Chapter 1

Introduction

The history of computing to date has been marked by five important, and continuing, trends:

- *ubiquity*
- *interconnection*
- *intelligence*
- *delegation*
- *human-orientation*.

By ubiquity, I simply mean that the continual reduction in cost of computing capability has made it possible to introduce processing power into places and devices where it would hitherto have been uneconomic, and perhaps even unimaginable. This trend will inevitably continue, making processing capability, and hence intelligence of a sort, ubiquitous. UBIQUITY

While the earliest computer systems were isolated entities, communicating only with their human operators, computer systems today are usually *interconnected*. They are *networked* into large *distributed* systems. The Internet is the obvious example; it is becoming increasingly rare to find computers in use in commercial or academic settings that do not have the capability to access the Internet. Until a comparatively short time ago, distributed and concurrent systems were seen by many as strange and difficult beasts, best avoided. The very visible and very rapid growth of the Internet has (I hope) dispelled this perception forever. Today, and for the future, distributed and concurrent systems are essentially the norm in commercial and industrial computing, leading some researchers and practitioners to revisit the very foundations of computer science, seeking theoretical models that reflect the reality of computing as primarily a process of interaction. INTERCONNECTION

The third trend is towards ever more *intelligent* systems. By this, I mean that the *complexity* of tasks that we are capable of automating and delegating to computers has also grown steadily. We are gaining a progressively better understanding of how to engineer computer systems to deal with tasks that would have been unthinkable only a short time ago. INTELLIGENCE

DELEGATION

The next trend is towards ever-increasing delegation. For example, we routinely delegate to computer systems such safety-critical tasks as piloting aircraft. Indeed, in fly-by-wire aircraft, the judgement of a computer program is trusted over that of experienced pilots. Delegation implies that we *give control* to computer systems.

HUMAN-ORIENTATION

The fifth and final trend is the steady move away from machine-oriented views of human–computer interaction towards concepts and metaphors that more closely reflect the way in which we ourselves understand the world. This trend is evident in every way that we interact with computers. For example, in the earliest days of computers, a user interacted with a computer by setting switches on the machine. The internal operation of the device was in no way hidden from the user – in order to use it successfully, one had to fully understand the internal structure and operation of the device. Such primitive – and unproductive – interfaces gave way to command-line interfaces, where one could interact with the device in terms of an ongoing dialogue, in which the user issued instructions that were then executed. Such interfaces dominated until the 1980s, when they gave way to graphical user interfaces, and the direct manipulation paradigm in which a user controls the device by directly manipulating graphical icons corresponding to objects such as files and programs (the 'desktop' metaphor). Similarly, in the earliest days of computing, programmers had no choice but to program their computers in terms of raw machine code, which implied a detailed understanding of the internal structure and operation of their machines. Subsequent programming paradigms have progressed away from such low-level views: witness the development of assembler languages, through procedural abstraction, to abstract data types, and most recently, objects. Each of these developments has allowed programmers to conceptualize and implement software in terms of higher-level – more human-oriented – abstractions.

These trends present major challenges for software developers. With respect to ubiquity and interconnection, we do not yet know what techniques might be used to develop systems to exploit ubiquitous processor power. Current software development models have proved woefully inadequate even when dealing with relatively small numbers of processors. What techniques might be needed to deal with systems composed of 10^{10} processors? The term *global computing* has been coined to describe such unimaginably large systems.

GLOBAL COMPUTING

The trends to increasing delegation and intelligence imply the need to build computer systems that can act effectively on our behalf. This in turn implies two capabilities. The first is the ability of systems to operate *independently*, without our direct intervention. The second is the need for computer systems to be able to act in such a way as to *represent our best interests* while interacting with other humans or systems.

The trend towards interconnection and distribution has, in mainstream computer science, long been recognized as a key challenge, and much of the intellectual energy of the field throughout the past three decades has been directed towards developing software tools and mechanisms that allow us to build distributed systems with greater ease and reliability. However, when coupled with the need for systems that can represent our best interests, distribution poses other fundamental problems. When a computer system acting on our behalf must interact with another computer system that represents the interests of another, it may well be (indeed, it is likely) that these interests are not the same. It becomes necessary to endow such systems with the ability to *cooperate* and *reach agreements* with

other systems, in much the same way that we cooperate and reach agreements with others in everyday life. This type of capability was not studied in computer science until very recently.

Together, these trends have led to the emergence of a new field in computer science: *multiagent systems*. The idea of a multiagent system is very simple. An agent is a computer system that is capable of *independent* action on behalf of its user or owner. In other words, an agent can figure out for itself what it needs to do in order to satisfy its design objectives, rather than having to be told explicitly what to do at any given moment. A multiagent system is one that consists of a number of agents, which *interact* with one another, typically by exchanging messages through some computer network infrastructure. In the most general case, the agents in a multiagent system will be representing or acting on behalf of users or owners with very different goals and motivations. In order to successfully interact, these agents will thus require the ability to *cooperate*, *coordinate*, and *negotiate* with each other, in much the same way that we cooperate, coordinate, and negotiate with other people in our everyday lives.

MULTIAGENT SYSTEMS

This book is about multiagent systems. It addresses itself to the two key problems hinted at above.

- How do we build agents that are capable of independent, autonomous action in order to successfully carry out the tasks that we delegate to them?
- How do we build agents that are capable of interacting (cooperating, coordinating, negotiating) with other agents in order to successfully carry out the tasks that we delegate to them, particularly when the other agents cannot be assumed to share the same interests/goals?

The first problem is that of *agent design*, and the second problem is that of *society design*. The two problems are not orthogonal – for example, in order to build a society of agents that work together effectively, it may help if we give members of the society models of the other agents in it. The distinction between the two issues is often referred to as the *micro/macro* distinction.

AGENT DESIGN
SOCIETY DESIGN
MICRO/MACRO LEVELS

Researchers in multiagent systems may be predominantly concerned with engineering systems, but this is by no means their only concern. As with its stablemate, artificial intelligence (AI), the issues addressed by the multiagent systems field have profound implications for our understanding of ourselves. AI has been largely focused on the issues of intelligence in individuals. But surely a large part of what makes us unique as a species is our *social ability*. Not only can we communicate with one another in high-level languages, we can cooperate, coordinate, and negotiate with one another. While many other species have social ability of a kind – ants and other social insects being perhaps the best-known examples – no other species begins to approach us in the sophistication of our social ability. In multiagent systems, we address ourselves to such questions as:

SOCIAL ABILITY

- How can cooperation emerge in societies of self-interested agents?
- How can self-interested agents recognize when their beliefs, goals, or actions conflict, and how can they reach agreements with one another on matters of self-interest, without resorting to conflict?

- How can autonomous agents coordinate their activities so as to cooperatively achieve goals?
- What sorts of common languages can agents use to communicate their beliefs and aspirations, both to people and to other agents?
- How can agents built by different individuals or organizations using different hardware or software platforms be sure that, when they communicate with respect to some concept, they share the same interpretation of this concept?

While these questions are all addressed in part by other disciplines (notably economics and the social sciences), what makes the multiagent systems field unique and distinct is that it emphasizes that the agents in question are *artificial computational* entities.

The remainder of this chapter

The purpose of this first chapter is to orient you for the remainder of the book. The chapter is structured as follows.

- I begin, in the following section, with some scenarios. The aim of these scenarios is to give you some feel for the long-term visions that are driving activity in the agents area.
- As with multiagent systems themselves, not everyone involved in the agent community shares a common purpose. I therefore summarize the different ways that people think about the 'multiagent systems project'.
- I then present and discuss some common objections to the multiagent systems field.

1.1 The Vision Thing

It is very often hard to understand what people are doing until you understand what their motivation is. The aim of this section is therefore to provide some motivation for what the agents community does. This motivation comes in the style of long-term future visions – ideas about how things might be. A word of caution: these visions are exactly that – visions. None is likely to be realized in the immediate future. But for each of the visions, work *is* underway in developing the kinds of technologies that might be required to realize them.

Due to an unexpected system failure, a space probe approaching Saturn loses contact with its Earth-based ground crew and becomes disoriented. Rather than simply disappearing into the void, the probe recognizes that there has been a key system failure, diagnoses and isolates the fault, and correctly reorients itself in order to make contact with its ground crew.

The key issue here is the ability of the space probe to act autonomously (see text box on p. 8). First, the probe needs to recognize that a fault has occurred, and it must then figure

out what needs to be done and how to do it. Finally, the probe must actually do the actions it has chosen, and must presumably monitor what happens in order to ensure that all goes well. If more things go wrong, the probe will be required to recognize this and respond appropriately. Notice that this is the kind of behaviour that we (humans) find easy: we do it every day – when we miss a flight or have a flat tyre while driving to work. But, as we shall see, it is *very* hard to design computer programs that exhibit this kind of behaviour.

A key air-traffic control system at the main airport of Ruritania suddenly fails, leaving flights in the vicinity of the airport with no air-traffic control support. Fortunately, autonomous air-traffic control systems in nearby airports recognize the failure of their peer, and cooperate to track and deal with all affected flights. The potentially disastrous situation passes without incident.

There are several important issues in this scenario. The first is the ability of systems to take the initiative when circumstances dictate. The second is the ability of agents to cooperate to solve problems that are beyond the capabilities of any individual agent. The kind of cooperation required by this scenario was studied extensively in the Distributed Vehicle Monitoring Testbed (DVMT) project undertaken between 1981 and 1991 (see, for example, [Durfee, 1988]). The DVMT simulates a network of vehicle monitoring agents, where each agent is a problem solver that analyses sensed data in order to identify, locate, and track vehicles moving through space. Each agent is typically associated with a sensor, which has only a partial view of the entire space. The agents must therefore cooperate in order to track the progress of vehicles through the entire sensed space. Air-traffic control systems have been a standard application of agent research since the work of Cammarata and colleagues in the early 1980s [Cammarata et al., 1983]; an example of a multiagent air-traffic control application is the OASIS system implemented for use at Sydney airport in Australia [Ljungberg and Lucas, 1992].

Well, most of us are not involved in either designing control systems for space probes or the design of safety-critical systems such as air-traffic controllers. So let us now consider a vision that is closer to most of our everyday lives.

After the wettest and coldest UK winter on record, you are in desperate need of a last-minute holiday somewhere warm and dry. After specifying your requirements to your personal digital assistant (PDA), it converses with a number of different websites, which sell services such as flights, hotel rooms, and hire cars. After hard negotiation on your behalf with a range of sites, your PDA presents you with a package holiday.

This example is perhaps the closest of the three scenarios to actually being realized. There are many websites that will allow you to search for last-minute holidays, but at the time of writing, to the best of my knowledge, none of them engages in active real-time negotiation in order to assemble a package specifically for you from a range of service providers. There are many basic research problems that need to be solved in order to make such a scenario work, such as the examples that follow.

Autonomous Systems in Space

Space exploration is proving to be an important application area for autonomous systems. Current unmanned space missions typically require a ground crew of up to 300 staff to continuously monitor flight progress. This ground crew usually makes all the necessary control decisions on behalf of the space probe, and painstakingly transmits these decisions to the probe, where they are then blindly executed. Given the length of typical planetary exploration missions, this procedure is expensive and, if decisions are ever required *quickly*, it is simply not practical. Moreover, in some circumstances, it isn't possible at all. For example, in the first serious feasibility study on interstellar travel, [Bond, 1978] notes that an extremely high degree of autonomy would be required on the proposed mission to Barnard's star, lasting more than 50 years:

> [The control system] must be capable of reacting autonomously in the best way possible to a set of circumstances which is indeterminate at launch. . . . Goals may be implanted in the [system] prior to flight, but rigid seeking of those goals may result in total mission failure; those implanted goals may have to be expanded, contracted, or superseded in the light of unanticipated circumstances.
>
> [Bond, 1978, p. S131]

An extremely clear description of the type of system that this book is all about, written in 1978! Sadly, interstellar travel of the type discussed in [Bond, 1978] is a long way off, if it is ever possible at all. But the idea of autonomy in space flight remains very relevant for real space missions today. Launched from Cape Canaveral on 24 October 1998, NASA's DS1 was the first space probe to have an autonomous, agent-based control system [Muscettola et al., 1998]. The autonomous control system in DS1 was capable of making many important decisions itself. This made the mission more robust, particularly against sudden unexpected problems, and also had the very desirable side effect of reducing overall mission costs. NASA (and other space agencies) are currently looking beyond the autonomy of individual space probes, to having teams of probes cooperate in space exploration missions [Jonsson et al., 2007].

- How do you state your preferences to your agent?
- How can your agent compare different deals from different vendors?
- What algorithms can your agent use to negotiate with other agents (so as to ensure that you are not 'ripped off')?

The ability to negotiate in the style implied by this scenario is potentially very valuable indeed. Every year, for example, the European Commission puts out thousands of contracts to public tender. The bureaucracy associated with managing this process has an enormous cost. The ability to automate the tendering and negotiation process would save enormous sums of money (*taxpayers'* money!). Similar situations arise in government organizations everywhere – a good example is the US military. So the ability to automate the process of

software agents reaching mutually acceptable agreements on matters of common interest is not just an abstract concern – it may affect our lives (the amount of tax we pay) in a significant way.

1.2 Some Views of the Field

The multiagent systems field is highly interdisciplinary: it takes inspiration from such diverse areas as economics, philosophy, logic, ecology, and the social sciences. It should come as no surprise that there are therefore many different views of what the 'multiagent systems project' is all about.

1.2.1 Agents as a paradigm for software engineering

Software engineers have derived a progressively better understanding of the characteristics of complexity in software. It is now widely recognized that interaction is probably the most important single characteristic of complex software. Software architectures that contain many dynamically interacting components, each with their own thread of control, and engaging in complex, coordinated protocols, are typically orders of magnitude more complex to engineer correctly and efficiently than those that simply compute a function of some input through a single thread of control. Unfortunately, it turns out that many (if not most) real-world applications have precisely these characteristics. As a consequence, a major research topic in computer science over at least the past three decades has been the development of tools and techniques to model, understand, and implement systems in which interaction is the norm. Indeed, many researchers now believe that, in the future, computation itself will be understood chiefly as a process of interaction. Just as we can understand many systems as being composed of essentially passive objects, which have a state, and upon which we can perform operations, so we can understand many others as being made up of interacting, semi-autonomous agents. This recognition has led to the growth of interest in agents as a new paradigm for software engineering.

As I noted at the start of this chapter, the trend in computing has been – and will continue to be – towards ever more ubiquitous, interconnected computer systems. The development of software paradigms that are capable of exploiting the potential of such systems is perhaps the greatest challenge in computing at the start of the 21st century. Agents seem a strong candidate for such a paradigm.

It is worth noting that many researchers from other areas of computer science have similar goals to those of the multiagent systems community.

Self-interested computation

First, there has been a dramatic increase of interest in the study and application of *economic mechanisms* in computer science. For example, auctions are a well-known type of economic mechanism, used for resource allocation, which have achieved particular prominence in computer science [Cramton et al., 2006; Krishna, 2002]. There are a number of reasons for this rapid growth of interest, but perhaps most fundamentally, it is increasingly recognized

that a truly deep understanding of many (perhaps most) distributed and networked systems can only come after acknowledging that they have the characteristics of economic systems, in the following sense. Consider an online auction system, such as eBay [eBay, 2001]. At one level of analysis, this is simply a distributed system: it consists of various nodes, which interact with one another by exchanging data, according to some protocols. Distributed systems have been very widely studied in computer science, and we have a variety of techniques for engineering and analysing them [Ben-Ari, 1990]. However, while this analysis is of course legitimate, and no doubt important, it is surely missing a big and very important part of the picture. The participants in such online auctions are *self interested*. They are acting in the system *strategically*, in order to obtain the best outcome for themselves that they can. For example, the seller is typically trying to maximize selling price, while the buyer is trying to minimize it. Thus, if we only think of such a system as a distributed system, then our ability to predict and understand its behaviour is going to be rather limited. We also need to understand it from an *economic* perspective. In the area of multiagent systems, we take these considerations one stage further, and start to think about the issues that arise when the participants in the system *are themselves computer programs*, acting on behalf of their users or owners.

The Grid

THE GRID

The long-term vision of the *Grid* involves the development of large-scale open distributed systems, capable of being able to effectively and dynamically deploy and redeploy computational (and other) resources as required, to solve computationally complex problems [Foster and Kesselman, 1999]. To date, research in the architecture of the Grid has focused largely on the development of a software *middleware* with which complex distributed systems (often characterized by large datasets and heavy processing requirements) can be engineered. Comparatively little effort has been devoted to *cooperative problem solving* in the Grid. But issues such as cooperative problem solving are exactly those studied by the multiagent systems community:

MIDDLEWARE

COOPERATIVE PROBLEM SOLVING

> The Grid and agent communities are both pursuing the development of such open distributed systems, albeit from different perspectives. The Grid community has historically focussed on . . . 'brawn': interoperable infrastructure and tools for secure and reliable resource sharing within dynamic and geographically distributed virtual organisations (VOs) [Foster et al., 2001], and applications of the same to various resource federation scenarios. In contrast, those working on agents have focussed on 'brains', i.e. on the development of concepts, methodologies and algorithms for autonomous problem solvers that can act flexibly in uncertain and dynamic environments in order to achieve their objectives.
>
> [Foster et al., 2004]

Ubiquitous computing

The vision of *ubiquitous computing* is as follows:

> [P]opulations of computing entities – hardware and software – will become an effective part of our environment, performing tasks that support our broad purposes without

> our continual direction, thus allowing us to be largely unaware of them. The vision arises because the technology begins to lie within our grasp. This tangle of concerns, about future systems of which we have only hazy ideas, will define a new character for computer science over the next half-century. What sense can we make of the tangle, from our present knowledge?
>
> [Milner, 2006]

This vision is clearly connected with the trends that we discussed at the opening of this chapter, and makes obvious reference to cooperation and autonomy. We might expect that the ubiquitous computing and multiagent systems communities will have something to say to one another in the years ahead.

The semantic web

Tim Berners-Lee, inventor of the worldwide web, suggested that the lack of *semantic markup* on the current worldwide web hinders the ability of computer programs to usefully process information available on web pages. The 'markup' (HTML tags) used on current web pages only provides information about the layout and presentation of the web page. While this information can be used by a program to present a page, these tags give no indication of the *meaning* of the information on the page. This led Berners-Lee to propose the idea of the *Semantic Web*:

SEMANTIC MARKUP

> I have a dream for the Web [in which computers] become capable of analysing all the data on the Web – the content, links, and transactions between people and computers. A 'Semantic Web', which should make this possible, has yet to emerge, but when it does, the day-to-day mechanisms of trade, bureaucracy and our daily lives will be handled by machines talking to machines. The 'intelligent agents' [that] people have touted for ages will finally materialise.
>
> [Berners-Lee, 1999, pp. 169–170]

The semantic web vision thus explicitly proposes the use of agents. In later chapters, we will see how the kinds of technologies developed within the semantic web community have been used within multiagent systems.

Autonomic computing

Autonomic computing is described as:

AUTONOMIC COMPUTING

> [T]he ability to manage your computing enterprise through hardware and software that automatically and dynamically responds to the requirements of your business. This means self-healing, self-configuring, self-optimising, and self-protecting hardware and software that behaves in accordance to defined service levels and policies. Just like the nervous system responds to the needs of the body, the autonomic computing system responds to the needs of the business.
>
> [Murch, 2004]

Systems that can heal themselves and adapt autonomously to changing circumstances clearly have the character of agent systems.

1.2.2 Agents as a tool for understanding human societies

In Isaac Asimov's popular *Foundation* science fiction trilogy, a character called Hari Seldon is credited with inventing a discipline that Asimov refers to as 'psychohistory'. The idea is that psychohistory is a combination of psychology, history, and economics, which allows Seldon to predict the behaviour of human societies hundreds of years into the future. In particular, psychohistory enables Seldon to predict the imminent collapse of society. Psychohistory is an interesting plot device, but it is firmly in the realms of science fiction. There are far too many variables and unknown quantities in human societies to do anything except predict very broad trends a short term into the future, and even then the process is notoriously prone to embarrassing errors. This situation is not likely to change in the foreseeable future. However, multiagent systems do provide an interesting and novel tool for simulating societies, which may help shed some light on various kinds of social processes. A nice example of this work is the EOS project [Doran and Palmer, 1995]. The aim of the EOS project was to use the tools of multiagent systems research to gain an insight into how and why social complexity emerged in a Palaeolithic culture in southern France at the time of the last ice age. The goal of the project was not to directly simulate these ancient societies, but to try to understand some of the factors involved in the emergence of social complexity in such societies. (The EOS project is described in more detail later.)

1.3 Frequently Asked Questions (FAQ)

At this point, you may have some questions in mind, about how multiagent systems stand with respect to other academic disciplines. Let me try to anticipate them.

Isn't it all just distributed/concurrent systems?

The concurrent systems community has for several decades been investigating the properties of systems that contain multiple interacting components, and has been developing theories, programming languages, and tools for explaining, modelling, and developing such systems [Ben-Ari, 1990; Holzmann, 1991; Magee and Kramer, 1999]. Multiagent systems are – by definition – a subclass of concurrent systems, and there are some in the distributed systems community who question whether multiagent systems are sufficiently different from 'standard' distributed/concurrent systems to merit separate study. My view on this is as follows. First, it is important to understand that when designing or implementing a multiagent system, it is *essential* to draw on the wisdom of those with experience in distributed/concurrent systems. Failure to do so invariably leads to exactly the kind of problems that this community has been working for so long to overcome. Thus it is important to worry about such issues as mutual exclusion over shared resources, deadlock, and livelock when implementing a multiagent system.

In multiagent systems, however, there are two important twists to the concurrent systems story.

- Because agents are assumed to be autonomous – capable of making independent decisions about what to do in order to satisfy their design objectives – it is generally

assumed that the synchronization and coordination structures in a multiagent system are not hardwired in at design time, as they typically are in standard concurrent/distributed systems. We therefore need mechanisms that will allow agents to synchronize and coordinate their activities *at run-time*.

- The encounters that occur among computing elements in a multiagent system are *economic* encounters, in the sense that they are encounters between *self-interested* entities. In a classic distributed/concurrent system, all the computing elements are implicitly assumed to share a common goal (of making the overall system function correctly). In multiagent systems, it is assumed instead that agents are primarily concerned with their own welfare (although, of course, they will be acting on behalf of some user/owner).

For these reasons, the issues studied in the multiagent systems community have a rather different flavour from those studied in the distributed/concurrent systems community. We are concerned with issues such as how agents can reach agreement through negotiation on matters of common interest, and how agents can dynamically coordinate their activities with agents whose goals and motives are unknown.

Isn't it all just artificial intelligence?

The multiagent systems field has enjoyed an intimate relationship with the AI field over the years. Indeed, until relatively recently it was common to refer to multiagent systems as a subfield of AI, although multiagent systems researchers would indignantly – and perhaps accurately – respond that AI is more properly understood as a subfield of multiagent systems. More recently, it has become increasingly common practice to define the endeavour of AI itself as one of constructing an intelligent agent (see, for example, the enormously successful introductory textbook on AI by Stuart Russell and Peter Norvig [Russell and Norvig, 1995]). There are several important points to be made here:

- AI has largely (and, perhaps, mistakenly) been concerned with the *components* of intelligence: the ability to learn, plan, understand images, and so on. In contrast, the agent field is concerned with entities that *integrate* these components, in order to provide a system that is capable of making independent decisions. It may naively appear that, in order to build an agent, we need to solve *all* the problems of AI itself: in order to build an agent, we need to solve the planning problem, the learning problem, and so on (because our agent will surely need to learn, plan, and so on). This is not the case. As Oren Etzioni succinctly put it: 'Intelligent agents are ninety-nine percent computer science and one percent AI' [Etzioni, 1996]. When we build an agent to carry out a task in some environment, we will very likely draw upon AI techniques of some sort – but most of what we do will be standard computer science and software engineering. For the vast majority of applications, it is not necessary that an agent has all the capabilities studied in AI – for some applications, capabilities such as learning may even be undesirable. In short, while we may draw upon AI techniques to build agents, we do not need to solve all the problems of AI to build an agent.

- Classical AI has largely ignored the *social* aspects of agency. I hope you will agree that part of what makes us unique as a species on Earth is not simply our undoubted ability to learn and solve problems, but our ability to communicate, cooperate, and reach agreements with our peers. These kinds of social ability – which we use every day of our lives – are surely just as important to intelligent behaviour as are components of intelligence such as planning and learning, and yet they were not studied in AI until about 1980.

Isn't it all just economics/game theory?

Game theory is a mathematical theory that studies interactions among self-interested agents [Binmore, 1992]. It is interesting to note that von Neumann, one of the founders of computer science, was also one of the founders of game theory [von Neumann and Morgenstern, 1944]; Alan Turing, arguably the other great figure in the foundations of computing, was also interested in the formal study of games, and it may be that it was this interest that ultimately led him to write his classic paper *Computing Machinery and Intelligence*, which is commonly regarded as the foundation of AI as a discipline [Turing, 1963]. However, after these beginnings, game theory and computer science went their separate ways for some time. Game theory was largely – though by no means solely – the preserve of economists, who were interested in using it to study and understand interactions among economic entities in the real world.

Recently, the tools and techniques of game theory have found many applications in computational multiagent systems research, particularly when applied to problems such as negotiation. Indeed, at the time of writing, game theory seems to be the predominant theoretical tool in use for the analysis of multiagent systems. An obvious question is therefore whether multiagent systems are properly viewed as a subfield of economics/game theory. There are two points here.

- Many of the solution concepts developed in game theory (such as Nash equilibrium, discussed later) were developed without a view to computation. They tend to be *descriptive* concepts, telling us the properties of an appropriate, optimal solution *without* telling us how to compute a solution. Moreover, it turns out that the problem of computing a solution is often computationally very hard (e.g. NP-complete or worse). Multiagent systems research highlights these problems, and allows us to bring the tools of computer science (e.g. computational complexity theory [Garey and Johnson, 1979; Papadimitriou, 1994]) to bear on them.

- Some researchers question the assumptions that game theory makes in order to reach its conclusions. In particular, debate has arisen in the multiagent systems community with respect to whether or not the notion of a rational agent, as modelled in game theory, is valid and/or useful for understanding human or artificial agent societies.

Note that all this should *not* be construed as a criticism of game theory, which is without doubt a valuable and important tool in multiagent systems, likely to become much more widespread in use over the coming years.

Software Agents in Popular Culture

Software technologies do not seem the most obvious subject matter for novels or films, but autonomous software agents have a starring role surprisingly often. Part of the reason may be that agents are seen as an 'embodiment' of the artificial intelligence dream, which has long been a subject for story makers. The computer Hal, in the film *2001: A Space Odyssey* is the best-known example. However, the kinds of issues addressed in this book have also made other appearances in film and fiction. One of the earliest mentions that I am aware of was in Douglas Adams' novel *Mostly Harmless*, where he imagines software agents cooperating to try to control a damaged spacecraft:

> Small modules of software – agents – surged through the logical pathways, grouping, consulting, re-grouping. They quickly established that the ship's memory, all the way back to its central mission module, was in tatters.

Some authors like to play on the fact that the word 'agent' has multiple meanings: in the Wachowski brothers' *Matrix* trilogy of films, the character Neo must do battle in a virtual world with 'agents' that are clearly intended to be of both the autonomous software and the secret variety.

Michael Crichton's novel *Prey* is based on the premise of agents, embodied in nano-machines, going (badly!) wrong. He clearly did some research about multiagent systems:

> Basically, you can think of a multiagent environment as something like a chessboard, the agents like chess pieces. The agents interact . . . to achieve a goal. . . . The difference is that nobody is moving the agents. They interact on their own to produce the outcome.

Finally, the main character of the David Lodge novel *Thinks* is an artificial intelligence researcher, who has an affair with a student, who subsequently blackmails him to publish her scientific paper – entitled 'Modelling Learning Behaviours in Autonomous Agents'! I am happy to report that, in my experience at least, this kind of behaviour really is limited to fiction.

Isn't it all just social science?

The social sciences are primarily concerned with understanding the behaviour of human societies. Some social scientists are interested in (computational) multiagent systems because they provide an experimental tool with which to model human societies. In addition, an obvious approach to the design of multiagent systems – which are artificial societies – is to look at how human societies function, and try to build the multiagent system in the same way. (An analogy may be drawn here with the methodology of AI, where it is quite common to study how humans achieve a particular kind of intelligent capability, and then to attempt to model this in a computer program.) Is the multiagent systems field therefore simply a subset of the social sciences?

Although we can usefully draw insights and analogies from human societies, it does not follow that we should build artificial societies in the same way. It is notoriously hard to model precisely the behaviour of human societies, simply because they are dependent on so many different parameters. Moreover, although it is perfectly legitimate to design a multiagent system by drawing upon and making use of analogies and metaphors from human societies, it does not follow that this is going to be the *best* way to design a multiagent system: there are other tools that we can use equally well (such as game theory – see above).

It seems to me that multiagent systems and the social sciences have a lot to say to each other. Multiagent systems provide a powerful and novel tool for modelling and understanding societies, while the social sciences represent a rich repository of concepts for understanding and building multiagent systems – but they are quite distinct disciplines.

Notes and Further Reading

There are now many introductions to intelligent agents and multiagent systems. [Ferber, 1999] is an undergraduate textbook, although it was written in the early 1990s, and so (for example) does not mention any issues associated with the Web. A first-rate collection of articles introducing agent and multiagent systems is [Weiß, 1999]. Many of its articles address issues in much more depth than is possible in this book. I would certainly recommend this volume for anyone with a serious interest in agents, and it would make an excellent companion to the present volume for more detailed reading.

Three collections of research articles provide a comprehensive introduction to the field of autonomous rational agents and multiagent systems: Bond and Gasser's 1988 collection, *Readings in Distributed Artificial Intelligence*, introduces almost all the basic problems in the multiagent systems field, and although some of the papers it contains are now rather dated, it remains essential reading [Bond and Gasser, 1988]; Huhns and Singh's more recent collection sets itself the ambitious goal of providing a survey of the whole of the agent field, and succeeds in this respect very well [Huhns and Singh, 1998]. Finally, [Bradshaw, 1997] is a collection of papers on software agents.

For a general introduction to the theory and practice of intelligent agents, see [Wooldridge and Jennings, 1995], which focuses primarily on the theory of agents, but also contains an extensive review of agent architectures and programming languages. A short but thorough roadmap of agent technology was published as [Jennings et al., 1998].

Class reading: introduction to [Bond and Gasser, 1988]. This article is probably the best survey of the problems and issues associated with multiagent systems research yet published. Most of the issues it addresses are fundamentally still open, and it therefore makes a useful preliminary to the current volume. It may be worth revisiting when the course is complete.

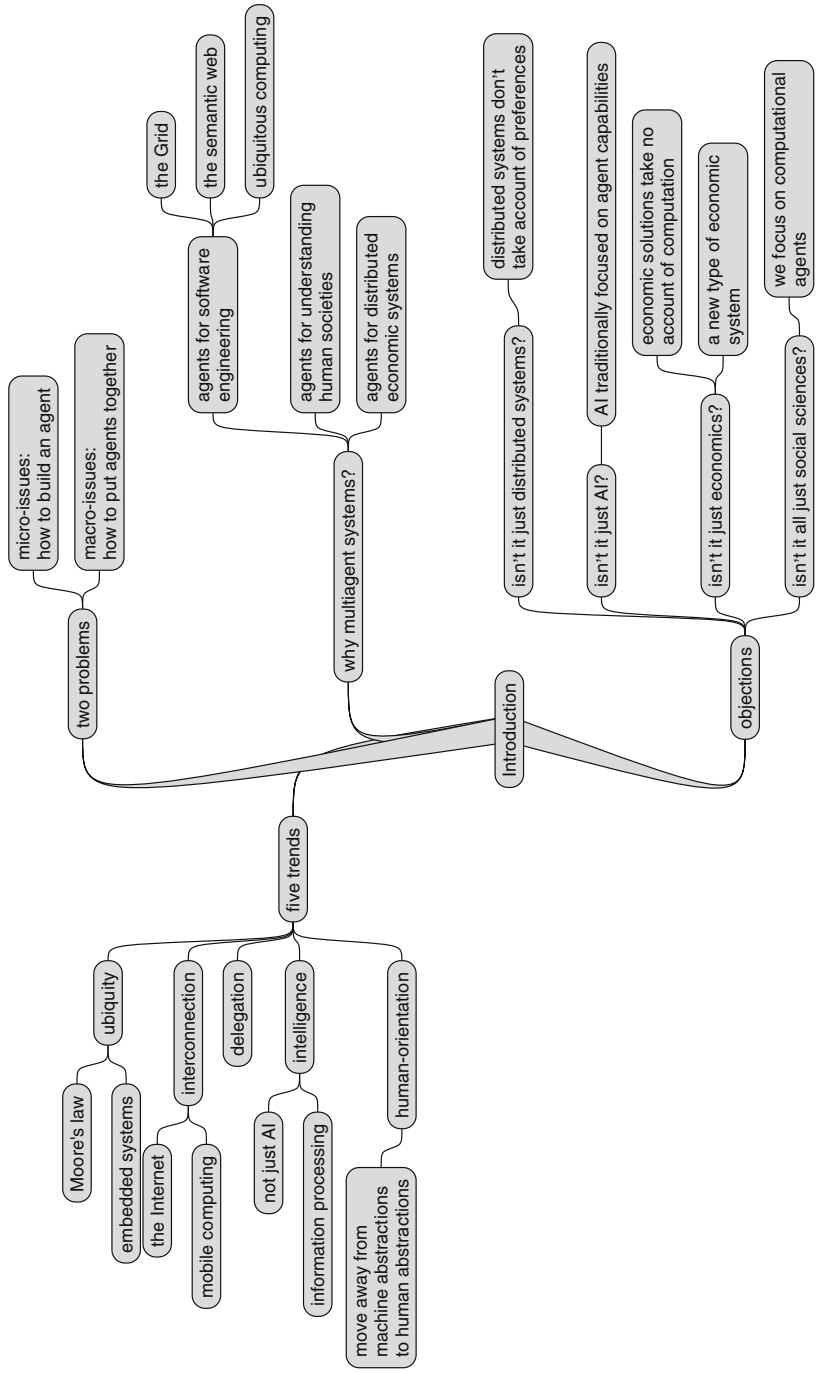

Figure 1.1: Mind map for this chapter.