scheme can be accomplished in the network by connecting the output elements to each other via previewed negative, or inhibitory, connections. The output element with the most activation along its connection becomes the highest active one and in due course forces all other output elements to be inactive. A simple competitive algorithm given by Rich and Knight (1991) is as follows.

(1) Provide a sample input from the training.
(2) Compute the initial activation for each output element.
(3) Let the output elements compete with each other for the input vector until only one is active.
(4) Increase the weight on the connection between the active output element and active input elements. This makes it more likely that the output element will be active next time the pattern is repeated.

One problem with this algorithm is that one output element may learn to be active all the time. A solution to this problem is described in Rumelhart and Zipser (1986).

11.9. CONCLUSION

The ability to learn is a fundamental attribute of intelligent behaviour. Progress in the theory and practice of computer modelling of learning is not only relevant to understanding intelligence but, if successful, should have a profound impact on the use of computing in scientific, industrial and commercial applications. The field of machine learning is an interdisciplinary subject which includes computing, artificial intelligence, cognitive science, information science, psychology, philosophy and other related disciplines.

There is as yet no single unifying theory for machine learning; instead there exists a range of methods, techniques and theories that have formed much of the research into machine learning. In this chapter we have described a range of methods with particular emphasis on induction learning and neural networks. This is because these two approaches are well established and well understood and promise to make a significant contribution to the area.

Induction learning is based on firm mathematical foundations of classical logic and its extensions. The new area of induction logic has the potential of an exciting future. Further work here still needs to be done, and in particular on the possible integration of PAC-learning and inductive Logic Programming.

As neural network development move away from ad-hoc heuristics and brute-force computation approaches to a more rigorous foundation based on dynamic system theory, nonlinear mathematics, system science and statistical physics, and developing learning formalisms and neural learning algorithms that can maintain performance, then neural networks can become extremely useful in solving difficult problems in nonlinear adaptive control, object recognitions and behavioural conditioning.

Chapter 12
Multiagent Systems

12.1. INTRODUCTION

Many important computer applications such as planning, cooperating robotics, process control, manufacturing, distributed sensing, avionics, collaborative design, health care and diagnostis, require construction of computational processes, both hardware and software, that are part of a larger system embedded in a physical environment. In such a framework the computational process must be made aware of its environment, continually monitoring the state of its world, choosing an appropriate action and reacting to changed conditions of this world. These computational processes are largely distributed and autonomous: we shall refer to them as agents.

In investigating the practical and theoretical foundations of these agents, a number of critical issues need to be considered. These include: What representational formalisms are to be developed and used to characterise the agent and the world? How are activities and tasks are to be usefully distributed across cooperating agents? How do agents communicate and cooperate to solve a goal? How does an agent generate a plan and carry out actions to complete a task? What resources are available to an agent and how are they used efficiently? How is it decided that a goal has been reached? What is the relationship between an agent and the world? This chapter attempts to answer some of these questions.

Distributed artificial intelligence (DAI) is a branch of AI that is concerned with the cooperative solution of problems by a decentralised group of processes or agents. These agents are loosely coupled but are logically independent of each other; they are capable of sophisticated problem solving and are able to reason, plan and communicate. DAI branches into three areas; distributed problem solving (DPS), multiagent systems (MAS) and parallel artificial intelligence (PAI).

DPS is concerned with decomposing a problem among a number of cooperating and knowledge-sharing modules which are specifically designed for a particular problem. In contrast, MAS are characterised by a number of autonomous generally heterogeneous and potentially independent agents working together to solve a problem. These agents are able to adapt to their environment, reacting to it and making changes to it. Agents should be able to receive knowledge from other agents and the environment, interpret and understand this knowledge, reformulate it if necessary, and store it internally in its own knowledge base. In real-world domains, agents typically perform complex tasks requiring a degree of cooperation, communication and joint activities taking into account the world around them, the temporal deadlines and resource limitation. These intelligent agents will require a range of skills to respond to unexpected events and

resolve conflicts, and to be able to continue their joint effort to carry out assigned tasks. The third area (PAI) is concerned more with performance problems than with conceptual advances. The emphasis is to develop parallel computation languages and algorithms. However, a sharp distinction between these areas cannot be drawn as there are no clear, commonly accepted definitions of these concepts. In this chapter we focus on MAS.

12.2. WHY MAS?

There are many good reasons for studying MAS:

- Distribution is a useful approach to controlling complexity. Large systems can be decomposed into multiple cooperating agents such that control can be decentralised and rendered easier to deal with.
- Joint activities, interactions and cooperation are a natural approach for many large evolutionary systems. These systems are subject to continuous change and extension. MAS facilitate the design and implementation of such systems.
- MAS provide foundations for increased reliability and robustness. They normally have a certain amount of redundancy, that is some agents can solve the same task as others or pieces of knowledge are known by several agents. Hence the system becomes more robust against external influences or breakdown of some agents. Generally the prospect for graceful degradation (soft fail) increases and the system can guarantee higher reliability.
- MAS provide insights and understanding about information processing phenomena occurring in the real world. Research into computational methods that take the social interaction between the agents themselves and with their environment may shed light on how activities and actions are achieved in the face of enormous complexity.
- The MAS approach is potentially more efficient, taking advantage of parallelism.

12.3. BASIC ISSUES AND FOUNDATION OF MAS

The traditional AI approach to agents has largely focused on a single-agent concept. Limited considerations were given to the social structure of agents' society, the world, and the interaction between them. Further, when the world was considered, it was assumed to be stable, predictable and certain. In the real world, however, these assumptions simply do not hold, and the approach has resulted in systems that are inflexible, and unable to respond to changing environments. Joint activities and actions were severely restricted. Recently the area has made an important move towards a theory of *agency* with more realistic assumptions, taking into consideration that the world is dynamic, changing, uncertain and unpredictable and that the agent does not have complete knowledge of its world. Research aimed at this new approach includes proposals such as the BDI models (Rao and Georgeff, 1991), situated automata (Rosenschein and Kaelbling, 1986), agency and structure (Agre and Chapman, 1987), subsumption architectures (Brooks, 1986), reactive planning (Georgeff and Lansky, 1987; Fox and Smith, 1984), action networks (Nilsson, 1989), universal plans (Schoppers, 1990), social conception of knowledge and action (Gasser, 1991) and open information systems semantics (Hewitt, 1991).

12.4. THE SOCIAL NATURE OF MAS

In attempting to build firm theoretical and practical foundations for MAS, using the structure of human society as the basis for these foundations, it is important to remember that almost all human activities and behaviour are social in character. A human being can neither act nor survive in isolation. A human being is in continuous interaction with the world it inhabits. Its knowledge, beliefs, activities and actions are totally shaped by this world. Understanding the social structure of agenthood provides practical and theoretical guidelines for defining principles for an agent. In the rest of this chapter we will address these principles, develop an agent architecture, and provide a formal model for multiagent systems.

12.5. CHARACTERISATION OF MAS

The foundations for MAS, based on their social structure, the world they inhabit, and their interaction with this world has recently been studied by a number of researchers (Agre and Chapman, 1987; Bond, 1990; Gasser, 1991; Durfee and Lesser, 1987; Hewitt, 1991; Singh, 1990; Werner, 1990a). These researchers and others in the field have identified a number of lexicons such as commitment, cooperation, social plans, joint activities, conflict, negotiation, belief and actions. These primitives are used in forming foundational principles on which computational theories and practices of MAS are to be developed. These foundational principles are addressed in Section 12.6, but first we will now examine these lexicons.

12.5.1. The New Lexicon

The new lexicon gives rise to the following issues:

(a) What role do cooperation and joint activities play in MAS?
(b) Conflict is identified as a key aspect of MAS. Where and why does it arise? And what is to be done about it?
(c) How can the semantics of negotiations be formalised?
(d) What role does commitment play? How is joint commitment arrived at? How does commitment effect agent behaviour?
(e) What are the atomic activities and actions in MAS? How can they be synthesised into larger activities and actions? How do these actions affect the world?
(f) Communications: how do agents communicate with each other and the world, and at what level? What is the semantics of a message and how do we ensure that an agent understands the message?

However, before examining these lexicons in more details, it may help the reader to see how these notions appear and interact in a typical application of MAS. The application chosen is car manufacturing, representing a class of organisations which market, design and produce product of any type.

In this scheme of things, the design of product, the car, requires experts on design, manufacture, safety regulations, costs, potential markets, and so on. The type of car to be produced is first identified by a business plan, which would normally include marketing, sales and manufacture. The business plan is further expanded into a design plan, and the design plan in turn into a process plan and a production plan. Execution of the process plan and the production plan requires the purchase and installation of resources such as machines, materials, labour and energy. These activities proceed in parallel and interact strongly. Each task is normally decomposed into a number of subtasks which are then translated into a set of activities and actions carried out by a set of agents. These agents share a common world, but each may have a different objective and view of it. Because of these different objectives and views, agent's propose different actions of their own which may lead to conflict; for example, conflict may arise by one agent wanting to cut manufacturing cost when another agent's objective is to increase car reliability. These conflicts are negotiated and resolved among agents and are then jointly committed to a joint plan. The joint plan is a set commitments of actions and beliefs at different levels of abstraction and for different time intervals. Commitments have associated resources; execution of a commitment uses the resources associated with it. Joint actions often require stringent synchronisation; agents need to track the success or failure of their planned actions and inform other agents if something goes wrong, an unexpected event occurs, or the environment changes.

We are now ready to examine these lexicons in more details.

(a) *Cooperation*

We define cooperation as the process by which participant agents generate mutually dependent roles in joint activities. For example, in car manufacturing, business, design and production agents cooperate to produce a car. Another example is hospitals where patients, medical people and managers cooperate in the treatment of a patient. Agents receive problems at a certain level of abstraction. The agent then decomposes a problem into those that can be carried out at this location with resources and knowledge available locally (those that can be solved by cooperation with other agents at the same level) and those that involve sending goals, ideas and plans to agents at other levels of abstraction.

A successful cooperation strategy must ensure a steady convergence towards a solution. Werner (1990b,c) gives algorithms for cooperating agents.

(b) *Conflicts*

Conflict occurs when agents cooperate to solve a problem. It can occur as a result of incorrect or incomplete local knowledge, different goals, priorities, solution evaluation criteria, and resource contention. It should be viewed as a positive part of the problem-solving process. The resolution of conflict involves information

exchange among agents, and this communication should provide improved robustness, breadth and balance in the integrated solution.

The traditional approach to managing conflict is to avoid potential conflicts by thorough analysis and consistency checking of knowledge at development time. This approach, however, is difficult and costly; in addition, as agents' knowledge increases and becomes more diverse and complex, the approach will no longer be viable. Another model is the blackboard system, which provides a model of a number of experts working together. Cooperation among the experts occurs implicitly through the incremental extension of globally available hypotheses. Conflicts are not resolved explicitly, instead competing hypotheses coexist and compete for processing resources to improve their viability. This model has been shown to be too rigid and too sensitive to the problem-solving context, and may not actually resolve a conflict.

Human negotiation and creative problem-solving models (Dean and Wellman, 1991; Fisher and Ury, 1981; de Bono, 1971) offer insight into possible strategies for conflict resolution and for the use of conflict as a platform for creativity. However, they cannot be applied directly to computational models because they use a human motivation factor which cannot be either applied to or is relevant to machine agents. Sycara (1989) presents a negotiation model which applies case based and utility reasoning methods to conflicts. Klein (1990) has developed a hierarchy of conflict types and resolution strategies in which conflicts are classified and mapped to specific resolutions by a global controller. In these systems, conflict resolution is not sensitive to the problem-solving context. Lander, Lesser and Connell (1990) describe a system, Conflict Expert Framework (CEF), in which conflict resolution techniques are chosen based on the characteristics of the problem-solving state such as the flexibility of a particular agent and the amount of effort that has been expended on a particular solution to date. The approach is discussed further in Section 12.6.7.

(c) *Negotiation*

The subject of negotiation between agents has been of continuing interest in the MAS community [Durfee and Lesser, 1988; Thomas and Martial, 1990; Smith, 1978; Sycara, 1989; Conry, 1988; Zlotkin and Rosenschein, 1990). The need for negotiation is triggered by detection of a conflict between the actions of different agents' plans and goals. Conflict may prevent one or more of the plans being executed. The purpose of negotiation is to resolve the conflict such that a joint plan for conflicting agents can be executed. Depending on the particular domain and goals involved, there may be the possibility that agents will actually be able to help each other and thereby achieve goals with lower overall cost. Interaction between agents occurs in two consecutive stages: first the agents negotiate, then they execute the joint plan that has been agreed upon. No divergence from the negotiated plan is allowed. This sharp separation of stages has consequences, in that it rules out certain negotiation on tactics that might be used in an interleaved process. A more general negotiation framework, however, allows concurrent negotiation and execution.

We assume that negotiation is an iterative process; at each step an agent offers a proposal and at no point does an agent demand more than it did previously. In other

words, each offers either repeats the previous offer or makes a concession to the other agents. Negotiation can end up in one of the following ways:

- Agreement, which creates a new joint plant.
- Deadlock, when agents cannot reach an agreement. As a result of deadlock, another negotiation may start with a different issue or alternative goal.
- Appeal, when the participating agents appeal to another agent acting as a coordinator. The coordinator may modify goals, or offer comprises until a final agreement is reached.

When agents negotiate, it is desirable that they converge to a Pareto optimal solution, meaning that the only way the situation could improve for one agent would be to worsen the situation of the other agent. It is also highly desirable that an agent's negotiation strategy be in equilibrium: a strategy S is in equilibrium if, assuming your opponent is using S, the best you can do is also to use S. Thus, no other agent will be able to take advantage of that agent using a different negotiating strategy.

Zlotkin and Rosenschein (1990) present a scheme for negotiation and goal relaxation.

(d) *Commitments*

We define commitments as a set of constraints on actions and belief. Commitment is a basic concept in MAS and forms the foundation of many other concepts such as cooperation, negotiation, plans and goals. A number of researchers (Cohen and Levesque, 1984, 1990; Fikes, 1982; Winograd and Flores, 1986; Levesque et al., 1990) have studied commitment as an individual choice. For example, Levesque et al. (1990) developed a notion of commitment based on what they called joint persistent goals. Informally, a group of agents are committed to a common goal until one agent in the group

- reaches the goal
- believes the goal can never be achieved, or
- believes motivation to achieve the goal is no longer valid,

whereupon the agent informs all other agents in the group that it is no longer committed to the goal. Although this approach has the advantage of specifying how agents should behave in a joint activity and commitment to a common goal, its disadvantage is that it does not specify how the goal state is reached. Further, this definition of commitment is not a social commitment; for example, agents may agree on a common goal but not agree upon a common solution. In contrast to Cohen and Levesque, we believe that commitment is a social concept. An agent engages in several but separate activities such as work, social life, travel, eating and sleeping over a long period (a lifetime). In carrying out other activities, the agent is in continuous interaction with other agents and objects in the world it inhabits. Commitment, in this sense, is the outcome of joint activities. In this framework, goals reflect commitment that must be accommodated regularly and steadily over an interval of time and in the face of many constraints. We concur with Gasser's (1991) definition of commitment: "In this setting the notion of commitment is distributed or social".

Commitment from the social prospective is grounded in the actions of the agent's many activities taken together. It is not a matter of individual choice; it is agents' actions in relation to those of others (and vice versa) that maintain the agent's participation in a course of action. Moreover, this social notion of commitment is a basic concept, it does not rely on more primitive mental states such as belief, goal or choice. This notion of commitment cannot be located within the individual, it is a product of the interaction of agents and the world. Because of this, it extends in varying degrees to objects as well as intelligent agents as active participants in situations. For example, an agent wishing to travel by plane from one place to another is engaged in a setting where the agent, the pilot, the plane and airport authorities and facilities must together enter into a course of action that involves the agent having a boarding pass and the plane, the pilot and air traffic control all being in place. In other words, all these entities are committed to acting in that way for the agent to travel from one place to another.

To compute commitment, Cohen and Levesque (1989) represent commitment using the primitives "belief" and "goal"; then based on logical entailment of its belief (i.e. persistence), an agent deduces whether it is still committed to a goal. This approach, however, envisages commitment as private and local to the agent. In contrast, commitment from the social perspective is based on system commitment; that is, a group of agents and the world are committed together in that way. From this perspective, computation of commitment will require that agents have models of themselves and of each other and the world. This approach is a foundation of the MACE system (Bond, 1990; Gasser et al., 1987, 1989), it has been exploited in DATMS (Mason and Johnson, 1989), PGP (Durfee and Lesser, 1987), and has foundation in what Mead (1934) saw as a concept that could unify the individual and society. This modelling can vary from a simple index, to a message received, to a rich and complex model of another agent.

(e) *Agents' interactions*

Problems that need solutions are decomposed into subproblems for agents to solve. To solve these problems and subproblems, agents generate plans, either statically or dynamically. A plan for a goal is a set of commitments for an agent to discharge. The plan then guides the activities and actions that must be performed. Actions can themselves be decomposed into further subactions. Issues that need to be considered are actions' precedence, identical actions and simultaneous actions, and how actions are to be combined to produce a higher level action. Most agents need to engage in several separate but interacting activities over a period of time. Interaction among activities include

- subgoal interactions,
- competition for resources,
- sharing of resources, and
- joint action.

A central problem in this area is to understand how an agent can organise and manage these activities in an orderly manner to discharge its commitments. In short, how does an agent organise and get on with its daily life in a social setting?

(f) *Communication*

Communication may be viewed as the maintenance of a collection of commitments among agents. Agents make commitments using locally available knowledge as well as information communicated by other agents. The goal is to provide a means by which agents of different capabilities can communicate. Communication must, therefore, be at several levels. To be of use to each other, agents must be able to participate in a dialogue. Their role in this dialogue may be active, passive, or both. In this way agents can function as master, slave or peer, respectively.

There is a link between representation and communication in that any knowledge intended to be used non-locally must be converted into stable transportable FORM (represented) and (re)interpreted in the local context where it is delivered. Werner (1988) provides a theory of communication between agents by relating communication to the intentions of agents. He developed a high level language that allows communication between agents through directives that include commands, demands and requests. Another model of communication, proposed by Gasper (1990) presents a model relating communication to changes of belief of an agent.

12.6. CHARACTERISTICS OF AGENCY

The concept of agency is defined as a society of agents embedded in a dynamic world. An autonomous agent is a system of sensors, computational processes, and actions structured in such a way that the sensors monitor the world, continuously measuring the state of affairs. The computational processes interpret and reason with information gathered and direct the next activities and actions to be carried out by the agent, actions that affect the world. Changes in the world close the loop for agent's sensors, introducing further sensing, computation, and action by the agent. An agent cannot and does not operate in isolation; an agent is embedded in the world it inhabits. This world contains other agents, objects, concepts, laws, space and time relationships, etc. Activity of the agent is to be understood and is organised in large measure through the social structure of this world.

In this section we will sketch some of the issues that a computational theory of agency must face. These ideas are neither comprehensive or complete, they are rather an attempt to bring together some of the latest research and development in this important area. They include the following:

(1) Relation and interaction between the agency and the world.
(2) Formal representation and reasoning methods from a social point of view.
(3) Integration and adaptation of agency.
(4) The relation between local knowledge and action and global or common knowledge and action.
(5) How an agent reasons about knowledge and action of other agents.
(6) How an agent operates in the face of resource limitation and constraints.
(7) Coherence and conflict resolution.
(8) Decomposition, task allocation, and synthesis of solution.

(9) How sensors (vision, sound, etc.) affect agent activities.
(10) The joint qualification problem.
(11) How an abstract notion of action can be translated into action in the world.
(12) Planning.
(13) Engineering methods and practices in constructing MAS.

We now give a more detailed description of these issues.

12.6.1. Agency and the World: The Individual in the Mass

An important characterisation of MAS is that agents are autonomous: they have their own goals, capabilities and knowledge. Agents with their own responsibility are aware of their own actions, the world they inhabit, and the effect their actions have on their world. Relationships and interactions between an agent and the world are an on-going process; both the agent and the world affect and shape each other.

Recent work on agenthood has begun to address these issues. Agre and Horswill (Agre, 1991) sketched out a computational theory of agency based on ideas of cultural support for improvisation. In particular, two aspects were investigated: do cultural artefacts (e.g. kitchen tools) have special properties that make them easy to generate plans for them; that is do they fit our minds specially well. Drawing on recent work in anthropology, they investigated formal properties of the tools and materials found in kitchens and garages, together with some of the invariants of the activities in these places (cooking an omelette). Another aspect of the investigation was what we learn from other people that is easy enough to learn in one or two demonstrations (apprenticeship) and yet complex enough to constrain our planning. In addition to these cultural ideas, they drew on the more general idea of how the environment external to the planner might support the planning process. Initial results suggest that these aspects are all well-adapted to simple, computationally straightforward policies for improvisation that probably achieve their goals without any specific knowledge of the locations or states of objects.

Gasser (1991) looked at social conception of knowledge and action. His work introduced a number of principles that underlie the social aspect of multiagent systems. Hewitt (1991) was concerned with open information system (OIS) semantics. OIS semantics provide a characterisation of deductive inference built on concepts such as self-reliance and interdependence, trials of strength, commitments, negotiation and conflict. OIS is inherently more "social", "grounded in large-scale information systems" rather than individual agents, and provides a different account of representational processes. Rosenschein and Kaelbling (1986) introduced situated automata built on concrete models of epistemic logic. In the situated automata framework, the concept of knowledge is analysed in terms of logical relationships between the state of a process (an agent) and its environment; not every state of the process–environment pair is possible, in general. This world was based on axioms of modal system S5, including consequential closure and positive and negative introspection. Other researchers in the area include reactive planning (Georgeff and Lansky, 1987), Nilsson active networks (Nilsson, 1989), and Schoppers universal plans (Schoppers, 1990).

12.6.2. Knowledge Representation and Reasoning in Multiagent Systems

MAS semantics uses concepts such as belief, commitment, conflict and negotiation. To make these semantics computational and to prove their properties, they need to be formalised and represented in a mathematical language. This language is principal based on mathematical logic. This mathematical logic should, in addition to classical logic, include modal logic, concept logic, epistemic and autoepistemic logic, non-monotonic logic, meta-theories and constraint theories. Section 12.8 gives a glimpse of a formal model for MAS.

Reasoning and deductive theories within this paradigm of logic, bearing in mind the social structure of MAS, should take into account that agents are constrained in their behaviour by other agents, by the world and by resources available to them. For example, an agent's commitment is constrainted by commitments of other agents and resources available. This social commitment limits the agent's field of choice. A promising approach to computation in MAS is based on agents modelling one another by exchanging self-description. This approach is the foundation to the MACE system (Gasser, 1992), DATMS (Mason and Johnson, 1989) and PGP (Durfee and Lesser, 1989). An agent will also need to model the world it inhabits, which includes objects, time and space, relationships, and interactions. An alternative reasoning for MAS which is not based on mathematical logic is to reason probabilistically and to use game theory (Russell and Lehner 1990; Jones, 1980; Laskey, 1989; Pearl, 1990; de Kleer, 1986).

12.6.3. Integration and Adaptation

To survive in its environment, an autonomous agent needs to find its role and purpose in this environment (society); it needs what is called an integrated way of life (Agre, 1991). In meeting its responsibilities and goals, an agent needs to be engaged in a number of activities, some of which may be concurrent. The behaviour of an agent requires capabilities for perception, cognition and action. Furthermore, in a complex dynamic environment an agent's ability to respond to changes will always be limited by the computational resources available to it. This is because the computational requirement of tasks to be performed often exceeds the computational resources available to the agent. While more efficient hardware and software may solve this problem in a specific application, they will not solve the general problem of limited resources. Because of these constraints, an agent needs to determine which operations to perform next, which one to postpone and which one to disregard. Moreover, we are interested in flexible agents that can accomplish a range of tasks rather than in a rigid agent that has been optimised to perform a specific task and is unable to respond to changes in its environment.

In effect an agent is continuously making decisions, within its role in the society, requiring sophisticated control structure to guide its activities. It is this control structure and adaptability that allows the agent to integrate and survive in its environment over a long period.

12.6.4. Local and Global Knowledge

How does knowledge become global? How do rules, regulations and laws emerge and how are they then used by agents across the same or different space and time? An agent's knowledge and actions are concepts that are local and situated (Gasser et al., 1989; Gasser and Hill, 1990). But for knowledge to be useful, i.e. to be used by other agents in different settings, different locations and different times, the knowledge must be made general. It is this generality that allows the propagation and distribution of knowledge; and yet for general knowledge to be useful, it has to be applied in a local setting, and made local again by instantiating the general variables which made distribution of knowledge possible. This scenario of local and global knowledge would appear to lead to inconsistency. The problem is this: If the meaning of knowledge is always local and situated, how can global knowledge become coherent and consistent? One approach to this problem has been proposed by Rosenschein and Kaelbling (1986); called local utility, it is based on agents building models of other agents. The problem with this approach is how one does ensures the stability of these models: that is, if the meaning of knowledge is local and situated, how can we be satisfied that uprooting this knowledge and transporting it to another situation has not somehow changed the meaning of this knowledge. Latour (1987, 1988) views knowledge as an artefact, a utility, a tool that can be transported, copied and interpreted in a local situation. Taking this viewpoint, for knowledge to be non-local, hence transportable, it must be made into a stable (preserving its useful qualities in new contexts) mobile (transportable across contexts) form and then made local again by reinterpreting it in a local situation where it is to be used. Information represented digitally meets these requirements: that is, it is stable, mobile and combinable.

12.6.5. Reasoning about Knowledge and Actions of Other Agents

In coordinating their actions, agents have to take into account, and reason about, knowledge and actions of other agents. In this way an agent is constrained by other agents and by its own environment. It is this web of constraints that shapes an agent's commitments into a social commitment. A possible approach to this problem is to pass a model of one agent to another. The model would include the agent's beliefs, plans and resources. Models of agents are employed to predict their behaviour, using search methods and/or game theory. Models of agents are also useful for evaluating the credibility, usefulness and reliability of similar data. In this framework, it is important for an agent to avoid compromising its set of beliefs and assumptions by integrating faulty or unreliable messages from other agents. Thus each agent must maintain its local autonomy and arms-length relationships while incorporating useful information generated by others. Mason and Johnson's (1989) approach is to let each agent use non-local knowledge for the local focus of attention. Another approach is the partial global planning (Durfee and Lesser, 1987, 1989; Durfree et al., 1987) that was developed as a distributed control technique to ensure coherent network problem-solving behaviour. It is a flexible approach to

coordination that does not assume any particular distribution of subproblems, expertise or other resources, but instead lets nodes coordinate in response to the current situation. Each node can represent and reason about the actions and interactions of groups of nodes and how they affect local activities. These representations are called partial global plans (PGPs) because they specify how different parts of the network plan to achieve more global goals. Each node can maintain its own set of PGPs that it may use independently and asynchronously to coordinate its activities.

A PGP contains a objective, a plan-activity map, a solution-construction graph and a status:

- The *objective* contains information about *why* the PGP exists, including its eventual goal (the larger solution being formed) and its importance (a priority rating or reasons for pursuing it).
- The *plan-activity map* represents *what* the nodes are doing, including the major plan steps the nodes are concurrently taking, their costs, and expected results.
- The *solution-construction graph* contains information about *how* the nodes should interact, including specifications about what partial results to exchange and when to exchange them.
- The *status* contains book-keeping information for the PGP, including pointers to relevant information received from other nodes and when that information was received.

A PGP is a general structure for representing coordinated activity in terms of goals, actions, interactions and relationships.

When in operation, a node's PGPlanner scans its current network model (a node's representation of the goals, actions and plans of other nodes in the system) to identify when several nodes are working on goals that are pieces of some larger network goal (partial global goal). By combining information from its own plan and those of other nodes, a PGPlanner builds practical global plans (PGPs) to achieve the partial global goals. A PGPlanner forms a plan-activity map from the separate plans by interleaving the plans' major steps using the predictions about when those steps will take place. Thus, the plan-activity map represents concurrent node activities. To improve coordination, a PGPlanner reorders the activities in the plan-activity map using expectations or predictions about their costs, results and utilities. Rather than examining all possible orderings, a PGPlanner uses a hill-climbing procedure to cheaply find a better (though not always optimal) ordering. From the reordered plan-activity map, a PGPlanner modifies the local plans to pursue their major plan steps in a more coordinated fashion. A PGPlanner also builds a solution-construction graph that represents the interactions between nodes. By examining the plan-activity map, a PGPlanner identifies when and where partial results should be exchanged in order for the nodes to be integrated into a complete solution, and this information is represented in the solution-construction graph.

The PGPlanner, as was used for coordination in a distributed vehicle monitoring task, relied on the fact that the level of abstraction at which the node plans were communicated was a sequence of intermediate goals (times and locations in which to extend a vehicle track). Each intermediate goal was an abstraction of the processing and integration work that each node planned locally. These intermediate goals were ordered by the local node planners on the basis of several criteria (Durfee and Lesser, 1989), but these relationships were not transmitted in the node plans. There was no representation of temporal relationships between intermediate goals. The PGPlanner reorders node activities by hill-climbing in the space of costs of the present ordering of activities. The cost of an ordering is computed from relationships such as redundancy reliability, predictiveness and independence of the activities.

12.6.6. Resource Limitations and Constraints

A critical consideration in the design of MAS is that agents are resource bound. All resources are finite and agents must reason and act within the resources available to them. In addition to resource limitation, the MAS environment is characterised by the presence of temporal constraints that give rise to tightly interacting subproblems.

Managing and optimising resources in a multiagent environment is characterised by the following:

- Allocation of limited resources among agents may give rise to conflicts.
- Conflicts over resources and criteria for optimality may result in a state where optimal solution is not possible.
- There is no single agent which has a global system view.
- Agents do not have complete knowledge of their environment or other agents' constraints.
- Because of the dynamic nature of world structure, an agent's problem-solving context is continuously changing. In addition an agent's decision can produce constraint violations for other agents which may lead to backtracking (Sycara et al., 1990). Backtracking can have major ripple effects on the multiagent system because it may invalidate resource allocations that have been made.

Allocating resources to agents' activities is essentially a scheduling task that synchronises activities to avoid and resolve conflicts. The schedule is built in a cooperative fashion through local computation and communication.

One approach to resource allocation was proposed by Bond (1990). In this approach, commitments are used to allocate resources and optimise the overall cost of projects. Agents have a joint plan which is a set of commitments to actions and beliefs. Each commitment in the plan has an associated set of resources; these resources are consumed when commitments are discharged. The joint plan is used to compute a total estimated cost of the problem, and it is this total cost that the set of agents jointly will try to optimise, irrespective of the cost to any individual agent. Based on this optimised plan, each agent then updates its view of the joint plan and continues to search for further commitment that will move the goal closer to solution. The agent is now constrained in its activity by the updated joint plan. Commitment in this sense has future implications. An agent's use of resource may constrain its, and others', choices in the future.

Another approach to resource allocation in MAS has been proposed by Sycara, Roth, Sadeh and Fox (1990). Their approach is conducted in the domain of job-shop scheduling using constrained heuristic search (CHS), and relies on a set of textures of the problem space being searched. Textures provide a probabilistic, graph-theoretic definition of the complexity and importance of decisions in the local problem space of each agent. In addition, textures provide a good predictive measure

of the impact of local decisions on system goals. As a result, textures can be used to make control decisions that reduce the search space required.

12.6.6.1. *Constrained Heuristic Search (CHS)*

CHS provides a methodology for solving constraint satisfaction problems (CSPs) and constrained optimization problems (COPs).A CSP is defined by a set of variables, each with a predefined domain of possible values, and a set of constraints restricting the values that can simultaneously be assigned to these variables (Montanari, 1971; Mackworth, 1985; Decker 1987; Decker et al., 1989). A solution to a CSP is a complete set of assignments that satisfies all the problem constraints. COPs are CSPs with an objective function to be optimized. The general CSP is a well-known NP-complete problem (Garvey, 1979). There are, however, classes of CSPs and COPs that do not belong to NP and for which efficient algorithms exist. The CHS methodology is meant for those CSPs/COPs for which there is no efficient algorithm. A general paradigm for solving these problems consists in using Backtrack Search (BT) (Golomb, 1965; Bitner and Reingold, 1975). BT is an enumerative technique that incrementally builds a solution by instantiating one variable after another. Each time a new variable is instantiated, a new search state is created that corresponds to a more complete partial solution. If, in the process of building a solution, BT generates a partial solution that it cannot complete (because of constraint incompatibility), it has to undo one or several earlier decisions. Partial solutions that cannot be completed are often referred to as dead-end states (in the search space). Because the general CSP is NP-complete, BT may require exponential time in the *worst-case*. CHS provides a methodology to reduce the average complexity of BT by interleaving search with *local constraints propagation* and the computation of *texture-based heuristics*. More specifically, we have the following.

(1) *Local propagation techniques*. Local constraint propagation techniques are used to prune the search space of alternatives that have become impossible owing to earlier decisions made to reach the current search state. By propagating the effects of earlier commitments as soon as possible, CHS reduces the chances of making deductions that are incompatible with these earlier commitments (Mackworth, 1985).

(2) *Texture-based heuristics*. Typically, pruning the search space can only be done efficiently on a local basis (Nadel, 1988). Hence, local constraint propagation techniques are not sufficient to guarantee backtrack-free search. In order to avoid backtracking as much as possible, as well as reduce its impact when it cannot be avoided, CHS analyses the pruned problem space in order to determine critical variables, promising values for these variables, promising search states to backtrack to, etc. The results of this analysis are summarised in a set of textures that characterise different types of constraint interactions in the search space. These textures are operationalised by a set of heuristics to decide which variable to instantiate next (so-called variable ordering heuristics), which value to assign to a variable (so-called value ordering heuristics), which assignment to undo in order to recover from a dead end, and so on.

For example, in the factory-scheduling domain, variables are activities whose values are reservations consisting of a start time and a set of resources (e.g. a human operator, a milling machine and a set of fixtures). Local constraint propagation techniques are used to identify reservations that have become unavailable for an unscheduled activity owing to the scheduling of another activity (e.g. a resource that has been allocated to an activity over some time interval, or a start time that has become infeasible owing to the scheduling of an earlier activity in a process plan). Within this context, texture-based heuristics are concerned with such decisions as which activity to schedule next, which reservation to assign to an activity, and which reservation assignments to undo if the current partial schedule cannot be completed.

12.6.7. Coherence and Conflict Resolution

The traditional approaches to managing conflict is to avoid potential conflicts by thorough analysis and consistency checking of knowledge at development time. This approach, however, is difficult and costly, and in addition as agents' knowledge increases and become more diverse and complex, the approach will no longer be viable. Blackboard systems provide a model of a number of experts working together. Cooperation among the experts occurs implicitly through the incremental extension of globally available hypotheses. Conflicts are not resolved explicitly; instead competing hypothesis coexist and compete for processing resources to improve their viability.

Human negotiation and creative problem-solving models (Dean, 1981; Fisher and Ury, 1981; de Bono, 1971) offer insight into possible strategies for conflict resolution and for the use of conflict as a platform for creativity. However, they cannot be applied directly to computational models because they use a human motivation factor which cannot either be applied or be relevant to machine agents. Sycara (1989) present a negotiation model which applies case-based and utility-reasoning methods to conflicts. Klein (1990) has developed a hierarchy of conflict types and resolution strategies in which conflicts are classified and mapped to specific resolutions by a global controller. In these systems, conflict resolution is not sensitive to problem-solving contexts. Lander, Lesser and Connell (1990) describe a system, Conflict Expert Framework (CEF); in this work they identify a number of conflict resolution strategies listed below.

- *General random alternatives*. In some types of problem solving, multiple solutions exist and can be generated with little extra computational overhead. For example, in a typical blackboard system, multiple solutions may exist at any given time. A highly rated one is chosen as "the answer" but there may be others that are rated equally or just slightly lower.
- *Compromise*. Find an intermediate proposal that is within the acceptable range of all agents using variable value relaxations. This strategy is the typical compromise that is used in buy/sell or other numeric transactions. Numerical optimisations or techniques based on the type of dimension can result in quick and fair results.
- *Generate constrained alternatives*. Generate new alternatives based on constraints that are received from an inflexible agent or based on some other agent's partial solution.
- *Generate goal alternatives*. The original proposals are abandoned. Alternative proposals are generated by looking for alternative goal expansions. If necessary,

some goals can be relaxed or relinquished. This can lead to substantially different proposals being generated at the level in which the conflict occurred. This strategy is useful for changing the focus of the system from a new plateau to a new area of the search space.

- *Case-based parameter set retrieval.* Find a previous solution that succeeded in resolving a conflict involving a similar set of parameters. Make the set of changes rather than isolated modifications. This approach minimises oscillation that occurs in the overall proposal rating when dependencies among parameter values are not well understood.
- Revise and merge goals. Prioritise goals to relinquish unimportant subgoals. Build a mutually defined goal structure that incorporates the most important goals of all agents involved in the conflict. Generate a solution guided by the new structure. This approach is computationally expensive and will only be used in situations where no other technique seems promising, the system cannot produce a feasible design, or where an innovative proposal is explicitly requested by the user. It is hoped that the mutual goal structure will cause a jump into a new area of the search space which would not be explored during the normal course of problem solving.

Conflict resolution protocols are realised as formal dialogues with specific actions that can be taken at each processing step. All agents know the protocols and can formulate the messages required for their role in a particular conflict situation. For example, an agent that is beginning a new *respond to conflict task* first analyses the conflict from its own point of view and suggests a particular resolution method and possibly a set of resolution values. It sends this message to all other agents involved in the conflict. It then waits for confirmation from those agents. The other agents must respond to the message; they accept the resolution method, accept the suggested resolution, or propose a different solution or method. The method is sometimes changed because another agent has a local view which makes the suggested method inappropriate. For example, the originating agent may suggest using *compromise*, but the receiving agent may be too inflexible. The receiving agent might then suggest *generate constrained alternatives* and send its inflexible constraints to the originating agent. Sometimes the method is acceptable but a different solution is suggested. The originating agent cannot consider the conflict resolved until all participating agents have confirmed that a suggested solution is acceptable to them.

12.6.8. Decomposition, Task Allocation and Synthesis of Solution

Many methods have been suggested for task decomposition and task allocation, including functional, abstraction level and information hiding, data, or control dependencies and interaction density.

Partial global planning (PGP) (Durfee and Lesser, 1989) was developed as a distributed control technique to ensure coherent network solving behaviour. It was designed to have a flexible approach to coordination that does not assume any particular distribution of subproblems. Nodes coordinate in response to current situation, each node can represent and reason about the actions and interactions of

groups of nodes and how they effect local activities. These representations specify how different parts of the network plan to achieve more global goals. A partial global plan (PGP) is a general structure for representing coordinated activities in terms of goals, actions, interactions and relationships. The PGP maintains a solution-activity map which represents what the nodes are doing, including the major plan steps the nodes are concurrently taking, their costs and expected results. A solution-construction plan is used to maintain information about how nodes should interact, including specification of what partial results to exchange and when to exchange them. By combining information from its own plans and those of other nodes, a PGPlanner builds a PGP to achieve partial global goals. By examining the plan-activity-map, a PGPlanner identifies when and where goal partial results should be exchanged in order for the nodes to integrate them into a complete solution.

12.6.9. Sensors and Agent Activities

Sensors (vision, sound, touch, etc.) provides an agent with information about its world. These senors, affect agents' actions, which in turn influence the sensor. An example is moving a camera to examine different scenes. Treisman (1985) has suggested that visual systems provide the agent with a set of operators that can participate in visual routines. Agre and Chapman (1987) have shown how visual routing might fit into an architecture for improvisation, and Whitehead and Ballard (1990) have shown that visual routines can arise from unsurprised learning. Ballard (1989) and Horswill and Brooks (1988) have proposed that the visual system as participating in time-extended dynamics is integrated into the larger structure of the activity.

12.6.10. The Frame Problem

Agents operate in a changing world; changes occur as a result of an action or of an event. In a theory of change, one should specify what changes as a result of an action and what remains unchanged. The problem of characterising the aspects of a state that has not been changed by an action is called the frame problem, the frame problem is a special case of default reasoning. The frame problem can and does lead to prohibitively large numbers of consistency checks, rendering the system unusable in practice. The reason for this may be demonstrated by the frame axiom

$$holds\ (A, s) \wedge \neg ab(a, s) \Rightarrow holds\ (A, result(a, s)).$$

Informally, this axiom states that if a formula A holds in state s, and action a is not abnormal in that it changes A when executed in state s, then A will continue to hold after the action is completed. Ignoring the technical construction of the frame axiom, the axiom can cause immense computational difficulties. Consider propagating a number of formulae through a long sequence of actions. The application of the axiom for each action and to each formula will result in a very large computation. There have been various attempts at devising logics for default and non-monotonic reasoning; Chapter 7 gives details. Recently Ginsberg (1990) provided a scheme which reduces the computational demand of the frame problem. The scheme views

the truth values assigned to formulae as functions for temporal elements as a set of basic values and views temporal operators as functions on these functional truth values. This is done in two separate ways. First the actual default behaviour assigned to some formula A is encoded in a truth value such as that shown in Figure 12.1, which explicitly records the default truth of the formula at intervals following times when it is known to be true with certainty.

Consider the following sentence:

> If an agent places a block on a table, then the block is on the table at that time, and can be assumed by default to be on the table at subsequent times.

Note that although the truth value to be assigned to the conclusion of the rule (that the block is on the table) is not the same as the truth value of the antecedent (that the block is being placed on the table), this holds only instantaneously, whereas the conclusion holds over a wide range of times. The future behaviour of any particular formula (in this case, the location of the block) is determined not by applying a frame axiom such as that in Figure 12.1, but instead by an axiom describing this future behaviour when the formula is first asserted. This situation is handled by introducing a modal operator m and writing

$$m(a) \Rightarrow c,$$

where a is the antecedent and c is the consequent. The modal operator m changes the truth value of a so that the truth value of c is modified correctly by the above rule. The idea is that modal operators can be viewed truth-functionally, that is as functions on the truth values of the formulae on which they operate. In Ginsberg, (1990) it was pointed out that if truth values are taken from an arbitrary bilattice instead from the set $\{T, F\}$, it becomes practical to view modal operators truth-functionally. This construction is a generality of Kripke's structures. It extends the notion of a modal operator to include temporal operators such as *propagate* and *delay*. These modal operators may be used as semantic markers for points at which the inference process can be suspended and an approximate answer computed. The event-driven nature of the approach allows us to reason in a computationally viable way about formulae that change value only infrequently.

12.6.11. Abstraction Translated into Actions in the Real World

For an agent to usefully operate in the actual world, it must be able to interact with and manipulate physical objects that exist in this world. To do so the agent must somehow associate an abstract symbol in its memory to a particular object in the real world. For example, in the block world, to determine whether the relation *on* (a, b)

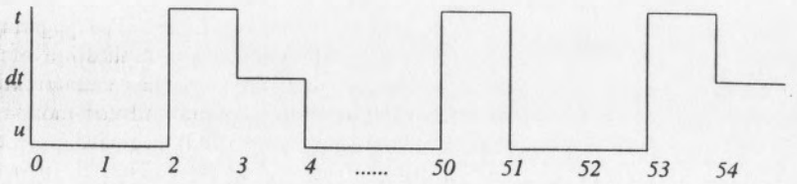

FIGURE 12.1 A default frame axiom.

is true or false, the agent must know what the symbols a and b refer to in the real world, and will need to have access to the physical world in looking for a block with label a on top of a block with label b. In addition it needs to be able to dynamically create, delete, compare, reason with, and manipulate references to these objects. The problem will increase in complexity if the real-world object is allowed to have identical objects; for example, if the world has more than one block with identical size, shape, colour and label. The traditional approach to solving this problem has been to create unique names that identify objects in the real world. However, if these objects are not displaying their unique name in any way, the traditional approach simply will not work. The problem of associating an abstract symbol in agent memory with a physical object is not addressed by logic-based representations.

Agre and Chapman (1987), and Agre (1990) proposed deictic representation to address the problem of how an embedded agent could manipulate objects. They showed how to do this by allowing an agent to establish indexical/functional reference to external objects, and introduced interactionist deictic representation over mentalistic logic-based representation. They provided the agent with a set of capabilities called marker control operators. These capabilities fell into five groups: marker comparison, marker inspection, marker assignment, indexing, and object comparison. Marker comparison and inspection operators provide such abilities as thresholding and distance between two tracked objects, testing whether two objects are approaching each other, and testing whether two markers refer to the same object. The indexing and assignment operators caused markers to pick up objects of a specified type, or to pick up objects having a specified position relative to another object already being tracked. The object comparison operators check whether objects are adjacent, whether objects are separated by empty space, and whether they are in a given direction from each other.

Schoppers (1990) described how to modify a universal plan execution engine to provide indexical/functional reference capabilities, thereby allowing universal plans to interact with physical objects. In their work Schoppers and Shu set up a modified block world in which blocks have names, labels, colours and shapes. Each block's name is unique and serves to identify the block. Block labels, e.g. a, are written on the blocks and there may be any number of blocks with the same letter. Hence an agent plan can name a specific block as usual, or can describe a desired block as one having a given colour – there may be many blocks with the same colour. Again blocks may be spherical, pyramidal or box-shaped. If a plan referred to a desired colour or shape, the plan would be describing, not naming, its objects. Similarly, if a plan referred to the labels printed on the blocks, the plan would be describing, not identifying, blocks. Only the block names serve as designators in the logical objective sense. The agent was given a descriptor-record that contained spatial location information of objects. This descriptor was dynamically updated by the agent's perceptual interface using sensors such as a camera and contact and position sensors to reflect the current real-world situation. The agent was allowed access to the world, to store beliefs about the positions of blocks, and to make use of existing beliefs to re-locate blocks. This use of beliefs solved two problems – a performance problem and a competence problem – reducing backtracking to a minimum. Object descriptions were posed as a goal; for example the goal

$$colour\ (X, red) \wedge shape\ (X, sphere) \wedge colour\ (Y, \text{blue}) \wedge shape(Y, box) \wedge on(X, Y)$$

is to find suitable objects for X and Y to refer to. The constraints on objects were indefinite descriptions, or, in logical terms, existentially quantified formula. This

means that, for the planner, object variables could have logico-objective semantics, while for the plan executor they have indexical/functional semantics.

Other research in this area has included Rosenschein and Kaelbling's (1986) description of situation semantics which maintains the model-theoretic framework of traditional logic, as well as proposing axiomatisation of the causal networks in which the agent is embedded in terms of information carried out by various physical world states.

12.6.12. Planning

Planning is a key component in the design and construction of an intelligent autonomous agent. Given a problem to solve, the agent (except in the most simple cases) will be interested in how to arrive at a solution, that is how to develop a plan. A plan describes a way to arrive at a goal state from a given initial state.

The planning problem may be expressed as follows. Given a set of goals $G = \{g_1, \ldots, g_n\} n \geq 1$, an initial state of the world s_0, and a set of operators $\{a_i\}$, find a sequence of the a_i that will cause all the g_i to be true if executed with the initial state at s_0. The operations can be external (moving objects) or internal (manipulating symbols), and either primitive (move an arm up or down) or complex (stack one block on another). The agent proceeds by iterating through the following steps:

(1) *Generation*: Generate a set of candidate operations.
(2) *Selection*: Select an operation from the set of candidate.
(3) *Relevance*: Determine whether the selected operation will achieve the required goal/subgoal.
(4) *Execution*: Execute the selected operation.

Different planning systems depend on how the four steps are to be performed. In general, however, planning systems may be grouped into four broad categories: classical planning, reactive planning, combined systems, and case planning.

12.6.12.1. *Classical Planning*

In this approach, sometimes referred to as the static approach, there is a strict separation between plan generation and plan execution. Before execution, the plan is created explicitly and handed out to the executor. Planning algorithms will try to prove (in principle at least) that the resulting state either probably satisfies or probably fails to satisfy all goals. Traditional development of the theory of planning emphasised exhaustive pre-planning with view to being able to ensure that an optimal or near-optimal plan would be found if one existed; thus the problem faced by traditional planners is one of search. The generated plan consists of a set of operations or actions, which can be specified at various levels of detail; for example, part of the plan for STACK (a,b) is shown in Figure 12.2 below. Included in the plan are constraints or conditions which should not be satisfied for an action to be executed. For example, constraints may determine the resource allocated to the action or its spatial location may identify shared objects, or may establish temporal relationships between objects.

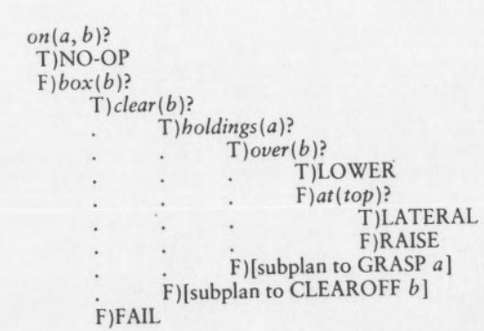

FIGURE 12.2 Decision tree equivalent of the STACK(a,b) plan.

As stated, in the classical development of a theory of planning, emphasis was put on exhaustive pre-planning, with the view to being able to ensure that an optimal or near optimal plan would be found if one existed. Planning in this paradigm required certain assumptions to hold:

- The world will be stable; it will behave as projected.
- The time allocated to the planning stage is independent of the time that can be allocated to execution.
- The information available to the planner is complete.
- Any initially correct plan will remain correct and can be carried out.

In the real world, however, these assumptions simply do not hold. As the assumptions are relaxed, new issues arise which lead to a new set of constraints, which include the following:

- An agent does not have complete knowledge of the world and the effects of its own actions.
- An agent does not always know all of its goals in advance.
- Planning time is limited, and is shared with execution time.
- The mapping from an action in a plan to an action in the world is non-trivial.
- Projection over all possible worlds is theoretically and practically intractable.
- The goal of an agent is to act, not simply to plan.

More sophisticated systems of classical planning interleave planning and execution; they perform "execution monitoring" in which the planning phase records the expected state of the world between actions. The execution phase then monitors the execution of the plan, making sure that the world satisfies the descriptions of the expected intermediate states. If it does not, the system reverts to the planning phase with a description of the current state of the world as the initial state. Chapman (1987) has shown that the planning phase in such a system is, in the general case, undecidable. However, this problem can be solved, in particular cases, by restricting operator description (Kaelbling, 1990).

Examples of classical planners include the following.

- *STRIPS* (Nilsson, 1980), generates a sequence of actions for manipulating objects. The current state of the world is kept in a database and a goal description

on a goal stack. Using search, entries of the goal stack are matched with entries in the database. The computation terminates when all entries in the goal stack are matched.

- *NOAH* (Sacerdoti, 1975, 1977) uses a *least commitment* search strategy based on a *hierarchical* representation of plans in which actions may be *partially ordered.*
- *Nonlin*: introduced the notion of *goal structure* as a means of recording the rationale behind actions in the plan, and also the use of *typed preconditions* as an aid to search space control. A declarative task formalism (TF) was also used to provide a description of applications to the planner.
- *Deviser* (Vere, 1981) was derived from Nonlin but was extended to handle *time* and *events*.
- *Molgen* (Stefik, 1981) is notable for its ability to perform object selection using *least commitment principles*. This is supported by constraint formulation and propagation techniques.
- *McDermott* (McDermott, 1978) provides the notion of defining a plan to encompass the decisions on plan structure already taken and outstanding problems still to be handled by the planner.
- *OPM* (Hayes-Roth and Hayes-Roth, 1979) provided an *opportunistic* planning framework in a *blackboard* architecture. It introduced the concepts of *cognitive specialists* that can make certain kinds of decisions to alter the plan as it is being built and showed how a measure of the worth of invoking these specialists could be utilised.
- *O-Plans* (Tate, 1990) seeks to provide a more coherent set of mechanisms to enable the planning and control system builder to select suitable implementation methods for controlling the flow and ordering of choices.

12.6.12.2. *Reactive Systems*

Classical planning systems have focused entirely on a single agent, which operates in an unchanging world, has complete knowledge of its environment, and has relatively few constraints on planning and execution times. In MAS, when the concept of agency (a society of agents embedded in a dynamic world structure) becomes a central issue, agents' plans have to contain steps of sensing the world and reacting to it. It has been argued that such systems are more suited to guiding the agents' behaviour over long stretches of time.

Reactive systems are characterised by the following general procedure: At regular intervals (the granularity of time may be varied), the agent scans its environment, or a specific part of it, selects an action from a finite set of actions, and then executes the action. The action could be an internal action (changing the state of the agent) or an external action (changing the state of the world). These steps are iterated until a goal is satisfied, the goal fails, or the agent has exhausted its resources. Reactive planning systems are implemented by increasing the capabilities of the execution module, allowing plan generation to provide a high-level description of operations required to achieve a goal, and assuming that the execution model will fill in the required details. The generation of such high-level plans should be less computationally complex than the generation of detailed plans. Increasing the power of the execution model allows it to deal with contingencies that cannot be accounted for by the planning module.

Reactive planning has a number of drawbacks, including the following.

- How do we know that the agent will achieve its goal? With classical planners, the goal is proved to be satisfied. In a dynamic, changing world such a proof is much harder.
- In a complex environment, the execution module cannot determine the effect of its actions and whether some actions that it decides to execute may fail. Hence it cannot rely on the results of those actions.
- Unless the situation is routine, the execution module will be unable to focus on which aspects of a high-level operations are most appropriate.

Some examples of reactive systems are listed and described below.

SAMUEL (Grefenstertte, 1990). SAMUEL Is a system that uses competition-based machine learning to develop reactive plans. It incorporates several assumptions selected to make the system broadly applicable to real-world problems. The system's perception facilities are limited to a fixed set of discrete, possibly noisy, sensors. There is also a fixed a set of control variables that may be set by the decision-making agent. The system's decision rules are limited to simple condition/actions of the form

$$\begin{aligned} &\text{if (and } c_1 \ldots c_n) \\ &\text{then (and } a_1 \ldots a_m) \end{aligned}$$

where each c_i is a condition on one of the sensor and each action a_j specifies a setting for one of the control variables. A reactive plan in SAMUEL comprises a set of such decision rules.

Phoenix (Moehlman and Lesser, 1990). Phoenix provides a real-time environment for study of cooperative planning and decentralised negotiation. Phoenix is a fire-fighting system and is concerned with bringing about the actions needed to assess and contain simulated fires. Each agent is responsible for fires occurring in a predefined geographical area. Spatially distributed agents having only local views negotiate to plan a globally acceptable resource configuration. To realise the negotiation, a three-phase framework was created: negotiation to find bulldozers (to fight the fire), decision to delay goals, and negotiation for delaying goals.

Explanation-based control (de Jong, 1990). The approach here relied on explanation-based learning (EBL) over a plausible and qualitative domain theory to learn about and exploit domain characteristics. Learning produces new planning constructs which are the continuous analogue of EBL-acquired schemata.

12.6.12.3. *Combined Systems*

Recently much effort has been made in combining classical planning and reactive planning with a view to bringing positive aspects of the two systems into one (Agre and Chapman, 1987; Brooks, 1986; Hammond et al., 1990; McDermott, 1990; Mitchell, 1990; Schoppers, 1987; Zweben, 1990). Examples of combined systems include the following.

Theo-Agent (Mitchell, 1990). This is a robot control architecture which combines a stimulus–response subsystem for rapid reaction with a search-based planner for handling unanticipated situations. The robot agent continually chooses which actions to perform, using the stimulus–response subsystem when possible, and falling back on the planning subsystem when necessary.

The design of the Theo-Agent architecture is primarily driven by the goal of combining the complementary advantages of reactive and search-based systems. Reactive systems offer the advantage of quick response. Search-based planners offer the advantage of broad scope for handling a more diverse range of unanticipated worlds. The Theo-Agent architecture employs both, and uses explanation-based learning to incrementally augment its reactive component whenever forced to plan. In addition, the architecture makes widespread use of caching and dependency maintenance in order to avoid needless recomputation of repeatedly accessed beliefs. The primary characteristics of the Theo-Agent are as follows.

(1) It continually reassesses what action it should perform. The agent runs in a tight loop in which it repeatedly updates its sensor inputs, chooses a control action, begins executing it, then repeats this loop.
(2) It reacts when it can, and plans when it must. Whenever it must choose an action, the system consults a set of stimulus–response rules which constitute its reactive component. If one of these rules applies to the current sensed inputs, then the corresponding action is taken. If no rules apply, then the planner is invoked to determine an appropriate action.
(3) Whenever forced to plan, it acquires a new stimulus–response rule. The new rule recommends the action the planner has recommended in the same situations (i.e. those world states for which the same plan justification would apply) but can be invoked much more efficiently. Learning is accomplished by an explanation-based learning algorithm EBG (Mitchell et al., 1986) and provides a demand-driven incremental compilation of the planner's knowledge into an equivalent reactive strategy, guided by the agent's experiences.
(4) Every belief that depends on sensory input is maintained as long as its explanation remains valid. Many beliefs in the Theo-Agent, including its belief of which action to perform next, depend directly or indirectly on observed sense data. The architecture maintains a network of explanations for every belief of the agent, and deletes beliefs only when their support ceases. This caching of beliefs significantly improves the response time of the agent by eliminating recomputation of beliefs in the face of unchanging or irrelevant sensor inputs.
(5) It determines which goal to attend to on the basis of the perceived world state, a predefined set of goal activation and satisfaction conditions, and given priorities among goals.

TCA (Simmons, 1990). The task control architecture extends the classical planning framework to include capabilities for interleaving, planning and execution, monitoring, error recovery, and handling multiple tasks. The system was designed to facilitate building and the control of mobile robot systems that have multiple complex tasks, have limited sensors relative to their task, and operate in dynamic but relatively benign environments.

TCA consists of a task-independent central control process and utilities for communicating between the central control and task-specific process. More importantly, TCA provides facilities for maintaining, scheduling and executing hierarchical plans, for coordinating concurrent monitors and exception-handling strategies, and for managing physical and computational resources. The facilities were designed by analysing the requirements for several mobile robot systems. Several important capabilities were needed to extend the planning framework to achieve the necessary reactivity. These capabilities include the following.

(1) *Interleaving planning and execution.* While the world is in general too complex and uncertain to plan down to primitive actions, there are often times when advance planning is desirable, or even necessary. Robot systems need flexibility in specifying when to plan and when to act. This flexibility can be achieved in a hierarchical planning framework by placing temporal constraints on the planning and execution of tasks.
(2) *Detecting changes.* Reacting to change is basic to survival. In rich environments, however, it is often difficult to continuously check all relevant features. To manage with limited sensors, systems must select which features to monitor on the basis of their current tasks and environment.
(3) *Error recovery.* Purely reactive systems do not perform error recovery since they treat each situation afresh. Planning systems, however, must notice when plans are going astray and modify them accordingly. In addition, reflexive behaviours should be provided to safeguard the robots.
(4) *Coordinating multiple tasks.* Unexpected opportunities and contingencies may give rise to multiple tasks. Robot systems must decide whether tasks can occur concurrently and, if not, in which contexts one task has priority over another. In addition, they should be able to interrupt lower-priority task and move smoothly to new ones.

XFRM (McDermott, 1990). XFRM provides a transformation to planning capabilities on top of a reactive plan interpreter for a robot delivery truck. Planning is implemented by way of a set of critics and schedulers that anticipate problems with the plan by projecting it ahead of time and seeking to transform the plan to remove these problems.

Decomposition abstractions (Martin and Allen, 1990). This approach combines reactive and classical planning systems. Statistical methods are used to gather data from executions of plans to guide the appropriate level of plan description. The plan is elaborated until the strategic planner is sufficiently confident that this plan will indeed achieve its goals on the basis of the previous behaviour of the executor. The plan is then given to the execution module. To address the problem of how to decide what the planner needs to reason about and what problems the reactive execution system can deal with, a technique based on decomposition abstraction was used. This technique provides a method of deciding what aspects of the planning problem the reactive execution module is capable of handling on its own on the basis of the prior performance of that execution module. It uses statistics on the execution module's prior performance to constrain the probability that the execution module can accomplish a particular task. If it can, the planner need not reason about that task. If

it cannot, the planner must discover the likely causes of failure and specify a plan that avoids them.

12.6.12.4. *Case-Based Planning*

The case-based approach to planning is an attempt to deal with interactability of proofs in complex domains present in classical planning. The idea is to develop a case database of plans which can be constructed and used incrementally. In this way a search is used to locate an appropriate stored plan that can satisfy the current goal. One way that a case-based planner can improve itself over time is to combine planning and learning such that the agent would have an understanding of its own success and failure within a given domain, then either modify or increment an existing plan to satisfy an unexpected goal or changes in the environment (Hammond, 1989; Kolodner and Simpson, 1984; Martin, 1990; Owens, 1990; Schank, 1982). Case-based planning suggests that the way to approach the combinatorics of planning and projection is to let experience (past histories) guide the planner as to when and where things work and do not work: rather than re-planning, re-use plans. This framework suggests seven basic case-based planning process modules (Hammond et al., 1990):

(a) Prediction: predicts planning problems on the basis of similar past cases.
(b) Retrieve: to search memory for a plan that satisfies, or partially satisfies current goals.
(c) Modify: to modify a retrieved plan, and satisfy remaining goals not yet satisfied by the current plan.
(d) Projection: uses cases indexed by planning solutions, rather than problem based as in the case with the predication module.
(e) Indexing: places new plans in memory, indexed by the goals they satisfy and the problems to be avoided.
(f) Re-plan: this module is called if a plan fails; causal knowledge may be applied here.
(g) Assign: uses causal explanation built during re-plan to determine features which will predict this failure in the future. This knowledge is used to index the failure for later anticipation.

In conclusion, planning is an important but still not soundly developed area of artificial intelligence. It needs to be integrated into a theory of agency which incorporates the different planning behaviour discussed above. We need to remove the sharp distinction of planning and execution phases prominent in classical planning, learning methods and experimental methods.

12.6.13. Engineering Methods and Practices in MAS

How do we design and develop practical MAS systems? How do we develop methodologies and tools in support of the construction of such systems? A number of technologies have been built, including RATMAN (Bürckert et al., 1991), which consist of a workbench of the definition and testing of rational agents in multiagent

environments. The special feature of RATMAN is the specification of such agents with hierarchical knowledge bases. AOP (Shoham, 1990) developed a language called agent-oriented programming which can be viewed as a specialisation of object-oriented programming. In this language, the state of an agent consists of components called beliefs, choices, capabilities and commitments. The language uses standard epistemic logic. MACE (Gasser et al., 1987) stands for multi-agent computing environment. The computing units in MACE, the agents, build an organisation of problem solvers. Agents run in parallel, communicate via messages and have facilities for knowledge representation. Other systems include test beds such as DUMT (Durfee and Lesser, 1987), integrative systems like ABE (Erman, 1988), and reflective systems such as BBI (Hayes-Roth, 1990), SOAR (Rosenbloom et al., 1990), and CAGE/POLIGON (Nilsson, 1989).

12.7. AGENT ARCHITECTURE

In this section we define agent architecture as a number of components representing the mental state of an agent and the interaction between them. We emphasise that this is not a unique correct model, and that different applications may require specific agent structure. The architecture comprises a number of different hierarchical levels as shown in Figure 12.3.

12.7.1. Sensors and Effectors

Sensors are the means by which an agent senses its world; they include vision, sound, touch, and signals, the form of input to the agent. Effectors are the means by which an agent affects the world, e.g. moving an object from one place to another, sending a message to another agent. Effectors form agent output.

12.7.2. Communication Processor and Perception Interfaces

In this component, input and output information are analysed, sorted and sent to the relevant knowledge base of agent memory structure.

12.7.3. Knowledge Bases

Knowledge bases comprise the following:

- *Objects*. This knowledge base stores information about objects in the world and relationships between them. For example, in the block world, knowledge about shape, size, colour, weight, material, and their special relationship are represented.
- *Time*. This knowledge base represents information about the topology of time – linear, branching, point, interval, bounded, etc. Axioms characterise the properties

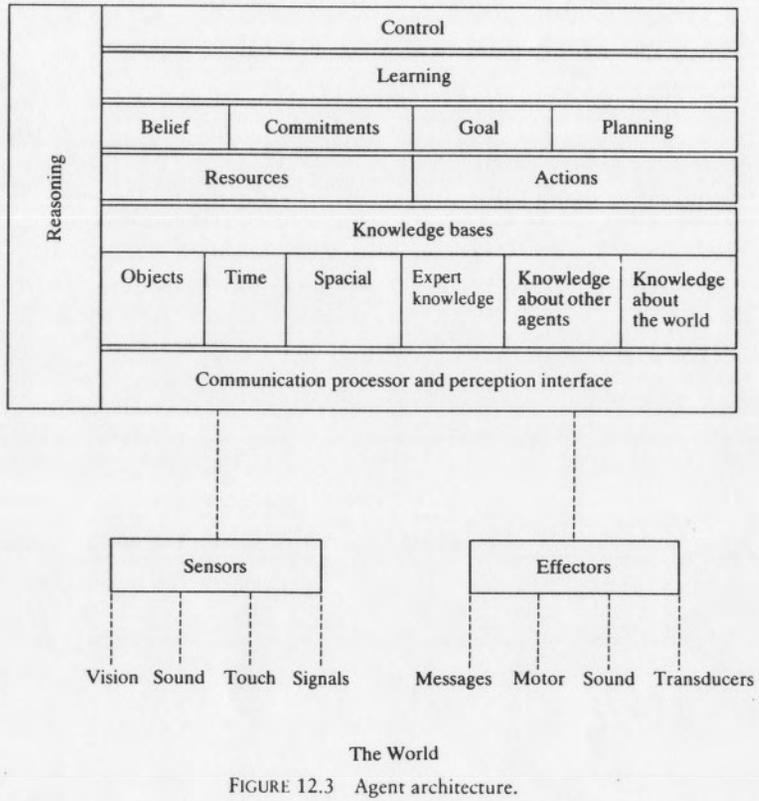

FIGURE 12.3 Agent architecture.

of time, the temporal constraints on objects, and temporal relationships between agents, objects and the world.

- *Spatial.* This is similar to the time knowledge base but is applied to spatial information.
- *Expert knowledge.* This knowledge base contains expert knowledge about a specific domain. It is the agent's "know-how".
- *Knowledge about other agents.* This knowledge base contains information about other agents, i.e. a model of other agents, that this agent needs to know.
- *Knowledge about the world.* This knowledge base describes what may be termed common-sense information (e.g. if you drop a cup of coffee, then it is very likely that the coffee will be spilled; or the fact that a car is used for transport).
- *Resources.* This knowledge base comprises resources available to the agent, including computational resources.
- *Actions.* This knowledge base contains information on the set of primitive actions (operations) that the agent is capable of performing.
- *Belief, commitments, goals.* These knowledge bases contains primitive modalities of belief, commitments and goals describing the agent's mental state. These primitive modalities are the basis by which other mental states of agents are defined; they include mutual and common belief, commitment and action.

- *Planning.* This component specifies planning methods (e.g. classical, reactive, case, combined) available to the agent.
- *Learning.* This component contains algorithms by which an agent can update its knowledge bases (e.g. induction, explanation-based learning).
- *Control.* This top-level control can be thought of as a meta-system and contains information on what the agent knows, control the search space, and how to apply integrity constraints that ensure consistency of knowledge bases.
- *Reasoning.* This is a complex component made of several subcomponents that allow the agent to reason at different levels of the hierarchy. The reasoning mechanism should in addition to classical logic reasoning, include non-monotonic reasoning and reasoning about beliefs, time, space and actions.

12.8. A FORMAL MODEL FOR MULTIAGENT SYSTEMS

The formal model presented in this section is very close to similar work of Rao, Georgeff and Sonnenberg (Rao and Georgeff, 1991), Singh (1990), Shoham (1990) and Rosenschein and Kaelbling (1986). The model is based on the three primitive modalities of belief, goal and commitment. More complex modalities are defined in terms of these primitive modalities. Two kinds of agents are considered: single agents and group agents. A group agent is an abstract entity that not only denotes individual agents but may also include other subgroups. From the outside, a group agent may be thought of as a single social agent. We use standard first order logic, modal logic, and branching-time modal temporal logic to describe situations, and dynamic logic to describe planning. The semantics of the model is based on an extension of standard Kripke interpretation of possible worlds, where each possible world is a temporal structure. A joint plan by a group agent involves a commitment from all members of the group that each one will satisfactorily discharge its commitment. Such joint commitments are formalised as constraints on actions and belief. A joint plan is a set of joint actions; joint actions often involve strict synchronisation conditions.

12.8.1. Syntax

12.8.1.1. *Joint Plans*

A joint plan is a pair ⟨precondition, body⟩. The precondition is a set of well-formed formulae that have to be satisfied before the body of the plan is executed. The body of the plan is a set of plan expressions; plan expressions are formulae in dynamic logic with extension to specify the agent having the plan. Plans, or more specifically a plan type, is an abstract structure that when executed by an agent results in the occurrence of an action in the real world; this action will often result in a state change of the world.

A primitive plan expression is a pair consisting of a plan type and an agent. More complex plan expressions may be constructed using the dynamic logic operator (;) for

sequencing; ($\|$) for parallelism; (*) for iteration and (|) for non-deterministic choice. The operators ? and !, operate on well-formed state formulae to convert them into plan types. $?\alpha$ is a plan that tests whether the condition α is true and $!\alpha$ is a plan type that achieves α.

Definition 12.1 (Rao and Georgeff, 1991)

- If p is a plan type and y is an agent (an agent is either a single or group agent) then $(p\,y)$ is a well-formed plan expression.
- If α is a well-formed formula and y is an agent, then $(!\alpha y)$ and $(?\alpha y)$ are well-formed plan expressions.
- If x_1 and x_2 are well-formed plan expressions, then $(x_1; x_2)$, $(x_1 \| x_2)$, $(^* x_1)$, $(x_1 | x_2)$ are well formed-plan expressions.

Action formulae are a result of the execution of a plan by an agent. These action formulae describe whether the execution was a success or a failure, and whether it occurred in the past or will occur in the future.

Definition 12.2. If x is a well-formed plan expression then $\langle x\rangle$, $\langle x\rangle_s$, $\langle x\rangle_f$, $[x]$, $[x]_s$ and $[x]_f$ are well-formed formulae.

The first three action formulae denote immediate future executions, and the last three indicate immediate past executions. The subscripts "s" and "f" denotes success and failure of execution, respectively. Without the subscript, the execution can be either a success or a failure.

12.8.2. Temporal and Modal Operators

The temporal structure of the model is described using computation tree logic CTL* (Emerson, 1989). The temporal structure CTL* is a tree with branching future and a single past. A distinction is made between state formulae and path formulae: state formulae are evaluated at specified time points in a time tree, whereas path formulae are evaluated over a specified path in a time tree. Two modal operators, **optional** and **inevitable**, operate on path formulae. A path formula ψ is said to be **optional** if at a particular time point in a time tree, ψ is true of at least one path emanating from that point, it is **inevitable** if ψ is true of all paths emanating from that point.

The standard temporal operators O (next), ◊ (eventually) and □ (always) operate over state and path formulae. Two types of arcs between time points are introduced: success arcs and failure arcs. If the arc is a success arc, the primitive plan type is said to be successful; if it is in a failure arc, the primitive plan is considered to have failed.

The modal operators **BEL**, **GOAL** and **COMT** are used to denote individual beliefs, goals and commitments. The corresponding joint attitudes – namely, mutual beliefs, joint goals, and joint commitment – are denoted by **MBEL**, **JGOAL**, and **JCOMT**, respectively. We also use the operators **EBEL**, **EGOAL** and **ECOMT** to denote the beliefs, goals and commitment of all the members of a social agent. All joint propositional attitudes are defined in terms of the individual propositional attitudes.

State formulae are defined as follows.

- Any first order formula is a state formula.
- If ϕ_1 and ϕ_2 are state formulae and x is an individual or plan variable, then $\neg\phi_1$, $\phi_1 \vee \phi_2$, and $\exists x \phi_1(x)$ are state formulae.
- If ϕ is a well-formed action formula then ϕ is also a state formula.
- If ϕ is a state formula and y is an individual agent then **BEL**$(y\,\phi)$, **GOAL**$(y\,\phi)$ and **COMT**(ϕ) are state formulae.
- If ϕ is a state formula and y is a social agent then **MBEL**$(y\,\phi)$, **JCOMT**$(y\,\phi)$ **EBEL**$(y\,\phi)$, **EGOAL**$(y\,\phi)$, **EGOAL**$(y\,\phi)$, and **ECOMT**$(y\,\phi)$ are state formulae.
- If ψ is a path formula, then **optional** (ψ) and **inevitable** (ψ) are a state formulae.

Path formulae can be defined as follows.

- Any state formula is also a path formula.
- If ψ_1 and ψ_2 are path formulae, then $\neg\psi_1$, $\psi_1 \vee \psi_2$, $\Diamond\psi_1$, $O\psi_1$ are path formulae.

12.8.3. Semantics

Definition 12.3. An interpretation M is defined to be a tuple, $M = \langle W, IA, GA, PP, P, PLANS, MEMBERS, T, <, U, JPS, PSA, B, G, I, \Phi\rangle$ where W is a set of worlds, IA is a set of individual agents, GA is a set of group agents, PP is a set of primitive plan types, P is a set of plan types, T is a set of time points, $<$ is a binary relation on time points, U is the universe of discourse, and JPS is the set of all joint plan structures. PSA is a plan structure assignment function that maps a plan type to a joint plan structure. $PLANS$ is a function from individual or joint agents to a set of plan types. Intuitively, this function provides the plan library of the agent. $MEMBERS$ is a relation between group agents and other groups and individual agents. More formally, $MEMBERS \subseteq GA\{GA \cup IA\}$. The accessibility relations, B, G and I map an individual agent's current situations to his belief, goal, and commitment accessible worlds, respectively. More formally, $B \leftarrow IA \times W \times T \times W$ and similarly for G and I. Φ is mapping of first-order entities to elements in U for any given world and time point.

Definition 12.4. A joint plan structure is a tuple $\langle \phi_{pre} P_{body}\rangle$, where ϕ_{pre} is any well-formed formula and P_{body} is any well-formed plan expression. We also have the functions *pre* and *body* which, given a plan type, return the appropriate argument of the above tuple.

Definition 12.5. Each world w of W, called a *time tree*, is a tuple $\langle T_w, <_w, S_w, F_w\rangle$, where $T_w \subseteq T$ is a set of time points in the world w and $<_w$ is the same as $<$, restricted to time points in T_w. A *fullpath* in a world w is an infinite sequence of time points $(t_0, t_1, \ldots)$ such that $\forall i (t_i, t_{i+1}) \in A_w$. We use the notation $(w_{t_0}, w_{t_1}, \ldots)$ to make the world of a particular fullpath explicit. The arc functions S_w and F_w map time points to a primitive plan type. More formally, S_w: $T_w \times t_w \mapsto 2^{PP}$ and similarly for F_w. The domains for S_w and F_w are disjoint. Intuitively, for any two time points for which the arc function S_w is defined, its value represents the primitive plan that successfully occurred (or was performed by agent(s)) between those time points. Similarly, the

value of the arc function F_w represents the failure of a primitive plan occurring between those time points.

12.8.4. Semantics of Temporal Modalities

The semantics of temporal modalities is straightforward. Both $O\psi$ and $\Diamond\psi$ are path formulae and are evaluated over a particular path. The formula **optional**(ψ) is a state formula and is true if there is at least one path where ψ is true. More formally, we have

$M, v, (w_{t_0}, w_{t_1}, \ldots) \models O\psi$ iff $M, v(w_{t_1}, \ldots) \models \psi$.
$M, v, (w_{t_0}, w_{t_1}, \ldots) \models \Diamond\psi$ iff $\exists k, k \geq 0$ such that $M, v, (w_{t_k}, \ldots) \models \psi$.
$M, v, w_{t_0} \models$ **optional**(ψ) iff there exists a fullpath $(w_{t_0}, w_{t_1}, \ldots)$ such that $M, v(w_{t_1}, \ldots) \models \psi$.

The formula **inevitable**(ψ) is defined as $\neg$**optional**$(\neg\psi)$ and $\Box\psi$ is defined as $\neg\Diamond\neg\psi$.

12.8.5. Semantics of Joint Plan Executions

A group or individual agent has a library of plans. All plans serve a purpose, which is either to achieve a certain condition (as in $!\alpha$) or to test for a certain condition (as in $?\alpha$).

We say that an agent y has a plan type p to achieve the condition α if, whenever the plan has been successfully executed, the condition α holds. We also require that the plan be in the agent's plan library.

$M, v, w_{t_0} \models$ *has-plans*$(p(!\alpha\, y))$ iff
(a) $p \in PLANS(y)$ and
(b) $M, v, w_{t_0} \models$ **inevitable** $\Box([(p\, y)]_s \Rightarrow \alpha)$.
(*has-plan*$(p\, x)$, p denotes a plan, and x denotes what the plan accomplishes, and who has the plan.)

Having a plan to test for a certain condition is very similar. We say that an agent y has a plan type p to test for condition α if, prior to the successful execution of the plan, the condition α holds. As before we require that the plan be in the agent's plans library.

$M, v, w_{t_0} \models$ *has-plan*$(p(?\alpha\, y))$ iff
(a) $p \in PLANS(y)$ and
(b) $M, v, w_{t_0} \models$ **inevitable** $\Box(\langle(p\, y)\rangle_s \Rightarrow \alpha)$.

Next we consider what it means for an agent to execute a plan type. We say that an agent y successfully executes a plan type p if the precondition of the plan is satisfied and the body is executed successfully.

$M, v, (w_{t_0}, w_{t_1}, \ldots) \models \langle(p\, y)\rangle_s$ iff
$M, v(w_{t_0}, w_{t_1}, \ldots) \models pre(p) \wedge \langle body(p)\rangle_s$.

The past execution of plans is somewhat more complicated to specify. We say that a plan type p has been successfully executed by agent y if the body of the plan has been executed successfully, the precondition held at some time in the past when the execution of the body started, and there was no other successful execution of the body in between.

$M, v, w_{t_n} \models [(p\, y)]_s$ iff
(a) $\exists t_0, t_0 < t_n$ such that $M, v, (w_{t_0}, \ldots) \models \langle pre(p) \wedge (p\, y)\rangle_s$;
(b) $M, v, w_{t_n} \models [body(p)]_s$; and
(c) $\nexists t_i, t_0 < t_i < t_n$ such that $M, v, w_{t_i} \models [body(p)]_s$.

The body of a plan could contain an expression to achieve or test for a certain condition. An agent y is said to achieve successfully the condition α if there is plan type p whose purpose is to achieve α, and the plan is executed successfully. Similarly for testing a condition. More formally, if x stands for $(!\alpha\, y)$ or $(?\alpha\, y)$, we have

$M, v, (w_{t_0}, w_{t_1}, \ldots) \models \langle x\rangle_s$ iff there exists a plan type p such that
(a) $M, v, w_{t_0} \models$ *has-plan*$(p(p\, x))$ and
(b) $M, v, (w_{t_0}, w_{t_1}, \ldots) \models \langle p\, y\rangle_s$.

We say that a sequence of two primitive plans is successfully executed if each primitive is executed successfully one after the other. Two parallel primitive plans are successfully executed if both of them are successfully executed at the same time, i.e. both label the same arc. Two non-deterministic primitive plans are successfully executed if either one of them is successfully executed. More formally, the successful future executions can be stated as follows:

$M, v, (w_{t_0}, w_{t_1}, \ldots) \models \langle(e_1\, a_1);\ (e_2\, a_2)\rangle_s$ iff
$M, v, (w_{t_0}, w_{t_1}, \ldots) \models \langle(e_1\, a_1)\rangle_s$ and $M, v, (w_{t_1}, \ldots) \models \langle(e_2\, a_2)\rangle_s$.
$M, v, (w_{t_0}, w_{t_1}, \ldots) \models \langle(e_1\, a_1) \parallel (e_2\, a_2)\rangle_s$ iff
$M, v, (w_{t_0}, w_{t_1}, \ldots) \models \langle(e_1\, a_1)\rangle_s$ and $M, v, (w_{t_0}, w_{t_1}, \ldots) \models \langle(e_2\, a_2)\rangle_s$.
$M, v, (w_{t_0}, w_{t_1}, \ldots) \models \langle(e_1\, a_1) \mid (e_2\, a_2)\rangle_s$ iff
$M, v, (w_{t_0}, w_{t_1}, \ldots) \models \langle(e_1\, a_1)\rangle_s$ or $M, v, (w_{t_0}, w_{t_1}, \ldots) \models \langle(e_2\, a_2)\rangle_s$.

The failure of execution and past executions of a sequence of primitive plans, parallel primitive plans and non-deterministic primitive plans can be stated in a similar manner.

Finally, we consider the success or failure of execution of primitive plan types. This is straightforward: a primitive plan is successfully executed if it labels a success arc and fails if it labels a failure arc. If e is a primitive plan type then we have the following semantics:

$M, v, (w_{t_0}, w_{t_1}, \ldots) \models \langle e\rangle_s$ iff for some t_1, $e \in S_w(t_0 t_1)$.
$M, v, w_{t_1} \models [e]_s$ iff for some t_0, $e \in S_w(t_0 t_1)$.
$M, v, (w_{t_0}, w_{t_1}, \ldots) \models \langle e\rangle_f$ iff for some t_1, $e \in F_w(t_0 t_1)$.
$M, v, w_{t_1} \models [e]_f$ iff for some t_0, $e \in F_w(t_0 t_1)$.

We define attempting an execution as either a successful execution or a failure of execution, i.e. $\langle e\rangle \Leftrightarrow \langle e\rangle_s \vee \langle e\rangle_f$ and similarly for past executions.

Using these definitions we can distinguish between having and executing a social plan. More formally, the following formulae are satisfiable in our logic:

- Having a plan and not executing the body of the plan:

$$has\text{-}plan(p\, x) \wedge \neg\langle body(p)\rangle.$$

- Executing the body of the plan but not executing it successfully:

$$\langle body(p)\rangle \wedge \neg\langle body(p)\rangle_s.$$

In the above case x can be a plan expression to achieve or test a certain condition. The last property is the same as execution failure of the body of p.

12.8.6. Semantics of Mutual Beliefs and Joint Goals

Belief is modelled in the conventional way using KD45 modal logic. That is, instead of one world, we have a set of different possible worlds. A particular time point in a particular world is called a *situation*. For each situation we associate a set of *belief-accessible*, *goal-accessible* and *intention-accessible* worlds; intuitively, those worlds that the agent *believes* to be possible, *desires* to bring about, or *commits* to achieving, respectively. Unlike most conventional models of belief, however, each belief-, goal- and intention-accessible world is a time tree. Multiple possible worlds result from the agent's lack of knowledge about the state of the world. But within each of these possible worlds the branching future represents the *choice* of actions available to the agent. Moving from belief to goal to intention worlds amounts to successively pruning the paths of the time tree; intuitively, to making increasingly selective choices about one's future actions.

The belief relation maps a possible world at a time point for a particular agent to a set of belief-accessible worlds. We say that an agent a has a belief ϕ, denoted $\mathbf{BEL}(a\ \phi)$, at time point t if and only if ϕ is true in all the belief-accessible worlds of the agent at time t. We use $\mathbf{B}_t^w(a)$ to denote the set of belief-accessible worlds of agent a from world w and time t, i.e.

$$\mathbf{B}_t^w(a) = \{w' \mid \mathbf{B}(a\, w\, t\, w')\}.$$

The semantics for beliefs can be defined formally as follows:

$$M, v, w_t \models \mathbf{BEL}(a\ \phi) \quad \text{iff} \quad \forall w' \in \mathbf{B}_t^w(a)\, M, v, w_t' \models \phi.$$

The semantics of goal and intentions are defined analogously by using the relations G and I.

The main semantic constraint imposed on the belief, goal and intention relation is that for each belief-accessible worlds there exists a sub-world which is goal-accessible and, in turn, for each goal-accessible world there exists a sub-world which is intention-accessible. This semantic constraint is called strong realism and is formalised elsewhere. Defining O-formulae to be well-formed formulae that contain no positive occurrences of **inevitable** (or negative occurrences of **optional**) outside the scope of belief, goal, or modal operators, we have the following axiom of strong realism.

Having commitments towards the body of a plan is different from *having* a plan and also different from *executing* the body of the plan. In other words, having a plan does not entail intention to execute the body of the plan structure and executing the body of the plan structure does not entail an intention to do so. More formally, the following formulae are satisfiable in this logic.

- Having a plan and not intending to execute the body of the plan:

$$ha\text{-}plan(p!(\alpha\, a)) \wedge \neg\mathbf{COMT}(a\langle body(p)\rangle).$$

- Executing the body of the plan and not intending to execute the body of the plan:

$$\langle body(p)\rangle \wedge \neg\mathbf{COMT}(a\langle body(p)\rangle).$$

Next we examine the semantic of all members of a social agent believing a formula. The formula $\mathbf{EBEL}(y\ \phi)$ is satisfiable if all members of the social agent y believe in ϕ. If the member is an individual agent, the agent believes in it; if the member is another social agent, all members of that social agent believe in it. As previously, the definition of "everyone believes" is recursive.

$$\mathbf{EBEL}(y\ \phi) \equiv \wedge_{[a|\, members(y\, a)\ and\ a\ \mathrm{NIA}|}\mathbf{BEL}(a\ \phi) \wedge \wedge_{[z|\, members(y\, z)\ and\ z\ \mathrm{NSA}|}\mathbf{EBEL}(z\ \phi).$$

The satisfaction of **EGOAL** and **COMT** is defined likewise.

Now the mutual belief ϕ of a group agent is defined as all members of the group agent believing ϕ and all of them believing that x is mutually believed. The joint goal ϕ of a group agent is defined as all members of the group agent having the goal ϕ and mutually believing that ϕ is held as a joint goal. Joint commitments are defined in the same way as joint goals.

$$\mathbf{MBEL}(y\ \phi) \equiv \mathbf{EBEL}(y\ \phi) \wedge \mathbf{EBEL}(y\, \mathbf{MBEL}(y\ \phi)).$$
$$\mathbf{JGOAL}(y\ \phi) \equiv \mathbf{EGOAL}(y\ \phi) \wedge \mathbf{MBEL}(y\, \mathbf{JGOAL}(y\ \phi)).$$
$$\mathbf{JCOMT}(y\ \phi) \equiv \mathbf{ECOMT}(y\ \phi) \wedge \mathbf{MBEL}(y\, \mathbf{JCOMT}(y\ \phi)).$$

Note the asymmetry between the definitions of **MBEL** and **JGOAL**; while **MBEL** allows arbitrary nestings of **BEL** operators, **JGOAL** allows arbitrary nestings of **BEL** operators with the innermost operator being a **GOAL** operator. However, there is a symmetry between the definitions of **JGOAL** and **JCOMT**, both allow arbitrary nestings of **BEL** operators with the innermost operator being **GOAL** and **COMT**, respectively.

The above definitions together with the strong realism axiom yields the following important theorem (Rao and Georgeff, 1991).

Theorem 12.1. $\models \mathbf{JCOMT}(y\ \psi) \rightarrow \mathbf{JGOAL}(y\ \psi) \Rightarrow \mathbf{MBEL}(y\ \psi)$, where ψ is any O-formula.

This theorem states that, if a group agent is jointly committed to an O-formula, the group agent also has it as a joint goal and also mutually believes it.

As with individual commitment, it can be shown that having a *joint commitment* towards the body of a joint plan is different from *having* a joint plan and also different from *executing* the body of the joint plan. More formally the following formulae are satisfiable in the logic.

- Having a plan and not jointly intending to execute the body of the plan:

$$has\text{-}plan(p!(a\, y)) \wedge \neg\mathbf{JCOMT}(y\langle body(p)\rangle).$$

- Executing the body of the plan and not jointly committed to execute the body of the plan:

$$\langle body(p)\rangle \wedge \neg\mathbf{JCOMT}(y\langle body(p)\rangle).$$

12.8.7. Multiagents

Whenever an agent is committed to the body of a plan structure, then the agent must have a goal towards the purpose of the plan and the preconditions must be believed.

$$\rightarrow \textit{has-plan}(p(!a\,a)) \wedge \mathbf{COMT}(a\langle \textit{body}(p)\rangle) \Rightarrow \mathbf{GOAL}(a\langle(!a\,a)\rangle_s) \wedge \mathbf{BEL}(a\langle \textit{pre}(p)\rangle)$$

However, this requirement alone is not sufficient for the agent to form commitments and act on them. We need additional constraints that would force the agent to form commitments. The stronger version of the means–end reasoning axiom can be stated as follows: If an individual agent has a plan p and has acquired the goal towards the purpose of this plan, and believes in the precondition of the plan, he is committed to execute the body of the plan. The body of the plan may contain other achievement plan expressions. An agent committed to such an achievement plan expression would then be forced to have a goal to achieve it. This goal may result in further commitments to execute the body of other plan structures. This hierarchical planning proceeds until the agent has executed the body of its top-level plan structure. Thus, for an individual agent a, we have the following axiom for means–end reasoning:

$$\vdash \textit{has-plan}(p(!a\,a)) \wedge \mathbf{GOAL}(a\langle(!a\,a)\rangle_s \wedge \mathbf{BEL}(a\,\textit{pre}(p)) \Rightarrow \mathbf{COMT}(a\langle \textit{body}(p)\rangle)$$

Note that, whenever the premise of the axiom is true, the agent is going to **COMT** the body of the plan structure. However, the agent may not act on all such commitments: an agent acts only if its immediate commitment is towards a non-deterministic action. In other words, if the agent has multiple present-directed commitment it needs to deliberate and choose the best possible action before acting. The agent is allowed to have multiple future-directed commitments as it can continue postponing deliberation until forced to act.

The scenario for multiple agents is very similar – one considers joint attitudes rather than individual attitudes. Thus, if a group agent has a plan p and has acquired the joint goal towards the purpose of this plan, and mutually believes in the precondition of the plan, then the agent will jointly be committed to execute the body of the plan. This joint intention would trigger the group agent to acquire other joint and individual goals, which might trigger further joint commitments, and so on. As before, if the group agent has successfully executed the body of the group plan structure we can say that the group agent mutually believes the p condition. Thus for a group agent y we have the following axiom for hierarchical planning:

$$\vdash \textit{has-plan}(p(!\alpha\,y)) \wedge \mathbf{JGOAL}(y\langle(!\alpha\,y)\rangle_s) \wedge \mathbf{MBEL}(y\,\textit{pre}(p)) \Rightarrow \mathbf{JCOMT}(y\langle \textit{body}(p)\rangle)$$

The above axioms also hold when the agent has a plan to test for a certain condition.

This exposition of a formal model is only preliminary, as Rao indicates, and much work needs to be done to formalise joint commitments, negotiations and conflicts.

12.9. CONCLUSION

In this chapter we described the new and exciting area of multiagent systems (MAS). We argued that many important applications ranging from process control and manufacturing to health care and diagnostics can only be tackled, from computation

point of view, by the construction of computational processes – both hardware and software – that are part of a larger system embedded in a physical environment. We argued that the behaviour of agents should be viewed as an on-going attempt to discover, create, and maintain a stable relationship with the world they inhabit. The relationship between the agent and the world is necessary for the generation of goals, plans and actions such that agents can affect and be affected by their world. The concept of agency rests on the idea that general-purpose intelligence is only possible if agents can be autonomous and capable of learning, planning, and work to establish functional correspondence between the world and their conception of it. This approach is different from traditional AI, which tended to produce either a "general-purpose" system that proved to be inadequate in solving problems in different domains, or to sacrifice generality by concentrating on domain-dependent programs that are effective in their own areas because of the skills of programmers rather than their own internal structures.

This chapter should be viewed as an attempt to study agency. Much work is still awaited and needs to be done. The study must rest on the premise that intelligent behaviour is a long-term activity, and effort must be aimed at developing a theory of knowledge, learning, action and communication. Some work toward this end has already started as shown in this chapter.