Agent-Based Computing

MICHAEL WOOLDRIDGE

Dept. of Electronic Engineering, Queen Mary & Westfield College
University of London, London E1 4NS, UK
E-mail: m.j.wooldridge@qmw.ac.uk

This article is an introductory survey of *agent-based computing.* The article begins with an overview of *micro*-level issues in agent-based systems: issues related to the design and construction of individual intelligent agents. The article then goes on to discuss some *macro*-level issues: issues related to the design and construction of agent *societies.* Finally, the key application areas for agent technology are surveyed.

1 Introduction

It is probably quite rare for a software technology to seize the imagination of the computer science community at large. And yet this is precisely what has happened with *intelligent agents and multi-agent systems.* In the past few years, interest in agents has grown at an astonishing rate, and while the current hype means that there is sure to be a backlash eventually, this backlash is not yet in evidence.

The aim of this article is to survey some key research issues and developments in the area of intelligent agents and multi-agent systems. While an article like this cannot hope to act as an introduction to *all* the issues in a field as rich and diverse as multi-agent systems, the aim is nevertheless to point the reader at the main areas of interest. Note that the article is intended as an *introduction*, not as a specialist, advanced survey.

The article starts — inevitably — by asking the question *what is an agent*? This leads to a brief discussion on the topic of what sort of computer systems are most appropriately conceived and implemented as multi-agent systems. A crude classification scheme is then introduced, whereby the issues relating to the design and implementation of multi-agent systems are divided into *micro* (agent-level) issues and *macro* (society-level) issues. In section 2, micro-level issues (essentially, what software structure should an agent have?) are discussed in more detail. Traditional symbolic AI architectures for agents are reviewed, as well as alternative, *reactive* architectures, and finally, various *hybrid* architectures. One particularly well-known agent architecture is discussed in detail: the *Procedural Reasoning System* (PRS) [50]. Section 3 considers macro, or society-level aspects of multi-agent systems. It begins by identifying a research area known as cooperative distributed problem solving (CDPS), and goes on to consider the issues of *coordination* and *coherence*. This section also mentions issues such as communication and negotiation. Section 4 discusses the applications to which the technology of intelligent agents and multi-agent systems seems likely to be applied. Finally, in section 5, some concluding remarks are presented.

1.1 What are Agents?

Like the question *what is intelligence?* in mainstream AI, the question *what is an agent?* is the most frequently-asked question in multi-agent systems research. Everyone working in the field has their own interpretation of the term, and their own ideas about what the important issues are. Rather than enter a debate about the issue, we shall here simply give an example scenario describing computer systems that it seems useful to think of as agents (this example is borrowed from [126]):

> The key air-traffic control systems in the country of Ruritania suddenly fail, due to freak weather conditions. Fortunately, computerized air-traffic control systems in neighbouring countries negotiate between themselves to track and deal with all affected flights, and the potentially disastrous situation passes without major incident.

The computer systems — agents — operating in this scenario ...

- ... *are situated in a constantly changing* environment;

  Unlike the theorem provers and expert systems of early AI research, agents operate both *in* and *on* some environment, which may be (in this case) the world of air-traffic control, the real world (in the case of a physically embodied robot), the INTERNET (in the case of network agents [118]), or a software environment such as UNIX (in the case of softbots [33]).

- ... *have only partial, possibly incorrect information about the environment, and are able to make (at best) limited predictions about what the future will hold*;

  The agents in this example are able to perceive their environment through radar and radio contact with pilots; clearly, the information they obtain in this way, (particularly about variables like the weather), is limited and prone to error. Moreover, any predictions the agent makes (e.g., about tomorrow's weather) will be liable to errors.

- ... *are able to act upon the environment in order to change it, but have at best partial control over the results of their actions*;

  The agents in this scenario do not have complete control over their environment: in particular, they have no control over 'nature' (in the form of, for example, weather). They do have *limited* control over *aspects* of the environment (in particular, they can instruct pilots about what course to fly, altitude, speed, and so on), but they cannot rely upon this control being perfect (pilots can ignore or misunderstand instructions).

- ... *have possibly conflicting tasks they must perform*;

  More than one re-routed aircraft may wish to land on the same runway at the same time. In circumstances such as this, where the agent cannot achieve all the tasks it has been allocated, it must fix upon some subset of these tasks and *commit* to realizing them (see the discussion on BDI architectures in section 2).

- ... *have available many different possible courses of action*;

  There will typically be many different ways that an agent can achieve its tasks; the agent must select some procedures, that it believes will achieve its selected tasks, and commit to performing them. Moreover, the agents should pick the procedures that are in some sense the *best*.

- . . . *are required to make decisions in a timely fashion*.

  Real agents do not have infinite resources: the world changes in real time, and agents must make the best (most *rational*) decisions possible, given the resources (information, time, computational power) available to them [91].

(These points are borrowed from [84, p313], where they are discussed in more detail.) It should be clear to anyone with more than a passing appreciation of software engineering that the design and implementation of computer systems that can operate under these constraints is *extremely* difficult. Some researchers (e.g., [84]) believe that such computer systems may usefully be considered as *multi-agent* systems, and that the tools and techniques of multi-agent systems research may fruitfully be used to develop them. Clearly, the scenario above is rather extreme (it is hard to think of many more difficult computer systems to build!), but the agent concept is likely to be useful even in domains that do not exhibit such extreme properties; some of these domains (such as user interfaces) will be discussed in section 4.

Let us try to identify the key properties enjoyed by agents such as those in the scenario above [126]:

- *autonomy*: agents operate without the direct intervention of humans or others, and have control over their actions and internal state;
- *social ability*: agents are able to cooperate with humans or other agents in order to achieve their tasks;
- *reactivity*: agents perceive their environment, and respond in a timely fashion to changes that occur in it;
- *pro-activeness*: agents do not simply act in response to their environment, they are able to exhibit goal-directed behaviour by *taking the initiative*.

For me, an intelligent agent is a system that enjoys these properties. Other researchers argue that different properties should receive greater emphasis. Some other properties discussed in the context of agency are:

- *mobility*: the ability of an agent to move around an electronic network [118];
- *veracity*: the assumption that an agent will not knowingly communicate false information [43];
- *benevolence*: the assumption that agents share common goals, and that every agent will therefore always try to do what is asked of it [88];
- *rationality*: the assumption that an agent will act in order to achieve its goals, and will not act in such a way as to prevent its goals being achieved — at least insofar as its beliefs permit [92];
- *learning*: the assumption that an agent will adapt itself to fit its environment.

For others — particularly those working in AI — the term ‘agent’ has a stronger meaning. They mean an agent to be a computer system that, in addition to having the properties specified above, is either designed or implemented in terms of concepts more usually applied to humans. One manifestation of this idea, called *agent-oriented programming*, is discussed in section 2.

1.2 Micro and Macro Issues

If one aims to build a computer system such as that indicated in the example scenario, then there are at least two sorts of issues that one needs to address [51, p145]:

- *Micro-issues.* How does one build an agent that has the kind of properties listed above? This is the area that has become known as *agent architectures*.
- *Macro-issues.* How does one design an agent society that can (co-)operate effectively? This latter area has been the primary focus of research in *Distributed AI (DAI)* [55, 10, 46].

These two sets of issues are by no means disjoint: the traditional AI approach to building a system that can operate intelligently in some environment proposes giving that system some symbolic representation of the environment. If an agent is to co-operate effectively with other agents, therefore, one might wish to give that agent a symbolic representation of the cooperative process (see, e.g., Jennings' *cooperation knowledge level* [56], and the co-operation level in the INTERRAP architecture [79]).

The micro/macro distinction has been criticized by a number of researchers. One objection, for example, is that agents can be comprised of a number of other agents, in the same way that any complex system can be decomposed into a number of other subsystems [54]. The micro/macro distinction makes little sense if one takes such a view. As the preceding paragraph illustrated, there is certainly a somewhat grey area between micro and macro issues. However, for the purposes of this review, it seems a useful classification scheme.

2 Micro (Agent Level) Issues

This section presents a short survey of the area known as *agent architectures*. Researchers working in this area are concerned with the design and construction of agents that enjoy the properties of autonomy, reactivity, pro-activeness, and social ability, as discussed earlier. Following [126], three classes of agent architecture are identified: deliberative, or symbolic architectures, are those designed along the lines proposed by traditional, symbolic AI; reactive architectures are those that eschew central symbolic representations of the agent's environment, and do not rely on symbolic reasoning; and finally, hybrid architectures are those that try to marry the deliberative and reactive approaches.

2.1 Traditional AI — Symbolic Architectures

The traditional approach to designing agents is to view them as a type of knowledge-based system. Such architectures are often called *deliberate* or *deliberative*. The key questions to be answered when developing an agent using traditional knowledge-based systems techniques are thus what knowledge needs to be represented, how this knowledge is to be represented, what reasoning mechanisms are to be used, and so on.

The symbolic approach to building agents has traditionally been very closely associated with the *AI planning* paradigm [49]. This paradigm traces its origins to Newell and Simon's GPS system [81], but is most commonly associated with the STRIPS planning system ([38]) and its descendents (such as [93, 94, 19, 119]). A typical STRIPS-style planning agent will have at least the following components:

- a symbolic model of the agent's *environment*, typically represented in some limited subset of first-order predicate logic;
- a symbolic specification of the actions available to the agent, typically represented in terms of PDA (pre-condition, delete, add) lists, which specify both the circumstances under which an action may be performed and the effects of that action;
- a planning algorithm, which takes as input the representation of the environment, a set of action specifications, and a representation of a goal state, and produces as output a plan — essentially, a program — which specifies how the agent can act so as to achieve the goal.

Thus, planning agents decide how to act from *first principles*. That is, in order to satisfy a goal, they first formulate an entirely new plan or program for that goal. We can thus think of planning agent continually executing a cycle of picking a new goal ϕ_1, generating a plan π for ϕ_1, executing π, picking a new goal ϕ_2, and so on. Unfortunately, first-principles planning of the type just described has associated with it a number of difficulties. The most obvious of these is that the processes of finding a goal, generating a plan to achieve it, and executing it are not atomic: they take time — in some cases, a considerable amount of time. Yet there is an assumption implicit in this scheme, that the environment in which the agent is situated does not change so as to invalidate the pre-conditions of the plan either while the plan is being formed or while it is being executed. Clearly, in any even moderately dynamic domain, this assumption simply will not hold.

There are a number of theoretical results which indicate that first-principles planning is not a viable option for agents that operate in time-constrained environments. For example, Chapman demonstrated that in many circumstances, first-principles planning is *undecidable* [19]. So, building *reactive* agents, that can truly respond to changes in their environment, is not likely to be possible using first-principles planning techniques. Chapman's negative results, (and other intractability results), have caused many researchers to look for alternative paradigms within which to construct agents; such attempts are reviewed below.

Despite the negative results of Chapman and others, planning is still regarded as an important ability for agents, and many attempts have been made to build planners that, for example, interleave planning, plan execution, and monitoring of plans. One example of such a system is Ambros-Ingerson and Steel's IPEM system [4]. Another example is Vere and Bickmore's HOMER system, in which a simulated submarine agent is given English-language instructions about goals to achieve in a changing environment [110].

As well as AI planning systems, a great deal of work has been carried out on deliberative agents within the paradigm known as *agent-oriented programming*. Within AI, it is common practice to characterize the state of an agent in terms of *mentalistic* notions such as *belief*, *desire*, and *intention*. The rationale is that these notions are *abstraction tools* used by us in everyday language to explain and predict the behaviour of complex intelligent systems: people. Just as we can use these notions to explain the behaviour of people, so, the reasoning goes, we can use them to predict, explain, and, crucially, even *program* complex computer systems. This proposal, in its best-known form, was made around 1989 by Yoav Shoham, building on ideas from John McCarthy and others [97, 98], and is known as *agent-oriented programming* (AOP). In order to demonstrate the AOP paradigm, Shoham defined a simple experimental language, AGENT0. This language allowed one to specify the behaviour of agents in terms of rules that define how an agent generates *commitments* from the messages it receives and beliefs it holds. The agent continually executes a cycle of receiving messages, updating beliefs, generating commitments, and attempting to discharge current commitments. Building on Shoham's work, many others have

developed agent programming environments based on similar ideas. For example: Becky Thomas described an extension to AGENT0 called PLACA [106, 107]; Wooldridge and Vandekerckhove developed an AGENT0-like multi-agent testbed called MYWORLD [123]; Poggi described an agent-oriented extension to the CUBL concurrent object language [83]; Weerasooriya *et al* discuss a (hypothetical) agent-oriented programming language called AGENTSPEAK [115]; Fisher has developed a multi-agent programming environment based on executable temporal logic, which enjoys many of the properties of AOP [41, 42]; Burkhard discusses some issues in the design of agent-oriented environments [17]; and finally, Lespérance *et al* describe a logic-based multi-agent programming environment called CONGOLOG [67], which incorporates many ideas from AOP.

2.2 Reactive Architectures

In addition to the criticisms outlined above, there are many other objections to symbolic to AI and deliberative agents, some of which have led researchers to seek alternative approaches for building agents. Perhaps the best-known proponent of this 'alternative AI' is Rodney Brooks, who, starting in about 1985, began development of the *subsumption architecture* [12, 13, 15, 14]. The basic idea of the subsumption architecture is to design an agent as a set of *task accomplishing behaviours*, arranged into a *subsumption hierarchy*. Each task behaviour is implemented as a simple finite-state machine, which directly maps sensor input to effector output. The layers interact with each other via suppression and inhibition actions. For example, a lower layer in the hierarchy, representing low-level behaviour (such as avoiding obstacles) may suppress higher layers that represent more abstract behaviours (such as exploring or avoiding obstacles). The process of designing an agent becomes one of systematically adding and experimenting with behaviours.

It should be stressed that Brooks' systems do no symbolic reasoning at all. They are, in a sense, extremely simple in computational terms. And yet experiments have shown that agents implemented using Brooks' scheme can achieve near optimal results [105].

A number of other researchers have developed approaches to agent design that borrow from Brooks' work. Some well-known examples are Pattie Maes' agent network architecture [73, 74], and the ABLE and REAL-TIME ABLE (RTA) languages developed at Philips research labs [24, 112, 113]. Other approaches to reactive architectures are [63, 40, 3, 90]; the book edited by Maes contains many relevant papers and references [72].

2.3 Hybrid Architectures

Some researchers strongly believe that the traditional, symbolic approach is the best candidate for the future development of AI; others (such as Brooks) just as strongly assert that symbolic AI is a dead end, and that alternative approaches are required. Still others accept that both arguments have their merits, and suggest that the best direction for future research is to try to marry the two styles of architecture. This has led to work on *hybrid architectures*.

Perhaps the best-known agent architecture is the *Procedural Reasoning System (PRS)* developed by Georgeff *et al* [50]. The PRS is illustrated in Figure 1. The PRS is an example of a currently popular paradigm for agent design known as the *belief-desire-intention* (BDI) approach [11]. As this figure shows, a BDI architecture typically contains four key data structures. An agent's *beliefs* correspond to information the agent has about the world, which may be incomplete or incorrect. Beliefs may be as simple as variables, (in the sense of, e.g., PASCAL programs), but implemented BDI agents typically represent beliefs symbolically (e.g., as PROLOG-like facts [50]). An agent's *desires* intuitively correspond to the tasks allocated to it.

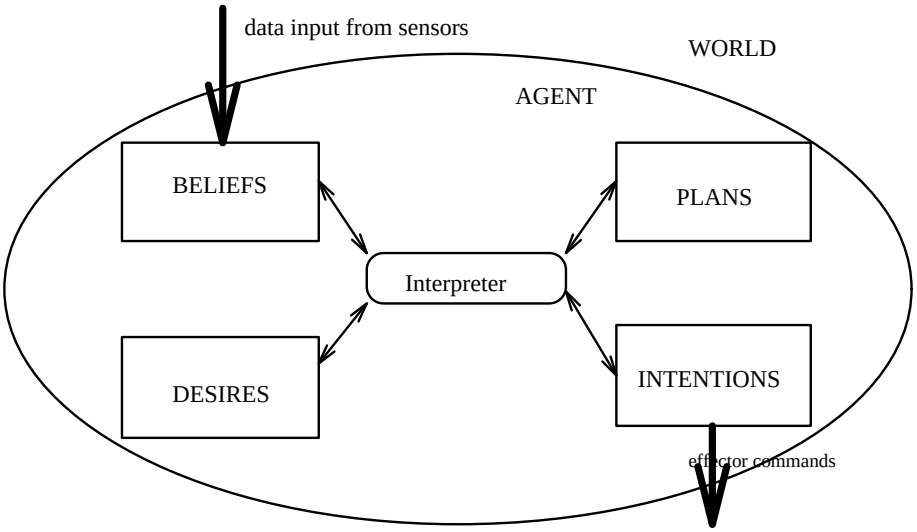

Figure 1: The PRS — A BDI Agent Architecture

(Implemented BDI agents require that desires be logically consistent, although *human* desires often fail in this respect.)

An agent's *intentions* represent desires that it has committed to achieving. The intuition is that an agent will not, in general, be able to achieve *all* its desires, even if these desires *are* consistent. Agents must therefore fix upon some subset of available desires and commit resources to achieving them. These chosen desires, which the agent has committed to achieving, are *intentions* [20]. An agent will typically continue to try to achieve an intention until either it believes the intention is satisfied, or else it believes the intention is no longer achievable [20].

The final data structure in a BDI agent is a *plan library*. A plan library is a set of plans (a.k.a. *recipes*) which specify courses of action that may be followed by an agent in order to achieve its intentions. An agent's plan library represents its *procedural knowledge*, or *know-how*. A plan contains two parts: a *body*, or *program*, which defines a course of action; and a *descriptor*, which states both the circumstances under which the plan can be used (i.e., its pre-condition), and what intentions the plan may be used in order to achieve (i.e., its post-condition). PRS agents do no first-principles planning at all.

The *interpreter* in Figure 1 is responsible for updating beliefs from observations made of the world, generating new desires (tasks) on the basis of new beliefs, and selecting from the set of currently active desires some subset to act as intentions. Finally, the interpreter must select an action to perform on the basis of the agent's current intentions and procedural knowledge.

In order to give a formal semantics to BDI architectures, a range of *BDI logics* have been developed by Rao and Georgeff [85, 86]. These logics are extensions to the branching time logic CTL* [31], which also contain normal modal connectives for representing beliefs, desires, and intentions. Most work on BDI logics has focussed on possible relationships between the three 'mental states' [85], and more recently, on developing proof methods for restricted forms of the

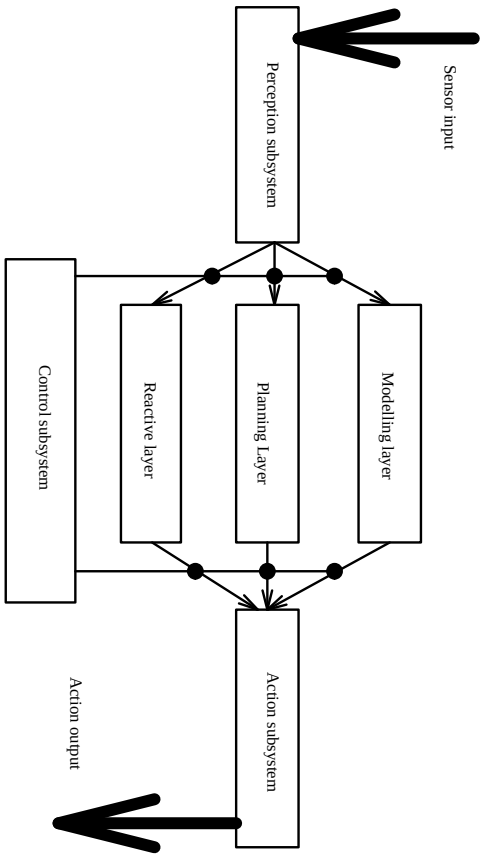

Figure 2: TOURINGMACHINES: A Layered Agent Architecture

logics [86]. In related work, attempts have been made to graft a logic of plans onto the basic BDI framework, in order to represent an agent's procedural knowledge [87, 64].

An increasingly popular approach to designing hybrid agents is to use a *layered* architecture. The idea of layered architectures is well-established in traditional systems control theory; in AI, it has been popularized by the work of Brooks (see above) and Kaelbling [63], among others. The basic idea of a layered architecture is that of integrating different agent control subsystems by *layering* them. Figure 2 illustrates a good example of a layered agent architecture: Innes Ferguson's TOURINGMACHINES [36, 35, 37]. As this Figure shows, TOURINGMACHINES consists of three *activity producing layers*. That is, each layer continually produces 'suggestions' for what actions the agent should perform. The *reactive layer* provides an immediate response to changes that occur in the environment — it is implemented as a set of situation-action rules, like the behaviours in Brooks' subsumption architecture. These rules map sensor input directly to effector output. The planning layer contains a plan library (much like the PRS — see above), which the agent can use in order to achieve goals. The *modeling* layer represents the various entities in the world (including the agent itself, as well as other agents); the modeling layer predicts conflicts

between agents, and generates new goals to be achieved in order to resolve these conflicts, which are posted down to the planning layer, which then determines how to satisfy them. The three layers are embedded within a control subsystem, which is responsible for deciding which of the layers should have control over the agent.

Another example of a layered architecture is INTERRAP [79]. Like TOURINGMACHINES, the INTERRAP architecture has three layers, with the lowest two layers corresponding fairly closely to Ferguson's reactive and planning layers. However, the highest layer in INTERRAP deals with *social* aspects of the system: it not only models other agents, but also explicitly represents social activities (such as cooperative problem solving). The 3T architecture is another layered architecture, that has been used in several real-world robotic agents [9].

3 Macro (Societal) Issues

So far, this article has considered only the issues associated with intelligent agent design. We have been concerned with *individuals*, but not with *societies*. The design of 'an agent' is a classic AI pursuit: in a sense, this is what the whole of the AI endeavour has been directed towards. However, AI has traditionally not considered the *societal* aspects of agency. In the late 1970s and early 1980s, these aspects of agency began to be studied in a subfield of AI known as Distributed AI (DAI). In this section, our aim is to briefly review work in DAI: we begin by considering the distinction between distributed problem solving and multi-agent systems, and go on to examine the issues of coherence and coordination, communication, cooperation, and negotiation. The best reference to these various issues is the collection of papers edited by Bond and Gasser [10]; the first two subsections that follow borrow from the survey article that opens [10].

3.1 Distributed Problem Solving and the Contract Net

Early research in Distributed AI focussed on problem solving by a group of logically distinct computational systems (agents), that 'cooperate at the level of dividing and sharing knowledge about the problem and its developing solution' [10, p3]. This research area is known as distributed problem solving (DPS). It is *not* assumed that the computing components in DPS systems are autonomous agents, in the sense described earlier. Thus these components are often assumed to be 'benevolent': willing partners to the development of a solution, typically designed by the same individual. In DPS, the main issues to be addressed include: How can a problem be divided into smaller tasks for distribution among agents? How can a problem solution by effectively synthesized from sub-problem results?

The best-known framework for distributed problem solving is the *contract net*, developed by Reid Smith for his doctoral thesis [100, 102, 101, 103]. The contract net proposes the following process for distributed problem solving. Problem solving begins when some agent has a problem that it either cannot solve on its own (for example, because it lacks the expertise), or would prefer not to (for example, because it would take too long) — see Figure 3(a). This agent (A1 in Figure 3) then divides the problem up in to sub-tasks (the mechanism via which this is achieved is not considered part of the contract net). The agent must then *advertise* the task(s), by doing a *task announcement* (Figure 3(b)). A task announcement is a message broadcast to a group of agents, which contains a specification of the task to be solved. Again, the exact form of the specification is application specific, and is not considered part of the protocol. Every agent, upon receiving the

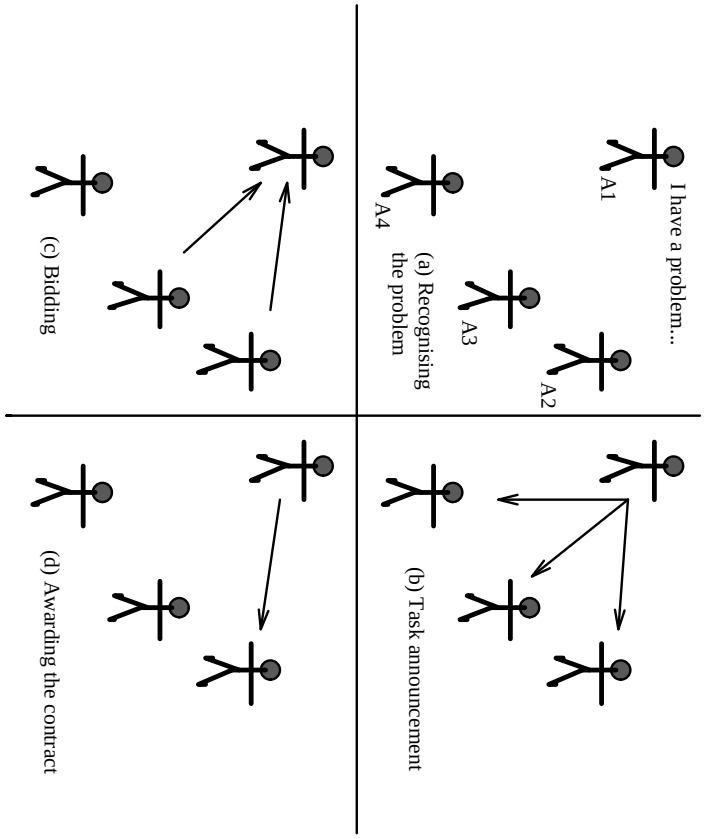

Figure 3: The Contract Net Protocol

task announcement, decides whether or not to *bid* for the task. If the agents are considered to be autonomous, then this stage involves every agent deciding whether or not it would like to perform the task on behalf of the announcer. If an agent *does* decide it would like to perform the task for the announcer, it must then decide whether or not it is *capable* of performing the task. This is done by matching the task specification encapsulated within the announcement against the agent's own abilities. Agents who reach the conclusion that they *do* wish to perform the task then make this known to the announcer by submitting a *bid*. A bid may simply indicate that the announcer is willing to do the task, or it may contain other information (such as how soon the bidder is able to expedite the task, what kind of solution quality is expected, and so on). The bidding process is illustrated in Figure 3(c), where agents A2 and A3 bid for the task announced by A1; agent A4 decides not to bid, either because it does not want to help A1, or else because it is unable to help. Finally, the agent that originally announced the task must decide which agent it wants to perform it: once it does this, the relevant agent must be 'awarded the contract' (in Figure 3(d), agent A2 is awarded the contract).

Despite (or perhaps because of) its simplicity, the contract net has become the most im-

plemented and best-studied framework for distributed problem solving. Many variations on the contract net theme have been described, including the effectiveness of a contract net with 'consultants', which have expertise about the abilities of agents [108], and a sophisticated variation involving marginal cost calculations [95]. Several formal specifications of the contract net have been developed, using basic set theoretic/first-order logic constructs [117], temporal belief logics [121], and the Z specification language [27]. Surveys of distributed problem solving techniques may be found in [30, 26].

In contrast to work on distributed problem solving, the more general area of *multi-agent systems* has focussed on the issues associated with societies of *autonomous* agents, (i.e., agents with the type of properties listed in section 1.1). Thus agents in a multi-agent system, (unlike those in typical distributed problem solving systems), cannot be assumed to share a common goal, as they will often be designed by different individuals or organizations in order to represent their interests. One agent's interests may therefore conflict with those of others, just as in human societies. Despite the potential for conflicts of interest, the agents in a multi-agent system will ultimately need to cooperate in order to achieve their goals; again, just as in human societies. Multi-agent systems research is therefore concerned with the wider problems of designing societies of autonomous agents, such as why and how agents cooperate [125]; how agents can recognise and resolve conflicts [1, 43, 44, 65, 66]; how agents can negotiate or compromise in situations where they are apparently at loggerheads [32, 89]; and so on.

Having implemented an artificial agent society, how does one assess the success (or otherwise) of the implementation? What criteria can be used? Distributed AI has proposed two types of issues need to be considered:

- *coherence* refers to 'how well the [multi-agent] system behaves as a unit, along some dimension of evaluation' [10, p19]; coherence may be measured in terms of solution quality, efficiency of resource usage, conceptual clarity of operation, or how well system performance degrades in the presence of uncertainty or failure; a discussion on the subject of when multiple agents can be said to be acting coherently appears as [122];

- *coordination* is 'the degree ... to which [the agents] ... can avoid "extraneous" activity [such as] ... synchronizing and aligning their activities' [10, p19]; in a perfectly coordinated system, agents will not accidentally clobber each other's sub-goals while attempting to achieve a common goal; they will not need to explicitly communicate, as they will be mutually predictable, perhaps by maintaining good internal models of each other; the presence of conflict between agents (which requires time and effort to resolve) is an indicator of poor coordination.

It is probably true to say that coherence and, in particular, coordination, have been the focus of more attention in multi-agent systems research than any other issues [29, 28, 45, 52, 57, 116].

3.2 Communication

Communication is not universally assumed in multi-agent research: some systems achieve a high-degree of efficiency without explicitly communicating at all [105]. However, communication via some form of message passing is a widely used approach. Messages may be as simple as signals, carrying only synchronization information (playing a role analogous to that of semaphores in traditional concurrent systems) [48]. However, it is more usual to have agents send messages in a richer, high-level communication language.

The inspiration for much multi-agent systems research on message passing comes from research in the philosophy of language on *speech act theory*. Speech act theories are *pragmatic* theories of language, i.e., theories of how language is used by people every day to achieve their goals. The origin of speech act theory is usually attributed to Austin, who, in his 1962 book *How to Do Things With Words* noted that certain sorts of utterances are *actions* that change the state of the world in a way analogous to that in which 'physical' actions (such as lifting a book from a table) do [6]. The paradigm examples of such actions would be statements such as 'I declare war', or 'I now pronounce you man and wife'. Uttered under the appropriate circumstances, these statements change the state of the world. More generally, it can be seen that *every* utterance one makes is an action performed with the objective of satisfying one of our goals or intentions. Thus, speech act theories are theories of communication as action. Austin did not attempt to rigorously analyze the nature of speech acts; the first and best known attempt to do so was by the philosopher John Searle in his 1969 book *Speech Acts* [96]. Searle attempted to formulate the conditions under which a speech act could be said to have occurred. He identified a number of different classes of *performative verbs*, including *representatives* (such as informing), *directives* (attempts to get the hearer to do something), *commisives* (which commit the speaker to doing something), *expressives* (whereby a speaker expresses a mental state), and finally, *declarations* (such as declaring war, or christening). There is some debate about whether this (or any) typology of speech acts is appropriate (see [69, pp226–283]).

In the late 1970s, speech act theory was brought into AI primarily through the work of Cohen and colleagues [23]. In their *plan based theory of speech acts*, Cohen and Perrault formalised a number of speech acts (such as *requesting* and *informing*) using the PDA-list formalism developed by AI planning researchers (see section 2.1, above). This work made it possible to develop systems that plan to perform speech acts [5]. Recently, Cohen and Levesque have presented a more sophisticated theory of speech acts in which requesting, informing, and so on are represented as actions performed by an agent with the intention of bringing about some state of affairs [21].

Following research in speech act theory, communications in multi-agent systems are most usually treated as messages with a performative verb (such as *request* or *inform*) defining the message type, and with the message *content* defined in some other language. KQML (the Knowledge Query and Manipulation Language) is a good example of such a communication scheme [76]. KQML defines a number of performative verbs, and allows message content to be represented in a first-order logic-like language called KIF (Knowledge Interchange Format). Here is a simple KQML/KIF exchange between two agents, A1 and A2:

```
A1 to A2:   (ask-if (> (size chip1) (size chip2)))
A2 to A1:   (reply true)
A2 to A1:   (inform (= (size chip1) 20))
A2 to A1:   (inform (= (size chip2) 18))
```

In the first message, A1 asks A2 whether chip 1 is larger than chip 2, using the performative `ask-if`; in the second message, agent A2 replies to A1 that the answer to the query is true, using the performative `reply`. In the third and fourth messages, agent A2 tells agent A1 the size of the respective chips, using the `inform` performative. A critique of KQML/KIF, focussing particularly on the lack of a formal semantics, is given in [22].

3.3 Cooperation

So far, this article has only briefly mentioned the notion of *cooperation*. This is such a key idea that it deserves further discussion. What does it mean for two agents to *cooperate* in order to solve a problem? Intuitively, cooperation means more than just coordinated effort to achieve the goal. It implies a *shared goal* by the agents, and even more than that, *willingness* to work together. The problem of how cooperation can occur in a world populated entirely by self-interested agents has sparked interest not just in the multi-agent systems field, but also in political science, economics, and game theory [71]. Perhaps the best-known work in this area was done by Robert Axelrod. He carried out a series of experiments into a scenario known as the *prisoner's dilemma* [7]. The prisoner's dilemma is summed up in the following scenario:

> Two men are collectively charged with a crime and held in separate cells. They are told that: (i) if one of them confesses and the other does not, the confessor will be freed, and the other will be jailed for three years; and (ii) if both confess, then each will be jailed for two years. Both prisoners know that if neither confesses, then they will each be jailed for one year.

What should a prisoner do? The problem is that traditional game and decision theory says that the *rational* thing to do is to *confess*. This is because confession guarantees a jail sentence of no worse than two years, whereas not confessing can potentially lead to three years in jail. Since the prisoners cannot communicate, they have no way of reaching any kind of binding mutual agreement. Hence, by this reasoning, the prisoner should confess. If both prisoners follow this reasoning, then both will confess, and each will get two years in jail. Intuitively, however, this makes no sense: surely they should *cooperate* by not confessing, and reduce their respective sentences to just one year? This is the prisoner's dilemma: the strictly rational course of action achieves an outcome that intuition tells us is sub-optimal.

The prisoners dilemma may seem an abstract problem, but it turns out to be ubiquitous. In the real world, the prisoners dilemma appears in situations ranging from nuclear weapons treaty compliance to negotiating with one's children. Many researchers have considered the circumstances under which the cooperative outcome (i.e., neither agent confessing) can arise. One idea is to *play the game more than once*. If you know you will be meeting an opponent in future rounds, the incentive to 'defect' appears to vanish, as your opponent is not likely to cooperate subsequently if you do not cooperate now. So, if you play the prisoner's dilemma game repeatedly, then it appears that cooperation is the rational outcome. However, if you agree with an opponent that you will only play a *fixed* number of times, (say 100), then the dilemma reappears. This is because on the last round — the 100th game — you know that you will not have to play any more. The last round thus becomes a non-iterated prisoner's dilemma game, and as we saw above, in such a game, the strictly rational thing to do is defect. But your opponent will reason in the same way, and so he will also defect on the 100th round. This means that the last real round is number 99; it is not difficult to see that in this situation, we end up where we started from, with no cooperation. Various researchers have considered this 'backwards induction' problem.

Axelrod carried out a series of experiments to determine the best strategy for playing the iterated prisoner's dilemma. That is, what strategy could you use, in order to decide whether to cooperate or defect on any given round, such that this strategy would guarantee you the best possible outcome? Surprisingly, he found that the best strategy is as follows: on round 1, cooperate (i.e., do not confess); on round $t > 1$, do what your opponent did on round $t - 1$. This TIT-FOR-TAT strategy is surprisingly robust, and overall, it performs better than many other much more

complex strategies under most assumptions. Space prevents us from embarking on a more detailed discussion of the prisoner's dilemma and Axelrod's results: [7] is recommended as a point of departure for further reading; [77] provides pointers into recent prisoner's dilemma literature.

Axelrod's work, (and that of others working in related areas), is of great importance in multi-agent systems. It helps us to understand what cooperation is, under what circumstances it can arise, and what the limits to cooperative behaviour are. But for the multi-agent system builder, it probably seems somewhat abstract. In order to actually *build* multi-agent systems, practitioners have tended to make many simplifying assumptions. Very little work has considered systems in which the possibility of cooperation between truly self-interested agents is addressed: most researchers tend to make the 'benevolence' assumption, that agents will help each other wherever possible. A major activity has been the development of *cooperation protocols*, that define how cooperation may begin and proceed. Examples include: Durfee's partial global planning paradigm [28, 25]; the cooperation frameworks developed at Daimler-Benz [18, 53]; and the COOL language, which provides 'structured conversation frameworks' within which agents can cooperate using KQML/KIF messages [8]. Other work on cooperation has considered how the process can be guided by the use of norms or social laws. Shoham and Tennenholtz [99] discuss how such norms can come to exist in a multi-agent society ([111] builds on this work); and Goldman and Rosenschein evaluate the efficacy of 'being nice' in a multi-agent society [52].

An obvious problem, related to the issue of cooperation, is that of finding consensus in a society of self-interested agents. Multi-agent systems researchers have investigated various ways by which such consensus may be reached. Perhaps the best-known work in this area is that of Rosenschein and Zlotkin [89], who, using the terminology and techniques of game theory, defined a range of negotiation techniques for multi-agent domains.

4 Application Areas

Agents are currently the focus of much attention in several application areas. In the subsections that follow, these areas are briefly reviewed (see [61] for further pointers to applications). We begin by considering some of the domain attributes that indicate the appropriateness of an agent-based solution.

4.1 When is an Agent-Based Solution Appropriate?

There are a number of factors which point to the appropriateness of an agent-based approach (cf. [10, 62]):

- *The environment is open, or at least highly dynamic, uncertain, or complex.*

  In such environments, systems capable of flexible autonomous action are often the only solution.

- *Agents are a natural metaphor.*

  Many environments (including most organisations, and any commercial or competitive environment) are naturally modelled as societies of agents, either cooperating with each other to solve complex problems, or else competing with one-another. Sometimes, as in intelligent interfaces, the idea of an agent is seen as a natural metaphor: [75] discusses agents as

'expert assistants', cooperating with the user to work on some problem.

- *Distribution of data, control or expertise.*

  In some environments, the distribution of either data, control, or expertise means that a centralised solution is at best extremely difficult or at worst impossible. For example, distributed database systems in which each database is under separate control do not generally lend themselves to centralised solutions. Such systems may often be conveniently modelled as multi-agent systems, in which each database is a semi-autonomous component.

- *Legacy Systems.*

  A problem increasingly faced by software developers is that of *legacy software*: software that is technologically obsolete but functionally essential to an organisation. Such software cannot generally be discarded, because of the short-term cost of rewriting. And yet it is often required to interact with other software components, which were never imagined by the original designers. One solution to this problem is to *wrap* the legacy components, providing them with an 'agent layer' functionality, enabling them to communicate and cooperate with other software components [47].

In the sub-sections that follow, we will discuss some of the applications to which agent technology has been put.

4.2 Agents and Distributed Systems

In distributed systems, the idea of an agent is often seen as a natural metaphor, and, by some, as a development of the concurrent object programming paradigm [2]. Specifically, multi-agent systems have been applied in the following domains:

- *Air traffic control.*

  Air-traffic control systems are among the oldest application areas in multi-agent systems [104, 39]. A recent example is OASIS (*O*ptimal *A*ircraft *S*equencing using *I*ntelligent *S*cheduling), a system that is currently undergoing field trials at Sydney airport in Australia [70]. The specific aim of OASIS is to assist an air-traffic controller in managing the flow of aircraft at an airport: it offers estimates of aircraft arrival times, monitors aircraft progress against previously derived estimates, informs the air-traffic controller of any errors, and perhaps most importantly, finds the optimal sequence in which to land aircraft. OASIS contains two types of agents: *global* agents, which perform generic domain functions (for example, there is a 'sequencer agent', which is responsible for arranging aircraft into a least-cost sequence), and *aircraft agents*, one for each aircraft in the system airspace. The OASIS system was implemented using the PRS agent architecture — see section 2.3.

- *Business process management.*

  Workflow and business process control systems are an area of increasing importance in computer science. Workflow systems aim to automate the processes of a business, ensuring that different business tasks are expedited by the appropriate people at the right time, typically ensuring that a particular document flow is maintained and managed within an organisation. The ADEPT system is a current example of an agent-based business process management system [59]. In ADEPT, a business organisation is modelled as a society of

negotiating, service providing agents. ADEPT is currently being tested on British Telecom (BT) business process which involves some nine departments and 200 different tasks.

- *Industrial systems management.*

  The largest and probably best-known European multi-agent system development project to date was ARCHON [120, 60, 58]. This ESPRIT-funded project developed and deployed multi-agent technology in several industrial domains. The most significant of these domains was a power distribution system, which was installed and is currently operational in northern Spain. Agents in ARCHON have two main parts: a *domain* component, which realises the domain-specific functionality of the agent, and a *wrapper* component, which provides the agent functionality, enabling the system to plan its actions, and to represent and communicate with other agents. The ARCHON technology has subsequently been deployed in several other domains, including particle accelerator control.

- *Distributed sensing.*

  The classic application of multi-agent technology was in distributed sensing [68, 28]. The broad idea is to have a system constructed as a network of spatially distributed sensors. The sensors may, for example, be acoustic sensors on a battlefield, or radars distributed across some airspace. The global goal of the system is to monitor and track all vehicles that pass within range of the sensors. This task can be made simpler if the sensor nodes in the network *cooperate* with one-another, for example by exchanging predictions about when a vehicle will pass from the region of one sensor to the region of another. This apparently simple domain has yielded surprising richness as an environment for experimentation into multi-agent systems: Lesser's well known *Distributed Vehicle Monitoring Testbed* (DVMT) provided the proving ground for many of today's multi-agent system development techniques [68].

- *Space shuttle fault diagnosis.*

  It is difficult to imagine a domain with harder real-time constraints than that of in-flight diagnosis of faults on a spacecraft. Yet one of the earliest applications of the PRS architecture was precisely this [50]. In brief, the procedures that an astronaut would use to diagnose faults in the space shuttle's reaction control systems were directly coded as PRS plans, and the PRS architecture was used to interpret these plans, and provide real-time advice to astronauts in the event of failure or malfunction in this system.

- *Factory process control.*

  As we observed in section 4.1, organisations can be modelled as societies of interacting agents. Factories are no exception, and an agent-based approach to modelling and managing factories has been taken up by several researchers. This work began largely with Parunak [109], who, in YAMS (*Y*et *A*nother *M*anufacturing *S*ystem) used the Contract Net protocol (see section 3) for manufacturing control. More recently, Mori *et al* have used a multi-agent approach to controlling steel coil processing plant [78], and Wooldridge *et al* have described how the process of determining an optimal production sequence for some factory can naturally be viewed as a problem of negotiation between the various production cells within the factory [124].

4.3 Agents in the INTERNET

Much of the hyperbole that currently surrounds all things agent-like is related to the phenomenal growth of the INTERNET [34, 16]. In particular, there is a lot of interest in *mobile* agents, that can move themselves around the INTERNET operating on a user's behalf. This kind of functionality is achieved in the TELESCRIPT language developed by General Magic, Inc., for *remote programming* [118]; related functionality is provided in languages like JAVA and Safe-TCL. There are a number of rationales for this type of agent:

- *Electronic commerce.*

  Currently, commercial activity is driven primarily by humans making decisions about what goods to buy at what price, and so on. However, it is not difficult to see that certain types of commerce might usefully be automated. A standard motivating example is that of a 'travel agent'. Suppose I want to travel from Manchester to San Francisco. There are many different airlines, price structures and routes that I could choose for such a journey. I may not mind about the route, as long as the aircraft involved is not 'fly-by-wire'; I may insist on a dietary option not available with some airlines; or I may not want to fly with Ruritanian airlines after I had a bad experience once. Trying to find the best flight *manually* given these preferences is a tedious business, but a fairly straightforward one. It seems entirely plausible that this kind of service will in future be provided by agents, who take a specification of your desired flight and preferences, and, after checking through a range of on-line flight information databases, will return with a list of the best options.

- *Hand-held PDAs with limited bandwidth*.

  Hand-held 'personal digital assistants' (in the spirit of Apple's NEWTON) are seen by many as a next step in the laptop computer market. Such PDAs are often provided with limited-bandwidth links to telecommunications networks. If a PDA has a query that needs to be resolved, that will require network information resources, it may be more efficient to send out an agent across the network whose purpose is to resolve this query remotely. The searching process is done by the agent at a remote site, and only the final result of the query need be sent back to the PDA that originated the query.

- *Information gathering*.

  The widespread provision of distributed, semi-structured information resources such as the world-wide web obviously presents enormous potential; but it also presents a number of difficulties, (such as 'information overload'); agents are seen as a natural tool to perform tasks such as searching distributed information resources, and filtering out unwanted news and email [75].

At the time of writing, most interest in mobile agents is centred around the JAVA programming language, which, in the form of applets (portable downloadable programs embedded within WWW pages), already provides a very widely used mobile object framework. Also of relevance is the work of the Object Management Group (OMG), a consortium of computer manufacturers who are developing, amongst other things, a mobile agent framework based on their well known CORBA (*C*ommon *O*bject *R*equest *B*roker *A*rchitecture) distributed object standard [82].

4.4 Agents in Interfaces

Another area of much current interest is the use of agent in *interfaces*. The idea here is that of the agent as an *assistant* to a user in some task. The rationale is that current interfaces are in no sense *pro-active*: things only happen when some user initiates a task. The idea of an agent acting in the way that a good assistant would, by *anticipating* our requirements, seems very attractive. Nicholas Negroponte, director of the MIT Media Lab, sees the ultimate development of such agents as follows [80]:

> 'The "agent" answers the phone, recognizes the callers, disturbs you when appropriate, and may even tell a white lie on your behalf. The same agent is well trained in timing, versed in finding opportune moments, and respectful of idiosyncrasies. ' (p150)
>
> 'If you have somebody who knows you well and shares much of your information, that person can act on your behalf very effectively. If your secretary falls ill, it would make no difference if the temping agency could send you Albert Einstein. This issue is not about IQ. It is shared knowledge and the practice of using it in your best interests.' (p151)
>
> 'Like an army commander sending a scout ahead . . . you will dispatch agents to collect information on your behalf. Agents will dispatch agents. The process multiplies. But [this process] started at the interface where you delegated your desires.' (p158)

Some prototypical interface agents of this type are described in [75].

5 Conclusions

It is probably fair to say that (multi-)agent technology is one of the most hyped software technology of the late 1990s. While the technology has much to offer, it is important not to oversell it. Multi-agent technology is likely to be most useful for a specific class of applications (albeit an important class), which exhibit, to some extent, the kind of properties listed in section 1.1. (Thus agents are *not* likely to replace objects!) One of the most worrying aspects of the current interest in agents is to label *everything* as an agent: one finds academic articles describing theorem provers as 'theorem proving agents', planners as 'plan development agents', and even numerical algorithms as 'numerical algorithm agents'. (In [114], Wayner jokes that sooner or later, some enterprising manufacturer will start calling on-off switches 'empowerment agents'.) Many people strongly believe that the notion of an agent as an independent rational decision maker is of great value not just in AI, but also in mainstream computer science. Similarly, the techniques being developed in multi-agent systems research, to enable agents to cooperate and negotiate, are surely of fundamental importance for the future.

References

[1] M. R. Adler, A. B. Davis, R. Weihmayer, and R. W. Worrest. Conflict resolution strategies for non-hierarchical distributed agents. In L. Gasser and M. Huhns, editors, *Distributed Artificial Intelligence*

Volume II, pages 139–162. Pitman Publishing: London and Morgan Kaufmann: San Mateo, CA, 1989.

[2] G. Agha, P. Wegner, and A. Yonezawa, editors. *Research Directions in Concurrent Object-Oriented Programming*. The MIT Press: Cambridge, MA, 1993.

[3] P. Agre and D. Chapman. PENGI: An implementation of a theory of activity. In *Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-87)*, pages 268–272, Seattle, WA, 1987.

[4] J. Ambros-Ingerson and S. Steel. Integrating planning, execution and monitoring. In *Proceedings of the Seventh National Conference on Artificial Intelligence (AAAI-88)*, pages 83–88, St. Paul, MN, 1988.

[5] D. E. Appelt. *Planning English Sentences*. Cambridge University Press: Cambridge, England, 1985.

[6] J. L. Austin. *How to Do Things With Words*. Oxford University Press: Oxford, England, 1962.

[7] R. Axelrod. *The Evolution of Cooperation*. Basic Books, 1984.

[8] M. Barbuceanu and M. S. Fox. Cool: A language for describing coordination in multiagent systems. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, pages 17–24, San Francisco, CA, June 1995.

[9] R. P. Bonasso, D. Kortenkamp, D. P. Miller, and M. Slack. Experiences with an architecture for intelligent, reactive agents. In M. Wooldridge, J. P. Müller, and M. Tambe, editors, *Intelligent Agents II (LNAI 1037)*, pages 187–202. Springer-Verlag: Heidelberg, Germany, 1996.

[10] A. H. Bond and L. Gasser, editors. *Readings in Distributed Artificial Intelligence*. Morgan Kaufmann Publishers: San Mateo, CA, 1988.

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

[12] R. A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–23, 1986.

[13] R. A. Brooks. Elephants don't play chess. In P. Maes, editor, *Designing Autonomous Agents*, pages 3–15. The MIT Press: Cambridge, MA, 1990.

[14] R. A. Brooks. Intelligence without reason. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 569–595, Sydney, Australia, 1991.

[15] R. A. Brooks. Intelligence without representation. *Artificial Intelligence*, 47:139–159, 1991.

[16] C. Brown, L. Gasser, D. E. O'Leary, and Alan Sangster. AI on the WWW: Supply and demand agents. *IEEE Expert*, 10(4):44–49, August 1995.

[17] H.-D. Burkhard. Agent-oriented programming for open systems. In M. Wooldridge and N. R. Jennings, editors, *Intelligent Agents: Theories, Architectures, and Languages (LNAI Volume 890)*, pages 291–306. Springer-Verlag: Heidelberg, Germany, January 1995.

[18] B. Burmeister, A. Haddadi, and K. Sundermeyer. Generic, configurable cooperation protocols for multi-agent systems. In C. Castelfranchi and J.-P. Müller, editors, *From Reaction to Cognition — Fifth European Workshop on Modelling Autonomous Agents in a Multi-Agent World, MAAMAW-93 (LNAI Volume 957)*, pages 157–171. Springer-Verlag: Heidelberg, Germany, 1995.

[19] D. Chapman. Planning for conjunctive goals. *Artificial Intelligence*, 32:333–378, 1987.

[20] P. R. Cohen and H. J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42:213–261, 1990.

[21] P. R. Cohen and H. J. Levesque. Rational interaction as the basis for communication. In P. R. Cohen, J. Morgan, and M. E. Pollack, editors, *Intentions in Communication*, pages 221–256. The MIT Press: Cambridge, MA, 1990.

[22] P. R. Cohen and H. J. Levesque. Communicative actions for artificial agents. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, pages 65–72, San Francisco, CA, June 1995.

[23] P. R. Cohen and C. R. Perrault. Elements of a plan based theory of speech acts. *Cognitive Science*, 3:177–212, 1979.

[24] D. Connah and P. Wavish. An experiment in cooperation. In Y. Demazeau and J.-P. Müller, editors, *Decentralized AI — Proceedings of the First European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW-89)*, pages 197–214. Elsevier Science Publishers B.V.: Amsterdam,

The Netherlands, 1990.

[25] K. Decker and V. Lesser. Designing a family of coordination algorithms. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, pages 73–80, San Francisco, CA, June 1995.

[26] K. S. Decker, E. H. Durfee, and V. R. Lesser. Evaluating research in cooperative distributed problem solving. In L. Gasser and M. Huhns, editors, *Distributed Artificial Intelligence Volume II*, pages 487–519. Pitman Publishing: London and Morgan Kaufmann: San Mateo, CA, 1989.

[27] M. d'Inverno and M. Luck. Formalising the contract net as a goal-directed system. In W. Van de Velde and J. Perram, editors, *Agents Breaking Away — Proceedings of the Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World, MAAMAW-96 (LNAI 1038)*, pages 72–85. Springer-Verlag: Heidelberg, Germany, 1996.

[28] E. H. Durfee. *Coordination of Distributed Problem Solvers*. Kluwer Academic Publishers: Boston, MA, 1988.

[29] E. H. Durfee and V. R. Lesser. Using partial global plans to coordinate distributed problem solvers. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence (IJCAI-87)*, Milan, Italy, 1987.

[30] E. H. Durfee, V. R. Lesser, and D. D. Corkill. Cooperative distributed problem solving. In *Handbook of Artificial Intelligence Volume IV*. Addison-Wesley: Reading, MA, 1989.

[31] E. A. Emerson and J. Y. Halpern. 'Sometimes' and 'not never' revisited: on branching time versus linear time temporal logic. *Journal of the ACM*, 33(1):151–178, 1986.

[32] E. Ephrati and J. S. Rosenschein. Multi-agent planning as a dynamic search for social consensus. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI-93)*, pages 423–429, Chambéry, France, 1993.

[33] O. Etzioni, N. Lesh, and R. Segal. Building softbots for UNIX. In O. Etzioni, editor, *Software Agents — Papers from the 1994 Spring Symposium (Technical Report SS–94–03)*, pages 9–16. AAAI Press, March 1994.

[34] O. Etzioni and D. S. Weld. Intelligent agents on the internet: Fact, fiction, and forecast. *IEEE Expert*, 10(4):44–49, August 1995.

[35] I. A. Ferguson. *TouringMachines: An Architecture for Dynamic, Rational, Mobile Agents*. PhD thesis, Clare Hall, University of Cambridge, UK, November 1992. (Also available as Technical Report No. 273, University of Cambridge Computer Laboratory).

[36] I. A. Ferguson. Towards an architecture for adaptive, rational, mobile agents. In E. Werner and Y. Demazeau, editors, *Decentralized AI 3 — Proceedings of the Third European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW-91)*, pages 249–262. Elsevier Science Publishers B.V.: Amsterdam, The Netherlands, 1992.

[37] I. A. Ferguson. Integrated control and coordinated behaviour: A case for agent models. In M. Wooldridge and N. R. Jennings, editors, *Intelligent Agents: Theories, Architectures, and Languages (LNAI Volume 890)*, pages 203–218. Springer-Verlag: Heidelberg, Germany, January 1995.

[38] R. E. Fikes and N. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 5(2):189–208, 1971.

[39] N. V. Findler and R. Lo. An examination of Distributed Planning in the world of air traffic control. *Journal of Parallel and Distributed Computing*, 3, 1986.

[40] J. A. Firby. An investigation into reactive planning in complex domains. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence (IJCAI-87)*, pages 202–206, Milan, Italy, 1987.

[41] M. Fisher. A survey of Concurrent METATEM — the language and its applications. In D. M. Gabbay and H. J. Ohlbach, editors, *Temporal Logic — Proceedings of the First International Conference (LNAI Volume 827)*, pages 480–505. Springer-Verlag: Heidelberg, Germany, July 1994.

[42] M. Fisher. Representing and executing agent-based systems. In M. Wooldridge and N. R. Jennings, editors, *Intelligent Agents: Theories, Architectures, and Languages (LNAI Volume 890)*, pages 307–323. Springer-Verlag: Heidelberg, Germany, January 1995.

[43] J. R. Galliers. *A Theoretical Framework for Computer Models of Cooperative Dialogue, Acknowledg-*

ing Multi-Agent Conflict. PhD thesis, Open University, UK, 1988.

[44] J. R. Galliers. The positive role of conflict in cooperative multi-agent systems. In Y. Demazeau and J.-P. Müller, editors, *Decentralized AI — Proceedings of the First European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW-89)*, pages 33–48. Elsevier Science Publishers B.V.: Amsterdam, The Netherlands, 1990.

[45] L. Gasser and R. W. Hill Jr. Coordinated problem solvers. *Annual Review of Computer Science*, 4:203–253, 1990.

[46] L. Gasser and M. Huhns, editors. *Distributed Artificial Intelligence Volume II*. Pitman/Morgan Kaufman, 1989.

[47] M. R. Genesereth and S. P. Ketchpel. Software agents. *Communications of the ACM*, 37(7):48–53, July 1994.

[48] M. P. Georgeff. Communication and interaction in multi-agent planning. In *Proceedings of the Third National Conference on Artificial Intelligence (AAAI-83)*, Washington, D.C., 1983.

[49] M. P. Georgeff. Planning. *Annual Review of Computer Science*, 2:359–400, 1987.

[50] M. P. Georgeff and A. L. Lansky. Reactive reasoning and planning. In *Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-87)*, pages 677–682, Seattle, WA, 1987.

[51] N. Gilbert. Emergence in social simulation. In N. Gilbert and R. Conte, editors, *Artificial Societies: The Computer Simulation of Social Life*, pages 144–156. UCL Press: London, 1995.

[52] C. V. Goldman and J. S. Rosenschein. Emergent coordination through the use of cooperative state-changing rules. In *Proceedings of the Twelfth International Workshop on Distributed Artificial Intelligence (IWDAI-93)*, pages 171–186, Hidden Valley, PA, May 1993.

[53] A. Haddadi. *Communication and Cooperation in Agent Systems (LNAI Volume 1056)*. Springer-Verlag: Heidelberg, Germany, 1996.

[54] B. Hayes-Roth, M. Hewett, R. Washington, R. Hewett, and A. Seiver. Distributing intelligence within an individual. In L. Gasser and M. Huhns, editors, *Distributed Artificial Intelligence Volume II*, pages 385–412. Pitman Publishing: London and Morgan Kaufmann: San Mateo, CA, 1989.

[55] M. Huhns, editor. *Distributed Artificial Intelligence*. Pitman Publishing: London and Morgan Kaufmann: San Mateo, CA, 1987.

[56] N. R. Jennings. Towards a cooperation knowledge level for collaborative problem solving. In *Proceedings of the Tenth European Conference on Artificial Intelligence (ECAI-92)*, pages 224–228, Vienna, Austria, 1992.

[57] N. R. Jennings. Commitments and conventions: The foundation of coordination in multi-agent systems. *The Knowledge Engineering Review*, 8(3):223–250, 1993.

[58] N. R. Jennings, J. M. Corera, and I. Laresgoiti. Developing industrial multi-agent systems. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, pages 423–430, San Francisco, CA, June 1995.

[59] N. R. Jennings, P. Faratin, M. J. Johnson, T. J. Norman, P. O'Brien, and M. e. Wiegand. Agent-based business process management. *International Journal of Cooperative Information Systems*, 5(2&3):105–130, 1996.

[60] N. R. Jennings and T. Wittig. ARCHON: Theory and practice. In *Distributed Artificial Intelligence: Theory and Praxis*, pages 179–195. ECSC, EEC, EAEC, 1992.

[61] N. R. Jennings and M. Wooldridge. Applying agent technology. *Applied Artificial Intelligence*, 9(6):357–370, 1995.

[62] N. R. Jennings and M. Wooldridge. Applications of intelligent agent agents. In N. R. Jennings and M. Wooldridge, editors, *Agent-Based Computing: Applications and Markets*. 1997. To appear.

[63] L. P. Kaelbling. An architecture for intelligent reactive systems. In M. P. Georgeff and A. L. Lansky, editors, *Reasoning About Actions & Plans — Proceedings of the 1986 Workshop*, pages 395–410. Morgan Kaufmann Publishers: San Mateo, CA, 1986.

[64] D. Kinny, M. Ljungberg, A. S. Rao, E. Sonenberg, G. Tidhar, and E. Werner. Planned team activity. In C. Castelfranchi and E. Werner, editors, *Artificial Social Systems — Selected Papers from the Fourth European Workshop on Modelling Autonomous Agents in a Multi-Agent World, MAAMAW-92 (LNAI*

Volume 830), pages 226–256. Springer-Verlag: Heidelberg, Germany, 1992.

[65] M. Klein and A. B. Baskin. A computational model for conflict resolution in cooperative design systems. In S. M. Deen, editor, *CKBS-90 — Proceedings of the International Working Conference on Cooperating Knowledge Based Systems*, pages 201–222. Springer-Verlag: Heidelberg, Germany, 1991.

[66] S. Lander, V. R. Lesser, and M. E. Connell. Conflict resolution strategies for cooperating expert agents. In S. M. Deen, editor, *CKBS-90 — Proceedings of the International Working Conference on Cooperating Knowledge Based Systems*, pages 183–200. Springer-Verlag: Heidelberg, Germany, 1991.

[67] Y. Lésperance, H. J. Levesque, F. Lin, D. Marcu, R. Reiter, and R. B. Scherl. Foundations of a logical approach to agent programming. In M. Wooldridge, J. P. Müller, and M. Tambe, editors, *Intelligent Agents II (LNAI 1037)*, pages 331–346. Springer-Verlag: Heidelberg, Germany, 1996.

[68] V. R. Lesser and L. D. Erman. Distributed interpretation: A model and experiment. *IEEE Transactions on Computers*, C-29(12), 1980.

[69] S. C. Levinson. *Pragmatics*. Cambridge University Press: Cambridge, England, 1983.

[70] M. Ljunberg and A. Lucas. The OASIS air traffic management system. In *Proceedings of the Second Pacific Rim International Conference on AI (PRICAI-92)*, Seoul, Korea, 1992.

[71] R. D. Luce and H. Raiffa. *Games and Decisions*. John Wiley & Sons, 1957.

[72] P. Maes, editor. *Designing Autonomous Agents*. The MIT Press: Cambridge, MA, 1990.

[73] P. Maes. Situated agents can have goals. In P. Maes, editor, *Designing Autonomous Agents*, pages 49–70. The MIT Press: Cambridge, MA, 1990.

[74] P. Maes. The agent network architecture (ANA). *SIGART Bulletin*, 2(4):115–120, 1991.

[75] P. Maes. Agents that reduce work and information overload. *Communications of the ACM*, 37(7):31–40, July 1994.

[76] J. Mayfield, Y. Labrou, and T. Finin. Evaluating KQML as an agent communication language. In M. Wooldridge, J. P. Müller, and M. Tambe, editors, *Intelligent Agents II (LNAI 1037)*, pages 347–360. Springer-Verlag: Heidelberg, Germany, 1996.

[77] Y. Mor and J. S. Rosenschein. Time and the prisoner's dilemma. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, pages 276–282, San Francisco, CA, June 1995.

[78] K. Mori, H. Torikoshi, K. Nakai, and T. Masuda. Computer control system for iron and steel plants. *Hitachi Review*, 37(4):251–258, 1988.

[79] J. P. Müller. *The Design of Intelligent Agents (LNAI Volume 1177)*. Springer-Verlag: Heidelberg, Germany, 1996.

[80] N. Negroponte. *Being Digital*. Hodder and Stoughton, 1995.

[81] A. Newell and H. A. Simon. GPS: A program that simulates human thought. In *Lernende Automaten*. R. Oldenbourg, KG, 1961.

[82] The Object Management Group (OMG). See `http://www.omg.org/`.

[83] A. Poggi. DAISY: An object-oriented system for distributed artificial intelligence. In M. Wooldridge and N. R. Jennings, editors, *Intelligent Agents: Theories, Architectures, and Languages (LNAI Volume 890)*, pages 341–354. Springer-Verlag: Heidelberg, Germany, January 1995.

[84] A. S. Rao and M. Georgeff. BDI Agents: from theory to practice. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, pages 312–319, San Francisco, CA, June 1995.

[85] A. S. Rao and M. P. Georgeff. Modeling rational agents within a BDI-architecture. In R. Fikes and E. Sandewall, editors, *Proceedings of Knowledge Representation and Reasoning (KR&R-91)*, pages 473–484. Morgan Kaufmann Publishers: San Mateo, CA, April 1991.

[86] A. S. Rao and M. P. Georgeff. Formal models and decision procedures for multi-agent systems. Technical Note 61, Australian AI Institute, Level 6, 171 La Trobe Street, Melbourne, Australia, June 1995.

[87] A. S. Rao, M. P. Georgeff, and E. A. Sonenberg. Social plans: A preliminary report. In E. Werner and Y. Demazeau, editors, *Decentralized AI 3 — Proceedings of the Third European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW-91)*, pages 57–76. Elsevier Science

Publishers B.V.: Amsterdam, The Netherlands, 1992.

[88] J. S. Rosenschein and M. R. Genesereth. Deals among rational agents. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence (IJCAI-85)*, pages 91–99, Los Angeles, CA, 1985.

[89] J. S. Rosenschein and G. Zlotkin. *Rules of Encounter: Designing Conventions for Automated Negotiation among Computers*. The MIT Press: Cambridge, MA, 1994.

[90] S. Rosenschein and L. P. Kaelbling. The synthesis of digital machines with provable epistemic properties. In J. Y. Halpern, editor, *Proceedings of the 1986 Conference on Theoretical Aspects of Reasoning About Knowledge*, pages 83–98. Morgan Kaufmann Publishers: San Mateo, CA, 1986.

[91] S. J. Russell, D. Subramanian, and R. Parr. Provably bounded optimal agents. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI-93)*, pages 338–344, Chambéry, France, 1993.

[92] S. J. Russell and E. Wefald. *Do the Right Thing — Studies in Limited Rationality*. The MIT Press: Cambridge, MA, 1991.

[93] E. Sacerdoti. Planning in a hierarchy of abstraction spaces. *Artificial Intelligence*, 5:115–135, 1974.

[94] E. Sacerdoti. The non-linear nature of plans. In *Proceedings of the Fourth International Joint Conference on Artificial Intelligence (IJCAI-75)*, pages 206–214, Stanford, CA, 1975.

[95] T. Sandholm and V. Lesser. Issues in automated negotiation and electronic commerce: Extending the contract net framework. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, pages 328–335, San Francisco, CA, June 1995.

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

[97] Y. Shoham. Agent-oriented programming. Technical Report STAN–CS–1335–90, Computer Science Department, Stanford University, Stanford, CA 94305, 1990.

[98] Y. Shoham. Agent-oriented programming. *Artificial Intelligence*, 60(1):51–92, 1993.

[99] Y. Shoham and M. Tennenholtz. On the synthesis of useful social laws for artificial agent societies. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)*, San Diego, CA, 1992.

[100] R. G. Smith. The CONTRACT NET: A formalism for the control of distributed problem solving. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence (IJCAI-77)*, Cambridge, MA, 1977.

[101] R. G. Smith. The contract net protocol. *IEEE Transactions on Computers*, C-29(12), 1980.

[102] R. G. Smith. *A Framework for Distributed Problem Solving*. UMI Research Press, 1980.

[103] R. G. Smith and R. Davis. Frameworks for cooperation in distributed problem solving. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(1), 1980.

[104] R. Steeb, S. Cammarata, F. A. Hayes-Roth, P. W. Thorndyke, and R. B. Wesson. Distributed intelligence for air fleet control. In A. H. Bond and L. Gasser, editors, *Readings in Distributed Artificial Intelligence*, pages 90–101. Morgan Kaufmann Publishers: San Mateo, CA, 1988.

[105] L. Steels. Cooperation between distributed agents through self organization. In Y. Demazeau and J.-P. Müller, editors, *Decentralized AI — Proceedings of the First European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW-89)*, pages 175–196. Elsevier Science Publishers B.V.: Amsterdam, The Netherlands, 1990.

[106] S. R. Thomas. *PLACA, an Agent Oriented Programming Language*. PhD thesis, Computer Science Department, Stanford University, Stanford, CA 94305, August 1993. (Available as technical report STAN–CS–93–1487).

[107] S. R. Thomas. The PLACA agent programming language. In M. Wooldridge and N. R. Jennings, editors, *Intelligent Agents: Theories, Architectures, and Languages (LNAI Volume 890)*, pages 355–369. Springer-Verlag: Heidelberg, Germany, January 1995.

[108] G. Tidhar and J. Rosenschein. A contract net with consultants. In *Proceedings of the Tenth European Conference on Artificial Intelligence (ECAI-92)*, pages 219–223, Vienna, Austria, 1992.

[109] H. Van Dyke Parunak. Manufacturing experience with the contract net. In M. Huhns, editor, *Dis-*

tributed Artificial Intelligence, pages 285–310. Pitman Publishing: London and Morgan Kaufmann: San Mateo, CA, 1987.

[110] S. Vere and T. Bickmore. A basic agent. *Computational Intelligence*, 6:41–60, 1990.

[111] A. Walker and M. Wooldridge. Understanding the emergence of conventions in multi-agent systems. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, pages 384–390, San Francisco, CA, June 1995.

[112] P. Wavish. Exploiting emergent behaviour in multi-agent systems. In E. Werner and Y. Demazeau, editors, *Decentralized AI 3 — Proceedings of the Third European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW-91)*, pages 297–310. Elsevier Science Publishers B.V.: Amsterdam, The Netherlands, 1992.

[113] P. Wavish and M. Graham. Roles, skills, and behaviour: a situated action approach to organising systems of interacting agents. In M. Wooldridge and N. R. Jennings, editors, *Intelligent Agents: Theories, Architectures, and Languages (LNAI Volume 890)*, pages 371–385. Springer-Verlag: Heidelberg, Germany, January 1995.

[114] P. Wayner. *Agents Unleashed: A Public Domain Look at Agent Technology*. Academic Press: London, 1995.

[115] D. Weerasooriya, A. Rao, and K. Ramamohanarao. Design of a concurrent agent-oriented language. In M. Wooldridge and N. R. Jennings, editors, *Intelligent Agents: Theories, Architectures, and Languages (LNAI Volume 890)*, pages 386–402. Springer-Verlag: Heidelberg, Germany, January 1995.

[116] G. Weiß. Learning to coordinate actions in multi-agent systems. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI-93)*, pages 311–316, Chambéry, France, 1993.

[117] E. Werner. Cooperating agents: A unified theory of communication and social structure. In L. Gasser and M. Huhns, editors, *Distributed Artificial Intelligence Volume II*, pages 3–36. Pitman Publishing: London and Morgan Kaufmann: San Mateo, CA, 1989.

[118] J. E. White. Telescript technology: The foundation for the electronic marketplace. White paper, General Magic, Inc., 2465 Latham Street, Mountain View, CA 94040, 1994.

[119] D. Wilkins. *Practical Planning: Extending the Classical AI Planning Paradigm*. Morgan Kaufmann Publishers: San Mateo, CA, 1988.

[120] T. Wittig, editor. *ARCHON: An Architecture for Multi-Agent Systems*. Ellis Horwood: Chichester, England, 1992.

[121] M. Wooldridge. *The Logical Modelling of Computational Multi-Agent Systems*. PhD thesis, Department of Computation, UMIST, Manchester, UK, October 1992. (Also available as Technical Report MMU–DOC–94–01, Department of Computing, Manchester Metropolitan University, Chester St., Manchester, UK).

[122] M. Wooldridge. Coherent social action. In *Proceedings of the Eleventh European Conference on Artificial Intelligence (ECAI-94)*, pages 279–283, Amsterdam, The Netherlands, 1994.

[123] M. Wooldridge. This is MYWORLD: The logic of an agent-oriented testbed for DAI. In M. Wooldridge and N. R. Jennings, editors, *Intelligent Agents: Theories, Architectures, and Languages (LNAI Volume 890)*, pages 160–178. Springer-Verlag: Heidelberg, Germany, January 1995.

[124] M. Wooldridge, S. Bussmann, and M. Klosterberg. Production sequencing as negotiation. In *Proceedings of the First International Conference on the Practical Application of Agents and Multi-Agent Technology (PAAM–97)*, pages 709–726, 1996.

[125] M. Wooldridge and N. R. Jennings. Formalizing the cooperative problem solving process. In *Proceedings of the Thirteenth International Workshop on Distributed Artificial Intelligence (IWDAI-94)*, pages 403–417, Lake Quinalt, WA, July 1994.

[126] M. Wooldridge and N. R. Jennings. Intelligent agents: Theory and practice. *The Knowledge Engineering Review*, 10(2):115–152, 1995.