Actions and Agents
Eduardo Alonso
Department of Computer Science
City University London, London EC1V 0HB
E.Alonso@city.ac.uk

1   Introduction

In this chapter the notion of agency in AI is presented..It has been argued that in order to behave rationally in prevalent software applications artificial entities would have to be autonomous and adaptive. Besides, rather than working with single, isolated systems the new trend in AI would need to focus on inherently social entities in the form of multi-agent systems.

The chapter begins by introducing the notion of action in traditional AI systems, deliberative and reactive. Next, the idea of agency is presented as complementary to classical AI approaches to action as well as a key factor in the development of applications and services that currently rely on technologies where software systems execute instructions automatically. In particular, the chapter highlights the importance of developing theories of action and learning in multi-agent scenarios such as the Internet. This introduction shall conclude with some considerations about the research areas that need to be targeted if the agent paradigm is to become the standard in the design, specification and implementation of intelligent systems.

2   Action in traditional AI

Historically, the "physical symbol system hypothesis" in traditional AI (Newell and Simon 1976) has been embedded in so-called deliberative systems. Such systems are characterized by containing symbolic models of the world, and decisions about which actions to perform are made via manipulation of these symbols. To get an AI system to "act" it is enough to give it a logical representation of a theory of action (how systems make decisions and act accordingly) and "get it to do a bit of theorem proving".

This approach to action is perhaps best illustrated in the *planning problem* where systems use symbolic manipulation to reason about which actions to execute to achieve their goals, that is, to reason about how to behave efficiently. Typically (Fikes and Nilsson 1971), the system will be given a description of the state of the world it is in (the initial state) and of the desired state of the world (the final state or goal). The system will also be provided with a set of actions, each accompanied with a list of pre-conditions for the action to be executed and a list of effects that result from the action being executed –that is, such lists encode how the world changes when the action is completed, which predicates are deleted and which added to the description of the world. For example, imagine that the world consists of two blocks and a table and that the initial state of the world is "block B on table, block A on block B, nothing on block A" or, as formally, `{OnTable(B), On(A, B), Clear(A)}`; also imagine that the goal is to have "block B on table and block A on table", that is, `{OnTable(A),OnTable(B)}` and that the system is able to execute two actions,

`UnStack(x, y)` and `PutDown(x)`. These actions are accompanied by the following lists of pre-conditions and effects, for `UnStack(x, y)` and `PutDown(x)` respectively:

```
Pre {On(x, y), Clear(x)}
Del {On(x, y)}
Add {Holding(x), Clear(y)}

Pre {Holding(x)}
Del {Holding(x)}
Add {OnTable(x)}
```

Clearly, in this example the plan consisting of the sequence of actions `{UnStack(A, B), PutDown(A)}` will bring the world from its initial state to the goal. At each step, the system that executes the planning algorithm (the planner) tries to match the preconditions for various actions to the description of the actual world. For example, the planner may begin by attempting `PutDown(A)`, but fail since the precondition for this action (holding `A`) does not obtain. On the other hand, the preconditions for the action `UnStack(A, B)` do hold (`A` is stacked on `B`), so this action can be executed. As a result of executing this second action in turn, the state of the world becomes `{OnTable(A), Clear(B), Clear(A), OnTable(B)}`, which satisfies the goal `{OnTable(A), OnTable(B)}`.

Unfortunately, given the computational complexity of theorem proving in even very simple logics this approach to the design and implementation of rational systems has not been widely applied in real-life scenarios. It has been proved (Chapman 1987) that even refined techniques will ultimately turn out to be unusable in any time-constrained system –as the extremely simple example above illustrates, it just takes too long to search through all possible combinations to deduce the goals (theorems) from a set of initial conditions (premises). These results had a profound influence on AI research, caused some researchers to question the symbolic AI paradigm and led to alternative approaches, reactive architectures particularly.

A reactive system is a system that does not use a symbolic model of the world nor symbolic reasoning to decide what to do next. Reactive architectures are modeled as *black boxes*: they follow if-then rules that map directly inputs, information received from their sensors, into actions. Without a model of the world or the task at hand such systems are cognitively elementary –they (re)act more like caterpillars rather than as human beings. Perhaps the most paradigmatic example of this type of system is the *subsumption architecture,* which establishes a hierarchy of competing behaviours where lower layers have precedence over higher ones (Brooks 1986).

For example, let's imagine a reactive robot that picks up samples from, say, the surface of Mars. Suppose the robot is given the following (situation → action) rules:

     (1)   `If` detect an obstacle `then` change direction.
     (2)   `If` carrying samples and at the base `then` drop samples.
     (3)   `If` carrying samples and not at the base `then` travel up gradient.
     (4)   `If` detect a sample `then` pick sample up.
     (5)   `If` true `then` move randomly.

Such rules form a hierarchy that ensures that the robot will turn if it finds an obstacle; if it is at the base and carrying samples, then it will drop them provided there is no immediate danger of crashing, and so on. The highest behavior –a random walk– will only be carried out if the agent has nothing more urgent to do: its "`If` true" precondition is assumed to always fire. It is a way of guaranteeing that, if rules (1)-(4) don't apply, the robot will still do something.

The resulting systems are, computationally speaking, extremely simple, and yet they are able to execute complex tasks. In addition, reactive systems are situated in real-life domains and able to display flexible behavior. In fact, actions are not planned ahead but are rather the emergent result of the system's "embeddedness" in a particular situation.

However interesting this approach may be, it presents several problems. Reactive systems learn procedures but no declarative knowledge; that is, they only learn values or attributes that are not easy to generalize to similar situations (or transmit to other systems). Besides, and perhaps more importantly, precisely because they show emergence properties there is no principled *methodology* for building such systems.

Regardless of the many attempts to combine deliberative and reactive architectures in hybrid systems (Ferguson 1992, Müller 1997), it seems that at the end of the day one is left to choose between theoretically sound but impractical deliberative systems and efficient yet loosely designed reactive systems. This may reflect the fact that each type of AI system was designed to solve related yet different problems: symbolic AI resulted from the effort to formalize and mechanize reasoning that blossomed with the development of expert systems; reactive systems on the other hand were often motivated by efforts to solve numerical, non-linear problems such as those associated with connectionism and Artificial Life.

Now, for the last couple of decades researchers have experienced the emergence of new technologies such as the Internet. These demand personal, continuously running systems for which older notions of action –those resulting from either cumbersome symbolic reasoning or ever-adaptive reflexes– may be insufficient. Indeed, many researchers believe that in the XXI century for AI systems to perform "intelligently" they must be able to behave in an *autonomous*, *flexible* manner in unpredictable, dynamic, typically *social* domains. In other words, they believe that the "new" AI should develop *agents* (Alonso 2002).

In fact, it can be argued that current trends in web development and web design as well as new applications in electronic commerce (for instance, PayPal) and social software (for example, facebook) will be only fully developed if an agents' perspective is adopted.

3   The three principles of agent-centered AI

This section examines in detail the main functionalities software systems would display in a social, agent-centered AI or, in other words, the principles of behavior of the "new" AI.

3.1   Autonomous behavior

By autonomy researchers mean the ability of the systems to make their own decisions and execute tasks on the designer's behalf. The idea of delegating some responsibility to the system to avoid tediously writing down code is certainly very attractive. Moreover, in scenarios where it is difficult to control directly the behavior of our systems, the ability of acting autonomously is essential. For example, space missions increasingly depend on their unmanned space-crafts and robots to make decisions on their own: this ability is paramount since the costs (in time and money) of communication between the space station and such systems can be prohibitive.

It is precisely this autonomy that defines agents. Traditionally, software systems execute actions (so-called methods) automatically: Imagine that the web application in your computer, the user or client, requests access the contents of a webpage that is stored in another software system elsewhere, the server or host. The server cannot deny access to the content of the webpage; it must execute the "send" method whenever it is requested to do so. On the contrary, agents decide by themselves whether to execute their methods according to their beliefs, desires and intentions (Bratman *et al.* 1988). Paraphrasing (Jennings *et al.* 1998), "what traditional software systems do for free, agents do for money".

3.2   Adaptive behavior

Secondly, agents must be flexible. When designing agent systems, it is impossible to foresee all the potential situations they may encounter and specify their behavior optimally in advance. For example, the components of interaction in the Internet (agents, protocols, languages) are not known *a priori*. Agents therefore have to learn from and adapt to their environment. This task is even more complex when Nature is not the only source of uncertainty, but the agent is situated in a multi-agent system (MAS) that contains other agents with potentially different capabilities, goals, and beliefs.

Besides, the new systems must be general. An agent must have the competence to display an action repertoire general enough to preserve its autonomy in dynamic environments. Certainly, an agent can hardly be called intelligent if it is not able to perform well when situated in an environment different from (yet in some ways similar to) the one it was originally designed for.

Indeed, there is no need to learn anything in static, closed domains where agents have perfect knowledge of state-action transitions. Nonetheless, intelligence and learning are tightly tied in domains where autonomous agents must make decisions with partial or uncertain information, that is, in domains where agents learn without supervision and without the luxury of having a complete model of the world, that is when facing the so-called *reinforcement learning problem* (Kaelbling *et al.* 1996). In such scenarios, each time an agent executes an action in a state it receives a numerical reward that indicates the immediate value of this state-action transition –how "good" it is. This produces a sequence of states, actions and rewards. The agent's task is to learn a policy that maximizes the expected sum of rewards, typically with future rewards discounted exponentially by their delay. In other words, the more into the future the predictions are, the less likely the rewards will count, a sensible principle since more distant rewards are less probable. Unlike supervised learning such as pattern recognition or

neural networks, the learner is not told which actions to take, but instead must discover which actions yield the most reward by exploiting and exploring their relationship with the environment. Actions may affect not only the immediate reward but also the next situation and, through that, all subsequent rewards. These two characteristics, trial and error search and delayed reward, are the two most important features of reinforcement learning.

This method has been successfully applied to several organizational problems in robotics, control, operation research, games, human computer interaction, economics/finance, complex simulation, and marketing.

3.3   Social behavior

Agents also show a social attitude. In an environment populated by heterogeneous entities, agents would need the ability to recognize their opponents, and to form groups when it is profitable to do so. It is not a coincidence that most agent-based platforms incorporate multi-agent tools (Luck *et al*. 2005). Indeed, some authors do state that agent-oriented software engineering needs to be developed precisely because there is no notion of organizational structure in the traditional software systems (Etzioni and Weld, 2007).

Generally speaking, the design and implementation of multi-agent systems is an attractive platform for the convergence of various AI technologies. That is the underlying philosophy of competitions such as RoboCup (http://www.robocup.org/) where teams of soccer agents must display their individual and collective skills in real-time. More importantly, multi-agent systems play several roles in IT and telecoms: for clients, they provide personalized, user-friendly interfaces; as middleware, they have been used extensively to implement electronic markets and electronic auctions.

The reasons for this happy marriage between MAS and new technologies are various. When the domain involves a number of distinct software systems that are physically or logically distributed (in terms of their data, expertise or resources), an agent-based approach can often provide an effective solution. Relatedly, when the domain is large, sophisticated, or unpredictable, the overall problem can indeed be partitioned into a number of smaller and simpler components, which are easier to develop and maintain, and which are specialized at solving the constituent problems. That is, in most real-life applications (single) agents can grow "too big" to work well, and a *divide and conquer* strategy, where qualified agents work in parallel, seems more sensible. Examples include the geographical distribution of cameras in a traffic network or the integrated approach required to solve complex tasks, for instance the collaboration between experts (surgeons, anesthetists, nurses) in an operating room.

To sum it up, it is widely accepted within the AI community that the "new" AI would need to design and implement multi-agent systems capable of acting and learning in a quick and efficient manner. The next two sections are dedicated to describing the basics of multi-agent behavior and multi-agent learning.

4   Multi-agent behavior

Approaches to multi-agent behavior differ mainly in regards to the degree of control that the designer should have over individual agents and over the social environment, *i.e.*, over the interaction mechanisms (Bond and Gasser 1988, Durfee 1988, Weiss 1999). In Distributed Problem Solving systems (DPS) a single designer is able to control (or even explicitly design) each individual agent in the domain –the task of solving a problem is distributed among different agents, hence the name; in MAS on the other hand, there are multiple designers and each is able to design only its agent and has no control over the internal design of other agents.

The design of interaction protocols is also tightly coupled to the issue of agents' incentives. When agents are centrally designed they are assumed to have a common general goal. As long as agents have to co-exist and cooperate in a single system, there is some notion of global utility that each agent is trying to maximize. Agents form teams that jointly contribute towards the overall goal. By contrast, in MAS each agent will be individually motivated to achieve its own goal and to maximize its own utility. As a result, no assumptions can be made about agents working together cooperatively. On the contrary, agents will collaborate only when they can benefit from that cooperation.

Research in DPS considers how work involved in solving a problem can be divided among several nodes so as to enhance the system's performance, that is, the aim is to make independent nodes solve a global problem by working together coherently, while maintaining low levels of communication. MAS researchers are also concerned with the coherence of interaction, but must build agents without knowing how their opponents have been designed. The central research issue in MAS is how to have these autonomous agents identify common ground for cooperation, and choose and perform coherent actions.

In particular, DPS researchers see negotiation as a mechanism for assigning tasks among agents and for allocating resources, using *automated contracting*: since all agents have a common goal and are designed to help one another (following the so-called *benevolence* assumption), there is no need to motivate an agent to agree to execute a set of actions. Alternatively, *multi-agent planning* is another DPS approach that avoids incoherent and inconsistent decisions by planning beforehand exactly how each agent will act and interact. Multi-agent planning has been formalized by extending single-agent planning languages and techniques to describe complex mental states –usually by defining social plans in terms of common beliefs and joint intentions (Rao *et al.* 1992).

On the other hand, MAS researchers have autonomous agents use negotiation to share the work associated with carrying out a previously agreed plan (for the agents' mutual benefit), or to resolve outright conflict. In MAS systems, agents typically make pair-wise agreements through negotiation about how they will co-ordinate, and there is no global control nor consistent knowledge nor shared goals or success criteria. So, the main purpose of this *incentive contracting* mechanism is to "convince" agents to reach reasonable agreements and do something in exchange for something else. In this case, AI researchers have followed the studies on bargaining with incomplete information developed in economics and game theory.

4.1   Negotiation

Since negotiation in MAS is probably the most common co-ordination technique, it is worth considering it in some detail (Rosenschein and Zlotkin 1994, Jennings *et al.* 2001, Kraus 2001).

In a MAS setting agents are given a negotiation mechanism consisting of a protocol and a set of strategies over a set of deals. Negotiation is defined as a process through which in each temporal point one agent proposes an agreement and the other agent either accepts the offer or does not. If the offer is accepted, then the negotiation ends with the implementation of the agreement. Otherwise, the second agent has to make a counteroffer, or reject its opponent's offer and abandon the process. So, the protocol specifies when and how to exchange offers (*i.e.*, which actions the agents will execute or abstain from executing and when) –for example, an `Offer(x, y, `δ_i`, t`$_1$`)` means that the negotiation process will start at time `t`$_1$ with agent `x` offering agent `y` a deal δ_i from the set of potential deals ($\delta_i \in \Delta$), typically of the from "I will do action 1 in exchange for action 2" or `{Do(x, a`$_1$`), Do(y, a`$_2$`)}`; then, in the next negotiation step, agent `y` will counteroffer with `Accept(y, `δ_i`, t`$_2$`)`, in which case the negotiation episode ends with the implementation of the agreement, δ_i; or with `Reject(y, `δ_i`, t`$_2$`)`, so that negotiation fails. Or, alternatively, agent `y` can send a respond, `Offer(y, x, `δ_j`, t`$_2$`)`, with say δ_j `= {Do(x, a`$_3$`), Do(y, a`$_2$`)}`, "I would prefer you to execute action `a`$_3$ rather than `a`$_1$", so that the negotiation progresses to the next stage in which the same routine applies.

Which specific offers the agents make depend on their negotiation strategy. A negotiation strategy is a function from the history of the negotiation to the current offer that is consistent with the protocol. It determines what move an agent should make to maximize its own utility, given the protocol, the negotiation up to this point, and the agent's beliefs and intentions. Such strategies also take into account how risk-averse the agent might be, that is, how reluctant it is to accept a bargain with an uncertain outcome rather than another bargain with a more certain, but possibly lower, outcome.

Usually strategies are demanded to be in what is called Nash-equilibrium: that is, no agent should have an incentive to deviate from agreed-upon strategies. Once a strategy is adopted, under the assumption that agent `x` uses it, agent `y` cannot do better by using a different strategy. Imagine the so-called *Prisoners' Dilemma:* two suspects are arrested by the police. The police have insufficient evidence for a conviction, and, having separated both prisoners, visit each of them to offer the same deal. If one testifies (defects from the other) and the other remains silent, the betrayer goes free and the silent accomplice receives the full 10-year sentence. If both remain silent, both prisoners are sentenced to only six months in jail for a minor charge. If each betrays the other, each receives a five-year sentence. Each prisoner must choose to betray the other or to remain silent. Obviously, the suspects cannot talk to each other or to reach an agreement. In this case, the Nash-equilibrium is that both testify. Each suspect knows that if one chose to remain silent the other one would do better by testifying, thus breaking the "remain silent equilibrium". Nash-equilibrium is a particularly important attribute, because it is seen as the only sustainable outcome of rational negotiation in the absence of externally enforceable agreements. Yet, this solution presents serious drawbacks:

Firstly, there are situations in which there is no Nash equilibrium. For instance, *Matching Pennies* is an example of games where one player's gain is exactly equal to the other player's loss.

There are other situations in which there are several pure Nash equilibria. In a simplified example, assume that two drivers meet on a narrow road. Both have to swerve in order to avoid a head-on collision. If both swerve to the same side they will manage to pass each other, but if they choose different sides they will collide. In this case there are two pure Nash equilibria: either both swerve to the left, or both swerve to the right. In this example, it doesn't matter which side both players pick, as long as they both pick the same. Since both strategies are equally good, one could just toss a coin to choose between the two alternatives. There are other situations, however, in which one would not have that choice: in the game *Battle of the Sexes* both players prefer engaging in the same activity over going alone, but their preferences differ over which activity they should engage in. Player 1 prefers that they both party while player 2 prefers that they both stay at home. In this case, there are two pure Nash equlibria but no agreement is reached.

Finally, accepting a Nash equilibrium solution both agents may lose more profitable agreements. This is the case of the *Prisoner's Dilemma*: the Nash-equilibrium for this game is a sub-optimal solution, the one that leads the two players to both play defect, even though each player's individual reward would be greater if they both played cooperatively and remained silent.

Thus, instead of Nash-equilibrium constraints and in order to prevent irrational attitudes the following assumptions about *social rationality* are typically made:
(1) Sincerity*:* no agent will attempt to have another believe a proposition that it either knows or believes to be false or a proposition it wants to be false. For example, agents cannot commit themselves to execute actions that they are not able to perform; (2) Honesty: agents have to act according to their beliefs; (3) Fair play: agents must abide by the agreed deals; (4) Sociability*:* in case of indifference, agents must accept others' offers. In any case, deals must always be individually rational.

4.2   Argumentation

The assumptions about social rationality required to make the previous approach work are not intuitive, and in any case, many real agents calculate their options individually in terms of self-interest, ignoring negotiations and agreed commitments. In response, many members of the MAS community have adopted alternative approaches to MAS co-ordination. In particular, several studies on argumentation-based negotiation have been presented as a powerful technique for cooperating and solving conflict situations (Rahwan *et al.* 2003). In this type of negotiation agents open up the agreement space by exchanging not only proposals and counterproposals but also reasons supporting them. Besides, agents commit themselves to accept the results of the argumentation, which follows strict rules regarding the validity and acceptability of the arguments and their ordering in argumentative types.

For instance, imagine the following situation: agent 1 has a hammer, a screw, a screwdriver and a picture it intends to hang by using the "plan" `{hammer + nail + picture}`; agent 2 on the other hand owns a mirror and a nail, its goal is to hang

the mirror and plans to execute `{hammer + nail + mirror}`. Now, agent 1 knows agent 2 has a nail and asks for it. Obviously, agent 2 cannot agree to such a request since it needs the nail to hang the mirror. Using a negotiation protocol, agent 2's rejection will end the episode and neither agent will achieve their respective goals. However, since they are allowed now to argue agent 2 can explain why it is rejecting agent's 1 offer ("I need it to hang my mirror"); with this information, agent 1 can persuade agent 2 that in fact there is another way to hang its mirror, a new plan that uses a screw and a screwdriver instead of a nail. Now, if agent 2 does not find a flaw in agent 1's argument it is forced to accept it. Since this seems to be the case, the agents agree to exchange the nail for the screw and the screwdriver and, as a consequence, both achieve their objectives.

This completes our account of the main issues and techniques in multi-agent behavior. Now, as introduced in section 3.3, behaving in complex dynamic scenarios such as MAS is not a one-shot task but a process of refinement through which agents adapt their strategies to each other's. Hence, dealing with multi-agent learning is paramount when studying multi-agent behavior.

5 Multi-agent learning

Machine learning has been mostly independent of agent research and only recently has it received attention in connection with agents and multi-agent systems (Stone and Veloso 2000, Alonso *et al.* 2001, Vohra and Wellman 2007). This is in some ways surprising because the ability to learn and adapt is arguably one of the most important features of intelligence. As discussed above, intelligence implies a certain degree of autonomy that in turn requires the ability to learn to make independent decisions in dynamic, unpredictable domains such as those in which agents co-exist.

Perhaps the two more important issues in multi-agent learning relate to which family of techniques should be used and, indeed, what multi-agent learning is.

At one level, agents and multi-agent systems can be viewed as yet another application domain for machine learning systems, admittedly with its own challenges. Research taking this view is mostly reduced to applying existing single-agent learning algorithms more or less directly to MAS, so that multi-agent learning is only seen as an emergent property. Even though this could be interesting from a MAS point of view, it does not seem overly interesting for machine learning research. Nevertheless, this is the direction most learning research for MAS has been following.

Existing learning algorithms have been developed for single agents learning separate and independent tasks. Alternatively, multi-agent systems pose the problem of *distributed learning*, that is, many agents learning separately to acquire a joint task. Once the learning process is distributed amongst several learning agents, such learning algorithms require extensive modification, or completely new algorithms need to be developed. In distributed learning, agents need to cooperate and communicate in order to learn effectively; these issues are being investigated extensively by MAS researchers but, to date, they have received little attention in the areas of learning.

Regarding learning techniques, supervised learning methods are not easily applied to multi-agent scenarios since they typically assume that the agents can be provided with the correct behaviour for a given situation. Thus most researchers have used reinforcement learning methods, to the point that the multi-agent learning problem can be re-defined as the reinforcement learning problem for multi-agent systems (Busoniu *et al.* 2008).

Specifically, the simplest way to extend single-agent learning algorithms to multi-agent problems is just to make each agent learn independently. Agents learn as if they were alone (Weiss and Dillenbourg, 1999). Communication or explicit co-ordination is not an issue therefore –co-operation and competition are not tasks to be solved but just properties of the environment. Likewise, agents do not have models of other agents' mental states or try to build models of other agents' behaviors. However simple this approach to multi-agent learning may be, the assumption that agents can learn efficient policies in a MAS setting independently of the actions selected by other agents is implausible. Intuitively, the most appealing alternative is to have the agents learn Nash-equilibrium strategies. However, as described in section 4.1 the concept of Nash equilibrium is problematic, and the methods formulated using such approach suffer from a plethora of technical difficulties that make their application rather restricted.

6   Challenges

Agent-based applications have enjoyed considerable success in manufacturing, process control, telecommunications systems, air traffic control, traffic and transportation management, information filtering and gathering, electronic commerce, business process management, entertainment and medical care (Jennings and Wooldridge 1998).

Nonetheless, one of the key problems has been the divide between theoretical and practical work, which have, to a large extent, developed along different paths. As a consequence, designers lack a systematic methodology for clearly specifying and structuring their applications as (multi-)agent systems. Most agent-based applications have been designed in an *ad hoc* manner either by borrowing a methodology from more traditional approaches or by designing the system on intuition and (necessarily limited) experience. At any rate, if agents and multi-agent systems are to become the standard in the development of emerging web-based application -- as their advocates believe they should -- then some important developments in agent-oriented methodologies and technologies will be needed.

First, an agent modeling language to specify, visualize, modify, construct and document (multi-)agent systems would have to be built. Agent developers still characterize their systems as extensions of traditional systems and thus UML is the facto standard language in the design and specification of agents and multi-agent systems. This drawback extends to the lack of proper verification methods and techniques for agent systems.

Second, while some programming features such as abstraction, inheritance and modularity make it easier to manage increasingly more complex systems, Java and other programming languages cannot provide a direct solution to agent

implementation. So far agent-oriented programs have been used mainly to test ideas rather than for developing any realistic systems –but see (Bordini *et al*. 2005) for a survey of multi-agent programming, languages, platforms and applications.

Third, standards for interoperability between agents will need to be established. The debate should not be focused exclusively on the pros and cons of different agent communication languages and protocols but also on ontologies, that is, on which types of entities and concepts define an agent domain and what are their properties and relations. Currently, ontologies are often specified informally or implicit in the agent implementation. For true interoperation, agents will need explicitly encoded, sharable ontologies.

A fourth issue is reusability. If multi-agents systems are to be sustainable, it will be necessary to develop techniques for specifying and maintaining reusable models and software for multi-agent systems, agents, and agent components. Reusability is also needed for mobility. If agents are to roam wide area networks such as the WWW, then they must be capable of being continuously reused in different scenarios.

Finally, if people are to be comfortable with the idea of delegating tasks to agents, then issues relating to trust will have be addressed. These include authentication, privacy of communication and user's personal profile information, auditing, accountability, and defense against malicious or incompetent agents.

All in all, although there is a need to keep theory and practice at the same pace agent-centered AI has already brought mature and integrative techniques and procedures that are ripe for exploitation. It can be claimed that the agent paradigm has served as a bridge between traditional AI systems and the software applications that have emerged in the last couple of decades. When on the occasion of *AI Magazine*'s twenty-fifth anniversary experts were asked about AI's state of the art, the shared feeling was that AI needed to get back to building *intelligent* systems of *general competence* (Leake 2005). It seems that agents and multi-agent systems may provide us with the concepts, methodologies and techniques necessary to realize AI's original goal in the services and applications that the Internet offers.

6   Conclusions

AI systems have to make intelligent decisions. But, most importantly, they must show that they do so by behaving accordingly. This chapter has focused on the role of agents in the analysis of the behavior of AI systems. After all, that is what agents do: they act. Hence, the study of behavior and action in AI does necessarily talk about agents. In fact, there are strong reasons for thinking that agents are the paradigm that will embody the "new" AI. More precisely, in the era of the Internet and web services, AI will come to focus on how collections of autonomous agents co-ordinate their behavior (multi-agent behavior) and on how they learn to do so (multi-agent learning).

References

Alonso, Eduardo 2002. AI and Agents: State of the Art, *AI Magazine* 23(3): 25-30.

Alonso, Eduardo, d'Inverno, Mark, Kudenko, Daniel, Luck, Michael and Noble, Jason 2001. Learning in Multi-Agent Systems, *Knowledge Engineering Review* 16 (3), 277-284.

Bond, Alan H. and Gasser, Less, (eds.) 1988. *Readings in Distributed Artificial Intelligence*. San Mateo, CA: Morgan Kaufmann Publishers.

Bordini, Rafael, Dastani, Mehdi, Dix, Jurgen. and El Fallah-Seghrouchni, Amal (eds.) 2005. *Multi-Agent Programming: Languages, Platforms and Applications*. Berlin: Springer-Verlag.

Bratman, Michael, Israel, David J., and Pollack, Martha E. 1988. Plans and resource-bounded practical reasoning, *Computational Intelligence*, 4 (3), 349-355.

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

Busoniu, Lucian, Babuska, Robert and De Schutter, Bart 2008. A Comprehensive Survey of Multi-Agent Reinforcement Learning, *IEEE Transactions on Systems, Man, and Cybernetics –Part C: Applications and Reviews* 38(2): 156-172.

Chapman, David 1987. Planning for conjunctive goals. *Artificial Intelligence*, 32 (3): 333-377.

Durfee, Edmund H. 1988. *Coordination for Distributed Problem Solvers*. Boston: MA: Kluwer Academic.

Etzioni, Oren and Weld, Daniel 2007. Intelligent Agents on the Internet: Fact, Fiction, and Forecast, *IEEE Expert: Intelligent Systems and Their Applications*, 10 (4): 44-49.

Ferguson, Ines A. 1992. TouringMachine: an Architecture for Dynamic, Rational, Mobile Agents. PhD thesis, Clare Hall, University of Cambridge, UK.

Fikes, Richard and Nilsson, Nils 1971. STRIPS: a new approach to the application of theorem proving to problem solving, *Artificial Intelligence*, 2:189-208.

Jennings, Nick and Wooldridge, Michael (eds.) 1998. *Agent Technology: Foundations, Applications, and Markets*. Berlin: Springer-Verlag.

Jennings, Nick, Faratin, Peyman, Lomuscio, Alessio, Parsons, Simon, Sierra, Carles and Wooldridge, Michael 2001. Automated negotiation: prospects, methods and challenges, *International Journal of Group Decision and Negotiation* 10(2): 199-215.

Jennings, Nick, Sycara, Katia and Wooldridge, Michael 1998. A roadmap of agent research and development, *Journal of Autonomous Agents and Multi-Agent Systems* 1(1): 7-38.

Kaelbling, Leslie Pack, Littman, Michael and Moore, Andrew 1996. Reinforcement Learning: A Survey, *Journal of Artificial Intelligence Research* 4: 237-285.

Kraus, Sarit 2001. *Strategic Negotiation in Multiagent Environments*, Cambridge, MA: The MIT Press.

Leake, David (ed.). 2005. Twenty-fifth anniversary issue, *AI Magazine* 26(4).

Luck, Michael, McBurney, Peter, Shehory, Onn, and Willmott, Steve (eds.) 2005. *Agent Technology: Computing as Interaction (A Roadmap for Agent Based Computing)*. AgentLink III.

Müller, Jörg 1997. A cooperation model for autonomous agents, in Jörg Müller, Michael Wooldridge and Nick Jennings (eds.), *Intelligent Agents III: Agent Theories, Architectures, and Languages* (pp. 245-260), LNCS 1193, Berlin: Springer.

Newell, Allen and Simon, Herbert A. 1976. Computer Science as Empirical Inquiry: Symbols and Search, *Communications of the ACM* **1**9 (3): 113–126.

Rao, Anand S., Georgeff, Michael P., and Sonenberg, Elizabeth A. 1992. Social Plans: a preliminary report, in Eric Werner and Yves Demazeau (eds.), *Decentralized AI 3: Proceedings of the 3rd European Workshop on Modelling Autonomous Agents in a Multi-Agent World* (pp. 57-76), Amsterdam: Elsevier.

Rahwan, Iyad, Ramchurn, Sarvapalid, Jennings, Nick, McBurney, Peter, Parsons, Simon and Sonenberg, Liz 2003. Argumentation-based negotiation, *The Knowledge Engineering Review* 18(4): 343-375.

Rosenschein, Jeffrey S. and Gilad Zlotkin 1994. *Rules of Encounter: Designing Conventions for Automated Negotiation among Computers*, Cambridge, MA: The MIT Press.

Stone, Peter and Veloso, Manuela 2000. Multiagent systems: a survey from a machine learning perspective, *Autonomous Robots* 8(3): 345-383.

Vohra, Rakesh, and Wellman, Michael (eds.) 2007. Foundations of multi-agent learning, *Artificial Intelligence* 171(7).

Weiss, Gerhard (ed.) 1999. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. Cambridge, MA: The MIT Press.

Weiss, Gerhard and Dillenbourg, Pierre 1999. What is 'multi' in multiagent learning?, in Pierre Dillenbourg (ed.), *Collaborative learning. Cognitive and computational approaches* (pp. 64–80), Oxford: Pergamon Press.

Further Reading

The best on-line references for further reading on agents are the AI Topics/Agents

web-page hosted by the Association for the Advancement of Artificial Intelligence (http://www.aaai.org/), the UMBC AgentWeb (http://agents.umbc.edu/), and AgentLink III, the European Co-ordination Action for Agent-Based Computing (http://www.agentlink.org/).

For those wishing to investigate agents and multi-agent systems further, the following two books are easy to read and full of useful references to specialized topics:

- Russell, Stuart and Norvig, Peter 2010. *Artificial Intelligence: A Modern Approach*. Upper Saddle River, NJ: Prentice Hall. The 3rd edition of first AI handbook that shamelessly introduced AI from an agent's perspective. See in particular the second chapter on Intelligent Agents.

- Wooldridge, Michael 2009. *An Introduction to Multiagent Systems*. Chichester, England: John Wiley & Sons. 2nd edition of an ideal introductory text on the agents and multi-agent systems, despite being somewhat limited in its coverage of learning.

Glossary terms

Agent: An agent is a software entity that acts autonomously, that is, that makes its own decisions on behalf of the designer –typically in dynamic environments from which it learns and to which it adapts. When applied to the development of new Internet technologies agents need also to show a social attitude.

Multi-agent system: A Multi-agent system is a collection of autonomous agents that need to get coordinated in order to achieve their individual goals. Coordination is achieved through negotiation or argumentation and, in most applications, requires that the agents learn to adapt to each other's strategies.

Short Bio

Dr. Eduardo Alonso is a Reader in Computing at the School of Informatics, City University London. His research has focused in the area of software agents, in particular in modeling complex multi-agent systems. In this respect, he has developed co-ordination mechanisms, negotiation and argumentation protocols specifically, as well as formal models of "right systems" –with applications to road traffic networks. He has published his research in journals such as the Knowledge Engineering Review and the Artificial Intelligence Review, and in various Springer LNAI and LNCS volumes. He has chaired several workshops in the field and acted as Student Scholarships Co-Chair for the *Fifth International Joint Conference on Autonomous Agents and Multiagent Systems* (AAMAS-06), held in Hakodate, Japan, May 2006. He has edited a special issue of the *International Journal of Autonomous Agents and Multi-Agent Systems* and was a member of the Management Committee of AgentLinkII (the European Network of Excellence for Agent-Based Computing, FP5 IST-1999-29003).