizational Knowledge

There are several approaches to representing organizational knowledge, such as:

Representation by the use of a particular architecture: Languages and shells which reflect particular architectures encode some implicit knowledge about the possible forms of interaction among problem-solvers, the possible roles each shall play, and how the roles and interaction can be modified. They include 1) the Contract Net Protocol, which organizes problem solvers into a strict virtual hierarchy - there is no lateral communication among several workers on related subtasks, and no preemption; 2) Blackboard shells such as Erasmus [Jagannathan87a], GBB [Corkill87, Corkill87] and BB1 [Johnson86, Hayes-Roth85, Hayes-Roth86], which organize interaction among expert knowledge sources sequentially, incorporate fixed problem-solving roles and

fixed communication paths, but allow temporal and control reconfiguration, using opportunism and control metaknowledge; 3) Concurrent object-oriented languages which provide maximally flexible interaction but no structure (see Section 7).

Representations for organization and interaction: Organizations are represented primarily in three ways. First, organizations are commonly seen as structural entities, and represented with graphs delineating functional roles or products connected in relationships of authority, communication, control, and information flow [Malone84, Malone87a, Malone87b, Malone88a, Pattison87]. Functional roles may be described constructively by listing capabilities, or by specifying constraints on activities of agents [Lesser83, Pattison87] or eligibility criteria for tasks [Davis83]. Second, they are represented as collections of tasks or actions in network, plan and process models such as Wesson et al.'s PAN, Gasser's Production Lattices, Bruce and Newman's social episodes, and the DVMT's Problem Solution Graphs, etc. [Bruce78, Corkill79, Durfee87d, Durfee87c, Gasser84, Gasser86, Gasser88b, Gasser88a, Wesson81], Third, organizations may also be described as collections of expectations, commitments, and/or behavioral defaults among agents [Barber80, Barber83, Fikes82, Gasser87d, Gasser88b, Gasser88a]. Finally, they may be represented as implicit knowledge encoded in agents' internal structures and capabilities, such as the agent structures found in [Agha85, Davis81, Davis83, Durfee87d, Erman75, Hewitt77a, Lenat75].

Some particular representation techniques include:

- Using delineated, fixed (but changeable) *roles and communication patterns.* For example, the EFIGE language represents particular organizations in the DVMT, by establishing fixed roles and communication patterns. In the DVMT, each agent has facilities for solving all problems; "roles" are actually static restrictions on the agents' capabilities and on data to which they will respond.

- MACE *acquaintance databases* contain models of other agents used in reasoning about interaction, and represent both implicit (goals and skills of agents) and explicit (named roles) organizational knowledge.

- The ICE problem-solving system uses localized *production lattice models* [Gasser86, Gasser88b, Gasser88a] which encode expectations of multi-agent interactions and which are used for default reasoning about the actions of other agents. Organizations are defined as hierarchically abstracted networks of settled and unsettled issues relating to the "basic organizational question:" *Who does what, when.* Temporary settlements provide expectations and default contexts for local decisions and action, but can be revised (with the right knowledge) to provide flexible organization.

- *Partial Global Plans* [Durfee87a, Durfee87d, Durfee87e] are representations of aggregated local plans of agents, with conflicts removed. They provide incrementally global or regional views of planned activity, enabling agents to allocate resources selectively.

- *Process Assembly Networks* (PANs). Introduced by Wesson, et al., PAN's are actually graphs of how the problem-solving processes will assemble partial results into global solutions, and are similar to the problem solution graphs of Hudlicka et al. [Hudlicka86a, Hudlicka86b]. Like PGPs and production lattices, PAN's enable agents to relate their planned local actions to the actions of others.

- *Problem-Solution Graphs* represent the actual set of steps which need to be taken (or which have been taken) by a set of agents to solve a particular problem, and can be seen as a goal network for the problem [Pavlin83, Pavlin84, Hudlicka86a, Hudlicka86b]. Problem-solution graphs may be generated statically or dynamically. In Pavlin and Hudlicka's work, actual problem-solution graphs were matched against optimal PSG's, to generate knowledge for analyzing performance, and for diagnosing and repairing problem-solving system failures.

6 Interagent disparities: uncertainty and conflict

Intelligent problem-solving agents must objectify portions of their world to reason about them. This objectification process is always subject to problems of abstraction and incompleteness, in part because no object or process can be fully described. Thus intelligent agents have to be able to cope with problems of disparity and uncertainty between their objectified representations and the affairs to which the representations refer. Sometimes these states of affairs involve other agents in the world; when agents act in coordinated ways, they (or some designer) must objectify each other as representations, and they must at some level align these representations to coordinate. This is a very difficult conceptual problem, in general, because alignment does *not* mean that they must have identical or "shared" representations, but rather that the representations must allow them to act so as to accomplish their individual ends. Establishing standard or common representations such as standard communication protocols or information exchange regimes (cf. [Durfee87a, Durfee87d, Durfee87e, Smith80]) seems like a reliable solution, but critically depends upon the agents having compatible ways of *interpreting* those protocols, and so on [Winograd86]. For example, in the DVMT, network nodes may compete or conflict because they must *locally interpret* the network goals [Durfee87d, Durfee87e]. It is the differing local interpretations which lead to competition or conflict (in the DVMT manifested as distraction, extraneous processing, or redundancy). As Hewitt has pointed out [Hewitt86], establishing a notion of semantics of an action as "effects on future behavior" rather than as changes in state eases the conceptual problem; Agha has developed such a semantics for concurrent Actor systems [Agha86a].

Nonetheless, in many extant DPS systems this is not a problem, because common interpretive frameworks and standard protocols are designed in. These systems still face problems of disparity, uncertainty, and conflict among agents, their beliefs, and the actual behaviors of the worlds they reason about. Agents may have knowledge bases in which beliefs are relatively *incomplete* (one KB contains some belief that another doesn't), *logically inconsistent*, *confidence inconsistent* (two KBs believe the same things to different degrees of certainty), or *incompatible* (two KBs use different representations). Incomplete viewpoints are evident in the DVMT (where each node has only partial information about global state), Contract Net (where each node has only partial information about potential contractors - those which are submitting bids) and Rand ATC [Steeb81] where each plane has incomplete viewpoint of total system airspace or plane plans. Incompatible viewpoints are rarely found in working DAI systems, largely because designers have installed common representational mechanisms in agents. For example, Lenat insisted on a common structure for all his agents in the PUP6 system, in order to combat the problems of representational incompatibility [Lenat75]. Inconsistent viewpoints are found for example in the plan conflicts studied by Georgeff and others [Georgeff83, Konolige80]. Inconsistencies in viewpoints in the Rand ATC [Steeb81] work were not explored - agents could have different and incompatible assumptions about each other's plans. Hewitt [Hewitt85, Hewitt86], Lenat and Feigenbaum [Lenat87], Gerson and Star [Gerson86], and others have proposed that *all* large-scale description systems have inherent inconsistencies, and distributed systems must be made to to reason with *local consistency* only.

Any disparity can lead to inter-agent goal conflict. An example of a goal conflict can be given for the Rand ATC system. The central (controlling or leader) plane may have an incomplete world view in having positions and routes for all planes except one. It may then generate a route plan for itself, based upon this world view. Another plane, unknown to the central plane, may have a disparate world view, possibly consisting of positions of other planes, but no knowledge of any

route plans except its own. This may lead it to generate subgoals incompatible with those of other planes, because its resulting route plan may be in conflict with that of the central plane. The Contract Net work on distributed sensing gives another clear illustration of disparity in goals and priorities, in this case for manager and worker nodes. A manager is interested in placing contracts with workers distributed in space over the monitoring region, to cover all subregions. A worker, on the other hand, is interested in bidding for contracts with managers whose communication distance is small, to preserve response time. This disparity can lead to global incoherence without effective policies for mediating the disparity.

Not all disparities must cause conflict. Indeed, there must be something different in agents' knowledge or structure to avoid duplication of work. If nodes have identical (i.e. overlapping or redundant) data, they must know which data is relevant to them, and which to another node, so as to avoid duplication [Lesser83]. The problem is really to keep the boundaries straight, and to allow and manage *appropriate differences* among agents.

Many agents pursuing duplicate, variant, different and even conflicting activities can be healthy, much like variety in a gene pool. Several authors have remarked on the potential benefits of alternative solution approaches. The use of alternative solution approaches has been a theme in literature on the fault tolerance of software in general. In the Rand ATC [Steeb81] plan-centered organization, alternative solution plans submitted by every plane are merged, (resolving their disparities by unspecified methods) to create the best overall plan. Similar merging occurs among partial global plans in the DVMT [Durfee87a, Durfee87d, Durfee87e]. Gasser [Gasser86] and Bond and Gasser [Bond87] have argued for multiple perspectives in organizational description, and for elucidating organizational problems from an examination of disparities among different perspectives. In research on scientific problem-solving, it has been argued, for example by Karl Popper, that it is essential for scientific progress that scientists hold conflicting explanations of phenomena. This promotes a natural selection mechanism for theories (but see dialectical theorists on this point). Kornfeld and Hewitt suggest that the scientific community is a useful metaphor organizing multi agent problem solvers [Kornfeld81b]. From their perspective, scientific research has three basic types of agent: *proposers* which put new theories, goals and techniques forward for critical assessment, and which also adjust theories to deal with anomalies pointed out by skeptics; *proponents* which try to substantiate new proposals; and *skeptics* which test proposals and attempt to establish inconsistency, anomalies, falsehood etc. These agents exchange two types of messages: conjectures that account for observations, and refutations which refute conjectures. New proposals are often based upon common sense knowledge, and sometimes on metaphor. Kornfeld and Hewitt's view is based on Popper's [Popper59] notion of *falsificationism*: no theory can be confirmed, but agents may believe it and rely upon it because a lot of agents have worked on it and none have overturned it (or at least overturning arguments have been successfully combatted or discredited). Adjustment of theories is an effort to protect a core of fundamental concepts which motivate work (Kuhn's "paradigm" [Kuhn70]).

6.1 Representing and Recognizing Disparities

Disparities can be recognized through processes of objectifying beliefs (fixing them by representing them), and comparing their representations. A prerequisite for reasoning about disparities among agents, then, is the ability to represent other agents' beliefs. A similar prerequisite for reasoning

about disparities in an agent's model of the world is the ability to create alternative models of the world - namely those produced, for example, by some sensing or execution monitoring component. Recognizing disparities requires representational compatibility. Two beliefs (or states) which are representationally indistinguishable are not disparate [Rosenschein87b].

Bruce and Newman [Bruce78] developed a *mutual belief space* mechanism for reasoning about disparate beliefs among agents. Each actor has a *belief space*, which includes (possibly nested) propositional beliefs of the forms: (bel P (bel Q R)) (meaning P believes that Q believes R). Nested beliefs may lead to arbitrary recursive levels, and so Bruce and Newman suggest dividing beliefs into two classes: those which are held commonly, and those which are disparate or incomplete. The mutual belief space is convenient, to avoid explicit representation of infinitely nested beliefs. The mutual belief space is used to represent the beliefs active in a cooperative interactive episode, i.e., social episode.

More recently, a number of researchers have used the paradigm of "reasoning about knowledge" to address problems of synchronization and control among concurrent processes, and specifically the notion of "common knowledge" which describes propositions which each agent knows, and which each agent knows everyone else knows. Halpern and Moses have shown the impossibility of achieving common knowledge when communication is not guaranteed, or when communication delay is not finite [Halpern84]. Their framework presumes common interpretive bases for representational structures, i.e., that a proposition refers to the same thing no matter which agent holds it.

Additional representational research on multiple belief spaces and the problems of common knowledge can be found in [Ballim86a, Ballim86b, Halpern84, Halpern86b, Konolige82, Konolige86, Wilks83, Wilks87]

6.2 Reconciling Disparities

To resolve disparities, agents must have some basis upon which they agree. Therefore, in open systems, not all disparities can be resolved. One way of establishing a basis is with a central or global controller with decisionmaking authority. In the Rand ATC work conflict resolution was assumed to be so difficult that negotiating a solution would be insurmountable in real time. This led Steeb et al. to cooperation regimes relying on centralized authority [Steeb81].

Given some bases (which in multiagent systems may have to be settled over longer reaches of time with negotiations at other levels), useful conflict resolution methods include those based upon power, where one agent can influence or dominate another, and those based upon agreed conflict resolution strategies, such as priorities set by convention or mediation procedures. Potential methods for conflict resolution include the following:

Abstraction of Common Frameworks: Any object or entity can be described to arbitrary levels of detail. Adding detail by *elaboration* or removing it by *abstraction and generalization* are two ways of realigning the representations of objects to make them compatible. Generalization is commonly used in machine learning systems for just this purpose. Finally, *translating* disparate representations into a common and comparable representation provides a basis for recognizing and removing conflicts. Approaches to natural language understanding, generation, and translation have relied on internal representations so that several disparate output sentences could be generated

from the same internal representation, and so that several input sentences could be represented using the same framework. In DAI, for example, common internode languages and interaction protocols such as the Contract Net Protocol [Smith80] or partial global plans [Durfee87a, Durfee87c] are common frameworks abstracted by designers into which possibly disparate views of situations, skills, goals, or plans can be fit for reconciliation.

Achieving Common Knowledge: For disparities which result from incomplete knowledge, the problem becomes identifying and communicating the appropriate knowledge to resolve the incompleteness. This may require reasoning about the knowledge state of separate agents; discussion of techniques and problems can be found in [Halpern84, Halpern86a, Halpern86b]. Achieving common knowledge among agents can be impossible in the face of communiction unreliability, as Halpern and Moses have illustrated [Halpern84].

Assumption Surfacing: Inconsistent propositions can sometimes be reconciled by backing up to the assumptions on which they rest to discover if the roots of disparity lie in assumptions. Disparities may be resolved at any point along this chain of support. This of course requires that an agent has (or can construct) knowledge of the supporting assumptions and their relationships. This is easiest in a system where shared and comparable knowledge representations are assumed, which is normally the case in DAI research. This assumption of sharing, however, has been challenged as impossible and unrealistic (see e.g. [Gerson86, Hewitt86, Suchman87, Winograd86]). Assumption Surfacing is an underlying technique found in many forms of argumentation and negotiation [Sycara85a, Sycara85b] and has been suggested as an organizational conflict resolution scheme by Mason and Mitroff [Mason69, Mason81]. It is the basis for "Truth Maintenance Systems" of Doyle and DeKleer [Doyle79, deKleer86], and is sometimes use as a basis of "belief revision." Distributed belief revision has been studied by Pearl [Pearl87]. Temporal and recency based approaches to resolving assumption conflicts are discussed in [Borchardt87].

Authority and Knowledgeable Mediation: Authority, possibly coupled with higher level knowledge can be used to to resolve conflicts. This can be seen in Hearsay-II [Erman80], where two KSs might set conflicting goals to solve the same problem. This conflict is then resolved by heuristic knowledge in the blackboard scheduler. Higher level knowledge can be centralized as in Hearsay-II, or distributed as in the Rand ATC case. In the latter, all planes could simultaneously and independently compute ratings of which goal to solve using conventionalized priorities. Similar shared conventions were used by planes in electing a new central planner. We view both centralized and decentralized conflict resolution strategies of this sort as mediation by higher authority, because decisionmaking rules are unchangeable by the agents whose conflicts are being resolved. It is important to note that, in general, rules and conventions are notoriously subject to local reinterpretation [Manning77]; in closed DAI systems this may not be the case, because reinterpretive capacity has been limited by design.

Constraint Resolution Conflicts and disparities which arise because of conflicting constraints can be resolved by relaxing those constraints, or by reformulating a problem to eliminate the constraints. Constraint relaxation requires prioritizing constraints. This process was elucidated by Goldstein in a scheduling domain by the use of preferences [Goldstein75]. Weaker preferences were relaxed first. More recently, Fox and others have studied constraints in scheduling for manufacturing [Fox84, Sathi86].

Evidential Reasoning and Argumentation: It may be possible to make arguments in

support of a particular perspective, sharing evidence. Evidence, methods, etc. may have to be justified, recursively. Hewitt and Kornfeld discuss argumentation as a basic method for organizing problem solving in [Kornfeld81a]. Modes of argumentation are discussed in [Alvarado86, Carbonell81, Gerson86, Hewitt86, Sycara85a, Sycara85b, Willer81]. Evidential reasoning is quite familiar in AI, particularly in the contexts of justifying conclusions made by knowldge-based systems to human users. Lesser and Corkill's *Functionally Accurate, Cooperative* (FA/C) processing [Lesser81] applied in the DVMT, supports convergence despite conflicting hypothesis by building evidence to strengthen the most favorable hypotheses. Related work on distributed evidential reasoning can be found in [Geffner87, Pearl82].

Goal Priority Conventions: Setting priorities on goals provides a way to reason about the impacts of particular disparities, and a way to make choices about which disparities are important enough to resolve. For example, conventions set by experiment designers are used in the DVMT to deal with the disparity of world views between two agents. A node distinguishes between internally and externally directed processes, and there is a (manually set) table of priorities for goals from different sources. Thus an agent had agreed priorities on all goals and could resolve conflicts. Similar mechanisms are incorporated in blackboard systems of many kinds by using scheduling metarules and scheduling knowledge sources which encode knowledge about settling conflicts among knowledge sources over which should be activated [Hayes-Roth85, Erman80, Nii86b, Nii86a]. These are typically called "conflict resolution mechanisms" for rule based systems.

Integration Without Conflict: For resolving conflicts of incompleteness, a simple approach is the integration of new knowledge by simple incorporation. This presumes common, compatible representations and mechanisms for adding new knowledge. Many DAI information exchange schemes use this method, including most of those discussed in work on common knowledge [Halpern86b].

Negotiation: Negotiation is often proposed in DAI research as a conflict resolution and information exchange scheme. Many of these models of negotiations fall short of providing detailed descriptions of the boundaries and assumptions of the processes of negotiations, and most make simple and computationally useful assumptions of negotiation procedures (e.g. what the bases, such as protocol or language, are). Discussions of varieties and models of negotiations can be found in [Strauss78, Pruitt81, Willer81]. Typically, negotiation involves some *context*, some (disparate) sets of *goals*, some (disparate) *information or knowledge* and some *procedure or protocol*. Negotiations depend upon bases, and in general may shift to multiple levels and to negotiating assumptions. Little DAI research has investigated the variety of negotiations. Davis and Smith, Durfee and Lesser, Gasser, Gerson and Star, Hewitt, and Sycara have pointed out the need for negotiated approaches for flexible coordination [Davis83, Durfee87a, Durfee87c, Gasser86, Gasser88a, Hewitt85, Hewitt86, Sycara87]. Recently there has been a resurgence of interest in topics in negotiation, including [Durfee87a, Durfee87c, Conry86]. Foundations for conflict resolution at various levels in negotiations may include all of the mechanisms discussed here, and much more work needs to go into control structures and flexible protocols for negotiation, especially those which treat negotiation as another type of goal directed activity.

Lewin and Moore's work on information-seeking, information-probing, instructing and griping "Dialogue Games" [Levin77] is one point where the context of negotiations is explicity considered. Dialogue games can represent negotiation in the sense of determining whether interaction should occur and in determining the roles of the participants as a stage of the interaction.

Davis and Smith [Davis83] viewed task distribution is viewed as an interactive process, treating it as a form of contract negotiation. In the Contract Net, negotiation referred to the the two way transfer of information, after which each party evaluated the information from its own perspective, Final agreement was achieved by mutual selection of a result (an allocation). As a basic framework for negotiation, the contract net was weak, as it incorporated no developed discussion of the nature of negotiation, such as finding common ground, judging of credibilities, weighing of priorities, compromising etc. This negotiation framework has been extended in new domains by Parunak and Ramamritham et al. [Parunak87, Ramamritham85].

Seeking Alternative Representations: Seeing a problem in a new way, by adding previously unknown knowledge, or changing the representational schema, may reconcile disparities. Also, exploring another (OR) branch of a goal graph may provide an alternative subproblem or operator description.

Standardization: The negotiation of alternatives over time, and the recognition of common requirements and routine behaviors, can lead to *standardization* as a basis mechanism for conflict avoidance. Over time, conflicts are resolved by common mechanisms and these are incorporated into standards and disseminated (or designed in). Conflicts are avoided by adherence to standards, but standards may impede adaptation [Durfee87d, Durfee87e, Lesser83], and they may be subject to local interpretation (cf. [Manning77]).

7 Tools for Distributed Artificial Intelligence

DAI researchers have built a variety of software tools which enable them to express solutions to the basic questions of DAI and to experiment with different approaches in different domains. There are several reasons why we are concerned with the particular tools currently being used. First, research tools help to verify theoretical insights through hard, real-world experimentation. Experimental studies have proven useful in some cases because of the difficulty of constructing complete theoretical analyses, and because some research issues (e.g., performance of problem-specific architectures) cannot practically be theoretically modeled due to their complexity. But in a more positive sense, experimentation is a useful way of generating new ideas and sometimes surprising results.

Some tools are designed to express ideas important to the domain. The ACTORS languages of Hewitt, Agha, and others [Agha85] are in part designed to capture the requirements of *open systems*. The MACE language includes explicit constructs for representing one agent's knowledge of the skills, roles, plans, etc. of other agents, to allow for reasoning about interaction. The blackboard system shells GBB and BB1 support the requirements of opportunistic problem-solving and integrated control knowledge [Corkill86, Hayes-Roth85]

The tools we survey here have each been driven by several of the following goals:

- To design and implement distributed object-oriented languages.
- To provide high-level environments for experimentation and simulation.
- To provide high-level knowledge representation schemes for representing world models, knowledge of other agents, control knowledge, etc.
- To provide a reusable shell for a particular problem-solving architecture.
- To provide a framework for integrating heterogeneous subsystems or different problem-solving architectures.
- To provide development tools for constructing large scale DAI systems, for experimentation or for applications.

Many of the DAI tools described in this section are implemented on serial machines. Often, tools designed for research experimentation simulate the concurrent communications and processing of a distributed system. Several of the tools are actually implemented on parallel architectures. ACTORS implementations exist for a network of LISP machines, and the highly concurrent Jellybean machine is now under development at MIT. In the DVMT, a simulator written in lisp simulates the concurrent execution of several DVMT nodes. The simulator itself is then distributed among a network of computers, providing a *distributed simulation of a distributed problem-solving system* [Durfee84a]. The USC MACE system is implemented on a heterogeneous network of parallel and sequential machines, including Lisp machines and a 16-processor Intel Hypercube.

There are basically four kinds of tools for DAI experimentation and development: Integrative Systems, Experimental Testbeds, Distributed, Object-Oriented Languages, and Paradigm-Specific Shells.

7.1 Integrative Systems

Integrative Systems provide the framework to combine a variety of paradigm-specific tools and methods into a useful whole. ABE [Erman85] is designed to be a framework for integrating a number of heterogeneous, independently developed problem-solving paradigms and software tools. It provides for several different subsystems, each possibly using a different method, to constitute a DAI system and to communicate. ABE is the glue that holds other systems together. It provides a window-based and menu-based front-end, and interfacing software.

AGORA [Bisiani85] is designed to be an opaque, high-level operating system. It has been designed for projects in speech understanding, and so that heterogeneous hardware systems could be integrated under a common operating system. In AGORA, the user software layer describes data structures and processes, and the operating system distributes these onto the appropriate portions of the distributed hardware.

MACE [Gasser87d, Gasser87c] is a generic testbed for building Distributed AI systems of varying levels of granularity. MACE allows for integration of different problem-solving and communication structures by supplying programmers with a collection of facilities (e.g., pattern-matchers, remote demons) and system agents (e.g. allocators, user-interfaces, command interpreters, dictionaries) and allowing arbitrary message formats and interpretations, using a user-supplied "engine" for each agent.

7.2 Experimental Testbeds

Several systems have been tailored for controlled experiments. This requires them to support both *measurement* and *monitoring*. It may also require *simulation*, if experiments must be exactly repeatable. Experimental testbeds for DAI research provide parameterization of control variables and experimental conditions. The DVMT and MACE systems are explicitly tailored for controlled, parameterized experimentation using simulation and instrumentation.

7.3 Distributed, Object-Oriented Languages

The family of languages which are coming to be termed *distributed, object-oriented languages* (DOO languages) are a natural framework for implementing truly concurrent distributed AI systems. These languages typically describe data and procedural abstractions in objects, and allow for inter-object interaction using message communication. Language processors and underlying kernels implement allocation, load-balancing, addressing and message routing schemes invisible to programmers. Synchronization primitives may be included as language constructs or left for the programmer to implement using messages.

A family of actor languages [Agha85] have been designed at MIT. The actor languages ETHER, PLASMA, ACT, ACT1, and ACT3 have been research attempts to implement systems for experimental programming using actors, and to our knowledge have not been applied to serious application developments. ABCL1[Yonezawa86] is in part an attempt to make actor-like languages more computationally practical, given current technology. As a consequence, not all activity is

concurrent in ABCL/1. The basic objects are serial at some medium grain of system granularity.

Though ACTORS is an elegant computational model, ACTOR-based systems appear to have some important practical limitations under current technology. In any Actor system, many messages are sent, many new actors are spawned, necessitating large and very dynamic routing tables and large amounts of overhead. Dynamic load balancing involves moving actors and forwarding messages, adding more overhead. These problems force the question of whether the level of concurrency (i.e. *total* is appropriate for current hardware technology. The answer of ABCL/1, MACE, and ORIENT84-K seems to be that we probably need to introduce larger granularity to build large-grain DAI systems with practical performance.

Of the other DOO languages, both OIL [Cohen85] and ORIENT84 [Tokoro84] are attempts to integrate logic programming and object-oriented programming in languages for message-passing distributed systems. VULCAN employs "logical concurrent objects" [Huberman88a]. MACE is most similar to ABCL/1, with the added capability that programmers may describe message formats and interpretations (i.e. they need not be patterns). ABCL/1 provides synchronization, while MACE relies on program-level synchronization using messages.

7.4 Blackboard Systems

Blackboard systems provide a central data structure called a *blackboard*, which is often divided into regions or *levels*. A collection of independent processes called *knowledge sources* may read and write one or more levels, under the supervision of a *control system*, which may be a synchronous global scheduler, a system of concurrency locks, or a collection of integrated control knowledge sources [Hayes-Roth85, Fennell77]. In the latter case, the blackboard is used for both problem-solving and control. Several DAI systems surveyed in this book have incorporated blackboard architectures (e.g., [Hayes-Roth85, Corkill86, Hayes-Roth79a, Lesser83, Yang85]). While numerous specialized blackboard systems have been built, the BB1 [Hayes-Roth85] and GBB [Corkill86] systems are high-level domain-independent blackboard shells.

Some of the key issues which blackboard-based systems must address include indexing schemes for retrieving knowledge from levels, granularity of representations, synchronization and conflict resolution among knowledge sources, and granularity of actions of the knowledge sources.

7.5 Remaining Problems

Several problems remain for the next generation of languages and systems for DAI. These include:

- *Facilities to represent knowledge of other agents and of the world of the agent.* Currently two languages have simple mechanisms to this - the actor languages and MACE. Actor languages have rather simple models of other agents: references to their acquaintances introduced the notion of acquaintance. MACE provides for more complex models, including the beliefs, plans, goals, location, and organizational roles of other agents. The DVMT incorporates models of other agents - the plans and activities of other nodes. However, there are no explicit language constructs for building this knowledge of other agents; it is simply programmed in.

- *Support for multi-grain problem-solving.* How to decide about *granularity* of agents in a DAI system is an important issue related to the basic problems of task decomposition and allocation. Languages and testbeds ought to accommodate agents of varying granularity for experimental purposes.

- *Integrating knowledge-base access with DAI.* The blackboard shells GBB and BB1 incorporate high-level knowledge structuring and access mechanisms, and pattern-directed retrieval of data. MACE provides associative databases within agents. But in general the problem of giving agents access to large and possibly shared knowledge bases is difficult.

- *Better development environments.* Understanding the operation of a concurrent system is extremely difficult; for large-scale experimental DAI systems with changing requirements and architectures, debugging, development, and evaluation are very hard. We need more automated (and intelligent) support for modeling, analyzing, and controlling the execution of concurrent DAI systems. This should include tools and support for profiling, tracing, and debugging systems of agents with confilcting behaviors and disparate world views.

9 Open Problems

In this survey, we have outlined a number of considerations in current Distributed AI systems and research. We believe that some of the more interesing open problems and avenues for research in Distributed AI include the following:

- More research needs to be done on linking theoretical results in distributed computing systems to DAI problems and approaches (e.g. fairness and starvation in market-based system organizations).
- We need more study of how to generalize and extend existing problem solvers and approaches to new domains - this is just beginning (cf. Parunak's and Ramamrithm et al.'s CNET extensions [Parunak87, Ramamritham85]).
- There has been very little research into the knowledge and structures required for automated task or problem description and decomposition. These will be necessary if agents in DAI systems are to begin jointly to construct and recognize their own problems. This knowledge would be useful, for example, for self-diagnosis, organizational self-design, adaptive control, and automated learning and discovery.
- There are interesting possibilities for research in collaborative and pluralistic learning among collections of agents, including the use of multiple perspectives to refine concepts, refinement of the organizational knowledge and behavior of agents, etc.
- We can envision greater connection between economic approaches to coherent behavior, resource allocation, and task distribution than has been exploited in contemporary DAI research.
- Large collections of intelligent agents, person machine collaborative systems, or any group of agents without global perspectives in design or knowledge, will be subject to *open systems* problems. Generating new knowledge and solving problems using mechanisms varieties of disparity resolution approaches such as *due process* deserve greater attention.
- The problems of building and diagnosing the behavior of collections of automated intelligent agents need greater attention. Most researchers with experience in experimental DAI projects, and especially those which involve actual concurrency (as versus simulation) can attest to the great difficulty of understanding and modeling the actual behavior of such systems. We need better models, abstraction mechanisms, and representations for concurrent activity.
- Much greater attention should be focused on the problems of method in DAI. The usefulness of metaphors, methods, and models from other fields such as sociology or economics needs to be carefully examined. The assumptions underlying the perspectives and methods we use and those we import need careful articulation and criticism. The research we do needs to include clear and carefully used criteria for evaluation, and tests which employ them.

Research on many of these open problems will interlock, and progress on one will often aid progress on the others. Research on many of these problems will be inherently dependent on integrating multiple perspectives and new ideas from other disciplines as "close" as distributed computing and

possibly as "distant" as history. We echo the voices of Chandrasekaran, Lesser and Corkill, Wesson et al. and others when we express the exciting potential for interaction between researchers and research results in DAI and cognate fields.