THE LOGICAL MODELLING OF COMPUTATIONAL MULTI-AGENT SYSTEMS

A THESIS SUBMITTED TO THE UNIVERSITY OF MANCHESTER
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
IN THE FACULTY OF TECHNOLOGY

MICHAEL JOHN WOOLDRIDGE
DEPARTMENT OF COMPUTATION
AUGUST 1992

DECLARATION

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institution of learning.

Abstract

THE aim of this thesis is to investigate logical formalisms for describing, reasoning about, specifying, and perhaps ultimately verifying the properties of systems composed of multiple intelligent computational agents. There are two obvious resources available for this task. The first is the (largely AI) tradition of reasoning about the intentional notions (belief, desire, etc.). The second is the (mainstream computer science) tradition of temporal logics for reasoning about reactive systems. Unfortunately, neither resource is ideally suited to the task: most intentional logics have little to say on the subject of agent architecture, and tend to assume that agents are perfect reasoners, whereas models of concurrent systems from mainstream computer science typically deal with the execution of individual program instructions.

This thesis proposes a solution which draws upon both resources. It defines a model of agents and multi-agent systems, and then defines two execution models, which describe how agents may act and interact. The execution models define what constitutes an acceptable *run* of a system. A run may then act as a model for a temporal logic; this logic can subsequently be used to describe and reason about multi-agent systems. A number of logics, with various properties, are developed in this way. Several detailed examples are presented, showing how the logics may be used for specifying and reasoning about multi-agent systems.

The thesis includes a detailed literature survey.

Acknowledgements

ONLY one name appears on the cover of this thesis, but a great many people have been indirectly involved in its production…

First, thanks to the SERC for financial support.

Second, thanks to Greg O'Hare, for all his enthusiasm and encouragement; it is difficult to imagine a friendlier or more supportive supervisor.

Third, thanks to everyone who has been involved in DAI/CSCW research at UMIST over the past three years. In particular, thanks must go to Paul Dongha, Rebecca Elks, Afsaneh Haddadi, Lynne Hall, Robin Pike, and Steve Viller.

Fourth, I would like to thank everyone else who has read my reports, commented on my ideas, engaged in protracted email debates, and generally helped me to understand. Special thanks here to Nigel Shardlow, whose discussions proved a fruitful source of ideas, and Daniel Mack, who carefully read and commented on a thesis draft.

Fifth, I would like to offer a special vote of thanks to Michael Fisher, who spent a great deal of time reading my reports, helping me to understand what I was doing, and helping me to sort out the technical details of my work. This thesis would quite simply not exist if it were not for Michael's help.

Finally, to my wife Janine, (who hopes everything will get back to normal now), for everything: thank you.

This thesis was submitted to the Faculty of Technology in the University of Manchester on September 4, 1992, and was successfully defended on October 23, 1992. It was typeset using LaTeX.

Contents

List of Tables

List of Figures

Part I

Introduction

Chapter 1

Introduction

DISTRIBUTED Artificial Intelligence (DAI) is an emerging subfield of Artificial Intelligence (AI) whose loose focus is the study of cooperative activity in systems composed of multiple intelligent computational agents. This thesis will use the expression "multi-agent system" to refer to such systems. The aim of the thesis is to investigate logic-based formalisms for describing, reasoning about, specifying and ultimately verifying the properties of multi-agent systems[1].

There is a very real need for formal tools with which to analyze and reason about multi-agent systems. This chapter discusses the motivations for developing such tools, and the various resources that are available for the task. The methodology of the thesis is then described. Finally, the structure of the remainder of the thesis is outlined.

1.1 Motivation

It is worth observing that a formal theory of multi-agent systems would be valuable in its own right. As Konolige observes [100, p5], *formal* models force us to make assumptions plain, and admit a rigorous examination of what predictions the model does and does not make.

Also, there is a pressing need for tools to help manage the complexity of multi-agent systems. Designing, implementing and debugging multi-agent systems is not easy. In a seminal 1987 paper, Gasser *et al* observed that:

> "Concurrency, problem domain uncertainty and non-determinism in execution together conspire to make it very difficult to understand the activity within a distributed intelligent system". [70]

To counter these problems, the authors advocated the development of:

> "…graphic displays of system activity linked to intelligent model based tools which help a developer reason about expected and observed behaviour". [70]

The work described by Gasser and colleagues is firmly and unashamedly in the empiricist tradition of AI: hence the call for practical tools to aid understanding. However, given the software experience of the past two decades, it is perhaps surprising that no similar plea was made for principled techniques for reasoning about, specifying, and verifying the properties of multi-agent systems.

Finally, it is to be expected that a formal theory of multi-agent systems would provide a framework in which general questions about cooperation and social interaction might be posed, and solutions developed.

To summarize, a suitable formal theory of multi-agent systems might fulfill the following roles:

- provide a tool with which to *describe* and *reason about* multi-agent systems;
- provide a tool with which to *specify* and, (ultimately), to *verify* the properties of multi-agent systems;
- provide a foundation upon which more profound theories of social action, interaction, and cooperation might be constructed.

[1] A detailed introduction to the scope, aims and methodologies of DAI is not possible given the limited space available. See [16] for a representative survey of DAI research up to 1988, and [90], [71], [36], [37] for subsequent work.

This thesis focuses on *logic based* methods for modelling systems. In general, a logic can be viewed as comprising:

- a well-defined *syntax*, (or *language*), which identifies a class of syntactically acceptable objects called the *formulae* of the logic;
- a well-defined *semantics*, (or *model theory*), the purpose of which is to assign each syntactic object of the language a formal *meaning*;
- a well-defined *proof theory* which is, broadly speaking, concerned with the manipulation of formulae.

The advantages of employing logic-based techniques for reasoning about multi-agent systems follow naturally from this definition. First, by fixing on a well-defined, formal, artificial language (as opposed to unstructured, ill defined, natural language), it is possible to investigate the question of *what* can be expressed in a rigorous, mathematical way (see, for example, [48], where a number of temporal logics are compared formally). Another major advantage is that any ambiguity can be removed (see, for example, proofs of the unique readability of propositional logic and first-order predicate logic [50, pp39–43 and pp97–100]).

Transparency is another advantage:

> "By expressing the properties of agents, and multi-agent systems as logical axioms and theorems in a language with clear semantics, the focal points of [the theory] are explicit. The theory is transparent; properties, interrelationships, and inferences are open to examination. This contrasts with the use of computer code, which requires implementational and control aspects within which the issues to be tested can often become confused". [66, p88]

Finally, by adopting a logic-based approach, one makes available all the results and techniques of what is (arguably) the oldest, richest, most fundamental, and best established branch of mathematics.

1.2 Background

If one aims to develop formal methods for modelling and reasoning about multi-agent systems, a good place to start is by observing how the mainstream AI/DAI community has gone about building intelligent (social) agents. Unfortunately, one immediately runs into difficulties, as the issue of "intelligent agent architecture" is the subject of a somewhat heated ongoing debate in AI. The chief protagonists in this debate may be crudely divided into two camps: the classical, symbolic, logicist camp, and the alternative, behavioural camp. Put crudely, the classical approach proposes giving agents symbolic "knowledge", which typically relates to the environment the agent occupies. This knowledge is commonly represented in the form of rules, frames, semantic nets, or, more generally, formulae of some logical language (e.g., first-order predicate logic). Reasoning, and hence, it is hoped, intelligence, is achieved by getting the agent to manipulate the knowledge it has in order to derive new knowledge. This manipulation usually involves theorem proving, or something very like it. The classical approach explains two of the main concerns of symbolic AI research over the past two decades: knowledge representation techniques, and automated theorem proving. The agent architectures described in the closing chapter of [73] are the canonical examples of the classical approach.

Alternatives to the classical approach are based on a diverse range of techniques, with the unifying principle that central, symbolic models are eschewed in favour of a closer relationship between the agent and the environment it occupies. Emphasis is placed on the *situated* nature of intelligence. Examples of the alternative approach are [3], [21], [32], [165].

Interesting and important though this debate is, it is not the aim of this thesis to become embroiled in it (though in the interests of completeness, the issues are briefly reviewed in the next chapter). Despite the intense interest — and controversy — that alternative approaches have evoked, the overwhelming majority of work in DAI lies firmly in the classical camp, and it is on such work that this thesis focuses. Some examples of classical approaches to DAI are AGENT0 [151], Concurrent METATEM [62], MACE [70], MCS/IPEM [42], RATMAN [22] and COSY [23].

How is one to go about reasoning about such systems? What techniques are appropriate, and/or available for the task? There are two obvious resources. The first is the well established tradition in AI/philosophy of devising logics of the mentalistic, *intentional* notions: belief, knowledge, intention, etc. Some notable examples of this work are [87], [98], [124], [100], [179] and [29]. Associated with this work are a set of theories of *speech acts*, for reasoning about communication: some notable examples of this work are [9], [145], [31], [8], [178] and [30].

Both the intentional logics, and the theories of speech acts, identify an agent with an *intentional system* [38], [39]. Crudely, an intentional system is one that is appropriately described in terms of the intentional notions (see [147], [149] for discussions of agency/intentionality).

Researchers have generally developed intentional logics with one of three aims in mind:

1. To investigate mentalistic properties of human activity. A good example is [29]; the aim of this work was to develop a working formal theory of human intention, and was largely motivated by the concerns of philosophy [18]. The work was avowedly *not* intended to be a theory of computational systems, and the authors took some trouble to distance themselves from the suggestion that the logic they developed might be automated in a computational agent [29, p257].

2. To serve as a knowledge representation formalism, perhaps in an automatic planning system. A good example of such work is [100].

3. To enable reasoning about distributed systems (see [81], and comments below).

The prevalent method for defining the semantics of intentional logics has been to give them a *possible worlds* interpretation (the idea was originated by Kripke [103], and proposed for intentional logics by Hintikka [87]). For belief logics, an agent is assigned a set of "possible worlds", or *epistemic alternatives*, each world representing one way the actual world might be, given what the agent believes. Something true in *all* epistemic alternatives could be said to be believed by the agent. Possible worlds semantics have the advantage of a well established, theoretically attractive mathematical foundation, (see, e.g., [28]), but also suffer from several disadvantages. Chief among these is the "logical omniscience" problem, which seems to imply that agents are perfect reasoners. Another problem is that unless the worlds in the semantics are in some way "grounded", (i.e., given some concrete interpretation), then they must remain a theoretical nicety, since it is not clear what worlds might correspond to in an agent (see, e.g., [147] for a discussion of this point). Most logics in category (1), above, adopt ungrounded possible worlds semantics, and are therefore not suited to the purposes of this thesis.

Given that ungrounded possible worlds semantics are not suited to the task of reasoning about classical computational multi-agent systems, one seems to be faced by two options. If one wishes to retain possible worlds semantics, (they are, after all, highly attractive from a theoretical point of view), then one might attempt to find some way of grounding the possible worlds. The second option is to reject possible worlds semantics altogether, and seek an alternative semantic base.

Epistemic logics with grounded possible worlds semantics have recently become the object of study for researchers in mainstream computer science, who found that the notion of knowledge is a valuable one for analyzing distributed systems and protocols (see [81] for an overview). Crudely, this work is based on the idea that a node in a distributed system could be said to "know" something if that thing were true in all the runs of the system that the node found indistinguishable. A similar grounding has been proposed independently by Rosenschein for his "situated automata" paradigm [142], [143]. However, it is still not clear how these approaches are related to the classical model of agents — if at all. So while distributed systems/situated automata models of knowledge are unquestionably a valuable and important research topic, they are only of tangential interest to this thesis.

Some researchers have rejected possible worlds semantics altogether, and looked instead to the possibility of developing an alternative semantics. The best known example of this work is the *deduction model of belief* developed by Kurt Konolige [99], [100]. The deduction model defines a belief system as a tuple containing a "base set" of formulae in some internal, cognitive, logical language of belief, together with a set of *deduction rules* for deriving new beliefs. Konolige argued that an agent with such a belief system could be said to believe something if it was possible to derive that thing from its base beliefs using its deduction rules. Logically incomplete reasoning may be modelled by giving the agent logically incomplete deduction rules. Interestingly, the deduction model can be viewed as an abstract model of the beliefs of classical AI systems. For this reason, the deduction model seems to be the best candidate out of all the intentional logics mentioned so far for the purposes of this thesis.

Let us now return to the question posed earlier: What resources are available for modelling multi-agent systems? The second resource is the Pnuelian tradition of using temporal logics to reason about *reactive systems*[2]:

> "Reactive systems are systems that cannot adequately be described by the *relational* or *functional* view. The relational view regards programs as functions … from an initial state to a terminal state.

[2]There are at least three current usages of the term *reactive system* in computer science. The first, oldest, usage is that by Pnueli and followers (see, e.g., [134], and the description above). Second, researchers in AI planning take a reactive system to be one that is capable of responding dynamically to changes in its environment — here the word "reactive" is taken to be synonomous with "responsive" (see, e.g., [94]). More recently, the term has been used to denote systems which respond directly to the world, rather than reason explicitly about it (see, e.g., [32]). In this thesis the term is used in its Pnuelian sense, except where otherwise indicated.

> Typically, the main role of reactive systems is to maintain an interaction with their environment, and therefore must be described (and specified) in terms of their on-going behaviour … [E]very concurrent system … must be studied by behavioural means. This is because each individual module in a concurrent system is a reactive subsystem, interacting with its own environment which consists of the other modules". [134]

There are good reasons for supposing that multi-agent systems of the type this thesis is interested in modelling are reactive:

- the applications for which a multi-agent approach seems well suited (e.g., distributed sensing [43], manufacturing control [173], air traffic control [24]) are non-terminating, and therefore cannot be described by the functional view;
- multi-agent systems are necessarily concurrent, and as Pnueli observes (above) each agent should therefore be considered a reactive system.

In a landmark 1977 paper, Pnueli proposed the use of temporal logic for reasoning about reactive systems [133]. An enormous amount of research effort has subsequently been devoted to investigating this possibility (see, for example, [134], [135], [47] for good overviews and references). Unfortunately, naive attempts to adapt such techniques seem doomed to failure, as Pnuelian models of concurrency typically deal with the execution of individual program instructions, a grain size too fine for the purposes of this thesis.

1.3 An Approach to Reasoning about Multi-Agent Systems

In order to develop methods for reasoning about multi-agent systems, the work described in this thesis draws upon both of the resources mentioned above. The work consists of three parts. The first part is an abstract *formal model* of computational multi-agent systems; this model aims to capture the key features of "typical" DAI systems. The main components of the model are, not surprisingly, agents:

- agents have a set of explicitly represented beliefs, which are formulae of some internal, logical language of belief;
- agents are able to derive some — though not necessarily all — of the logical consequences of their beliefs;
- agents have computational resources upon which they may draw, by performing private, internal, cognitive actions (for example, a database agent might perform a "retrieve" action);
- agents are able to affect the beliefs of other agents by communicating with them through message passing.

The second part of the work consists of two *execution models* for agents and groups of agents. The role of an execution model may be summarized as follows. During execution, an agent follows a continuous cycle of acting (by performing cognitive actions, and sending messages), updating beliefs, (perhaps as a result of receiving messages), acting, … In doing so, the agent traces out an *execution history* or *run*, which defines what beliefs it held, and what actions it performed at every moment during execution. Taken as a whole, a multi-agent system also traces out a run which defines what each agent's beliefs and actions were at every moment of execution. An *execution model* is a set of rules which define, for every system, the set of "legal" runs of that system. The simplest execution model is a *synchronous* model, where each agent is assumed to act simultaneously. A more complex, but more realistic, alternative model is an *interleaved* execution model, where at most one agent is allowed to act at any one time.

Taken together, the model of multi-agent systems and the associated execution models constitute a *theory* of multi-agent systems. It is important to understand at this stage what this theory is and is not intended to be. It *is* intended to be a plausible formal theory of multi-agent systems, in which each agent is a computational artifact built along classic AI lines. In contrast, the theory is emphatically *not* intended to be a model of *human* social systems[3]. Also, the model is *not* intended to be a *canonical* model of agents and multi-agent systems. *What we have aimed for is an idealized model that, we believe, captures some of the most important aspects of a wide class of agents.* The basic model described in this thesis can be taken as a starting point from which to develop finer grained, more realistic models of real DAI agents and systems.

[3]This will not prevent us from taking inspiration from human social systems, from using analogies with human social systems, or from being mildly anthropomorphic where it improves readability — for example by calling agents "Ralph" and "Freda". However it should be understood that analogies are just that.

	PROPOSITIONAL	FIRST-ORDER
LINEAR TIME	AL [187] (Chapter 5)	QAL (IAL) (Chapter 5)
BRANCHING TIME	BAL (Chapter 7)	QBAL [188] (Chapter 7)

Table 1.1: A Family of Logics for Multi-Agent Sytems

The third part of the work is a family of temporal logics, for reasoning about the theory of multi-agent systems. This family may be divided into two categories: *linear time*, and *branching time*, each of which may be subdivided into two further categories: propositional and first-order (see Table 1.1).

The linear time logics (AL, IAL, QAL) are so-called because they are based on a model of time in which time points are isomorphic with the natural numbers: each time point has just one "successor". The idea which informs the development of the linear time logics is that of allowing a run of a system to act as a *model* for a logic[4]. A relationship is thus established between multi-agent systems and models; that of a model representing a run of a multi-agent system. This relationship is easily formalized, and gives a kind of correspondence theory for the logics.

The branching time logics (BAL, QBAL) are based on a model of time which is viewed as a tree-like structure, branching infinitely into the future from every time point. Such a structure arises if one considers all the possible runs of a system collected together: the branching nature comes from the choices faced by each agent at each time point.

Axiomatizations of each of the logics are presented. These axiomatizations are sound, but are not shown to be complete, as completeness is an issue somewhat beyond the scope of this thesis.

1.4 Structure of the Thesis

The remainder of the thesis is divided into three parts. The first part, (Chapters 2 and 3), is a literature survey. Chapter 2 begins by addressing the question "What is an agent?", particularly in the context of AI, and concludes that an agent is generally taken to be synonomous with an *intentional* system — one which is most simply described in terms of beliefs, desires, etc. The chapter then investigates and critically assesses a number of formalisms developed for reasoning about the intentional notions. Additionally, the main approaches to building agents in AI are reviewed, and the main issues in the classical *versus* behavioural AI debate (mentioned earlier), are discussed.

Chapter 3 addresses the wider question of *social agency*, particularly in the context of DAI. Speech act theory is identified as the dominant paradigm for reasoning about communication in DAI, and various speech act theories are reviewed in detail. The chapter also briefly examines attempts within the DAI community to build multi-agent systems.

Chapters 4, 5, 6, and 7 represent the main contribution of the thesis. Chapter 4 develops a formal model of computational multi-agent systems that, it is argued, captures the main features of a wide range of "classical" DAI systems. Associated with this model are a number of *execution models*, which define how agents can act. Taken together, the model of multi-agent systems and the execution models constitute a *theory* of multi-agent systems. Chapter 5 introduces a family of linear discrete temporal logics for reasoning about systems modelled by the theory developed in Chapter 4. The key idea is that a model for a temporal logic can be shown to "correspond" to a multi-agent system of the type described by the theory. A number of issues associated with the logics are examined. Chapter 6 contains five case studies, which demonstrate how the logics developed in Chapter 5 may

[4] Actually, a model is a run together with some extra technical apparatus to make the logical machinery work.

be used to describe and reason about multi-agent systems. Chapter 7 is devoted to logics based on a *branching* model of time, which contrasts with the linear models used in Chapter 5. It is shown that branching models admit the possibility of reasoning about attributes of agents (such as goals) that cannot be described using linear models.

The final part of the thesis, (Chapter 8), presents some conclusions. Additionally, Appendix A presents a brief review of notation used in the thesis, Appendix B contains technical reviews of formalisms for reasoning about agency that were held over from earlier chapters in the interests of readability, and Appendix C contains an overview of the use of temporal logic for reasoning about Pnuelian reactive systems.

Throughout the thesis, strict formalism has been avoided wherever possible, and mathematical expositions have been eschewed wherever a natural language description would suffice. In order to make the text consistent, it was decided to "standardize" the presentation of language semantics, etc. The net result is that the description of some material may appear to differ from its description in the material's cited source. Such differences are (it is hoped) cosmetic only.

The reader is assumed to be familiar with the fundamentals of logic.

Part II

Background

Chapter 2

Agency and Artificial Intelligence

THE aim of this thesis is to investigate formal methods for modelling and reasoning about multi-agent systems. The building blocks of such systems are *agents*. An obvious point of departure would therefore be a detailed examination of the notion of agency. There are three obvious questions to ask (cf. [147, p1]):

1. What are agents?
2. How are we to formally describe and reason about agents?
3. How are we to construct agents?

This chapter addresses each of these questions in turn. The first question — an obvious pre-requisite for our study — is addressed in the next section, which finds that the commonest interpretation takes an agent to be an "intentional system": one whose behaviour is best explained and predicted in terms of the intentional notions: belief, desire, etc.

Given that the emphasis of this thesis is on methods for formally reasoning about agents and multi-agent systems, it is not surprising that the second question receives most attention. It is dealt with in sections 2.2–2.5, which examine in detail various proposals for the logical modelling of intentional notions. Modal logics with possible worlds semantics are identified as the dominant paradigm in this area, and the advantages and disadvantages of this approach are discussed at length. A number of variations on the possible worlds theme are reviewed, before a discussion of alternatives to possible worlds semantics is presented.

Finally, in section 2.6, the third of the above questions is discussed, and the various AI approaches to building intelligent agents are reviewed. Since the issue of building agents is not central to the purposes of this thesis, the subject is dealt with only briefly.

Note that, for the most part, this chapter deals with isolated agents; aspects of *social* agency are dealt with in the next chapter.

2.1 What is an Agent?

In the past ten years, the term "agent" has been adopted by a variety of sub-disciplines of AI and computer science. One now hears of "agents" in software engineering, data communications and concurrent systems research, as well as robotics, AI and distributed AI. A recent article in a British national daily paper made the following prediction:

> "Agent-based computing (ABC) is likely to be the next significant breakthrough in software development"[1].

A whole programming paradigm has been christened "Agent Oriented Programming"[2]. And yet each of these usages appeals to a quite different notion of agency. It is thus important to examine the question "what is an agent?" very carefully.

A dictionary defines an agent as: "one who, or that which, exerts power or produces an effect"[3]. While this definition is not terribly helpful, it does at least indicate that *action* is somehow involved, and indeed it does seem at first sight that the notion of action is inextricably bound to that of agency:

[1] *The Guardian*, Thursday, 12*th* March, 1992.
[2] See [151]; AOP is also used as a case study in Chapter 6.
[3] *The Concise Oxford Dictionary of Current English (7th edn)*, 1988

"Agents do things, they *act*: that is why they are called agents". [149]

A tacit assumption is that agents take an active role, originating actions that affect their environment, rather than passively allowing their environment to affect them: one often hears, in AI work, of *the agent of an action*. Two words often used to describe agentive action are *autonomy* and *rationality*. Autonomy generally means that an agent operates without direct human (or other) intervention or guidance. Rationality is not so easily tied down, but is often used in the pseudo-game-theoretic sense of an agent maximizing its performance with respect to some "valuation function" (see [66, pp49–54] for a discussion of rationality and agency). A measure of rationality implicitly assumes that an agent has independent goals that it "wants" to achieve.

Unfortunately, autonomous rational action, so defined, is not a sufficient criterion for agenthood, as it admits an unacceptably wide class of objects as agents. For example, it is perfectly consistent to describe a transistor — essentially the simplest form of electronic switch — as an autonomous rational agent by this definition.

Perhaps more troubling for an action-based analysis of agency is that the very notion of action is a slippery one. For example, almost any action can be described in a number of different ways, each seemingly valid. A classic example, due to the philosopher Searle, is that of Gavrilo Princip in 1914: did he pull a trigger, fire a gun, kill Archduke Ferdinand, or start World War I? Each of these seem to be equally valid descriptions of "the same event", and yet trying to isolate that event is notoriously difficult. Trying to describe actions in terms of causal links does not help, as it introduces a seemingly infinite regress. For example, in waving to a friend, I lift my arm, which was caused by muscles contracting, which was caused by some neurons firing, which was caused by…and so on. There is no easy way of halting this regress without appealing to a notion of primitive action, which, as the Gavrilo Princip example above illustrates, is philosophically suspect[4].

An action-based analysis of agency does not look like it is going to work. What other properties of agency might one consider? Shoham has suggested that the term "agent" in AI is often used to denote "high-level" systems, that employ symbolic representations, and perhaps enjoy some "cognitive-like" function, (such as explicit logical, or pseudo-logical reasoning) [151]. This "high-level" condition excludes entities such as actors [2], the neuron-like entities of connectionism [119], and the objects of object-oriented programming. It implies that agents possess significant computational resources (though these resources will, of course, be finite). However, the "high-level" property is a contentious one: a growing number of researchers vigorously argue that "high-level" agents are not the best way to go about AI. The chief protagonist in this debate is Brooks, who has built a number of robotic agents which are certainly not "high-level" by Shoham's definition, and yet are able to perform tasks that are impressive by AI standards (see [19], [21], [20] and the discussion in section 2.6). So a "high-level" condition does not seem to be useful for classifying agents, as it "unfairly" discriminates against objects that do not employ explicit cognitive-like functions.

Perhaps the most widely held view is that an agent is an entity "which appears to be the subject of beliefs, desires, etc." [147, p1]. The philosopher Dennett has coined the term *intentional system* to denote such systems.

2.1.1 Agents as Intentional Systems

When explaining human activity, it is often useful to make statements such as the following:

> Janine took her umbrella because she *believed* it was going to rain.
>
> Michael worked hard because he *wanted* to possess a PhD.

Each of these statements makes use of a *folk psychology*, by which human behaviour is predicted and explained through the attribution of *attitudes*, such as believing and wanting (as in the above examples), hoping, fearing, and so on. This folk psychology is well established: most people reading the above statements would say they found their meaning entirely clear, and would not give them a second glance.

The attitudes employed in such folk psychological descriptions are called the *intentional* notions[5]. The philosopher Daniel Dennett has coined the term *intentional system* to describe entities "whose behaviour can be predicted by the method of attributing belief, desires and rational acumen" [39, p49], [38]. Dennett identifies different "grades" of intentional system:

[4] See [4] for a classic AI attempt to deal with the notion of action, and [149] for an excellent analysis of the relationship between action and agency.

[5] Unfortunately, the word "intention" is used in several different ways in logic and the philosophy of mind. First, there is the everyday usage, as in "I intended to kill him". Second, an intentional notion is one of the attitudes, as above. Finally, in logic, the word intension (with an "s") means the internal content of a concept, as opposed to its extension. In the sequel, intended meaning will always be clear from context.

> "A *first-order* intentional system has beliefs and desires (etc.) but no beliefs and desires *about* beliefs and desires. …A *second-order* intentional system is more sophisticated; it has beliefs and desires (and no doubt other intentional states) about beliefs and desires (and other intentional states) — both those of others and its own". [39, p243]

One can carry on this hierarchy of intentionality as far as required.

An obvious question is whether it is legitimate or useful to attribute beliefs, desires, and so on, to artificial agents. Isn't this just anthropomorphism? McCarthy, among others, has argued that there are occasions when the *intentional stance* is appropriate:

> "To ascribe *beliefs, free will*, *intentions*, *consciousness*, *abilities*, or *wants* to a machine is legitimate when such an ascription expresses the same information about the machine that it expresses about a person. It is useful when the ascription helps us understand the structure of the machine, its past or future behaviour, or how to repair or improve it. It is perhaps never logically required even for humans, but expressing reasonably briefly what is actually known about the state of the machine in a particular situation may require mental qualities or qualities isomorphic to them. Theories of belief, knowledge and wanting can be constructed for machines in a simpler setting than for humans, and later applied to humans. Ascription of mental qualities is most straightforward for machines of known structure such as thermostats and computer operating systems, but is most useful when applied to entities whose structure is incompletely known". [117], (quoted in [151])

What objects can be described by the intentional stance? As it turns out, more or less anything can. In his recent thesis, Seel showed that even very simple, automata-like objects can be consistently ascribed intentional descriptions [147]; similar work by Rosenschein and Kaelbling, (albeit with a different motivation), arrived at a similar conclusion [143]. For example, consider a light switch:

> "It is perfectly coherent to treat a light switch as a (very cooperative) agent with the capability of transmitting current at will, who invariably transmits current when it believes that we want it transmitted and not otherwise; flicking the switch is simply our way of communicating our desires". [151, p6]

And yet most adults would find such a description absurd — perhaps even infantile. Why is this? The answer seems to be that while the intentional stance description is perfectly consistent with the observed behaviour of a light switch, and is internally consistent,

> "…it does not *buy us anything*, since we essentially understand the mechanism sufficiently to have a simpler, mechanistic description of its behaviour". [151, p6]

Put crudely, the more we know about a system, the less we need to rely on animistic, intentional explanations of its behaviour[6]. However, with very complex systems, even if a complete, accurate picture of the system's architecture and working *is* available, a mechanistic, *design stance* explanation of its behaviour may not be practicable. Consider a computer. Although we might have a complete technical description of a computer available, it is hardly practicable to appeal to such a description when explaining why a menu appears when we click a mouse on an icon. In such situations, it may be more appropriate to adopt an intentional stance description, if that description is consistent, and simpler than the alternatives.

Being an intentional system seems to be a *necessary* condition for agenthood, but is it a *sufficient* condition? In his recent Master's thesis, Shardlow trawled through the literature of cognitive science and its component disciplines in an attempt to find a unifying concept that underlies the notion of agenthood. He was forced to the following conclusion:

> "Perhaps there is something more to an agent than its capacity for beliefs and desires, but whatever that thing is, it admits no unified account within cognitive science". [149]

However, given the comments above, it seems reasonable to say that an agent is a system that is most conveniently described by the intentional stance; one whose simplest consistent description requires the intentional stance.

The next step is to investigate methods for reasoning about the intentional notions.

[6] Shoham makes the insightful observation that the move from an intentional stance to a technical description of behaviour correlates well with Piaget's model of child development, and with the scientific development of humankind generally. Children will use animistic explanations of objects — such as light switches — until they grasp the more abstract technical concepts involved. Similarly, the evolution of science has been marked by a gradual move from theological/animistic explanations to mathematical ones. The author's own experiences of teaching computer programming suggest that, when faced with completely unknown phenomena, it is not only children who adopt animistic explanations. Indeed, it often seems easier to teach some computer concepts by using explanations such as: "the computer doesn't know…", than to try to teach abstract principles first.

2.2 Reasoning about Intentional Notions

Suppose one wishes to reason about intentional notions in a logical framework. Consider the following statement (after [73, pp210–211]):

Janine believes Cronos is the father of Zeus. (2.1)

A naive attempt to translate (2.1) into first-order logic might result in the following:

$$Bel(Janine, Father(Zeus, Cronos)) \tag{2.2}$$

Unfortunately, this naive translation does not work, for at least two reasons. The first is syntactic: the second argument to the *Bel* predicate is a *formula* of first-order logic, and is not, therefore a term. So (2.2) is not a well-formed formula of classical first-order logic. The second problem is semantic, and is more serious. The constants *Zeus* and *Jupiter*, by any reasonable interpretation, denote the same individual: the supreme deity of the classical world. It is therefore acceptable to write, in first-order logic:

$$(Zeus = Jupiter). \tag{2.3}$$

Given (2.2) and (2.3), the standard rules of first-order logic would allow the derivation of the following:

$$Bel(Janine, Father(Jupiter, Cronos)) \tag{2.4}$$

But intuition rejects this derivation as invalid: believing that the father of Zeus is Cronos is *not* the same as believing that the father of Jupiter is Cronos.

So what is the problem? Why does first-order logic fail here? The problem is that the intentional notions — such as belief and desire — are *referentially opaque*, in that they set up *opaque contexts*, in which the standard substitution rules of first-order logic do not apply. In classical (propositional or first-order) logic, the denotation, or semantic value, of an expression is dependent solely on the denotations of its sub-expressions. For example, the denotation of the propositional logic formula $p \wedge q$ is a function of the truth-values of p and q. The operators of classical logic are thus said to be *truth functional*.

In contrast, intentional notions such as belief are *not* truth functional. It is surely not the case that the truth value of the sentence:

Janine believes p (2.5)

is dependent solely on the truth-value of p[7]. So substituting equivalents into opaque contexts is not going to preserve meaning. This is what is meant by referential opacity. The existence of referentially opaque contexts has been known since the time of Frege. He suggested a distinction between *sense* and *reference.* In ordinary formulae, the "reference" of a term/formula (i.e., its denotation) is needed, whereas in opaque contexts, the "sense" of a formula is needed (see also [147, p3]).

Clearly, classical logics are not suitable in their standard form for reasoning about intentional notions: alternative formalisms are required. A vast enterprise has sprung up devoted to developing such formalisms.

The field of formal methods for reasoning about intentional notions is widely reckoned to have begun with the publication, in 1962, of Jaakko Hintikka's book *Knowledge and Belief: An Introduction to the Logic of the Two Notions* [87]. At that time, the subject was considered fairly esoteric, of interest to comparatively few researchers in logic and the philosophy of mind. Since then, however, it has become an important research area in its own right, with contributions from researchers in AI, formal philosophy, linguistics and economics. There is now an enormous literature on the subject, and with a major biannual international conference devoted solely to theoretical aspects of reasoning about knowledge, as well as the input from numerous other, less specialized conferences, this literature is growing ever larger.

Despite the diversity of interests and applications, the number of basic techniques in use is quite small. Recall, from the discussion above, that there are two problems to be addressed in developing a logical formalism for intentional notions: a syntactic one, and a semantic one. It follows that any formalism can be characterized in terms of two independent attributes: its *language of formulation*, and *semantic model* [100, p83].

There are two fundamental approaches to the syntactic problem. The first is to use a *modal* language, which contains non-truth-functional *modal operators*, which are applied to formulae. An alternative approach involves the use of a *meta-language*: a many-sorted first-order language containing terms which denote formulae of some other *object-language*. Intentional notions can be represented using a meta-language predicate, and given

[7] Note, however, that the sentence (2.5) is itself a proposition, in that its denotation is the value true or false.

	MODAL LANGUAGE	META-LANGUAGE
POSSIBLE WORLDS	ungrounded [87] [52] [29] [83] grounded [142] [143] [58] [105] [53]	[124]
OTHER	[100] [153]	[121] [98] [125] [35]

Table 2.1: Some Intentional Logics

whatever axiomatization is deemed appropriate. Both of these approaches have their advantages and disadvantages, and will be discussed at length in the sequel.

As with the syntactic problem, there are two basic approaches to the semantic problem. The first, best known, and probably most widely used approach is to adopt a *possible worlds* semantics, where an agent's beliefs, knowledge, goals, etc. are characterized as a set of so-called *possible worlds*, with an *accessibility relation* holding between them. Possible worlds semantics have an associated *correspondence theory* which makes them an attractive mathematical tool to work with [28]. However, they also have many associated difficulties, notably the well-known logical omniscience problem, which implies that agents are perfect reasoners. A number of minor variations on the possible-worlds theme have been proposed, in an attempt to retain the correspondence theory, but without logical omniscience.

The commonest alternative to the possible worlds model for belief is to use a *sentential*, or *interpreted symbolic structures* approach. In this scheme, beliefs are viewed as symbolic formulae explicitly represented in a data structure associated with an agent. An agent then believes ϕ if ϕ is present in the agent's belief structure. Despite its simplicity, the sentential model works well under certain circumstances [100].

Table 2.1 characterizes a number of well known intentional logics in terms of their syntactic and semantic properties (after [100, p85]). The next part of this chapter contains detailed reviews of some of these formalisms. First, the idea of possible worlds semantics is discussed, and then a detailed analysis of normal modal logics is presented, along with some variants on the possible worlds theme. Next, some meta-language approaches are discussed, and one hybrid formalism is described. Finally, some alternative formalisms are described.

Before the detailed presentations, a note on terminology. Strictly speaking, an *epistemic logic* is a logic of knowledge, a *doxastic logic* is a logic of belief, and a *conative logic* is a logic of desires or goals. However, it is common practice to use "epistemic" as a blanket term for logics of knowledge and belief. This practice is adopted in this thesis; a distinction is only made where it is considered significant. Also, the reviews focus on knowledge/belief to the virtual exclusion of goals/desires; this is because most of the principles are the same, and little work has addressed the issue of goals (but see the comments on Cohen and Levesque's formalism, below).

2.3 Possible Worlds Semantics

The possible worlds model for epistemic logics was originally proposed by Hintikka ([87]), and is now most commonly formulated in a normal modal logic using the techniques developed by Kripke ([103])[8].

Hintikka's insight was to see that an agent's beliefs could be characterized in terms of a set of *possible worlds*, in the following way. Consider an agent playing the card game Gin Rummy[9]. In this game, the more one knows about the cards possessed by one's opponents, the better one is able to play. And yet complete knowledge of an opponent's cards is generally impossible, (if one excludes cheating). The ability to play Gin Rummy well thus depends, at least in part, on the ability to deduce what cards are held by an opponent, given the limited information available. Now suppose our agent possessed the ace of spades. Assuming the agent's sensory equipment was functioning normally, it would be rational of her to believe that she possessed this card. Now suppose she were to try to deduce what cards were held by her opponents. This could be done by first calculating all the various different ways that the cards in the pack could possibly have been distributed among the various players. (This is not being proposed as an actual card playing strategy, but for illustration!) For argument's sake, suppose that each possible configuration is described on a separate piece of paper. Once the process was complete, our agent can then begin to systematically eliminate from this large pile of paper all those configurations which are *not possible, given what she knows*. For example, any configuration in which she did not possess the ace of spades could be rejected immediately as impossible. Call each piece of paper remaining after this process a *world*. Each world represents one state of affairs considered possible, given what she knows. Hintikka coined the term *epistemic alternatives* to describe the worlds possible given one's beliefs. Something true in *all* our agent's epistemic alternatives could be said to be believed by the agent. For example, it will be true in all our agent's epistemic alternatives that she has the ace of spades.

On a first reading, this technique seems a peculiarly roundabout way of characterizing belief, but it has two advantages. First, it remains neutral on the subject of the cognitive structure of agents. It certainly doesn't posit any internalized collection of possible worlds. It is just a convenient way of characterizing belief. Second, the mathematical theory associated with the formalization of possible worlds is extremely appealing (see below).

The next step is to show how possible worlds may be incorporated into the semantic framework of a logic. This is the subject of the next section.

2.3.1 Normal Modal Logics

Epistemic logics are usually formulated as *normal modal logics* using the semantics developed by Kripke [103]. Before moving on to explicitly epistemic logics, this section describes normal modal logics in general.

Modal logics were originally developed by philosophers interested in the distinction between *necessary* truths and mere *contingent* truths. Intuitively, a necessary truth is something that is true *because it could not have been otherwise*, whereas a contingent truth is something that could, plausibly have been otherwise. For example, it is a fact that as I write, the Conservative Party of Great Britain hold a majority of twenty-one seats in the House of Commons. But although this is true, it is *not* a necessary truth; it could quite easily have turned out that the Labour Party won a majority at the last general election. This fact is thus only a contingent truth.

Contrast this with the following statement: *the square root of 2 is not a rational number*. There seems no earthly way that this could be anything *but* true, (given the standard reading of the sentence). This latter fact is an example of a necessary truth. Necessary truth is usually defined as something true in *all possible worlds*. It is actually quite difficult to think of any necessary truths other than mathematical laws.

To illustrate the principles of modal epistemic logics, a normal propositional modal logic is defined.

Syntax and Semantics

This logic is essentially classical propositional logic, extended by the addition of two operators: "$\Box$" (necessarily), and "$\Diamond$" (possibly). First, its syntax.

Definition 1 *Let Prop* $= \{p, q, \ldots\}$ *be a countable set of atomic propositions. The syntax of normal propositional modal logic is defined by the following rules:*

1. *If* $p \in$ *Prop then* p *is a formula.*

[8] In Hintikka's original work, he used a technique based on "model sets", which is equivalent to Kripke's formalism, though less elegant. See [88, Appendix Five, pp351–352] for a comparison and discussion of the two techniques.

[9] This example was adapted from [81].

$\langle M, w\rangle$	$\models$	true	
$\langle M, w\rangle$	$\models$	p	where $p \in \mathit{Prop}$, iff $p \in \pi(w)$
$\langle M, w\rangle$	$\models$	$\neg\phi$	iff $\langle M, w\rangle \not\models \phi$
$\langle M, w\rangle$	$\models$	$\phi \vee \psi$	iff $\langle M, w\rangle \models \phi$ or $\langle M, w\rangle \models \psi$
$\langle M, w\rangle$	$\models$	$\Box\phi$	iff $\forall w' \in W \cdot$ if $(w, w') \in R$ then $\langle M, w'\rangle \models \phi$
$\langle M, w\rangle$	$\models$	$\Diamond\phi$	iff $\exists w' \in W \cdot (w, w') \in R$ and $\langle M, w'\rangle \models \phi$

Figure 2.1: The Semantics of Normal Modal Logic

2. *If ϕ, ψ are formulae, then so are:*
   true $\quad \neg\phi \quad \phi \vee \psi$

3. *If ϕ is a formula then so are:*
   $\Box\phi \quad \Diamond\phi$

The operators "$\neg$" (not) and "$\vee$" (or) have their standard meaning; true is a logical constant, (sometimes called *verum*), that is always true. The remaining connectives of propositional logic can be defined as abbreviations in the usual way. The formula $\Box\phi$ is read: "necessarily ϕ", and the formula $\Diamond\phi$ is read: "possibly ϕ". Now to the semantics of the language.

Normal modal logics are concerned with truth at worlds; models for such logics therefore contain a set of worlds, W, and a binary relation, R, on W, saying which worlds are considered possible relative to other worlds. Additionally, a valuation function π is required, saying what propositions are true at each world.

Definition 2 *A model for a normal propositional modal logic is a triple $\langle W, R, \pi\rangle$, where W is a non-empty set of worlds, $R \subseteq W \times W$, and*

$$\pi\colon W \to \textit{powerset Prop}$$

is a valuation function, which says for each world $w \in W$ which atomic propositions are true in w. An alternative, equivalent technique would have been to define π as follows:

$$\pi\colon W \times \mathit{Prop} \to \{\mathsf{true}, \mathsf{false}\}$$

though the rules defining the semantics of the language would then have to be changed slightly.

The semantics of the language are given via the satisfaction relation, "$\models$", which holds between pairs of the form $\langle M, w\rangle$, (where M is a model, and w is a reference world), and formulae of the language. The semantic rules defining this relation are given in Figure 2.1. The definition of satisfaction for atomic propositions thus captures the idea of truth in the "current" world, (which appears on the left of "$\models$"). The semantic rules for "true", "$\neg$", and "$\vee$", are standard. The rule for "$\Box$" captures the idea of truth in all accessible worlds, and the rule for "$\Diamond$" captures the idea of truth in at least one possible world.

Note that the two modal operators are *duals* of each other, in the sense that the universal and existential quantifiers of first-order logic are duals:

$$\Box\phi \;\Leftrightarrow\; \neg\Diamond\neg\phi.$$

It would thus have been possible to take either one as primitive, and introduce the other as a derived operator.

Correspondence Theory

To understand the extraordinary properties of this simple logic, it is first necessary to introduce *validity* and *satisfiability*. A formula is *satisfiable* if it is satisfied for some model/world pair, and *unsatisfiable* otherwise. A formula is *true in a model* if it is satisfied for every world in the model, and *valid in a class of models* if it true in every model in the class. Finally, a formula is valid *simpliciter* if it is true in the class of all models. If ϕ is valid, we write $\models \phi$.

The two basic properties of this logic are as follows. First, the following axiom schema is valid.

$$\models \Box(\phi \;\Rightarrow\; \psi) \;\Rightarrow\; (\Box\phi \;\Rightarrow\; \Box\psi)$$

NAME	AXIOM	CONDITION ON R	FIRST-ORDER CHARACTERIZATION
T	$\Box\phi \Rightarrow \phi$	Reflexive	$\forall w \in W \cdot (w, w) \in R$
D	$\Box\phi \Rightarrow \Diamond\phi$	Serial	$\forall w \in W \cdot \exists w' \in W \cdot (w, w') \in R$
4	$\Box\phi \Rightarrow \Box\Box\phi$	Transitive	$\forall w, w', w'' \in W \cdot (w, w') \in R \wedge (w', w'') \in R \Rightarrow (w, w'') \in R$
5	$\Diamond\phi \Rightarrow \Box\Diamond\phi$	Euclidean	$\forall w, w', w'' \in W \cdot (w, w') \in R \wedge (w, w'') \in R \Rightarrow (w', w'') \in R$

Table 2.2: Some Correspondence Theory

This axiom is called *K*, in honour of Kripke. The second property is as follows.

If $\models \phi$ then $\models \Box\phi$

Proofs of these properties are trivial, and are left as an exercise for the reader. Now, since *K* is valid, it will be a theorem of any complete axiomatization of normal modal logic. Similarly, the second property will appear as a rule of inference in any axiomatization of normal modal logic; it is generally called the *necessitation* rule. These two properties turn out to be the most problematic features of normal modal logics when they are used as logics of knowledge/belief (this point will be examined later).

The most intriguing properties of normal modal logics follow from the properties of the accessibility relation, *R*, in models. To illustrate these properties, consider the following axiom schema.

$\Box\phi \Rightarrow \phi$

It turns out that this axiom is *characteristic* of the class of models with a *reflexive* accessibility relation. (By characteristic, we mean that it is true in all and only those models in the class.) There are a host of axioms which correspond to certain properties of *R*: the study of the way that properties of *R* correspond to axioms is called *correspondence theory*. In Table 2.2, we list some axioms along with their characteristic property on *R*, and a first-order formula describing the property. Note that the table only lists those axioms of specific interest to this thesis; (see [28] for others). The names of axioms follow historical tradition.

The results of correspondence theory make it straightforward to derive completeness results for a range of simple normal modal logics. These results provide a useful point of comparison for normal modal logics, and account in a large part for the popularity of this style of semantics.

A *system of logic* can be thought of as a set of formulae valid in some class of models; a member of the set is called a *theorem* of the logic (if ϕ is a theorem, this is usually denoted by $\vdash \phi$). The notation $K\Sigma_1 \ldots \Sigma_n$ is often used to denote the smallest normal modal logic containing axioms $\Sigma_1, \ldots, \Sigma_n$ (recall that any normal modal logic will contain *K*; cf. [78, p25]).

For the axioms *T*, *D*, *4*, and *5*, it would seem that there ought to be sixteen distinct systems of logic (since $2^4 = 16$). However, some of these systems turn out to be equivalent (in that they contain the same theorems), and as a result there are only eleven distinct systems. The relationships between these systems are described in Figure 2.2 (after [100, p99], and [28, p132]). In this diagram, an arc from *A* to *B* means that *B* is a strict superset of *A*: every theorem of *A* is a theorem of *B*, but not *vice versa*; $A = B$ means that *A* and *B* contain precisely the same theorems. Because some modal systems are so widely used, they have been given names:

KT	is known as	T
KT4	is known as	S4
KD45	is known as	weak-S5
KT5	is known as	S5

Normal Modal Logics as Epistemic Logics

To use the logic developed above as an epistemic logic, the formula $\Box\phi$ is read as: "it is known that ϕ". The worlds in the model are interpreted as epistemic alternatives, the accessibility relation defines what the alternatives

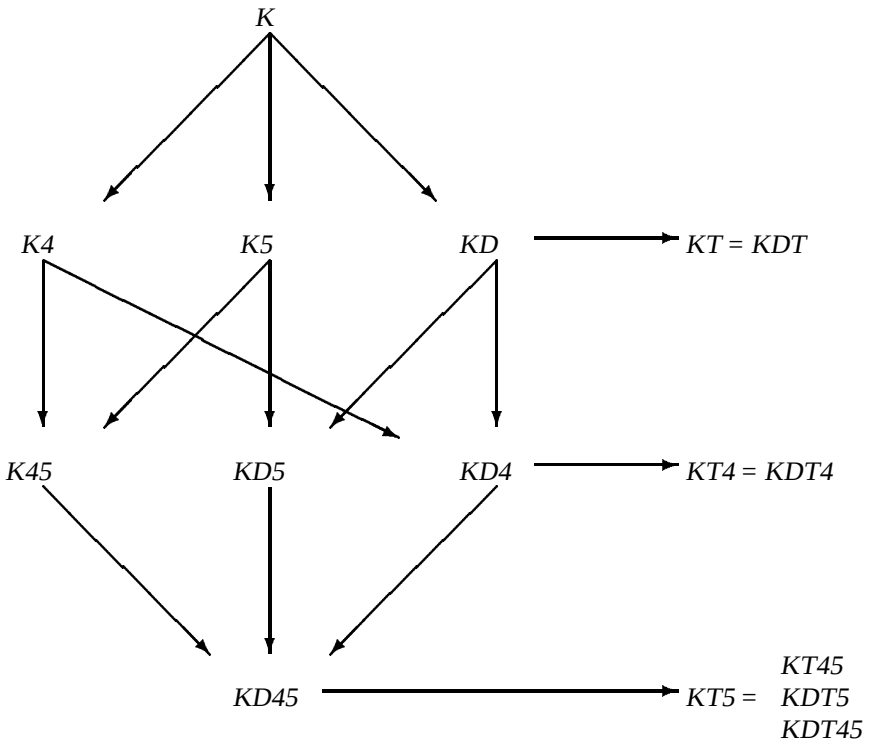


Figure 2.2: The Modal Systems Based on Axioms *T*, *D*, *4* and *5*

are from any given world. The logic deals with the knowledge of a single agent. To deal with multi-agent knowledge, one adds to a model structure an indexed set of accessibility relations, one for each agent. A model is then a structure:

$$\langle W, R_1, \ldots, R_n, \pi \rangle$$

where R_i is the knowledge accessibility relation of agent i. The simple language defined above is extended by replacing the single modal operator "$\Box$" by an indexed set of unary modal operators $\{\mathsf{K}_i\}$, where $i \in \{1, \ldots, n\}$. The formula $\mathsf{K}_i\phi$ is read: "i knows that ϕ". The semantic rule for "$\Box$" is replaced by the following rule:

$$\langle M, w \rangle \models \mathsf{K}_i\phi \quad \text{iff } \forall w' \in W \cdot \text{ if } (w, w') \in R_i \text{ then } \langle M, w' \rangle \models \phi$$

Each operator K_i thus has exactly the same properties as "$\Box$". Corresponding to each of the modal systems Σ, above, a corresponding system Σ_n is defined, for the multi-agent logic. Thus K_n is the smallest multi-agent epistemic logic and $\mathrm{S5}_n$ is the largest.

The next step is to consider how well normal modal logic serves as a logic of knowledge/belief. Consider first the necessitation rule and axiom *K*, since any normal modal system is committed to these.

The necessitation rule tells us that an agent knows all valid formulae. Amongst other things, this means an agent knows all propositional tautologies. Since there are an infinite number of these, an agent will have an infinite number of items of knowledge: immediately, one is faced with a counter intuitive property of the knowledge operator.

Now consider the axiom *K*, which says that an agent's knowledge is closed under implication. Suppose ϕ is a logical consequence of the set $\Phi = \{\phi_1, \ldots, \phi_n\}$, then in every world where all of Φ are true, ϕ must also be true, and hence

$$\phi_1 \wedge \cdots \wedge \phi_n \Rightarrow \phi$$

must be valid. By necessitation, this formula will also be believed. Since an agent's beliefs are closed under implication, whenever it believes each of Φ, it must also believe ϕ. Hence an agent's knowledge is closed under logical consequence. This also seems counter intuitive. For example, suppose, like every good logician,

our agent knows Peano's axioms. It may well be that Fermat's last theorem follows from Peano's axioms — although, despite strenuous efforts, nobody has so far managed to prove it. Yet if our agent's beliefs are closed under logical consequence, then our agent must know it. So consequential closure, implied by necessitation and the K axiom, seems an overstrong property for resource bounded reasoners.

The Logical Omniscience Problem

These two problems — that of knowing all valid formulae, and that of knowledge/belief being closed under logical consequence — together constitute the famous *logical omniscience* problem. This problem has some damaging corollaries.

The first concerns consistency. Human believers are rarely consistent in the logical sense of the word; they will often have beliefs ϕ and ψ, where $\phi \vdash \neg \psi$, without being aware of the implicit inconsistency. However, the ideal reasoners implied by possible worlds semantics cannot have such inconsistent beliefs without believing *every* formula of the logical language (because the consequential closure of an inconsistent set of formulae is the set of all formulae). Konolige has argued that logical consistency is much too strong a property for resource bounded reasoners: he argues that a lesser property, that of being *non-contradictory* is the most one can reasonably demand [100]. Non-contradiction means that an agent would not simultaneously believe ϕ and $\neg \phi$, although the agent might have logically inconsistent beliefs.

The second corollary is more subtle. Consider the following propositions (this example is from [100, p88]):

1. Hamlet's favourite colour is black.
2. Hamlet's favourite colour is black *and* every planar map can be four coloured.

The second conjunct of (2) is valid, and will thus be believed. This means that (1) and (2) are logically equivalent; (2) is true just when (1) is. Since agents are ideal reasoners, they will believe that the two propositions are logically equivalent. This is yet another counter intuitive property implied by possible worlds semantics, as: "equivalent propositions are *not* equivalent as beliefs" [100, p88]. Yet this is just what possible worlds semantics implies. It has been suggested that propositions are thus too *coarse grained* to serve as the objects of belief in this way.

The logical omniscience problem is a serious one. In the words of Levesque:

> "Any one of these [problems] might cause one to reject a possible-world formalization as unintuitive at best and completely unrealistic at worst". [112]

Axioms for Knowledge and Belief

We now consider the appropriateness of the axioms D_n, T_n, 4_n, and 5_n for logics of knowledge/belief.

The axiom D_n says that an agent's beliefs are non-contradictory; it can be re-written in the following form:

$$\mathsf{K}_i\phi \implies \neg \mathsf{K}_i \neg \phi$$

which is read: "if i knows ϕ, then i doesn't know $\neg \phi$". This axiom seems a reasonable property of knowledge/belief.

The axiom T_n is often called the *knowledge* axiom, since it says that what is known is true. It is usually accepted as the axiom that distinguishes knowledge from belief: it seems reasonable that one could believe something that is false, but one would hesitate to say that one could *know* something false. Knowledge is thus often defined as true belief: i knows ϕ if i believes ϕ and ϕ is true. So defined, knowledge satisfies T_n.

Axiom 4_n is called the *positive introspection axiom*. Introspection is the process of examining one's own beliefs, and is discussed in detail in [100, Chapter 5]. The positive introspection axiom says that an agent knows what it knows. Similarly, axiom 5_n is the *negative introspection axiom*, which says that an agent is aware of what it doesn't know. Positive and negative introspection together imply an agent has perfect knowledge about what it does and doesn't know (cf. [100, Equation (5.11), p79]). Whether or not the two types of introspection are appropriate properties for knowledge/belief is the subject of some debate. However, it is generally accepted that positive introspection is a less demanding property than negative introspection, and is thus a more reasonable property for resource bounded reasoners.

Given the comments above, the modal system $\mathrm{S5}_n$ is often chosen as a logic of *knowledge*, and weak-$\mathrm{S5}_n$ is often chosen as a logic of *belief*.

Computational Aspects

Before leaving this basic logic, it is worth commenting on its computational/proof theoretic properties. Halpern and Moses have established the following ([83]):

1. The provability problem for each of the key systems K_n, T_n, $S4_n$, weak-$S5_n$, and $S5_n$ *is* decidable. Halpern and Moses sketch some tableaux decision procedures for these logics.
2. The satisfiability and validity problems for K_n, T_n, $S4_n$, (where $n \geq 1$), and $S5_n$, weak-$S5_n$ (where $n \geq 2$) are PSPACE complete.

The first result is encouraging, as it holds out at least some hope of automation. Unfortunately, the second result is extremely discouraging: in simple terms, it means that in the worst case, automation of these logics is not a practical proposition.

Discussion

To sum up, the basic possible worlds approach described above has the following disadvantages as a multi-agent epistemic logic:

- agents believe all valid formulae;
- agents beliefs are closed under logical consequence;
- equivalent propositions are identical beliefs;
- if agents are inconsistent, then they believe everything;
- in the worst case, automation is not feasible.

To which many people would add the following:

- "[T]he ontology of possible worlds and accessibility relations …is frankly mysterious to most practically minded people, and in particular has nothing to say about agent architecture". [147]

Despite these serious disadvantages, possible worlds are still the semantics of choice for many researchers, and a number of variations on the basic possible worlds theme have been proposed to get around some of the difficulties. The following sections examine various topics associated with possible worlds semantics.

2.3.2 Common and Distributed Knowledge

In addition to reasoning about what one agent knows or believes, it is often useful to be able to reason about "cultural" knowledge: the things that everyone knows, and that everyone knows that everyone knows, etc. This kind of knowledge is called *common knowledge*. The famous "wisest man" puzzle — a classic problem in epistemic reasoning — is an example of the kind of problem that is efficiently dealt with via reasoning about common knowledge (see, e.g., [100, p58] for a statement of the wisest man problem)[10].

The starting point for common knowledge is to develop an operator for things that "everyone knows". A unary modal operator EK is added to the modal language discussed above; the formulae $\mathsf{EK}\phi$ is read: "everyone knows ϕ". It can be defined as an abbreviation:

$$\mathsf{EK}\phi \triangleq \mathsf{K}_1\phi \wedge \cdots \wedge \mathsf{K}_n\phi$$

or it can be given its own semantic rule:

$$\langle M, w\rangle \models \mathsf{EK}\phi \quad \text{iff } \langle M, w\rangle \models \mathsf{K}_i\phi \text{ for all } i \in \{1, \ldots, n\}$$

The EK operator does not satisfactorily capture the idea of common knowledge. For this, another derived operator CK is required; CK is defined, ultimately, in terms of EK. It is first necessary to introduce the derived operator EK^k; the formula $\mathsf{EK}^k\phi$ is read: "everyone knows ϕ to degree k". It is defined as follows:

$$\begin{array}{lcl} \mathsf{EK}^1\phi & \triangleq & \mathsf{EK}\phi \\ \mathsf{EK}^{k+1}\phi & \triangleq & \mathsf{EK}(\mathsf{EK}^k\phi) \end{array}$$

[10] The discussion that follows was adapted and expanded from [73, Chapter 9] and [83].

The common knowledge operator can then be defined as an abbreviation:

$$\mathsf{CK}\phi \triangleq \mathsf{EK}\phi \wedge \mathsf{EK}^2\phi \wedge \cdots \wedge \mathsf{EK}^k\phi \wedge \cdots$$

or it can be given its own semantic rule:

$$\langle M, w\rangle \models \mathsf{CK}\phi \quad \text{iff } \langle M, w\rangle \models \mathsf{EK}^k\phi \text{ for all } k \in \mathbb{N}_1$$

A classic problem in distributed systems folklore is the *coordinated attack problem*, where two generals on the same side wish to attack an enemy, but are unwilling to do so unless they are sure the other will also attack, i.e., it is common knowledge that they will attack. Halpern has shown that under certain circumstances, (notably, where communication is not guaranteed), common knowledge cannot be achieved [82].

A related issue to common knowledge is that of distributed, or implicit knowledge. Suppose there is an omniscient observer of some group of agents, with the ability to "read" each agent's beliefs/knowledge. Then this agent would be able to pool the collective knowledge of the group of agents, and would generally be able to deduce more than any one agent in the group. For example, suppose, in a group of two agents, agent 1 only knew ϕ, and agent 2 only knew $\phi \Rightarrow \psi$. Then there would be *distributed* knowledge of ψ, even though no agent explicitly knew ψ. Distributed knowledge cannot be reduced to any of the operators introduced so far: it must be given its own definition. The distributed knowledge operator DK has the following semantic rule:

$$\langle M, w\rangle \models \mathsf{DK}\phi \quad \text{iff} \quad \langle M, w'\rangle \models \phi \text{ for all } w' \text{ such that } (w, w') \in (R_1 \cap \cdots \cap R_n)$$

This rule might seem strange at first, since it uses set intersection rather than set union, which is at odds with a naive perception of how distributed knowledge works. However, a *restriction* on possible worlds generally means an *increase* in knowledge.

Distributed knowledge is potentially a useful concept in cooperative problem solving systems, where knowledge about a problem is distributed among a group of problem solving agents, which must try to deduce a solution through cooperative interaction.

Proof theoretically, the various group knowledge operators have straightforward properties. Note that they form a hierarchy:

$$\mathsf{CK}\phi \Rightarrow \cdots \Rightarrow \mathsf{EK}^k\phi \Rightarrow \cdots \Rightarrow \mathsf{EK}\phi \Rightarrow \mathsf{DK}\phi$$

See [83] for further discussion of these operators and their properties.

2.3.3 Quantified Epistemic Logics

A natural extension to the basic logic described above would be to allow quantification. In this section, we proceed, (as with the propositional modal logic described above), to define a quantified modal logic and then examine its suitability as a logic for reasoning about knowledge/belief. The discussion below is adapted and expanded from [88, Part Two], [138], and [100].

The syntax of quantified modal logic is that of classical first-order logic enriched by the addition of the unary modal operators "$\Diamond$" and "$\Box$", with the same meanings as above. These operators can be applied to arbitrary formulae of the modal or first-order language. So, for example, the following are syntactically acceptable formulae of quantified modal logic:

$$\forall x \cdot \Box P(x) \Rightarrow Q(x)$$
$$P(a) \wedge \Box \exists x \cdot P(x)$$

It is assumed that constants of the basic first-order language belong to a set *Const*, variables to a set *Var*, and predicate symbols to a set *Pred*. A model for quantified modal logic is a structure:

$$\langle W, R, D, I, \pi\rangle$$

where W and R are a set of worlds and a binary relation on W (as before), and:

- D is a non-empty set, called the *domain*;
- I maps *Const* to D;
- π maps $Pred \times W$ to powerset D^n, (where n is the arity of the predicate symbol, and D^n is the set of n-tuples over D); π gives the extension of each predicate symbol in each world.

As usual with a predicate logic, formulae can only be interpreted with respect to a variable assignment: a variable assignment V maps *Var* to D. A *term* is either a variable or a constant. It is useful to define a function $[\![\_]\!]_{I,V}$, which takes an arbitrary term and returns its denotation relative to I, V:

$$[\![\theta]\!]_{I,V} \triangleq \begin{array}{l} \textit{if } \theta \in \textit{Const} \\ \textit{then } I(\theta) \\ \textit{else } V(\theta) \end{array}$$

Reference to I and V is suppressed where these are understood.

The only semantic rule of interest is that for predicates; the remainder are essentially unchanged, (except that a formula must now be interpreted with respect to a variable assignment).

$$\langle M, V, w \rangle \models P(\theta_1, \ldots, \theta_n) \quad \text{iff } \langle [\![\theta_1]\!], \ldots, [\![\theta_n]\!] \rangle \in \pi(P, w)$$

So, despite the added complexity of predicates and terms, the model theory for quantified modal logics remains straightforward.

Now consider such a system as an epistemic logic. The semantics can easily be extended to the multi-agent case, as in propositional modal logic, by the addition of multiple accessibility relations, etc. One unfortunate property of quantified modal logics, as defined above, is that they make the *Barcan formulae* valid. The Barcan formula is:

$$\forall x \cdot \Box \phi(x) \Rightarrow \Box \forall x \cdot \phi(x)$$

and the converse Barcan formula is the reverse implication, i.e.:

$$\Box \forall x \cdot \phi(x) \Rightarrow \forall x \cdot \Box \phi(x)$$

(see e.g., [88, pp142–145] for a discussion).

If "$\Box$" is interpreted as "it is known that …", then the Barcan formula means that if it is known of all individuals independently that they have property ϕ, then it is known that all individuals have this property. The Barcan formula thus implies that knowers are somehow aware of the existence of all individuals, which seems to be an overstrong demand to make of resource bounded reasoners.

The converse Barcan formula says that if an agent knows that all individuals have property ϕ, then it will be known that each individual has property ϕ. In other words, it implies that agents will perform universal instantiation wherever possible. This is a weaker property than that implied by the Barcan formula, but still seems an overstrong demand to make of resource bounded reasoners.

If one refers back to model structures for quantified modal logics for a moment, it will be found that the domain of individuals D is fixed, for all worlds: this is called the *constant domain assumption*. If this assumption is dropped, so that the domain is indexed by worlds, and quantification is restricted to just those individuals in the domain of the reference world, then the Barcan formulae are no longer valid. As to whether this is a reasonable property for an epistemic logic, consider a semi-mythological figure such as Robin Hood, or King Arthur. An agent might reasonably be unsure as to whether these individuals ever actually existed, and so they might appear in the domain of some epistemic alternatives, but not all.

It is also worth noting that in the model structure above, constants were given a *rigid* interpretation: a constant has the same denotation in all worlds. What would it mean for this assumption to be dropped, and what would be the effect? It would be possible to drop the assumption by simply making I a map from $\textit{Const} \times W$ to D. Consider the following formula, which is intended to express the fact that an agent believes of the individual that is Prime Minister of Britain in the actual world, (i.e., the world as perceived by the author at the time of writing: late April 1992), that he is a cricket fan.

$$\Box \textit{Cricket\_Fan}(\textit{PM})$$

Here, *PM* is a constant which in the actual world denotes the individual John Major. But suppose our agent had been on a desert island since November 1990, and was somehow unaware that Margaret Thatcher was no longer Prime Minister. For this agent, the constant *PM* would denote someone entirely different to that which it denotes for us, and it is unlikely that a rational agent would believe of the individual Margaret Thatcher that she was a cricket fan. Note that the agent could still believe that the individual John Major was a cricket fan, but in its ignorance, it would use a different constant for this individual. The point is that the constant *PM* has a different denotation for us and the agent, and, more generally, a different denotation in different worlds. Such constants are said to be *fluent* expressions; non-fluent expressions are called *rigid designators*.

To deal with fluent expressions, Konolige has proposed the technical device of the *bullet operator*, ("•"). Suppose a is a constant, then $\bullet a$ is an expression returning a constant that always denotes what a denotes for us. The use of the bullet operator is somewhat complex, and is described in detail in [100, pp38–42 and pp100–104].

One difficult issue in the philosophy of modal logics is that of *quantifying-in* to modal contexts. Consider the following English sentence:

> It is known that there is a unicorn. (2.6)

Now consider the following quantified modal logic formulae, each representing one attempt to formalize (2.6):

$$\exists x \cdot \Box \mathit{Unicorn}(x) \tag{2.7}$$

$$\Box \exists x \cdot \mathit{Unicorn}(x) \tag{2.8}$$

In (2.7), a *de re* reading of (2.6) is presented. The formula implies the existence of some particular individual, which the believer has a name for, which the agent believes has the property of being a Unicorn. Note that *we* do not know who this individual is, or have a name for it; we simply represent the fact that the agent does.

In (2.8), a *de dicto* reading of (2.6) is presented. This reading expresses a weaker property than (2.7): namely, that the agent believes *some* individual is a Unicorn. The agent does not necessarily have a name for the individual. Formulae (2.7) and (2.8) thus express quite different properties.

This concludes the review of quantified modal epistemic logic.

2.3.4 Grounded Possible Worlds

Most people, confronted with possible worlds semantics for the first time are — initially at least — uncomfortable with the idea:

> "[The notion] of one possible world being accessible to another has at first sight a certain air of fantasy or science fiction about it". [88, p77]

The problem seems to be with the ontological status of possible worlds: Do they really exist? If so, where are they? How do they map onto an agent's physical architecture? If these questions cannot be answered, then one would be reluctant to treat epistemic alternatives as anything other than a theoretical nicety.

Some researchers have proposed *grounding* epistemic alternatives: giving them a precise meaning in the real world, thus overcoming any confusion about their status[11]. Two more or less identical groundings have been proposed: the first by the distributed systems community, the second by Rosenschein, for his situated automata paradigm. This section describes grounded possible worlds, and will focus on the distributed systems approach; the formal treatment is adapted from [53].

Using a logic of knowledge to analyze a distributed system may seem strange. However, as Halpern points out, when informally reasoning about a distributed system, one often makes statements such as: "processor 1 can't send a packet to processor 2 until it knows that processor 2 received the previous one" [81]. A logic of knowledge formalizes such reasoning.

The starting point for our study is to define a simple model of distributed systems. A system contains an *environment*, which may be in any of a set E of environment states, and a set of n processes $\{1, \ldots, n\}$, each of which may be in any of a set L of "local" states. At any time, a system may therefore be in any of a set G of global states.

$$G = E \times \underbrace{L \times \cdots \times L}_{n \text{ times}}$$

Next, a *run* of a system is a function which assigns to each time point a global state: time points are isomorphic to the natural numbers, (and time is thus discrete, bounded in the past, and infinite in the future).

$$\mathit{Run} = \mathbb{N} \to G$$

A *point* is a run together with a time:

$$\mathit{Point} = \mathit{Run} \times \mathbb{N}$$

A point implicitly identifies a global state. Points will serve as worlds in the logic of knowledge to be developed. A *system* is a set of runs.

$$\mathit{System} = \mathsf{powerset}\ \mathit{Run}$$

Now, suppose s and s' are two global states.

$$\begin{aligned} s &= \langle e, l_1, \ldots, l_n \rangle \\ s' &= \langle e', l'_1, \ldots, l'_n \rangle \end{aligned}$$

[11] To some extent, we have already seen an example of grounding in the card game example presented earlier.

Define a relation $\sim_i$, on states, for each process i:

$s \sim_i s'$ iff $(l_i = l'_i)$

Note that $\sim_i$ will be an equivalence relation. The terminology is that if $s \sim_i s'$, then s and s' are *indistinguishable* to i, since the local state of i is the same in each global state. Intuitively, the local state of a process represents the information that the process has, and if two global states are indistinguishable, then it has the same information in each.

> "The crucial point here is that since a processes [choice of] actions …are a function of its local state, if two points are indistinguishable to processor i, then processor i will perform the same actions in each state". [81, pp46–47]

The next step is to define a language for reasoning about such systems. The language is that of the multi-agent epistemic logic defined earlier (i.e., classical propositional logic enriched by the addition of a set of unary modal operators K_i, for $i \in \{1, \ldots, n\}$). The semantics of the language are presented via the satisfaction relation, "$\models$", which holds between triples of the form:

$\langle M, r, u \rangle$

and formulae of the language. Here, $\langle r, u \rangle$ is a point, and M is a structure:

$\langle \mathcal{R}, \pi \rangle$

where $\mathcal{R}$ is a system, and

π: *Point* $\rightarrow$ powerset *Prop*

returns the set of atomic propositions true at a point. The structure $\langle \mathcal{R}, \pi \rangle$ is called an *interpreted system.* The only non-standard semantic rules are for propositions and modal formulae.

$$\langle M, r, u \rangle \models p \quad \text{where } p \in \mathit{Prop}, \text{ iff } p \in \pi(\langle r, u \rangle)$$

$$\langle M, r, u \rangle \models \mathsf{K}_i\phi \quad \text{iff} \quad \langle M, r', u' \rangle \models \phi \text{ for all } r' \in \mathcal{R} \text{ and } u' \in \mathbb{N} \text{ such that } r(u) \sim_i r'(u')$$

Note that since $\sim_i$ is an equivalence relation (i.e., it is reflexive, symmetric, and transitive), this logic will have the properties of the system $S5_n$, discussed above. In what sense does the second rule capture the idea of a processes knowledge? The idea is that if $r(u) \sim_i r'(u')$, then *for all i knows*, it could be in either run r, time u, or run r', time u'; the process does not have enough information to be able to distinguish the two states. The information/knowledge it *does* have are the things true in all indistinguishable states.

> "In this model, knowledge is an *external* notion. We don't imagine a processor scratching its head wondering whether or not it knows a fact ϕ. Rather, a programmer reasoning about a particular protocol would say, from the outside, that the processor knew ϕ because in all global states [indistinguishable] from its current state (intuitively, all the states the processor could be in, for all it knows), ϕ is true". [80, p6]

A distributed knowledge operator can be defined in a similar way. Define a relation $\sim$ on global states.

$s \sim s'$ iff $\forall i \in \{1, \ldots, n\} \cdot s \sim_i s'$

The distributed knowledge operator can then be defined.

$$\langle M, r, u \rangle \models \mathsf{DK}\phi \quad \text{iff} \quad \langle M, r', u' \rangle \models \phi \text{ for all } r' \in \mathcal{R} \text{ and } u' \in \mathbb{N} \text{ such that } r(u) \sim r'(u')$$

This distributed knowledge operator will have exactly the same properties as the distributed knowledge operator developed earlier.

A number of researchers have investigated a variety of different properties of distributed systems using slight variations on the basic technique described here (see, e.g., [58], [105], [54], [46]). For example, Fagin and Vardi have demonstrated a *conservation principle* of distributed knowledge [54][12]. Briefly, they investigate the properties required for the distributed knowledge of a system to remain constant. A related issue is whether processes "learn" or "forget" over time.

Interestingly, an essentially equivalent definition of knowledge to that described above has been proposed independently by Rosenschein and Kaelbling for their situated automata paradigm [142], [143]. The details are

[12] Note that Fagin and Vardi use the term "implicit knowledge" in the way that we use "distributed knowledge".

discussed in Appendix B. Rosenschein's aim is to analyze the knowledge content of states of automata. In [143], an epistemic temporal logic is described, which is used to specify what a designer would have a machine "know". This intentional description is then compiled down to a gate level description of a digital machine, which satisfies the properties expressed in its intentional description. The situated automata paradigm is gaining some followers in AI: see, e.g., [150], [151], [97], and [127] for work based on, or inspired by, situated automata.

While the distributed systems/situated automata analysis of knowledge is a useful one, allowing us to characterize agent's states in terms of the information they implicitly carry, it is not clear how such a description relates to the class of agents under consideration in this thesis. Moreover, the analysis still seems too coarse grained to be useful. So while the area is of undoubted interest, it is only of minor relevance to this thesis.

2.3.5 Avoiding Logical Omniscience

A number of attempts have been made to develop epistemic logics that avoid the logical omniscience problem, and yet retain the theoretically attractive properties of possible worlds semantics. Two of the better known examples of this work are reviewed in this section. The first is Levesque's logic of implicit and explicit belief [112], the second is Fagin and Halpern's logic of general awareness, which was, in part, inspired by Levesque's work [52].

Levesque proposed making a distinction between *explicit* and *implicit* beliefs. To define the semantics of implicit and explicit beliefs, Levesque proposed the use of *situations*, a notion borrowed from situation semantics (see [14], [41], and below). A situation is best thought of as a fragment of a world, or a partial description of a world. Situations are similar to possible worlds, but differ in that whereas a world assigns to every primitive proposition either true or false, a situation may assign it true, false, neither, or both (in this last case, a situation is said to be *incoherent*).

Levesque proceeded by assigning an agent a set of situations, those compatible with its explicit beliefs. An agent could be said to explicitly believe ϕ if the truth of ϕ were supported by every situation compatible with that agent's explicit beliefs. Syntactically, Levesque's logic is classical propositional logic extended by two unary modal operators: *B*, for explicit belief, and *L*, for implicit belief. Nested beliefs are not allowed.

A model for Levesque's logic is a structure:

$$\langle \mathcal{S}, \mathcal{B}, \mathcal{T}, \mathcal{F} \rangle$$

where

- $\mathcal{S}$ is a set of situations;
- $\mathcal{B} \subseteq \mathcal{S}$ is the set of situations compatible with the agent's explicit beliefs;
- $\mathcal{T}$: *Prop* $\rightarrow$ powerset $\mathcal{S}$ takes a primitive proposition and returns the set of situations which support its truth;
- $\mathcal{F}$: *Prop* $\rightarrow$ powerset $\mathcal{S}$ takes a primitive proposition and returns the set of situations which support its falsity.

Rather than the single satisfaction relation of standard logics, Levesque defines two satisfaction relations, "$\models_{\mathcal{T}}$", and "$\models_{\mathcal{F}}$", which hold between structures of the form

$$\langle M, s \rangle$$

(where *M* is a model, and *s* is a situation), and formulae of the language. The relation $\models_{\mathcal{T}}$ is for "supporting the truth of…", the relation $\models_{\mathcal{F}}$ is for "supporting the falsity of…". The semantic rules for the main part of the language are given in Figure 2.3. The first six rules are straightforward: in particular, the first two clearly show what it means for a situation to support the truth or falsity of a proposition. But how does the seventh rule, that for explicit belief, overcome the problem of logical omniscience?

Consider the fundamental problem of possible worlds semantics, that of knowing all tautologies (valid formulae). A tautology is known because it is assigned the truth value true in all worlds. This is because, in effect, every world acts as a propositional valuation, which by definition assigns true to a tautology. In Levesque's formalism, however, a tautology is *not* necessarily assigned true in every situation, (since a situation may assign a primitive proposition both true and false, either, or neither). In short, situations do not act as propositional valuations, so the problem does not arise.

Similar arguments show that: 1) explicit belief, so defined, is not closed under logical consequence; 2) agents can be inconsistent without believing everything, and 3) logically equivalent formulae are not equivalent as explicit beliefs.

$\langle M, s\rangle$	$\models_{\mathcal{T}}$	p	where $p \in \mathit{Prop}$, iff $s \in \mathcal{T}(p)$
$\langle M, s\rangle$	$\models_{\mathcal{F}}$	p	where $p \in \mathit{Prop}$, iff $s \in \mathcal{F}(p)$
$\langle M, s\rangle$	$\models_{\mathcal{T}}$	$\phi \vee \psi$	iff $\langle M, s\rangle \models_{\mathcal{T}} \phi$ or $\langle M, s\rangle \models_{\mathcal{T}} \psi$
$\langle M, s\rangle$	$\models_{\mathcal{F}}$	$\phi \vee \psi$	iff $\langle M, s\rangle \models_{\mathcal{F}} \phi$ and $\langle M, s\rangle \models_{\mathcal{F}} \psi$
$\langle M, s\rangle$	$\models_{\mathcal{T}}$	$\neg\phi$	iff $\langle M, s\rangle \models_{\mathcal{F}} \phi$
$\langle M, s\rangle$	$\models_{\mathcal{F}}$	$\neg\phi$	iff $\langle M, s\rangle \models_{\mathcal{T}} \phi$
$\langle M, s\rangle$	$\models_{\mathcal{T}}$	$B\phi$	iff $\langle M, s'\rangle \models_{\mathcal{T}} \phi$ for all $s' \in \mathcal{B}$
$\langle M, s\rangle$	$\models_{\mathcal{F}}$	$B\phi$	iff $\langle M, s\rangle \not\models_{\mathcal{T}} B\phi$

Figure 2.3: Semantics for Levesque's Logic

Now for the implicit belief operator, L. For this operator, a function $\mathcal{W}$ is defined, which takes a situation s, and returns that set of situations which have the following properties: 1) they agree with s on the truth or falsity of propositions, and 2) they act as propositional valuations. If s is incoherent, then $\mathcal{W}(s) = \{\}$. The function can be extended to take a set of situations in the following way.

$$\mathcal{W}(\{s_1, \ldots, s_n\}) \triangleq \mathcal{W}(s_1) \cup \cdots \cup \mathcal{W}(s_n)$$

The semantic rules for the implicit belief operator can now be given.

$\langle M, s\rangle \models_{\mathcal{T}} L\phi$ iff $\langle M, s'\rangle \models_{\mathcal{T}} \phi$ for all $s' \in \mathcal{W}(\mathcal{B})$

$\langle M, s\rangle \models_{\mathcal{F}} L\phi$ iff $\langle M, s\rangle \not\models_{\mathcal{T}} L\phi$

The implicit belief operator thus picks out those situations which are both compatible with the agent's explicit beliefs, and act as classical propositional valuations; these situations then act as worlds in the definition of the implicit belief operator.

Now, suppose $B\phi$ is true in some model. Then every situation $s \in \mathcal{B}$ supports the truth of ϕ. So the subset picked out by the implicit belief operator will also support its truth. Thus $L\phi$ will also be true in the model. This gives the following validity:

$$\models B\phi \;\Rightarrow\; L\phi$$

In general, however, the reverse does not hold. Note that L has the properties of a normal modal necessity operator with logic S5.

Although Levesque's logic appears to solve the problem of logical omniscience, there seem to be four main objections to it (see [138, p135]). First, it does not allow quantification (but see [106], [107]). Second, it does not allow nested beliefs. Third, the notion of a situation in Levesque's logic is, if anything, even more suspect than the notion of a world in normal epistemic modal logic. Finally, Fagin and Halpern have shown that under certain circumstances, an agent modelled by Levesque's scheme must still be aware of propositional tautologies [52].

In an effort to recover from this last negative result, Fagin and Halpern have developed a logic of general awareness [52]. This logic contains the modal operator L, for implicit belief, an operator A, for "awareness", and a derived operator B, for explicit belief. Call the language of the logic LGA, for "logic of general awareness".

Semantically, the logic is much simpler than Levesque's. Models are structures of the form:

$$\langle \mathcal{S}, \mathcal{A}, \mathcal{B}, \pi\rangle$$

where

- $\mathcal{S}$ is a set of states;
- $\mathcal{A}: \mathcal{S} \rightarrow$ powerset $\mathit{Form}(\mathit{LGA})$ maps each state to the set of formulae that the agent is aware of in that state;
- $\mathcal{B} \subseteq \mathcal{S} \times \mathcal{S}$ is a transitive, serial, euclidean relation on $\mathcal{S}$;
- $\pi: \mathcal{S} \rightarrow$ powerset Prop interprets primitive propositions in states.

The only non-standard semantic rules are for the A and L operators.

$\langle M, s\rangle \models L\phi$ iff $\langle M, s'\rangle \models \phi$ for all $s' \in \mathcal{S}$ such that $(s, s') \in \mathcal{B}$

$\langle M, s\rangle \models A\phi$ iff $\phi \in \mathcal{A}(s)$

The L operator is thus an ordinary modal necessity operator; the restrictions on the belief accessibility relation mean its logic is KD45. The A operator says that an agent is "aware of" a formula. The explicit belief operator B is then introduced as an abbreviation.

$$B\phi \triangleq L\phi \wedge A\phi$$

So an agent explicitly believes ϕ if it is "aware of" ϕ and ϕ is true in all its epistemic alternatives. In effect, the awareness set $\mathcal{A}(s)$ of each state s acts as a filter, picking out those beliefs which an agent is aware of, as opposed to those the agent knows nothing about.

Konolige has presented a detailed critique of the logic of general awareness, in which he compares it with his own deduction model of belief (see below, and [100]). He concludes:

> "It does not seem that there is much to be gained by considering a logic of general awareness, at least as far as modelling resource limited reasoning from beliefs is concerned. It is no more powerful than [the deduction model], and can be re-expressed in those terms. The practice of mixing sentential and possible-worlds elements in the semantics does not preserve the elegance of the latter, or offer any insight into the psychological nature of explicit belief". [101, p248]

2.3.6 Beliefs, Goals, Intention and Rational Balance

All of the formalisms considered so far have focussed on just one aspect of intelligent agency: either knowledge or belief. However, it is to be expected that a general theory of agency must integrate reasoning about these facets of intentionality with other components of an agent's cognitive makeup: goals, intentions, etc. The best known, and most sophisticated attempt to show how this might be done is due to Cohen and Levesque [29]. Their *logic of rational agency* is the object of study in this section[13].

Cohen and Levesque's formalism was originally used to develop a theory of intention (as in "I intended to…"), which the authors required as a pre-requisite for a theory of speech acts (see next chapter for a summary, and [30] for full details). However, the logic has subsequently proved to be so useful for specifying and reasoning about the properties of agents that it has been used in an analysis of conflict and cooperation in multi-agent dialogue [66], [65], as well as several studies in the theoretical foundations of cooperative problem solving [174], [113], [91], [92]. This section will focus on the use of the logic in developing a theory of intention. The first step is to lay out the criteria that a theory of intention must satisfy.

When building intelligent agents — particularly agents that must interact with humans — it is important that a *rational balance* is achieved between the beliefs and goals of the agents.

> "For example, the following are desirable properties of intention: An autonomous agent should act on its intentions, not in spite of them; adopt intentions it believes are feasible and forego those believed to be infeasible; keep (or commit to) intentions, but not forever; discharge those intentions believed to have been satisfied; alter intentions when relevant beliefs change; and adopt subsidiary intentions during plan formation". [29, p214]

Following Bratman, ([17], [18]), Cohen and Levesque identify seven specific properties that must be satisfied by a reasonable theory of intention:

1. Intentions pose problems for agents, who need to determine ways of achieving them.
2. Intentions provide a "filter" for adopting other intentions, which must not conflict.
3. Agents track the success of their intentions, and are inclined to try again if their attempts fail.
4. Agents believe their intentions are possible.
5. Agents do not believe they will not bring about their intentions.
6. Under certain circumstances, agents believe they will bring about their intentions.
7. Agents need not intend all the expected side effects of their intentions.

[13] Unfortunately, the logic is too complex to allow a detailed presentation here. Instead, we outline its most significant components, and present a detailed technical review in Appendix B.

(BEL x ϕ)	Agent x believes ϕ
(GOAL x ϕ)	Agent x has goal of ϕ
(HAPPENS α)	Action α will happen next
(DONE α)	Action α has just happened

Table 2.3: Modalities in Cohen and Levesque's Formalism

Given these criteria, Cohen and Levesque adopt a two tiered approach to the problem of formalizing a theory of intention. First, they construct the logic of rational agency, "being careful to sort out the relationships among the basic modal operators" [29, p221]. On top of this framework, they introduce a number of derived constructs, which constitute a "partial theory of rational action" [29, p221]; intention is one of these constructs.

Syntactically, the logic of rational agency is a many-sorted, first-order, multi-modal logic with equality, containing four primary modalities; see Table 2.3. The semantics of BEL and GOAL are given via possible worlds, in the usual way: each agent is assigned a belief accessibility relation, and a goal accessibility relation. The belief accessibility relation is euclidean, transitive, and serial, giving a belief logic of KD45. The goal relation is serial, giving a conative logic KD. It is assumed that each agent's goal relation is a subset of its belief relation, implying that an agent will not have a goal of something it believes will not happen. Worlds in the formalism are a discrete sequence of events, stretching infinitely into past and future.

The two basic temporal operators, HAPPENS and DONE, are augmented by some operators for describing the structure of event sequences, in the style of dynamic logic [84]. The two most important of these constructors are ";" and "?":

$\alpha; \alpha'$	denotes α followed by α'
$\alpha?$	denotes a "test action" α

The standard future time operators of temporal logic, "$\Box$" (always), and "$\Diamond$" (sometime) can be defined as abbreviations, along with a "strict" sometime operator, LATER:

$$\begin{aligned} \Diamond\alpha &\triangleq \exists x \cdot (\text{HAPPENS } x; \alpha?) \\ \Box\alpha &\triangleq \neg\Diamond\neg\alpha \\ (\text{LATER } p) &\triangleq \neg p \wedge \Diamond p \end{aligned}$$

A temporal precedence operator, (BEFORE p q) can also be derived, and holds if p holds before q. An important assumption is that *all* goals are eventually dropped:

$$\Diamond\neg\,(\text{GOAL } x\ (\text{LATER } p))$$

The first major derived construct is a *persistent* goal.

$$\begin{aligned} (\text{P\_GOAL } x\ p) \triangleq\ & (\text{GOAL } x\ (\text{LATER } p)) \wedge \\ & (\text{BEL } x \neg p) \wedge \\ & \left[\begin{array}{l} \text{BEFORE} \\ \quad ((\text{BEL } x\ p) \vee (\text{BEL } x\ \Box\neg p)) \\ \quad \neg\,(\text{GOAL } x\ (\text{LATER } p)) \end{array}\right] \end{aligned}$$

So, an agent has a persistent goal of p if:

1. It has a goal that p eventually becomes true, and believes that p is not currently true.
2. Before it drops the goal, one of the following conditions must hold:
   (a) the agent believes the goal has been satisfied;
   (b) the agent believes the goal will never be satisfied.

It is a small step from persistent goals to a first definition of intention, as in "intending to act". Note that "intending that something becomes true" is similar, but requires a slightly different definition; see [29].

$$\begin{aligned} (\text{INTEND } x\ \alpha) \triangleq\ & (\text{P\_GOAL } x \\ & \quad [\text{DONE } x\ (\text{BEL } x\ (\text{HAPPENS } \alpha))?; \alpha] \\ &) \end{aligned}$$

Cohen and Levesque go on to show how such a definition meets many of Bratman's criteria for a theory of intention (outlined above).

A critique of Cohen and Levesque's theory of intention is presented in [155]; space restrictions prevent a discussion here.

2.4 Meta-Languages and Syntactic Modalities

There is a school of thought — almost a cult — in mathematics, computer science, and AI, which holds that classical first-order logic is in some sense "canonical": that anything which can be done in a non-classical (e.g., modal) logic can be reduced to a problem in first-order logic. In the area of reasoning about intentional notions, this thinking manifests itself in the use of first-order meta-languages to represent and reason about beliefs, goals, etc.

The basic idea of a meta-language is quite simple. Recall the naive attempt to formalize the statment "Janine believes Cronos is the father of Zeus" presented earlier:

$$Bel(Janine, Father(Zeus, Cronos)) \tag{2.9}$$

The syntactic problem with this pseudo-formalization is that $Father(Zeus, Cronos)$ is a formula of first-order logic, not a term, so the whole formula is ill-formed. Now suppose the domain of the first-order language (hereafter called the meta-language) contains formulae of another language (hereafter called the object-language). Suppose also that the meta-language contains terms which denote object-language formulae. Then it would be possible to write a meta-language formula capturing the sense of (2.9).

To make meta-language formulae readable, it is useful to employ a *quoting convention*. Suppose that ϕ is an object-language formula, then it is usual to let $\ulcorner \phi \urcorner$ be an abbreviation for a meta-language term standing for ϕ. A formulation of the above sentence might then be:

$$Bel(Janine, \ulcorner Father(Zeus, Cronos) \urcorner) \tag{2.10}$$

Since $\ulcorner Father(Zeus, Cronos) \urcorner$ is a meta-language term, (2.10) is a syntactically acceptable formula of the meta-language. Using this approach, belief, knowledge, etc. can be axiomatized using the first-order meta-language and given whatever properties are deemed appropriate. One obvious difficulty seems to be that it is not possible to quantify over object-language terms. This difficulty can be overcome by using an *un-quoting convention*: see [73, pp241–242] for a discussion of this point.

The quote marks, $\ulcorner \_ \urcorner$, are sometimes called *Frege quotes*, or *sense quotes*. The process of making formulae into objects in a meta-language domain is sometimes called *arithmetization*, after Gödel, who used a meta-language technique in his proof of the incompleteness of arithmetic; the names given to object-language formulae are sometimes called Gödel numbers.

It is often claimed that meta-language approaches enjoy the following advantages over modal languages:

1. **Expressive power.** The following meta-language formulae have no quantified modal logic counterparts:

$$\begin{array}{lll} \exists x \cdot Bel(i, x) & & \text{“}i \text{ believes something”} \\ \forall x \cdot Bel(i, x) & \Rightarrow \; Bel(j, x) & \text{“}i \text{ believes everything } j \text{ believes”} \end{array}$$

2. **Computational tractability.** Meta-languages are just many-sorted first-order languages, for which automated theorem provers exist. It should therefore be possible — in principle at least — to use existing theorem provers for meta-languages. However, this claim has yet to be satisfactorily demonstrated.

Unfortunately, meta-language approaches also suffer from severe problems. These problems are discussed in the next section. One meta-language approach is then described in detail [98]. Finally, a hybrid formalism, based in part on a meta-language approach, and in part on possible worlds, is briefly described [124].

2.4.1 Meta-Languages, Self Reference, and Inconsistency

The key difficulty with meta-language approaches is that many naive attempts to treat meta-language predicates as *syntactic modalities* run into inconsistency. These difficulties arise, for the large part, from the issue of *self reference*[14].

[14] A good discussion of these issues may be found in [131]; the issue of self reference is examined in detail in [129] and [130].

Suppose one wished to use a meta-language approach to represent and reason about knowledge and belief. Following "standard" practice, knowledge may be defined as true belief. One might begin by setting object-language = meta-language, so that the object- and meta-language are the same; in such circumstances, the meta-language is said to be *self referential*, since it may contain terms which refer to its own formulae. A *truth predicate* can then be defined.

$$True(\ulcorner \phi \urcorner) \Leftrightarrow \phi \tag{2.11}$$

Knowledge may then be defined as a composite concept.

$$Know(\ulcorner \phi \urcorner) \Leftrightarrow Bel(\ulcorner \phi \urcorner) \wedge True(\ulcorner \phi \urcorner) \tag{2.12}$$

The *Bel* predicate can then be given whatever axiomatization is deemed appropriate. This approach seems simple, intuitive, and satisfactory. Unfortunately, in general, any first-order meta-language theory containing axiom (2.11) turns out to be inconsistent. This problem was first recognized by Tarski; the difficulty is with the possibility of formulae asserting their own falsity (as in the famous "liar" paradox).

Several proposals have been put forward to remedy the problem. One possibility is to define *True* and *False* predicates, such that for some formulae ϕ, neither $True(\ulcorner \phi \urcorner)$ nor $False(\ulcorner \phi \urcorner)$ hold; in effect, one axiomatizes a system in which the law of the excluded middle does not hold. This solution is unintuitive, however, and seems rather *ad hoc*.

Another solution, due to Perlis [129], is to replace (2.11) with an axiom of the form:

$$True(\ulcorner \phi \urcorner) \Leftrightarrow \phi^* \tag{2.13}$$

where ϕ^* is the formula obtained from ϕ by replacing every occurrence of $\neg\, True(\ulcorner \psi \urcorner)$ in ϕ by $True(\ulcorner \neg\, \psi \urcorner)$. This simple expedient prevents inconsistency (see [129, pp312–315] for proof). This proposal has been criticized by Konolige, on a number of grounds, however: see [100, p116] for details.

A problem not alleviated by Perlis' schema was first recognized by Montague [121]. Consider the following axiom:

$$Bel(\ulcorner \phi \urcorner) \Rightarrow \phi$$

and the following inference rule:

From $\vdash \phi$ infer $\vdash Bel(\ulcorner \phi \urcorner)$

(cf. the modal system *KT*). Montague showed that any moderately sophisticated first-order theory (and in particular, basic arithmetic) containing the above axiom and inference rule is inconsistent. This result appears to be devastating, and for a long time dampened down research into first-order meta-languages. A similar result by Thomason ([169]) is similarly discouraging.

However, des Rivieres and Levesque have recently shown that, while Montague's results are technically correct, a careful reworking, with slightly different assumptions, leads back to consistency [40][15]. Also, Perlis has shown how a similar technique to that proposed by him for recovering from Tarski's result can be used to recover from Montague's results [130]. Despite these results, the author is not aware of any recovery from Thomason's negative results.

If nothing else, the above discussion illustrates that the whole issue of self-referential languages is a difficult one. It is hardly surprising, therefore, that some researchers have rejected self reference entirely. One critic is Konolige, who suggests that an observer of a system of agents can hardly have any use for a self referential language. He argues that the logical puzzles caused by self reference are just that: puzzles, which an agent will never usually need to consider (unless she is a philosophy or logic student) [100, p115].

One "standard" alternative to self-referential languages is to use an *hierarchical* language structure. Consider a "tower" of languages:

$$L_0 \text{ — } L_1 \text{ — } L_2 \text{ — } \cdots \text{ — } L_k \text{ — } \cdots$$

Let language L_0 be a non-self-referential "base" language; standard first-order logic, for example. Then let each of the languages L_k, where $k > 0$, be a meta-language containing terms referring only to formulae of languages lower down in the hierarchy. Thus no language in the hierarchy is self-referential. It is possible to write formulae such as (2.10) in any of the languages L_k, where $k > 0$, but the paradoxes of self-reference will not arise.

[15] In fact, they showed that it was a naive translation from a modal to meta-language that was causing difficulties.

The hierarchical approach has problems of its own, however. Consider the following scenario, adapted from [172]. There are two agents, A and B, each with equal expressive powers of belief: anything A can believe, B can believe, and *vice versa*. Consider the following statement.

> A believes that everything B believes is true. (2.14)

How might this statement be represented, using the hierarchical approach? It obviously cannot be represented in L_0. One obvious solution is to write:

$$Bel(A, \ulcorner \forall x \cdot Bel(B, x) \Rightarrow True(x) \urcorner).$$

Suppose that the inner *Bel* predicate belongs to language L_k, where $k > 0$. The variable x in this formula ranges over all formulae of languages L_j, where $j < k$. It does *not*, therefore range over the language of B's beliefs, but over some language lower in the hierarchy. Moving to a language further up the hierarchy does not solve the problem:

> "No matter how far we climb up the object/meta-language hierarchy, we will not be able to capture the intuitive content of [(2.14)]". [172, pp7–8]

Other objections to the hierarchical approach have been raised. For example, Morgenstern points out that the truth predicate paradox is only resolved by positing an infinite number of truth predicates, one for each language in the hierarchy. She argues that this is not a satisfactory solution, for two reasons:

> "In the first place, it is implausible that people are consciously aware of using different truth predicates when they speak. Secondly, we often do not know which truth predicate to use when we utter a particular statement. Finally, it is impossible …to construct a pair of sentences, each of which refers to the other, although such sentences may make perfect sense". [125, p104]

So both self-referential and hierarchical languages have their difficulties. One problem common to both is that meta-languages are, in general, notationally cumbersome, unintuitive, and complex: see [100, p111] for a discussion of this point. However, the difficulties described above have not prevented the development of some ingenious and useful meta-language formulations of intentional notions.

2.4.2 Konolige's First-Order Formalization of Knowledge and Action

In a 1982 paper, Kurt Konolige described a hierarchical meta-language formalism for describing and reasoning about the knowledge and actions of computational agents [98]. In this section, we briefly review this formalism.

We begin by describing the meta-/object-language relationship, before moving on to the specifics of Konolige's axiomatization. For the moment, assume that the object-language is a standard first-order language. For each primitive object-language expression e, there is assumed to be a corresponding meta-language term e'. Meta-language terms denoting compound object-language formulae are constructed using the meta-language functions *and*, *or*, *not*, and so on. For example, the object-language formula

$$p \Rightarrow (q \vee r)$$

is denoted by the meta-language term

$$imp(p', or(q', r')).$$

Since this construction is somewhat cumbersome, sense quotes are used as abbreviations:

$$\begin{array}{ll} \ulcorner \neg p \urcorner & \triangleq \; not(p') \\ \ulcorner p \vee q \urcorner & \triangleq \; or(p', q') \end{array}$$

etc.

To write that an object-language formula reflects the true state of the world, Konolige used a meta-language truth predicate *TRUE*, with the following axiomatization:

$$\begin{array}{lll} \forall f \cdot \neg TRUE(f) & \Leftrightarrow & TRUE(not(f)) \\ \forall f \cdot \forall g \cdot TRUE(f) \vee TRUE(g) & \Leftrightarrow & TRUE(or(f, g)) \end{array}$$

etc.

Konolige took a *syntactic* approach for describing the beliefs of agents; each agent is assigned a set of object-language formulae — its *theory* — and believes ϕ if ϕ is provable from its theory. A meta-language function *th* is assumed, which takes a term denoting an agent, and returns the set of object-language formulae representing that agent's theory. A meta-language predicate $FACT(t, f)$ is assumed, which says that f is a member of theory t. A *common fact* is a formula which is common to all agent's theories, and is also true.

$$\forall f \cdot CFACT(f) \;\Leftrightarrow\; \forall a \cdot FACT(th(a), f) \wedge TRUE(f)$$

The next step is to introduce a predicate *PR*, for provability. This is a binary predicate, taking a theory and a formula as arguments. It can be given any axiomatization desired, but Konolige suggests giving it one which makes provability complete with respect to the object-language. Its axiomatization will thus include modus ponens, reflexivity, etc.

$$\forall t \cdot \forall f \cdot \forall g \cdot PR(t, imp(f, g)) \wedge PR(t, f) \;\Rightarrow\; PR(t, g)$$
$$\forall t \cdot \forall f \cdot PR(t, f) \;\Rightarrow\; PR(t, f)$$
etc.

Belief is then defined using a meta-language predicate which holds between an agent and an object-language formula: an agent believes ϕ if ϕ is provable from its theory.

$$\forall a \cdot \forall f \cdot BEL(a, f) \;\Leftrightarrow\; PR(th(a), f)$$

Knowledge is defined as true belief, in the usual way. Konolige then goes on to extend his formalism in three ways:

- he introduces a standard name function η, and a denotation function Δ, to simplify some of the notational problems associated the meta-language approach;
- he deals with *nested belief*, (i.e., beliefs about beliefs) by extending the two-language hierarchy to a three-language hierarchy;
- he introduces situations (in the sense of the situation calculus [118]) into the domain of the meta-language for reasoning about a changing world.

The details of these extensions need not concern us here. A critique of Konolige's formalism has been developed by Davies [34]. In addition to the drawbacks associated with hierarchical languages generally, Davies suggests three criticisms of Konolige's formalism:

- the three layer hierarchy seems *ad hoc*;
- computationally, Konolige's formalism seems likely to be unwieldy, due to the "meta-level" overhead;
- some of the axioms for common facts are invalid.

2.4.3 Moore's Formalism

One of the most influential figures in the recent development of intentional logics has been R. Moore. For his 1980 doctoral thesis, he presented a detailed formal examination of the interaction between knowledge and action, framed in a multi-modal first-order logic [122], [124]. This logic has a possible worlds semantics, based on the semantics of normal modal logics as discussed earlier. However, Moore took the novel step of showing how these semantics could be axiomatized in a first-order meta-language. Modal formulae could then be translated to first-order meta-language formulae using the axiomatization. Theorem proving in the modal language is thereby reduced to a problem of theorem proving in the meta-language. Moore's formalism is rather complex, and details are therefore omitted; the interested reader is referred to the cited works.

The first part of the formalism is essentially a treatment of knowledge using possible worlds semantics, except that Moore axiomatized the semantics of the modal operators in a first-order meta-language. Moore then developed a treatment of action, using action constructors in the style of dynamic logic [84].

Moore went on to present a detailed examination of the way that knowledge and action interact. This culminated in a definition of ability: what it means for an agent to be able to do something. Moore began his analysis by noting that knowledge and action interact in two obvious ways:

1. As a result of performing an action, an agent can gain knowledge. Agents can perform "test" actions, in order to find things out.

2. In order to perform some actions, an agent needs knowledge: these are knowledge pre-conditions ([126]). For example, in order to open a safe, it is necessary to know the combination.

As a result of these considerations, Moore defined ability as either: 1) having a rigid designator for an action such that the agent knows that as a result of performing the action, the goal will be achieved, or 2) having some initial action which the agent may perform, such that after it is performed, the agent will be in a position to achieve the goal. This definition has subsequently been extended and refined by several workers (e.g., [126], [111]).

Moore's formalism has one major advantage: since it is ultimately based on a first-order meta-language, it should be possible to use a first-order theorem prover for it. Moore's formalism has, in fact, been implemented, by Appelt, in his natural language planning system KAMP [6], [7].

However, there also seem to be a number of difficulties with the formalism. The most important are as follows:

1. The process of translating the modal language into a first-order one and then theorem proving in the first-order language is inefficient. Reichgelt observes that "hard-wired" modal theorem provers will probably be more efficient [138, pp128–129].

2. The formulae resulting from the translation process are complicated and unintuitive.

3. Moore's formalism is based on possible worlds, and thus falls prey to logical omniscience. As a corollary, the definition of ability is somewhat vacuous (agents know all necessary truths, which weakens the knowledge precondition) [126, p152].

To conclude, the approach of using a first-order meta-language to axiomatize the semantics of a modal object-language was developed in order to provide a framework for efficient modal theorem proving. Since this approach seems to be ruled out in practice, it is not clear what advantages this technique has over a normal modal language.

2.5 Other Formalisms

In this section, we examine four formalisms for reasoning about intentional notions that do not fit neatly into any of the categories described above. First, Konolige's deduction model of belief is examined in detail [100]; Werner's formalism [179], situation semantics [14], [41], and Singh's formalism [153] are then briefly described.

2.5.1 The Deduction Model of Belief

In Konolige's 1982 first-order formalization of knowledge and action, described above, an agent's beliefs were modelled as a set of formulae and a provability relation. Motivated by some shortcomings of this formalism (see above), Konolige went on to develop a family of belief logics based on a semantic model similar to that described in the 1982 paper. These logics employed a modal language for belief, which is less cumbersome than the meta-language formalism, and used a limited form of deduction for the provability relation, thereby modelling resource bounded reasoning. The work was originally described in his 1984 doctoral thesis [99], and was published in a polished, revised form as [100].

The starting point for the *deduction model of belief* was the observation that: "[T]he most important properties of belief, for our purposes, are those that are necessary for typical robot planning and problem solving systems" [100, p1]; "The deduction model was developed in an effort to define accurate models of the beliefs of AI …systems" [100, p3]. This is an important point, often overlooked by critics of the deduction model: his aim was to develop a model of the beliefs of artificial, computational agents, and *not* a model of human believers. He argued that such a model should capture the following properties of beliefs:

- agents (may) derive some conclusions from their beliefs;
- they do not necessarily derive *all* the logically possible conclusions.

Possible worlds logics of belief capture the first property; they do not capture the second. So how is one to construct a model of the beliefs of AI systems that captures both properties? Konolige observed that:

> "Because [AI systems] have been constructed by AI researchers, we can actually look at their design and answer questions about their internal 'mental' structures. In effect, we are using these …systems as subjects in a discipline that might be appropriately termed *experimental robot psychology*". [100, p3]

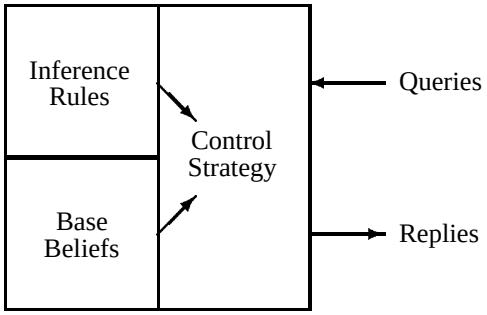


Figure 2.4: Structure of a Typical AI Belief System

Konolige then noted that a typical AI system has a structure such as that illustrated in Figure 2.4 (after [100, p19]). The "base beliefs" correspond to what is often called "knowledge" in an AI system (though "belief" is surely a more appropriate term). These beliefs are usually expressed in some symbolic language. This language may be one of semantic nets or frames, but is more generally a *logical language*: perhaps a subset of first-order predicate logic. Call this language the *internal* language. It is important to distinguish this language from the *external* language that will be used by an observer to describe the beliefs of agents. These languages may be the same, though this is not necessarily the case; there may be no way of talking about the beliefs of agents in the internal language.

The second feature is some set of *inference rules*. Most AI systems have some limited inferential ability, based on (pseudo-) logical inference. Such inference is inevitably limited by the computational resources available to the agent. It is not, in general, possible for an agent to derive *all* the possible consequences of a set of beliefs, as there will be an infinite number of such consequences. The application of inference rules is generally guided by some control strategy. Unfortunately, trying to mathematically model control strategies is often not possible, due to their inherent complexity. Konolige therefore proposed simplifying the modelling process by assuming that an agent possessed a finite set of rules, which were applied exhaustively to the agent's base beliefs. An agent then believes ϕ if it is possible to derive ϕ from the agent's base beliefs using its deduction rules. This is perhaps still an overstrong model of resource bounded reasoners, but it at least manages to capture *some* incomplete reasoning, and has the advantage of simplicity.

The model of belief systems outlined above was formalized in the *deduction structure*. A deduction structure is a pair:

$$\langle \Delta, \rho \rangle$$

where

- Δ is a database of base beliefs: a countable set of formulae in some internal language of belief;
- ρ is a set of *deduction rules*.

A deduction rule is a rule of inference that must satisfy the following properties:

- the number of premises of the rule is fixed and finite;
- the rule is an effectively computable function of those premises.

A relation $\vdash_\rho$ is defined between sets of formulae and formulae of the internal language, for each set of deduction rules ρ.

$$\Delta \vdash_\rho \phi \text{ iff } \begin{array}{l} \text{there is a proof of } \phi \text{ from } \Delta \\ \text{using only the rules in } \rho \end{array}$$

The relation "$\vdash_\rho$" enjoys many of the properties of the proof relation ("$\vdash$"), including transitivity and reflexivity. The *closure* of a set of formulae under a set of deduction rules is given by the following function[16].

$$close(\langle \Delta, \rho \rangle) \triangleq \{\phi \mid \Delta \vdash_\rho \phi\}$$

[16] Called bel in Konolige's work.

Suppose d_i is the deduction structure of agent i, then belief is given the following meaning.

$$\begin{array}{rcl} \phi \in \mathit{close}(d_i) & — & i \text{ believes } \phi \\ \phi \notin \mathit{close}(d_i) & — & i \text{ doesn't believe } \phi \\ \neg\phi \in \mathit{close}(d_i) & — & i \text{ believes } \neg\phi \\ \neg\phi \notin \mathit{close}(d_i) & — & i \text{ doesn't believe } \neg\phi \end{array}$$

Note that it may be the case that an agent's belief system simultaneously satisfies the second and fourth conditions, thus modelling the possibility that an agent has no opinion on a formula. Note also that deductive closure, as modelled by the *close* function, is a weaker property than consequential closure.

Konolige defines two modal "external" languages for describing beliefs: L^B, which does not allow quantifying-in to modal contexts, and L^{Bq}, which does. The language L^B is essentially based on an internal language L, assumed to be a superset of classical first-order logic, augmented by an indexed set of unary modal operators $\{[i]\}$, for agents $i \in \{1, \ldots, n\}$. A formula $[i]\phi$ is read: "i believes ϕ".

A model for L^B is a model for first-order logic, augmented by a set of agents $\{1, \ldots, n\}$ and an indexed set of deduction structures $\{d_i\}$, for $i \in \{1, \ldots, n\}$. The semantics of modal formulae of L^B are given by the following rule.

$$M, \ldots \models [i]\phi \quad \text{iff } \phi \in \mathit{close}(d_i)$$

Let the deduction rules of agent i, with respect to some model, be given by $\rho(i)$. It is not difficult to see that the following axiom will be sound for all of Konolige's models.

$$[i]\phi_1 \wedge \cdots \wedge [i]\phi_n \Rightarrow \phi \text{ where } \{\phi_1, \ldots, \phi_n\} \vdash_{\rho(i)} \phi$$

This axiom justifies the following, derived rule of inference.

$$\begin{array}{rcl} \text{From} & \vdash & [i]\phi_1 \\ & \vdots & \\ & \vdash & [i]\phi_n \\ \text{and} & & \{\phi_1, \ldots, \phi_n\} \vdash_{\rho(i)} \phi \\ \text{infer} & \vdash & [i]\phi \end{array}$$

Konolige develops a tableaux-based decision procedure and resolution system for his logics; the resolution system makes the formalism attractive from a computational point of view.

Having developed the basic logic, Konolige goes on to develop two further operators: a common knowledge operator, and a *circumscription* operator. The common knowledge operator is based on the idea of a fictitious agent whose beliefs are those that "any fool would know". The agent name "0" is reserved for this agent. The semantics of the common knowledge operator are as follows.

$$M, \ldots \models [0]\phi \quad \text{iff} \quad \begin{array}{l} \phi \in \mathit{close}(d_i) \text{ and } [0]\phi \in \mathit{close}(d_i) \\ \text{for all } i \in \{1, \ldots, n\} \end{array}$$

Common knowledge is, therefore, only possible for agents capable of having beliefs about beliefs. The following axiom will obviously be sound.

$$[0]\phi \Rightarrow [i]\phi \wedge [i][0]\phi$$

The circumscription operator is unusual. It is written $\langle i{:}\,\Gamma\rangle\phi$, and is read: "$\phi$ is derivable from Γ in i's deduction structure". The semantic rule for this operator is:

$$M, \ldots \models \langle i{:}\,\Gamma\rangle\phi \quad \text{iff } \Gamma \vdash_{\rho(i)} \phi$$

To understand how this operator might be used, consider trying to represent the following statement in the modal language of belief:

> The only facts i knows about ϕ are Γ.

Using the circumscription operator, this statement can be formalized as follows:

$$\langle i{:}\,\Gamma\rangle\phi \Leftrightarrow [i]\phi$$

The forward implication tells us that i can prove ϕ from Γ, (that is, if i believed Γ, it would believe ϕ). The reverse implication says that it cannot believe ϕ unless it believes Γ. In effect, the operator allows us to circumscribe an agent's beliefs.

The deduction model is undoubtedly a simple model of belief. For AI researchers, accustomed to building agents with an architecture along the lines of that described by Konolige, the deduction model seems a far more

down-to-Earth semantics than possible worlds. However, the deduction model does have its critics. The commonest criticism is that deductive closure is still an overstrong property for an agent's belief system to have [138]. Konolige gives two rejoinders to this criticism. First, he outlines a method of modelling belief systems with a local cost bound on derivations [100, p24]. Second, he argues that the deductive closure property is needed in order to alleviate the need for modelling the proof trees, agendas, and so on, that would otherwise be required to model an agent's belief system. The implication is that a finer grained model of belief *could* be developed if required, for specific applications.

Another criticism of the deduction model is that it is *ad hoc*: modelling belief systems as sets of formulae seems a naive approach (see, e.g., [156]). This criticism seems unfounded. First, Konolige shows that the deduction model can be treated as a generalization of possible worlds semantics, and that any of the "standard" possible worlds systems can be modelled using the deduction model (see [100, Chapter 6]). Second, the deduction model was developed not as a model of *human* belief, but as a model of the beliefs of AI systems.

To conclude, while the deduction model is arguably too simplistic to capture the intricacies of human belief, its serves as an adequate model of the beliefs of AI systems, and is therefore well suited to the purposes of this thesis.

2.5.2 Werner's Formalism

In an extensive series of papers published since 1988, Eric Werner has proposed a formalism for modelling and reasoning about multi-agent societies, which draws on work in game theory, temporal logic, and intentional logics [178], [176], [177], [179], [180], [181], [182], [183]. A detailed technical review of Werner's formalism and his logic LT $\Box$CAN (or CANPLAN, as it is known in [182]) is presented in Appendix B.

Werner defines the cognitive state of an agent as a triple:

$$\langle I, S, V \rangle$$

where

- I is the information state of the agent;
- S is the intentional state of the agent;
- V is the evaluative state of the agent.

The information state of an agent is a set of partial histories, corresponding to epistemic alternatives. Each partial history represents one way the world could have developed, given what the agent knows.

The intentional, or plan state of an agent is a set of strategies. Each strategy represents one course of action the agent might follow. A strategy is a function from information states to a set of histories leaving the state. So a strategy (presumably) maps an information state to the states that might result if the strategy were followed.

The evaluative state of an agent represents the agent's preferences; it may be a function which assigns to a particular history a numerical value, called the utility of the history.

From this simple starting point, Werner investigates a number of aspects of cooperative activity, including speech acts [176], planning [182], social structure [179], and so on. Werner's theory of speech acts is reviewed briefly in the next chapter; the model of social structure is outlined in Appendix B.

The most interesting aspect of Werner's work has been to model an agent's intentional state as a set of strategies. This approach seems more flexible than the goal accessibility relation proposed by Cohen and Levesque (see above).

2.5.3 Situation Semantics

Situation semantics were originally developed as a formalism for natural language semantics by the philosophers Barwise and Perry [14]. The chief motivation for their work was a deep dissatisfaction with Montague's grammar, the then-prevailing formalism for natural language semantics, which is based on possible worlds semantics.

> "[W]e felt that the possible worlds point of view is dead wrong, deeply unsatisfactory, both philosophically and mathematically". [14, p13]
>
> "The assumption …built into …[possible worlds semantics] is that each world provides total information about the extension of every piece of language in that world, for all time". [14, p14]

Since the publication of *Situations and Attitudes*, there has been intense interest in situation semantics in the linguistics community, but comparatively little interest by the mainstream AI community. Situation semantics have recently been reworked to some degree by Devlin, who was interested in building a foundational theory of information [41].

Interesting and valuable though situation semantics undoubtedly are, it is not yet clear how they might be employed to reason about multi-agent systems. The subject is, therefore largely tangential to the interests of this thesis, and will be not be discussed in any further detail; the interested reader is referred to the cited works.

2.5.4 Singh's Formalism

In a series of papers, Munindar Singh has outlined various aspects of a theory of multi-agent systems [152], [153], [156]. His motivation appears to be similar to that of this thesis: to investigate methods for reasoning about multi-agent systems. However, the techniques he adopts are quite different. In his most detailed analysis of the cognitive structure of agents, ([156]), he develops a branching time logic of multi-agent systems which includes Believes and Intends operators, to describe the beliefs and intentions of agents. The semantics of these operators are based on a unified view of cognition, action, and communication called *discourse representation theory* (DRT). The goal of DRT is to provide a theory of discourse meaning that can "capture aspects of information typically encoded in natural language utterances" [156, p474]. Unfortunately, an examination of DRT is quite beyond the scope of this thesis.

2.6 Agency and AI: Building Agents

This thesis has, so far, been largely concerned with the question of how to *reason about* agents, when agents are viewed as intentional systems. This section will move away from such considerations, and turn instead to the question of how agents have been *built*: after all, the aim of the thesis is to develop formalisms for reasoning about systems composed of "real" computational agents.

The question of how to build agents is, of course, a very big one: in a sense, this is what the whole AI project is about. Not only is the question big, it is also vexed. There is an ongoing debate in AI about the best way to go about building intelligent agents. The protagonists in this debate may be broadly divided into two camps: the traditional, symbolic, *logicist* camp, and the alternative, *behavioural* camp (connectionists have been somewhat sidelined in this debate)[17]. In this section, some of the key arguments from each camp will be described.

2.6.1 Symbolic AI and Agency

The foundation on which the whole corpus of symbolic AI has been constructed is the *physical-symbol system hypothesis*, formulated by Newell and Simon[18]. A physical symbol system is defined to be a physically realizable set of physical entities (symbols) that can be combined to form structures, and which is capable of running processes which operate on those symbols according to symbolically coded sets of instructions. The physical-symbol system hypothesis then says that such a system is capable of general intelligent action.

It is a short step from the notion of a physical symbol system to McCarthy's dream of a *sentential processing automaton*, or *deliberate agent* (the term "deliberate agent" was introduced by Genesereth, [73, pp325–327], but is here used in a slightly more general sense). A deliberate agent is one which satisfies the following properties:

1. It contains an explicitly represented database of formulae in some logical language, representing its beliefs.
2. It operates on a continuous cycle of *observe* — *deliberate* — *act* $\cdots$.
3. Reasoning and deciding upon actions are based on some form of logical inference.

The deliberate agent is the ultimate aim of the pure logicist AI project, but in a looser sense, it is a description of the agent in classical AI generally. It is also worth pointing out that it is the beliefs of such systems that Konolige was trying to model in his deduction model of belief, (see earlier, and [100]).

[17] It is worth pointing out that while the members of each camp might not be in complete agreement with each other, they certainly don't agree with the members of the opposing camp.

[18] See [149] for a detailed discussion of the way that this hypothesis has affected thinking in symbolic AI.

> "Supporters of classical AI have, in general, accepted the physical symbol system hypothesis …[C]omplacent acceptance of this hypothesis, or some variant of it, led researchers to believe that the appropriate way to design an agent capable of finding its way round and acting in the physical world would be to equip it with some formal, logic-based representation of that world and get it to *do a bit of theorem proving*". [149, §3.2]

If one accepts this doctrine, then there are at least two important problems to be solved before one can build an intelligent agent:

1. The transduction problem: that of translating the real world into an accurate, adequate symbolic description of the world, in time for that description to be useful.

2. The representation/reasoning problem: that of representing information symbolically, and getting agents to manipulate/reason with it, in time for the results to be useful.

The former problem has led to work on vision, speech understanding, learning, etc. The latter has led to work on knowledge representation, automated reasoning, automated planning, etc. Despite the immense volume of work that the problems have generated, many people would argue that neither problem is anywhere near solved. Even seemingly trivial problems, such as commonsense reasoning, have turned out to be extremely difficult. A detailed exposition of specific problems would be quite beyond the scope of this thesis, so to illustrate some problems, work on AI planning is briefly reviewed.

It is widely accepted within the symbolic AI community that a *planner* will be a central component in the structure of an artificially intelligent agent. Briefly, a planner takes a *goal*, (something an agent want to achieve), and a symbolic representation of the current state of the world, (the agent's beliefs), and generates a plan of action, which, if the agent follows it, will achieve the goal. Planning, it is argued, it a central component of everyday activity, that every intelligent agent must engage in (see [75], [184]).

The first real planner was the STRIPS system, developed by Fikes in the late 1960s/early 1970s [57]. The two basic components of STRIPS were a model of the world as a set of formulae of predicate logic, and a set of *action schemata*, which describe the pre-conditions and effects of all the actions available to the planning agent. This latter component has proved to be STRIPS' most lasting legacy in the AI planning community: nearly all implemented planners employ the "STRIPS formalism" for action, or some variant of it. The STRIPS planning algorithm was based on a principle of finding the "difference" between the current state of the world and the goal state, and reducing this difference by applying an action. Unfortunately, this proved to be an inefficient process for formulating plans, as STRIPS tended to become "lost" in low-level plan detail.

In an effort to overcome this problem, Sacerdoti developed an *hierarchical* planner called ABSTRIPS, which works by sorting out high-level plan components before moving to low-level details [144]. Despite the performance improvements gained by hierarchical planning, the planning process remained time consuming. Further efforts to improve performance were made [167], [5]. The apotheosis of this work came with Chapman's work on TWEAK, a non-linear planner that was proved *correct*, in the sense that if it produces a solution the solution is correct, and if it signifies failure then no solution is possible [26]. However, in the course of his work, Chapman established some results which delivered something of a body-blow to the AI planning community. It has long been known that in the worst case, the planning problem is NP-hard, which is a discouraging result in itself. But Chapman showed that correct planning is, in the general case, also *undecidable*.

So planning — an activity long regarded as central to the classical model of artificially intelligent agency — turns out not only to be a computationally intractable problem, but for planners of even moderate sophistication, undecidable. It is perhaps this result, more than any other, which has led a number of researchers to reject the classical AI model of agency altogether and seek an alternative.

2.6.2 Alternative Approaches

Probably the most vocal critic of the symbolic AI notion of agency has been Rodney Brooks, a researcher at MIT. Brooks was originally a robotic engineer, who apparently became frustrated by AI approaches to building control mechanisms for autonomous mobile robots. In a 1985 paper, he outlined an alternative architecture for building agents, the so called subsumption architecture, and began his attack on symbolic AI [19]. The analysis of alternative approaches begins with Brooks' work.

In recent papers, ([21], [20]), Brooks has propounded three key theses:

1. Intelligent behaviour can be generated *without* explicit representations of the kind that symbolic AI proposes.

2. Intelligent behaviour can be generated *without* explicit abstract reasoning of the kind that symbolic AI proposes.

3. Intelligence is an *emergent* property of certain complex systems.

Brooks identifies two key ideas that have informed his research:

1. **Situatedness and embodiment.** "Real" intelligence is situated in the world, not in disembodied systems such as theorem provers or expert systems.

2. **Intelligence and emergence.** "Intelligent" behaviour arises as a result of an agent's interaction with its environment (a point borne out in practical terms by [165]). Also, intelligence is "in the eye of the beholder".

If Brooks was just a Dreyfus-style critic of AI, his ideas would probably not have gained much currency. However, to demonstrate the validity of his claims, he has built a number of robots, based on the *subsumption architecture*. A subsumption architecture is a hierarchy of task-accomplishing *behaviours*. Each behaviour "competes" with the others to exercise control over the robot. Lower layers represent more primitive kinds of behaviour, (such as avoiding obstacles), and have precedence over layers further up the hierarchy. It should be stressed that the resulting systems are, in computational terms, *extremely* simple, with no explicit reasoning, or even pattern matching, of the kind found in symbolic AI systems. But despite this simplicity, Brooks has demonstrated the robots doing tasks that would be extremely impressive if they were accomplished by symbolic AI systems. For example, one robot goes around an office building, looking for soda-pop cans, lifting them to see if they are empty, and discarding them if they are. Similar work has been reported by Steels, who described simulations of "Mars explorer" systems, containing a large number of subsumption-architecture agents, that can achieve near-optimal performance in certain tasks [165].

At about the same time as Brooks was describing his first results with the subsumption architecture, Chapman was completing his Master's thesis, in which he reported the theoretical difficulties with planning described above, and was coming to similar conclusions about the inadequacies of the symbolic AI model himself. Together with his co-worker Agre, he began to explore alternatives to the AI planning paradigm [27].

Agre's work is conceptually more sophisticated than thats of Brooks. He observed that most everyday activity is "routine" in the sense that it requires little — if any — new abstract reasoning. Most tasks, once learned, can be accomplished in a routine way, with little variation. Agre proposed that an efficient agent architecture could be based on the idea of "running arguments". Crudely, the idea is that as most decisions are "routine", they can be encoded into a low-level structure (such as a digital circuit), which only needs periodic updating, perhaps to handle new kinds of problems. His approach was illustrated with the celebrated Pengi system [3].

Another sophisticated approach is that of Rosenschein and Kaelbling, who propose describing an agent in intentional terms, and compiling this intentional description into a low-level device which realises the description [142], [143]. Their *situated automata* paradigm has attracted much interest (see, e.g., [151], [97]).

Several other researchers have described similar work, most noticeably the Phillips group, who have worked on "behaviour-based" systems [32], [175]. Some "hybrid" systems have also been described (e.g., [76], [96] [56]), which try to marry classical and alternative approaches, with some success.

2.7 Summary

This chapter began by trying to come to some definition of an agent. This was found to be difficult, as action-based analyses, and "high-level" analyses of agenthood proved not to be useful. An agent was instead identified with Dennett's notion of an *intentional system*: one that is most simply described in terms of the intentional notions of belief, desire, and so on. This then motivated a detailed analysis of formal methods for reasoning about the intentional notions. This analysis began with a look at why classical first-order logic is not appropriate to the task. It then discussed at length the so-called possible worlds semantics. Some problems (notably logical omniscience) were identified with the possible worlds approach, and several variations on the possible worlds theme were discussed. Meta-language approaches to reasoning about knowledge and belief were then reviewed. Finally, Konolige's deduction model of belief was reviewed, and found to be a good model of the beliefs of AI systems.

The focus of the chapter then turned from methods for reasoning about AI systems to how agents are actually built. The classical AI model of agency was found to be in accord with the deduction model of belief, where an agent is equipped with a set of formulae of some logic, (its beliefs), which constitute a description of its world. Objections to the classical model of agency — notably theoretical difficulties associated with activities generally

taken to be at the heart of the classical AI model of agency — were then discussed, and some alternative work was mentioned. However, since the focus of this thesis is on classical models of agency, this alternative work will not be discussed further.

This chapter has focussed largely on the "isolated" aspects of agency: the knowledge and beliefs of single agents. The aim of the next chapter is to move on to aspects of *social* agency, and in particular, methods for reasoning about communication, and building social agents.

Chapter 3

Social Agency and Distributed Artificial Intelligence

BY definition, an agent in a DAI system is *social*: it exists in an environment containing other agents, with which it will generally be expected to interact in some way. Agents in DAI are thus *situated* in a multi-agent environment; they are not disembodied, isolated intelligences, as the early AI theorem provers and expert systems were. Consequently, DAI is largely concerned with the *social* aspects of agency. It is these aspects of agency which are the object of study in this chapter.

The chapter is structured as follows. First, the notion of communication is examined. It seems almost axiomatic that social agents are able to communicate with their peers (though this is not universally assumed in DAI). The various ways that communication has been treated in the DAI literature are examined in the next section. This leads to the recognition of *speech acts* as the most widely used method for describing and reasoning about communication in DAI. Section 3.2 examines various formal and semi-formal speech act theories in detail, from their origins with Austin, to recent treatments by Cohen and Levesque.

Section 3.3 echoes the structure of the preceding chapter by examining the various ways that social agents have been built in DAI. The chapter closes with a summary.

3.1 Communication and DAI

It is widely accepted that communication plays a central role in DAI systems. Why is this? Consider the twin problems of *coordination*, (that of ensuring that the actions of agents in a system do not conflict, and are not self-defeating), and *coherence*, (that of ensuring that global system performance is satisfactory), both of which arise in any non-trivial DAI system (the best currently available account of these problems is [16, pp19–24]). The root cause of these problems is lack of information: it is not generally feasible for agents to maintain complete, correct, current knowledge of their environments. In short, no agent ever knows precisely what every other agent is doing. Instead, agents generally maintain partial knowledge (or, as we shall see, in some limited cases, no knowledge). The basic role of communication is to provide a means of exchanging information: plans, partial results, synchronization information, and so on.

Werner has identified five ways that communication has been treated in DAI [179, pp5–8][1]:

1. **No communication.** The simplest kind of communication is no communication. Agents might infer each other's plans without communicating them, [141], or they might employ a structured environment, which alleviates the need for direct communication [165]. Another alternative is a "pre-established harmony", where every agent operates according to some pre-determined scheme to achieve a goal, with no need to communicate at "run-time".

2. **Signalling.** The synchronization of multi-agent activity may be achieved by the use of semaphore-like signals [74].

[1] It is worth pointing out that the classic computer science problems of communication, (e.g., the problem of efficiently transmitting a packet of information from one node to another), are not considered within the scope of DAI. These "low level" problems are generally considered "solved" for DAI purposes.

3. **Computation as message passing.** Hewitt and Agha have developed a computational paradigm based on the notion of message passing [2]; the so-called *actor* systems contain large numbers of very fine-grained agents working together. More generally, many DAI systems are based on communication via message passing [70], [189].

4. **Plan passing.** The problems of coordination and coherence, mentioned above, have been tackled through the use of plan-exchange, particularly in the partial global planning paradigm [45], [43], [44].

5. **Speech acts.** The basic axiom of speech act theories is that communication is a form of action, in much the same way that lifting a block from a table is action. Speech act theories were developed by philosophers/linguists in an attempt to understand the workings of human communication.

The first possibility, (no communication), is only practicable in certain circumstances: inferring each others plans is too time consuming to be practicably useful, and structured environments are only feasible for certain, highly stylized tasks. Similar comments apply to signalling, which can be used for synchronization in the operating system sense, but little else.

The actor paradigm deals with fine-grained, massively concurrent systems, and is thus of limited interest to this thesis. Plan passing is a very specific technique for coordinating multi-agent activity and is, again, only of minor interest given the purposes of this thesis. The best developed framework for reasoning about communication is therefore speech act theories: such theories are reviewed in detail in the next section.

3.2 Speech Acts

Speech act theories, originally a fairly obscure issue in linguistic philosophy, are now a central component of AI and linguistic theory. Speech act theories are *pragmatic* theories of language: they deal with language *use*, in everyday situations, to achieve everyday ends. Pragmatics can be distinguished from syntax (to do with language form), and semantics (to do with language meaning).

The basic axiom of speech act theories is that utterances made by humans in everyday situations are *actions*, typically performed by a speaker with some intention, which change the state of the world in a way analogous to the way "physical" actions do. Since utterances do not change the state of the physical world in any obvious way[2], it seems appropriate to ask: What exactly *is* changed by an utterance?

Utterances are always performed in some *context*, and alter that context. This is in essence the basic idea of pragmatic theories of language. Suppose $\mathcal{L}$ is the set of all possible utterances of some language, and $\mathcal{C}$ is the set of all contexts. Then a pragmatic theory can be viewed as a function:

$$f: \mathcal{L} \times \mathcal{C} \rightarrow \mathcal{C}$$

or, equivalently:

$$f: \mathcal{L} \rightarrow \mathcal{C} \rightarrow \mathcal{C}$$

(see, e.g., [114, pp30-32], and the function *Prag* in [179, pp15–18], [176] and below). A complete theory must therefore account for:

1. The set $\mathcal{C}$ of all possible contexts.

2. The "internals" of the function f.

It is generally accepted that the context of an utterance is composed of the mental states of the utterance participants: their beliefs, desires, etc. This is consistent with the view of agents as intentional systems, discussed in the preceding chapter.

The following subsections analyze various developments in speech act theories: the treatment is adapted from [114], [9], [145], [146], [31], [30], [132], [8], [67], [179], [176], and [154].

3.2.1 Speech Acts *à la* Austin

The origin of speech acts theory is usually attributed to Austin [9]. In his 1962 book, *How To Do Things With Words*, he noted that in everyday spoken language, humans use a class of sentences which do not simply assert

[2] Ignoring pathological cases, (such as shouting and causing an avalanche), and microsopic effects.

some fact, but actually perform some action. Austin called such sentences *performatives*. He also noted that performatives can *fail* in the same sense that non-linguistic actions can fail.

Austin called the conditions required for the successful completion of performatives *felicity conditions*. He recognized three important felicity conditions:

1. (a) There must be an accepted conventional procedure for the performative.
   (b) The circumstances and persons must be as specified in the procedure.
2. The procedure must be executed correctly and completely.
3. The act must be sincere, and any *uptake* required must be completed, insofar as is possible.

To illustrate the first condition, consider someone saying: "You are exiled from this country". Unless the utterer has gone through all the legal procedures required to exile the hearer, and moreover, has the legal authority to do so, then this act is not a performative. This is because there is no conventional procedure whereby someone can say: "you are exiled", and make it so. The second part of the first condition might arise if, for example a clergyman baptized the wrong baby.

The second condition is relatively straightforward. The third is more subtle. It requires, for example, that agents must have the appropriate mental state to perform the act, and must carry through any required behaviour. For example, asking for something while not wanting it, or promising something while having no intention of carrying out the promise would contravene this condition.

Austin postulated that in uttering a sentence, an agent performs three types of action:

- **Locutionary acts.** A locutionary act is performed simply by uttering a syntactically acceptable sentence.
- **Illocutionary acts.** An illocutionary act is often performed with a performative verb. Some examples of performative verbs are: "request", "inform", "insist", "state", "demand", and "argue". Austin claimed that the English language contains over a thousand such verbs. Each performative verb has an associated *illocutionary force*. Many performatives have a similar (if not identical) illocutionary force (e.g., "inform" and "tell"). Performatives can often be identified by seeing whether they take the adverb "hereby" (e.g., "I hereby inform you that …").
- **Perlocutionary Acts.** The perlocutionary act is the bringing about of an effect on the hearer of the utterance.

The term *speech act* has subsequently become synonomous with the illocutionary act.

3.2.2 Speech Acts *à la* Searle

The next important stage in the development of speech acts theory came in 1969, with the publication of John Searle's book, *Speech Acts* [145] (further developed in [146]). The main thesis of Searle's work was that conversing in a language is a rule-governed form of behaviour. Searle attempted to formulate the structure of speech acts by deriving the set of "necessary and sufficient" conditions for their successful completion. For example, the conditions that Searle suggested must hold in order for the successful completion of a *request* act are listed below: a speaker (SPEAKER) is uttering a sentence, which is a request for a hearer (HEARER) to perform ACTION.

1. **Normal I/O conditions.** Normal I/O conditions state that HEARER is able to hear the request, (thus HEARER must not be deaf, …), the act was performed in normal circumstances (not in a film or play, …), etc.
2. **Preparatory conditions.** The preparatory conditions state what must be true of the world in order that SPEAKER correctly choose the speech act. In this case, HEARER must be able to perform ACTION, and SPEAKER must believe that HEARER is able to perform ACTION. Also, it must not be obvious that HEARER will do ACTION anyway.
3. **Sincerity conditions.** These conditions distinguish sincere performances of the request; an insincere performance of the act might occur if SPEAKER did not really want ACTION to be performed.
4. **Essential conditions.** The act was an attempt by SPEAKER to get HEARER to do ACTION.

Searle listed similar sets of conditions for asserting, questioning, thanking, advising, warning, greeting, and congratulating.

As observed above, some speech acts have a similar illocutionary force. This begs the question: is it possible to classify illocutionary acts according to some abstract typology? Searle believed this was an essential part of understanding speech acts. He proposed the following, five-point typology:

1. **Representatives.** A representative act commits the speaker to the truth of an expressed proposition. A paradigm case is *asserting*.
2. **Directives.** A directive is an attempt on the part of the speaker to get the hearer to do something. Paradigm case: *requesting*.
3. **Commissives.** Commit the speaker to a course of action. Paradigm case: *promising*.
4. **Expressives.** Express some psychological state (e.g., gratitude). Paradigm case: *thanking*.
5. **Declarations.** Effect some changes in an institutional state of affairs. Paradigm case: *declaring war*.

Levinson has criticized this typology on the grounds that it lacks a principled basis; a number of other typologies have been proposed [114, pp240–242]. Whatever the merits of such typologies, it does seem that there are at least three *sentence types* common to most languages: declaratives, imperatives, and interrogatives.

3.2.3 Speech Acts *à la* Cohen and Perrault

In a landmark paper published in 1979, Cohen and Perrault [31] took Searle's work and attempted to re-formulate his necessary and sufficient conditions using techniques adapted from mainstream AI work on planning. The result was the *plan based theory of speech acts* (PBTSA).

Cohen and Perrault pointed out that one model of humans is as planners, continually generating, debugging, and executing the plans that constitute their behaviour. They suggested that speech acts could be viewed as operators in plans, in the same way that physical actions are viewed as operators in planning. The aim of their work was then to develop a theory of speech acts:

> "[B]y modelling them in a planning system as operators defined …in terms of speakers and hearers beliefs and goals. Thus speech acts are treated in the same way as physical actions". [31]

In other words, the *context* defined by Cohen and Perrault, upon which speech acts operate, is the mental state of the participants, where the mental states are defined in terms of beliefs and desires. This model is therefore consistent with the view of agents as intentional systems.

Consider first the *Request* act: this act is fundamental. The aim of the *Request* act will be for a speaker to get a hearer to perform some action. Table 3.1 defines the *Request* act. This definition requires some explanation. Two preconditions are stated: the "cando.pr" (can-do pre-conditions), and "want.pr" (want pre-conditions).

The cando.pr states that for the successful completion of the *Request*, two conditions must hold. First, the speaker must believe that the hearer of the *Request* is able to perform the action. Second, the speaker must believe that the hearer also believes it has the ability to perform the action. The want.pr states that in order for the *Request* to be successful, the speaker must also believe it actually wants the *Request* to be performed. If the pre-conditions of the *Request* are fulfilled, then the *Request* will be successful: the result (defined by the "effect" part of the definition) will be that the hearer believes the speaker believes it wants some action to be performed.

While the successful completion of the *Request* ensures that the hearer is aware of the speakers desires, it is not enough in itself to guarantee that the desired action is actually performed. This is because the definition of *Request* only models the illocutionary force of the act. It says nothing of the perlocutionary force. What is required is a *mediating act*. Table 3.1 gives a definition of *Cause_to_Want*, which is an example of such an act. By this definition, an agent will come to believe it wants to do something if it believes that another agent believes it wants to do it. This definition could clearly be extended by adding more pre-conditions, perhaps to do with beliefs about social relationships, power structures, etc.

The *Inform* act is as basic as *Request*. The aim of performing an *Inform* will be for a speaker to get a hearer to believe some statement. Like *Request*, the definition of *Inform* requires an associated mediating act to model the perlocutionary force of the act. The definitions of *Inform* and its associated mediating act *Convince* are given in Table 3.1. The cando.pr of *Inform* states that the speaker must believe ϕ is true. The effect of the act will simply be to make the hearer believe that the speaker believes ϕ. The cando.pr of *Convince* simply states that the hearer must believe that the speaker believes ϕ. The effect is simply to make the hearer believe ϕ.

Request(S, H, α)		
PRECONDITIONS	CANDO.PR	(*S BELIEVE* (*H CANDO* α)) $\wedge$ (*S BELIEVE* (*H BELIEVE* (*H CANDO* α)))
	WANT.PR	(*S BELIEVE* (*S WANT request_instance*))
EFFECT		(*H BELIEVE* (*S BELIEVE* (*S WANT* α)))

Cause_to_Want($A1, A2, \alpha$)		
PRECONDITIONS	CANDO.PR	(*A*1 *BELIEVE* (*A*2 *BELIEVE* (*A*2 *WANT* α)))
	WANT.PR	$\times$
EFFECT		(*A*1 *BELIEVE* (*A*1 *WANT* α))

Inform(S, H, ϕ)		
PRECONDITIONS	CANDO.PR	(*S BELIEVE* ϕ)
	WANT.PR	(*S BELIEVE* (*S WANT inform_instance*))
EFFECT		(*H BELIEVE* (*S BELIEVE* ϕ))

Convince($A1, A2, \phi$)		
PRECONDITIONS	CANDO.PR	(*A*1 *BELIEVE* (*A*2 *BELIEVE* ϕ))
	WANT.PR	$\times$
EFFECT		(*A*1 *BELIEVE* ϕ)

Table 3.1: Definitions from the Plan-Based Theory of Speech Acts

Note that the *Inform* and *Request* acts are, to some degree, interchangeable. Suppose an agent SPEAKER wants some task ACTION to be performed by some agent HEARER. The *Request* act, described above, provides the obvious method for causing HEARER to carry out the task. Examination of the *Inform* act, however, illustrates at least one other method of achieving the same result. Suppose SPEAKER performed the following act:

Inform(*SPEAKER, HEARER,* (*SPEAKER WANT ACTION*))

By the definition of *Inform* given above, the successful completion of this act would cause HEARER to believe that SPEAKER believed it wanted ACTION to be performed. This in turn would satisfy the pre-condition part of *Cause_to_Want*. Issuing an inform of a want thus appears to achieve the same result as issuing a request: this point is examined in [31].

The definitions expounded above have assumed that the result of a speech act may be modelled completely as the "effect" part of its definition. However, by virtue of the fact that an agent has performed a speech act, it is possible to make inferences about the cognitive state of the speaker. For example, the hearer may reasonably infer that the pre-conditions of the act held when the speaker performed the speech act. This tells the hearer about the cognitive state of the speaker. Such inferences are called "side effects".

The PBTSA enjoyed some success. For example, it was used in Appelt's celebrated natural language generation system, KAMP [6], [7].

Appealing though this formulation of speech acts is, it does have its problems. One of these is that it seems to make the recognition of illocutionary force a necessary component of language use: a point stated explicitly by Appelt [7, p87]. But it seems extremely unlikely that humans classify acts before understanding them[3]. So while a taxonomy of speech acts is useful in its own right, it does not constitute an *explanation* of a dialogue. There seems to be general agreement that a deep theory of speech acts should instead be rooted in a theory of rational action: this is the starting point for the work of Cohen and Levesque, described in the next section.

[3]Though there might be situations where some explicit reasoning about the type of an act *is* necessary — perhaps when a hearer is unsure of whether a speaker is being ironic.

3.2.4 Speech Acts *à la* Cohen and Levesque

Cohen and Levesque have developed a theory which arguably represents the state of the art in the logical analysis of speech acts [30]. Their work proceeds from two basic premises:

1. Illocutionary force recognition is unnecessary.

   > "What speakers and hearers have to do is only recognize each others intentions (based on mutual beliefs). [W]e do not require that those intentions include intentions that the hearer recognize precisely what illocutionary act(s) were being performed". [30, p223]

2. Illocutionary acts are *complex event types*, and not primitives.

Given this latter point, one must find some way of describing the actions that are performed. But actions, as the previous chapter mentioned, are a very slippery concept. Recall Searle's example of Gavrilo Princip in 1914, simultaneously pulling a trigger, firing a gun, killing Archduke Ferdinand, and starting World War I. How many actions are being performed here? Is there just one action, being described in many ways, or are there a number of actions? Cohen and Levesque's solution is to use their logic of rational action, (described briefly in the preceding chapter, and in more detail in [29] and Appendix B), which provides a number of primitive event types, which can be put together into more complex event types, using dynamic-logic-style constructions. Illocutionary acts are then defined as complex event types.

Their approach is perhaps best illustrated by giving their definition of a request. Some preliminary definitions are required. First, *alternating belief*.

$$(\mathsf{ABEL}\ n\ x\ y\ p) \triangleq \underbrace{(\mathsf{BEL}\ x\ (\mathsf{BEL}\ y\ (\mathsf{BEL}\ x\ \cdots (\mathsf{BEL}\ x}_{n \text{ times}}\ p\ \underbrace{)\cdots)}_{n \text{ times}}$$

And the related concept of *mutual belief*.

$$(\mathsf{BMB}\ x\ y\ p) \triangleq \forall n \cdot (\mathsf{ABEL}\ n\ x\ y\ p)$$

Next, an *attempt* is defined as a complex action expression — hence the use of curly brackets, to distinguish it from a predicate or modal operator.

$$\{\mathsf{ATTEMPT}\ x\ e\ p\ q\} \triangleq \left[\begin{array}{ll} (\mathsf{BEL}\ x \neg p) & \wedge \\ (\mathsf{GOAL}\ x\ (\mathsf{HAPPENS}\ x\ e; p?)) & \wedge \\ (\mathsf{INTEND}\ x\ e; q?) & \end{array}\right] ?; e$$

In English:

> "An attempt is a complex action that agents perform when they do something (e) desiring to bring about some effect (p) but with intent to produce at least some result (q)". [30, p240]

The idea is that p represents the ultimate goal that the agent is aiming for by doing e; the proposition q represents what it takes to at least make an "honest effort" to achieve p. A definition of *helpfulness* is now presented:

$$(\mathsf{HELPFUL}\ x\ y) \triangleq \forall e \cdot \left[\begin{array}{l} (\mathsf{BEL}\ x\ (\mathsf{GOAL}\ y\ \Diamond(\mathsf{DONE}\ x\ e))) \wedge \\ \neg\,(\mathsf{GOAL}\ x\ \Box\neg\,(\mathsf{DONE}\ x\ e)) \\ \quad \Rightarrow\ (\mathsf{GOAL}\ x\ \Diamond(\mathsf{DONE}\ x\ e)) \end{array}\right]$$

In English:

> "[C]onsider an agent [x] to be helpful to another agent [y] if, for any action [e] he adopts the other agent's goal that he eventually do that action, whenever such a goal would not conflict with his own". [30, p230]

The definition of requests can now be given (note again the use of curly brackets: requests are complex event types, not predicates or operators):

$$\{\text{REQUEST } spkr\ addr\ e\ \alpha\} \triangleq \{\text{ATTEMPT } spkr\ e\ \phi\ (\text{BMB } addr\ spkr\ (\text{GOAL } spkr\ \phi))\}$$

where ϕ is

$$\Diamond(\text{DONE } addr\ \alpha) \wedge (\text{INTEND } addr\ \alpha\ [(\text{GOAL } spkr\ \Diamond(\text{DONE } addr\ \alpha)) \wedge (\text{HELPFUL } addr\ spkr)])$$

In English:

> A request is an attempt on the part of *spkr*, by doing *e*, to bring about a state where, ideally, 1) *addr* intends α, (relative to the *spkr* still having that goal, and *addr* still being helpfully inclined to *spkr*), and 2) *addr* actually eventually does α, or at least brings about a state where *addr* believes it is mutually believed that it wants the ideal situation.

By this definition, there is no primitive request act:

> "[A] speaker is viewed as having performed a request if he executes any sequence of actions that produces the needed effects". [30, p246]

In short, any event, of whatever complexity, that satisfies this definition, can be counted a request. Cohen and Levesque show that if a request takes place, it is possible to infer that many of Searle's preconditions for the act must have held [30, pp246–251].

Using Cohen and Levesque's work as a starting point, Galliers has developed a more general framework for multi-agent dialogue, which acknowledges the possibility for conflict [66].

3.2.5 Non-monotonic Reasoning, Belief Revision, and Speech Acts

In a 1987 paper, (reprinted as [132]), Perrault described how Reiter's default logic, ([139]), could be used to reason about speech acts. Perrault's insight was to see that a scenario containing communicating participants could be represented as a *default theory*. A default theory is a pair, containing a set of formulae called *assumptions* and a set of *default rules*: the key difference between default and classical logic is that default logics do not obey the monotonicity principle.

For reasoning about speech acts, the assumptions part of the theory contains an observer's knowledge of the world *before* an utterance is made; the rules part contains a small set of default rules capturing assumptions about the scenario. An example of such a rule is the *belief transfer rule*: if *x* believes *y* believes ϕ, and ϕ is consistent with *x*'s beliefs, then *x* will also believe ϕ. If the default theory is augmented by the statement that an utterance has been performed, and statements describing who is observing who, then, Perrault claims:

> "[E]ach extension …of the [resultant] default theory …describes a consistent view of the observer's knowledge of the world both before and after the utterance". [132, p170]

One of the positive features of Perrault's theory is that it can deal with situations where the pre-conditions for a speech act do not hold: for example, where a speaker utters an assertion about the world but does not, in fact, believe the assertion, and the hearer fails to detect the lie.

However, Perrault's theory has been criticized on a number of grounds [8, pp167-168]:

1. The theory predicts that an agent can come to believe any unsupported proposition simply by uttering it. This is at odds with our everyday experiences.
2. The theory does not adequately deal with belief revision.
3. The theory cannot easily be integrated with a theory of action.
4. The belief transfer rule could not easily be implemented.
5. The default logic used in the formalization is not easily implemented.

Motivated by these points, Appelt and Konolige proposed using hierarchic auto-epistemic logic (HAEL) ([102]) for reasoning about speech acts. HAEL is essentially an extension of auto-epistemic logic ([123]) developed in order to allow reasoning about priorities between default theories. Syntactically, auto-epistemic logic is a classical logic augmented by a single unary modal operator, L, with the formula $L\phi$ having the intended reading: "I believe ϕ". Using this operator, it is possible to write formulae expressing: "unless I believe …", and thus achieve simple default reasoning. In contrast to auto-epistemic logic, HAEL contains an indexed set of operators $\{L_i\}$, for referring to different *evidence spaces*, τ_i. The evidence spaces are arranged in a hierarchy, defined by an irreflexive, well-founded partial order, $\prec$. If $\tau_i \prec \tau_j$, then the evidence space τ_i is said to be *more specific* than τ_j. Information from lower level evidence spaces "filters up" to higher levels. The formula $L_i\phi$ is read: "ϕ is in the i'th level evidence space". An operator L_i can only be applied to evidence spaces *lower* in the hierarchy than i. So it is possible to write formulae expressing: "unless I have more specific information …". Priorities between defaults are incorporated by putting them at different levels in the hierarchy; the higher the level, the lower the priority. So, it is possible to write formulae for reasoning about prioritized defaults. Moreover, it is readily demonstrated that a HAEL theory has just one extension, making it a more tractable system for automation than Reiter's logic.

HAEL can be used for reasoning about speech acts in the following way:

> "[A]ssume that there is a hierarchy of autoepistemic subtheories …[$\tau_0 \prec \tau_1 \prec \tau_2$ …]. The lowest subtheory, τ_0, contains the strongest evidence about the speaker's and hearer's mental states. For example, if it is known to the hearer that the speaker is lying, this information goes into τ_0.
>
> In subtheory τ_1, defaults are collected about the effects of the speech act on the beliefs of both speaker and hearer. These defaults can be overridden by the particular evidence of τ_0. Together, τ_0 and τ_1 constitute the first level of reasoning about the speech act. At level 2, the beliefs of the speaker and hearer that can be deduced in τ_1 are used as evidence to guide defaults about nested beliefs, that is, the speaker's beliefs about the hearer's beliefs, and *vice versa*. These results are collected in τ_2. In a similar manner, successive levels contain the result of one agent's reflection upon his, and his interlocutor's beliefs and intentions at the next lower level". [8, p170]

The authors show how Perrault's results can be reconstructed by following this scheme, and also show that their proposal is not subject to many of the problems of Perrault's technique.

Finally, Galliers has proposed that *autonomous belief revision* plays an important role in cooperative interaction. The idea is that an agent has a *preference ordering* on cognitive states: given a number of different, possible cognitive states, an agent will choose to be in its preferred state. Agents thus have control over their cognitive state.

> "Revision of another's cognitive state …[is] the motivating force for communicative behaviour. But if there are no specialized rules dictating what is a cooperative response, the success of an utterance is not guaranteed. Autonomous agents may or may not comply with the recognized intended effects of an utterance on their cognitive states. …Strategic interaction acknowledges all participants sharing control over the effects of a communication. The aim in utterance planning is to determine one's own actions according to one's own goals and the context. *But*, this context includes the other agent and her autonomy over presumed existing mental states. *…Strategic planning to achieve a desired change in another's belief states is therefore a matter of setting the goal state based upon a prediction of the context of that other agent, such that the general rules of rational autonomous belief revision would then dictate the desired change anyway*". [67, p156]

3.2.6 Werner's Formalism and Speech Acts

In several papers, ([176], [179]), Werner has used his formalism for describing the cognitive states of agents (see preceding chapter and Appendix B) as a canvas on which to paint a theory of speech acts. Central to his theory is the notion of a *pragmatic interpretation function*, *Prag*. Suppose *Rep* is the set of all cognitive, or representational states of an agent, and L is the set of all formulae of some language. Then *Prag* is a function:

$$Prag: L \rightarrow Rep \rightarrow Rep$$

That is, a pragmatic interpretation function takes a formula of the language, and returns a function which maps a cognitive state to a cognitive state. Since Werner has already described the elements of *Rep* at some length, his theory concentrates for the most part on describing the properties of this function, for assertions and directives.

1. **Assertions.** "[T]he pragmatic interpretation of the sentence ϕ = 'Jon opened the door' is arrived at as follows: ϕ refers to the event of Jon opening the door. *Prag*(ϕ) is an operator on the hearer's information state I such that *Prag*(ϕ)I is the reduction of the set I to just those histories where the event occurred. The hearer A knows ϕ if ϕ holds in all the worlds in I. Thus, A comes to know that ϕ as a result of receiving and interpreting the message ϕ" [179, p16]. If an agent's information set I is viewed as a set of epistemic alternatives, as Werner implies by the above, then on receiving, "interpreting" and "accepting" a message, an agent comes to know the propositional content.

2. **Directives.** "For example, if ϕ = 'Open the door', ϕ refers to the situation of the addressee A opening the door. *Prag*(ϕ) operates on A's intentional state S_A such that A opens the door. *Prag* does this by removing all those possible plans of A that do not force ϕ. …Viewed constructively, a plan is incrementally built up by the *Prag* algorithm. The result is that the agent performs the directive in parallel with other goals he may have" [179, p17].

So, Werner's theory tells us that an agent comes to believe assertions, and intend to do directives. Werner appears to envisage his theory in a computational setting:

> "…*Prag* describes the *pragmatic competence* of an ideal speaker, and not the actual performance. He may for various reasons not accept the message. There is nothing in our account that would force the recipient of a command to automatically or mechanically obey the command. …But for him to understand the conventional meaning of the assertion or directive, the agent must know what the effect of the message is supposed to be if he were to accept it". [179, p17]

So before an agent "accepts" a speech act, it computes its force; the force of the act may, or may not, then be acted upon. Since it seems unlikely that Werner is proposing that humans operate in this manner, it seems reasonable to suppose that Werner's theory is computationally motivated. Sadly, Werner gives no more details.

3.2.7 Singh's Formalism and Speech Acts

Munindar Singh has recently done some work on:

> "[F]ormally describing the conditions of *satisfaction* for the different kinds of speech acts". [154, p69]

Singh identifies three ways in which a speech act can be *satisfied*:

1. **Extensional satisfaction.** A speech act is extensionally satisfied when its propositional content becomes true — for whatever reason.

2. **Whole-hearted satisfaction.** "The whole hearted satisfaction of a directive requires not only that the specified proposition be made true, but made true in a sure-fire manner". [154, p71]

3. **Relevant satisfaction.** "This requires that not only must the proposition in the speech act be made true in a sure-fire way, it must be made true *because* of that speech act". [154, p71]

Singh's formal definitions of the three types of satisfaction are based on his own theory of "know how" for multi-agent systems, ([152]), and a branching time temporal logic CTL^* [48]. He defines three operators, one for each of the three types of satisfaction. The formal definitions are quite simple, but too involved to give here; instead, we informally state the semantics for the three operators.

Extensional satisfaction. A directive is extensionally satisfied if its proposition eventually becomes true. An assertion is extensionally satisfied if its proposition is currently true.

Whole-hearted satisfaction. A directive is whole-heartedly satisfied if it is extensionally satisfied, and until it is extensionally satisfied, the hearer of the act knows how to achieve the propositional content, and intends the propositional content. The condition for assertions is the same as for extensional satisfaction.

Relevant satisfaction. A speech act is relevantly satisfied if it is whole-heartedly satisfied because the hearer chose it to be (as a result of the speech act).

It is not difficult to derive some theorems capturing various properties of satisfaction. The most obvious of these is:

$$\begin{array}{c}\text{Relevant}\\ \text{satisfaction}\end{array} \Rightarrow \begin{array}{c}\text{Whole-hearted}\\ \text{satisfaction}\end{array} \Rightarrow \begin{array}{c}\text{Extensional}\\ \text{satisfaction.}\end{array}$$

3.3 Social Agency and DAI: Building Social Agents

As in the preceding chapter, where the various AI approaches to building intelligent agents were examined, this chapter will now examine various DAI approaches to building intelligent *social* agents. The majority of such work has been firmly in the classical/symbolic AI camp, and only such systems will be examined here.

3.3.1 The Blackboard Architecture

The discipline of DAI is generally reckoned to have begun with work on the HEARSAY speech understanding system [55]. HEARSAY was the first computer system to employ a *blackboard* architecture. A blackboard system contains at least three components:

- a globally accessible data structure, called the *blackboard*;
- a set of demon-like *knowledge sources,* which constantly watch all or part of the blackboard;
- a central control mechanism, or protocol, which determines the order in which knowledge sources may access the blackboard.

Knowledge sources are the basic problem solving elements in a blackboard system. They do not communicate directly with each other; they only interact with the central control mechanism, and through that with the blackboard. Each knowledge source continually scans the blackboard, (or at least that portion of the blackboard to which it is allocated read access), and when it can make a contribution to the problem, it applies to the central controller for write access to the blackboard. If/when access is granted, the knowledge source writes its contribution on the blackboard. In this way, problem solving proceeds with each knowledge source trying to contribute partial solutions wherever possible.

Many variations on the basic blackboard structure are possible. For example, goal blackboards may be introduced to facilitate goal-directed reasoning. A good reference is [51], where a number of blackboard systems are described. Some attempts have been made to formalize the blackboard architecture; see [33].

As a problem solving architecture, the blackboard has much to commend it. However, the centralized controller imposes a severe bottleneck on problem solving, which is never truly distributed: there is only ever a single thread of control. Knowledge sources are not truly autonomous, but act more like collections of production rules. The notion of agency is thus not well developed in blackboard systems.

3.3.2 Beings

A slightly different approach was taken by Lenat, in his *beings* concept [110]. This work was essentially an attempt to simulate problem solving by a community of experts, each with domain expertise in distinct areas. Lenat hypothesised that the dominant problem solving strategy in such systems was a process of question/answer. Each expert is modelled as a *being,* (a relatively small fragment of LISP code with a structure described below). In order to manage problem solving, some beings act as experts in management tasks. To test out the beings concept, Lenat implemented PUP6, a program generator which contained about one hundred beings. Structurally, each being in PUP6 resembles a frame, with twenty seven slots (called *parts* in Lenat's terminology), each slot corresponding to a question that may be asked of the being. Beings are activated in a pattern directed fashion, and are able to communicate by broadcasting messages. Problem solving proceeds by beings receiving messages, matching them against their slots, which causes other messages to be broadcast, and so on.

The notion of agency is clearly better developed in beings than in blackboard systems, and the beings concept in many ways anticipates current work on cooperative problem solving. However, Lenat saw beings as a knowledge representation technique, not as a research paradigm in its own right, and the concept was not developed any further (though many of the ideas were refined and later used in the hugely successful AM project).

3.3.3 Actors

At about the time Lenat proposed beings, Hewitt was developing the actor paradigm [85], [86], a concept which received its fullest expression in Agha's 1985 thesis (published as [2]). Unlike beings, actors are still the subject of ongoing research both in AI and computer science generally. The actor paradigm influenced the early development of object-oriented systems, and the actor formalism can be thought of as a framework for concurrent object-oriented programming. The actor paradigm has a well-developed mathematical basis, which makes it of special interest to this thesis.

An actor system contains a set of actors, which remain passive until they receive a message. When an actor receives a message, it tries to match it against a *behaviour*[4]. The behaviour of an actor is so called because it determines the actor's response to the message, which may be one of three things:

- send messages to itself or other actors;
- create more actors;
- specify a *replacement* behaviour.

Message passing in the actor formalism is asynchronous, with agents identified by unique mail addresses; each actor has its own message queue, containing messages waiting to be processed. The next message to be processed will be dealt with by a *replacement* actor, which the original actor specifies itself. As soon as an actor specifies its replacement, the replacement can begin work on the next message. This feature, together with the ability to create new actors, makes for a computational paradigm with a high potential for concurrency: in principle, the only constraints on execution speed stem from the logical dependencies in the computation itself, and any hardware restrictions.

The actor paradigm is an elegant model of concurrent computation. However, actors do not comfortably map onto the idea of agents as propounded in this thesis: actors are very *fine grained*.

3.3.4 The Contract Net

Another innovation in the late 1970s was the *contract net* protocol, developed by Smith [159], [161]. The aim of the contract net was to provide a framework for efficient cooperative problem solving. A contract net system contains a number of autonomous agents, or *nodes*, which communicate through the contract net protocol to perform cooperative tasks. Each node in a contract net system is capable of making independent decisions about when to engage in problem solving, and thus has a high degree of autonomy. Frustratingly, Smith gives no information about the structure of contract net agents, and so a detailed analysis is not possible.

The contract net protocol is the subject of a case study in Chapter 6, where more details are given.

3.3.5 MACE

As a discipline, distributed AI "came of age" with the publication in 1987 of the proceedings of the fifth US workshop on DAI, held in 1985 [90]. In these proceedings was a paper describing in detail the Multi-Agent Computing Environment (MACE) developed by Gasser and colleagues [70]. MACE can, with some justification, claim to be the first general experimental testbed for DAI systems.

A MACE system contains five components:

- a collection of *application agents*, which are the basic computational units in a MACE system (see below);
- a collection of pre-defined *system agents*, which provide services to users (e.g., user interfaces);
- a collection of facilities, available to all agents, (e.g., a pattern matcher);
- a *description database*, which maintains agent descriptions, and produces executable agents from those descriptions;
- a set of *kernels*, one per physical machine, which handle communication and message routing, etc.

Gasser *et al* identify three aspects of agents: they contain knowledge, they sense their environment, and they perform actions [70, p124]. Agents have two kinds of knowledge: specialized, local, domain knowledge, and *acquaintance knowledge* — knowledge about other agents. An agent maintains the following information about its acquaintances [70, pp126–127]:

- **class:** agents are organized in structured groups called *classes*, which are identified by a class name;
- **name:** each agent is assigned a name, unique to its class — an agent's address is a $\langle class, name \rangle$ pair;
- **roles:** a role describes the part an agent plays in a class;

[4] In early work, behaviours were called *scripts*.

- **skills:** skills are what an agent knows are the capabilities of the modelled agent;
- **goals:** goals are what the agent knows the modelled agent wants to achieve;
- **plans:** plans are an agent's view of the way a modelled agent will achieve its goals.

Agents sense their environment primarily through receiving messages. An agent's ability to act is encoded in its *engine*. An engine is a LISP function, evaluated by default once on every scheduling cycle. The only externally visible signs of an agent's activity are the messages it sends to other agents. Messages may be directed to a single agent, a group of agents, or all agents. The interpretation of messages is left to the programmer to define.

Gasser *et al* describe how MACE was used to construct blackboard systems, a contract net system, and a number of other experimental systems (see [70, pp138–140], [72]).

MACE has proved to be a hugely influential system. A number of subsequent systems have taken onboard many of MACE's innovations, (e.g., [164], [189]). The notion of autonomous agency is well defined in MACE, and the structure of agents is described more explicitly than in previous work.

3.3.6 After MACE

Since MACE, many testbeds for DAI have been described. For example, Doran *et al* describe MCS/IPEM [42]. The MCS/IPEM system is a multi-agent testbed built around the sophisticated IPEM non-linear hierarchical planner [5]. Each agent in an MCS/IPEM system has a database of beliefs, (represented as PROLOG facts), a set of operators, (corresponding to actions the agent can perform), and some "demons" (basically condition/action pairs, the action being performed when the condition is believed). At runtime, agents have goals, which they continually strive to achieve. A single "cycle" for an agent is a flaw-fix plan cycle of the IPEM planner, to which agents have virtual access. Limited communication is possible, by agents writing goals or facts on each other's databases. Agents thus have a simple structure, but have access to a powerful planner, making MCS/IPEM somewhat unusual in DAI terms.

Other recent testbeds for DAI are AGENT0, and the agent-oriented programming paradigm, [151], [171], and Concurrent METATEM [62]. Both of these systems are the subject of case studies in Chapter 6, where they are described in detail.

3.4 Summary

Whereas Chapter 2 examined essentially isolated aspects of agency, this chapter has examined aspects of *social* agency. Although it is not *necessarily* the case that agents must communicate in order to cooperate, this is widely assumed in DAI. The chapter thus began by examining the various ways that communication has been treated in DAI. This led to the recognition of speech act theories as being the dominant paradigm for reasoning about communication in DAI. A detailed examination of various speech act theories was then presented. Finally, various attempts to build DAI systems were discussed.

At this stage, it is worth making the point that, (as in AI generally), there is some distance between theory and practice in DAI. In particular, with the exception of the actor paradigm, there have been few attempts to incorporate formal theories of communication into real DAI systems. This situation is changing: Shoham's agent-oriented programming framework claims to be based on a theory of agency and speech acts [151]. Also, Cawsey *et al* describe a distributed expert system which employs Galliers' theory of autonomous belief revision [25]. However, this work is still at an experimental stage.

This concludes the literature survey. The next part of the thesis is concerned with constructing an abstract model of agents and DAI systems, and a number of logics for reasoning about such systems.

Part III

A Theory of Computational Multi-Agent Systems and Its Temporal Logics

Chapter 4

A Theory of Computational Multi-Agent Systems

THIS chapter presents a formal theory of computational multi-agent systems. The theory contains two components. The first is a *model* of multi-agent systems; the second is a set of *execution models*, each of which defines one way that agents can act and interact.

The chapter is structured as follows. The next section is an informal introduction to most of the ideas that are later developed in the chapter. Some assumptions are then described, and some comments are made on the techniques used in the presentation of the theory. Following this introductory material, the model of multi-agent systems is introduced. First, a model of agents is defined, by taking each component of an agent (belief, action, communication) in turn. Next, systems are defined to be a collection of named, interacting agents. Two execution models for multi-agent systems are then defined: a simple synchronous model, and a more realistic interleaved execution model. The chapter concludes with a brief look at some ways in which the basic theory may be adapted and extended.

4.1 Setting the Scene

It is important to understand what the theory developed in this chapter is and is not intended to be. The theory is emphatically *not* intended to model *human* social systems. The phenomena of human belief, communication and action are *not* the objects of study in this thesis.

The theory *is* intended to be a plausible formal model that captures the key features of a wide range of classical DAI agents and systems. This statement requires some qualification, however. Even a superficial reading of the literature demonstrates that everyone who builds a DAI system has their own ideas about what agents are, how to deal with communication, what beliefs agents have, and so on. Developing a canonical model of DAI systems is, therefore, out of the question (at least at the moment). *The best we can hope for is an idealized model which captures the most important properties of a wide range of systems.* This is what the theory described herein is intended to do.

Note that the theory is, of necessity, fairly coarse grained. However, it may be readily adapted, extended and refined, to model the features of specific systems. Some examples, illustrating how the basic theory can be adapted, are presented in section 4.6.

So what are the key features of a DAI system, that we hope to capture in the theory? Begin with *agents*. As Chapter 2 observed, workers in a variety of disciplines have made liberal use of the term "agent", and so we have an obligation to explain our usage of it.

First, agents have significant (but finite) computational resources. They are thus distinct from the fine grained neuron-like computational agents of connectionism, the algebraic agents of CCS [120], and actors [2].

Agents also have a set of explicitly represented *beliefs*, and are able to reason about these beliefs in accordance with the computational resources afforded them. These beliefs are generally taken to represent, for the believer, a model of the environment in which the believer resides (cf. the acquaintance models of MACE [70] and ARCHON [185], and beliefs of MCS/IPEM agents [42]). Beliefs are expressed in a well defined cognitive, or internal language, which shall be called L throughout this thesis. For the moment, no assumptions are made about L, other than that it is at least a *logical* language. The nature and role of L are explored in a later chapter. Agents

are also assumed to have some ability to reason about their beliefs: given a base set of beliefs, they will typically derive some of the logical consequences of this set. They will not generally derive all the logically possible ones: they are thus resource-bounded reasoners.

A model of belief with these properties is the *deduction model* developed by Konolige (see [99], [100], and Chapter 2). The deduction model is adopted virtually wholesale for the model of multi-agent systems. The term *cognitive state* will occasionally be used to refer to the set of beliefs possessed by an agent.

In addition to being believers, agents can *act*. Three types of action are distinguished (cf. [23]):

- *cognitive* actions correspond to an agent employing its own computational resources. An example might be a database agent performing a "retrieve" operation. The distinguishing feature of cognitive actions is that they are private, in that they cannot be observed by other agents. Also, the agent performing a cognitive act has "control" over it; it is not possible for another agent to "interfere" with the act, and prevent its successful completion.

- *communicative* actions correspond to sending messages to other agents. An example might be an agent sending another agent a message containing a request for some piece of information. It is through the communicative act that agents are able to affect the cognitive state of their peers. In contrast to cognitive acts, the effect of a communicative act is not under the complete control of the actor (i.e., the agent sending the message). So while an agent can *predict* what effect a communicative act will have, and plan its actions based on such predictions, the *actual* effect is under the control of the agent receiving the message.

- *effectoric* acts are performed in the "real world", and correspond to what are more normally regarded as actions. An example might be a robot agent lifting a block from a table. Such actions are not interference-free.

A philosopher might find this typology dubious; after all, an axiom of speech act theory is that an agent requesting or informing is performing an action just like any other. However, the typology suits the purposes of this thesis. It arises when one considers the *domain* of each action type. The domain of a cognitive act is the actor's own cognitive state; the domain of a communicative act is the cognitive state of the message recipient. Both of these domains will be well defined in our formalism, and so we can achieve an effective formalization of cognitive and communicative acts. In contrast, the domain of an effectoric act is the physical world, which extends beyond any one agent's unique control. It has proven extraordinarily difficult to devise realistic, tractable formalizations of effectoric acts performed in multi-agent environments, and for this reason they are not considered by this thesis (see, e.g., [109] for an attempt to formalize multi-agent scenarios where agents can perform effectoric acts).

Note that we are by no means unique in making a distinction between the different types of act: Shoham [151, p25] distinguishes private/cognitive and communicative acts and, (in his AGENT0 system at least), does not consider effectoric acts. In reactive systems research, (e.g., [12]), a distinction is made between the component part of a system, which is under the unique control of the agent, and the environment part of a system, over which the agent has at best partial control.

Agents can therefore perform two types of action: cognitive and communicative. They will possess a repertoire of possible actions.

A formalization of cognitive actions is straightforward. A cognitive action can be viewed as a function taking an agent's belief set as its argument. The result is an *epistemic input* [69]: a "new piece of evidence" which may be incorporated into subsequent belief sets. The mechanism via which epistemic inputs are incorporated into belief sets is described below.

A formalization of communicative acts is more problematic, since, as the preceding chapter points out, formal models of communication are rarely adopted within real DAI systems. In the theory of multi-agent systems, the communicative act is modelled as the exchange of messages whose content is a formula of some well defined common communication language. It is mostly assumed that the communication language is the same as — or at least a subset of — the internal language L. The effect of messages on their recipients is modelled via an *interpretation*. The idea is that a message is interpreted in the context of the recipient's cognitive state; the result is an epistemic input. An interpretation can be modelled as a function from belief sets and messages to epistemic inputs. Each agent will possess its own interpretation. While this scheme is perhaps somewhat arbitrary, it seems general enough to describe many models of communication. The idea of an interpretation is based on Werner's "pragmatic interpretation function", (*Prag*), described in the preceding chapter. Note that by this scheme, agents have autonomy over their cognitive state; this is consistent with Gallier's theory of autonomous belief revision [67].

Finally, to the processing of epistemic inputs. The idea is to give each agent a *belief revision function*. A belief revision function takes a belief set and a set of epistemic inputs and returns a new belief set. The idea of a belief revision function is similar to Gärdenfors' definition of an epistemic commitment function (see [69]).

To summarize, the model of agents has the following key features:

- agents maintain a set of beliefs, in the style of Konolige [100];
- agents can perform any of a repertoire of communicative actions (sending messages), and cognitive actions (using their own computational resources);
- the effect of a communicative act is modelled via an interpretation — each agent possesses an interpretation;
- the result of performing an action is an epistemic input: cognitive actions cause epistemic inputs for the actor — communicative actions cause epistemic inputs for the message recipient;
- epistemic inputs are incorporated into an agent's belief set via a belief revision function.

Having developed a model of agents, one is faced with the problem of defining an *execution model*, which says how the activity in a system composed of a number of such agents may proceed. Unfortunately, this involves modelling *concurrency*. This is by no means a trivial problem. The modelling of concurrency is very much an ongoing research area in its own right (see, e.g., [120]). In this thesis, we will ignore the difficult issue of "true" concurrency, and make do with unrealistic but simple execution models, which will allow us to focus on the properties of agents without requiring us to become bogged down in the deep theoretical issues associated with concurrency (see the comments on future work in Chapter 8).

Two execution models are defined. The first is a simple synchronous model, where each agent is considered to be acting simultaneously. The second is a finer grained interleaved model, where agent's actions are interleaved. Although the interleaved model is more realistic, it is more difficult to manage formally, and only the synchronous model is used in the remainder of the thesis.

4.2 Some Assumptions

Any formal theory that purports to model something existing in the real world must to some extent involve abstraction. This abstraction typically involves making assumptions about how unimportant, irrelevant, or technically intractable portions of the domain operate. The main assumptions made in the theory are as follows:

- **No model of control.** Control schemes for agents are generally extremely complex, with all but the simplest defying attempts at formalization. Moreover, the huge range of control schemes, (from production rule systems, to simple STRIPS-style planners [57], to interleaved plan/execute systems [5], and intelligent reactive systems [94]), prevent any unified description. For these reasons, no attempt is made to explicitly model control within the theory of multi-agent systems: agents are assumed to choose actions via some anonymous decision procedure (cf. [77]). However, there is nothing to prevent specific control schemes being modelled and grafted onto the basic theory presented here. Also, some control schemes can be effectively modelled using the logics developed in subsequent chapters; some examples are given in Chapter 6.
- **Standard names.** A *name* is a symbol used to denote some object. Both the observers of a system, and the agents within the system will use names. There are a number of technical problems associated with names. For example, what happens when:
  - one object has different names — perhaps within the same agent;
  - agents don't have names for all objects.

  This problem is essentially that of *fluent expressions*, mentioned in the review of quantified modal logics in Chapter 2. The solution adopted here is to assign each object a unique *standard name*, and demand that *only* standard names are used to refer to objects. Standard names correspond closely to rigid designators.

4.2.1 A Comment on Notation

Before proceeding with the technical details of the theory, a brief comment on notation. The mathematical parts of the theory are presented using techniques based on the VDM specification language — the first seven chapters of [93] cover all the required material. However, anyone familiar with basic set notation and logic should have no difficulty in understanding the notation; a summary is given in Appendix A. In order to further aid comprehension, a summary of types appears at the end of the chapter.

4.3 Agents

This section introduces the attributes of agents in four parts: first belief, then communication, and cognitive (internal) actions. Finally, the components of an agent are put together in an agent architecture.

4.3.1 Belief

As mentioned above, the deduction model of belief is adopted virtually wholesale for the model of agents (see [100] and Chapter 2). Beliefs are expressed in an *internal* language L, which is assumed to be a logical language. A type is introduced for belief sets.

$$Belset = \text{powerset } Form(L)$$

So a belief set is just a set of formulae of the internal language, L. The type for deduction rules is *Drule*. The symbol Δ, (with appendages: $\Delta_0, \Delta', \ldots$), is generally used to denote a belief set, and the symbol ρ is generally used to denote a set of deduction rules. It is assumed that as L is a logical language, the proof relation "$\vdash$" is well defined for L. For each set of deduction rules ρ, a relation $\vdash_\rho$ is defined.

Definition 3 *Let Δ be a set of formulae, ϕ a formula, and ρ a set of deduction rules, all of the logical language L. Then $\Delta \vdash_\rho \phi$ iff there is a proof of ϕ from Δ using only the rules ρ.*

Note that if ρ is complete with respect to the logical language in question (L), then $\vdash_\rho$ is equivalent to $\vdash$ (since everything provable using $\vdash$ will be provable using $\vdash_\rho$). The closure of a set of formulae under some deduction rules is given by the following function.

$$close : Belset \times \text{powerset } Drule \rightarrow Belset$$
$$close(\Delta, \rho) \quad \triangleq \quad \{\phi \mid \Delta \vdash_\rho \phi\}$$

Changes in an agent's belief state are caused by *epistemic inputs*. An epistemic input could be defined to be anything appropriate, but is here defined to be a set of formulae of the internal language. (Alternative definitions of epistemic inputs are discussed at the end of the chapter.)

$$Epin = \text{powerset } Form(L)$$

Agents are able to *revise* beliefs in order to accommodate new information. A rule determining a new belief set for every belief set and set of epistemic inputs is called a *belief revision function* (BRF). A BRF has the following type.

$$Brf = Belset \times \text{powerset } Epin \rightarrow Belset$$

The symbol β is generally used to denote a belief revision function. We will say a belief revision function is *consistent* if it never produces a logically inconsistent belief set.

4.3.2 Communication

Communication is modelled as the exchange of messages whose content is a formula of some well defined common communication language (generally assumed to be the internal language L). In the basic theory, it is assumed that messages are sent point-to-point, rather than broadcast (but see the comments on extensions to the basic theory, at the end of this chapter). Some method is required to identify agents uniquely, in order that messages can be "routed" to the intended recipient. The method chosen is to assign each agent an *agent identifier*, (or agent id). A type is introduced for agent ids.

$$Agid = \text{an arbitrary countable set.}$$

The symbols *i*, *j*, *k* and *l* are used for agent ids.

A *message* is defined to be a triple of sender, recipient, and content: the sender and recipient are agent ids, the content is a formula of the communication language L. Self addressed messages are not allowed (otherwise an agent could alter its own cognitive state by sending itself a message).

$$Mess = Agid \times Agid \times Form(L)$$

The symbol μ, (with appendages: $\mu', \mu_1, \ldots$), is used to denote a message. Two selector functions are assumed: the function *sender* takes a message and returns the id of the sender; the function *recvr* takes a message and returns the id of the receiver. The formal definitions of these functions are trivial, and are therefore omitted. Using these functions, the invariant condition for the message type (that agents cannot send messages to themselves) can be easily defined.

$$\forall \mu \in Mess \cdot (sender(\mu) \neq recvr(\mu))$$

A *message interpretation* is a function that takes a belief set and message, and returns an epistemic input.

$$Messint = Belset \times Mess \rightarrow Epin$$

The usual symbol for an interpretation is ι. Finally, a set $Mess_{\mathsf{nil}}$ of *nil* messages is assumed. The idea is that a nil message can be sent by any agent at any time, and will be guaranteed not to be received by any other agent. Nil messages thus do not affect the state of a system, and provide a conveniently simple method of modelling agents "doing nothing".

4.3.3 Action

Each agent possesses a repertoire of actions, both communicative and cognitive[1]. However, it is not always the case that all actions are *applicable* given an agent's cognitive state. Each action is therefore associated with a *condition*, represented as a formula of the internal language. If the condition is believed, then the associated action is applicable. A condition/action pair is called a *rule*. The idea of associating actions with conditions is quite common. For example, in the AOP paradigm, Shoham associates each private (cf. cognitive) action with a "mental condition" which plays precisely the same role as our conditions [151, p34]. And of course this is the standard treatment of action in the AI planning community (see e.g., [57]).

An agent possesses a number of rules, which implicitly define the actions available to it. An obvious condition to demand of rule sets is that they should be *weakly complete*, in the following sense: there should always be at least one applicable action/message available to every agent, whatever its beliefs. Presented with a number of applicable actions, an agent must choose between them. The question of how to choose then arises. This problem is not a concern of this thesis; see the comments earlier.

Our task is now to formalize these ideas. First, the type for actions.

$$Action = Belset \rightarrow Epin$$

The symbol α is generally used to denote an action. To model an agent "doing nothing", a *nil action* is assumed. This action produces an empty epistemic input, for all arguments. The idea is that this action can be used to model an agent "doing nothing".

A rule is a condition/action pair: a condition is either a formula of L, or the special condition, "true". The condition "true" is always satisfied, and is used for actions that are always applicable. The type for conditions is therefore as follows.

$$Cond = Form(L) \cup \{\mathsf{true}\}$$

Two types of rule are defined, (cognitive action rules and message action rules), one for each type of action. For convenience, "cognitive action rule" is usually abbreviated to "action rule", and "message action rule" is abbreviated to "message rule". The types are as follows.

$$Arule = Cond \times Action$$

$$Mrule = Cond \times Mess$$

The usual symbol for an action rule is *ar*; for a message rule, *mr*. The following boolean-valued function defines what it means for an action rule to be *applicable*, given some belief set.

[1] We will usually abuse terminology by calling a cognitive action an action, and a communicative action a message. Context should always make meaning clear, however.

$$ar\_applic : Arule \times Belset \rightarrow \mathbb{B}$$
$$ar\_applic(\langle \phi, \alpha \rangle, \Delta) \triangleq \phi \in (\Delta \cup \{\mathsf{true}\})$$

So an action rule will be applicable if its condition is believed, or true. A similar function, *mr_applic*, is assumed to be defined for message rules (the formal definition is essentially identical to that of *ar_applic*, and is therefore omitted). The following boolean valued function defines what it means for an action rule to be *sound* (compare this definition with that of soundness in [115]).

$$sound : Arule \rightarrow \mathbb{B}$$
$$sound(\langle \phi, \alpha \rangle) \triangleq \forall \Delta \in Belset \cdot ar\_applic(\langle \phi, \alpha \rangle, \Delta) \Rightarrow (\Delta \in \mathsf{dom}\, \alpha)$$

In general, an agent will possess a range of possible actions, represented by a set of action rules and a set of message rules. The symbol *AR* is used to denote a set of action rules; the symbol *MR* is used to denote a set of message rules.

An important property to demand of an agent's action/message rules is *weak completeness*: whatever the beliefs of an agent, it always has at least one action it can perform and message it can send. The following boolean valued function defines when a set of action rules is weakly complete[2].

$$ar\_wk\_cmplt : \mathsf{powerset}\ Arule \rightarrow \mathbb{B}$$
$$ar\_wk\_cmplt(AR) \triangleq \forall \Delta \in Belset \cdot \exists ar \in AR \cdot ar\_applic(ar, \Delta)$$

Once again, a similar function, *mr_wk_cmplt*, is assumed for message rule sets. The formal definition is essentially identical to that of *ar_wk_cmplt*, and it is therefore omitted.

The following boolean valued function says of any action whether it is *legal*, given a set of action rules and a belief set.

$$ac\_legal : Action \times \mathsf{powerset}\ Arule \times Belset \rightarrow \mathbb{B}$$
$$ac\_legal(\alpha, AR, \Delta) \triangleq \exists \langle \phi, \alpha' \rangle \in AR \cdot (\alpha = \alpha') \wedge ar\_applic(\langle \phi, \alpha' \rangle, \Delta)$$

The function *ms_legal* is assumed for messages and message rule sets; the definition is essentially identical, and is omitted.

Finally, it seems sensible to demand *honesty* of message rule sets; agents are not allowed to send messages which "lie" about the sender. Honesty is given by the following boolean valued function.

$$honest : Agid \times \mathsf{powerset}\ Mrule \rightarrow \mathbb{B}$$
$$honest(i, MR) \triangleq \forall \langle \phi, \mu \rangle \in MR \cdot (sender(\mu) = i)$$

4.3.4 Agent Architecture

The type for agents is called *Agent*. It is defined as follows.

Definition 4 *An agent is a structure:*

$$\langle \Delta_0, \rho, \beta, \iota, MR, AR \rangle$$

where

- $\Delta_0 \in Belset$ *is an initial belief set;*
- $\rho \subseteq Drule$ *is a set of deduction rules for* L*;*
- $\beta \in Brf$ *is a belief revision function;*
- $\iota \in Messint$ *is a message interpretation function;*
- $MR \subseteq Mrule$ *is a set of message rules such that*

  $mr\_wk\_cmplt(MR)$;

[2] *Strong* completeness would demand that there was always precisely one available action/message.

1. Interpret any messages received.
2. Update beliefs by processing epistemic inputs resulting from:
   - previous action;
   - message interpretation

   through the belief revision function.
3. Derive deductive closure of belief set.
4. Derive set of possible messages, choose one and send it.
5. Derive set of possible actions, choose one and apply it.
6. Goto (1).

Figure 4.1: Operation of Agents

- $AR \subseteq Arule$ *is a set of action rules such that*

  $ar\_wk\_cmplt(AR) \wedge \forall ar \in AR \cdot sound(ar)$.

The operation of an agent is summarized in Figure 4.1 (cf. the "basic loop" of AOP systems [151, p22]).

4.4 Systems

A group of named agents is called a *system.* The type for systems is called *System*, and is given in the following definition. (The definition abuses notation somewhat, for example by using ρ to denote a map from agent ids to sets of deduction rules, rather than — as before — a set of deduction rules. However, the abuse helps improve readability, and meaning will always be clear from context.)

Definition 5 *A system is a structure:*

$$\langle Ag, \Delta_0, \rho, \beta, \iota, MR, AR \rangle$$

where

- $Ag \subseteq Agid$ *is a countable set of agent ids;*
- $\Delta_0 = Ag \xrightarrow{m} Belset$ *maps each element of Ag to an initial belief set;*
- $\rho = Ag \xrightarrow{m} powerset\ Drule$ *maps each element of Ag to a set of deduction rules;*
- $\beta = Ag \xrightarrow{m} Brf$ *maps each element of Ag to a belief revision function;*
- $\iota = Ag \xrightarrow{m} Messint$ *maps each element of Ag to a message interpretation function;*
- $MR = Ag \xrightarrow{m} powerset\ Mrule$ *maps each element of Ag to a set of message rules;*
- $AR = Ag \xrightarrow{m} powerset\ Arule$ *maps each element of Ag to a set of action rules*

such that

$$\forall i \in Ag \cdot \langle \Delta_0(i), \rho(i), \beta(i), \iota(i), MR(i), AR(i) \rangle \in Agent$$

and

$$\forall i \in Ag \cdot honest(i, MR(i)).$$

$$\sigma_0 \xrightarrow{\tau_1} \sigma_1 \xrightarrow{\tau_2} \sigma_2 \xrightarrow{\tau_3} \sigma_3 \xrightarrow{\tau_4} \cdots \xrightarrow{\tau_u} \sigma_u \xrightarrow{\tau_{u+1}} \cdots$$

Figure 4.2: States and Transitions

It is convenient to define a function which takes an agent id and system, and extracts the agent associated with the id from the system.

$$agent : Agid \times System \to Agent$$
$$agent(i, sys) \triangleq$$
$$\quad \text{let } \langle Ag, \Delta_0, \rho, \beta, \iota, MR, AR \rangle = sys \text{ in}$$
$$\quad \langle \Delta_0(i), \rho(i), \beta(i), \iota(i), MR(i), AR(i) \rangle$$

4.5 Execution Models

The aim of this section is to define two *execution models* for multi-agent systems, which show how a group of agents with the structure described above can operate and interact with one another. The first, and simpler, of the two execution models assumes that agents act in synchrony: sending/receiving messages, and acting, at the same time. The second, more realistic model, assumes that at most one agent can act at any one time.

Both execution models hinge on the notion of the *state* of a system, and of changes in state being caused by *transitions*. Crudely, a state is a "snapshot" of the belief set of each agent in the system at some moment in time. These belief sets are assumed to be in some kind of "equilibrium" (cf. [69]). A state is denoted by the symbol σ. A state change, or transition, occurs when one or more agents receive some messages and perform cognitive actions. A transition is denoted by the symbol τ. The history of an executing system can be considered to be a sequence of state — transition — state — transition … (see Figure 4.2).

4.5.1 Synchronous Execution

Each agent (and by extension each system) has a defined initial state. For an agent, the initial state is its initial belief set, closed under its deduction rules. For a system, the initial state is a collection of initial belief sets, one for each member agent, each set closed under that agent's deduction rules.

Agents are able to change state by performing actions of various types. A tuple of actions (one of each type — communicative and cognitive) is called a *move.* The move an agent makes does not uniquely determine its next state. The moves of each agent in a system combine with those of others. A collection of moves, one for each agent, is called a *transition*. Given a system state and transition, the resultant state is uniquely defined.

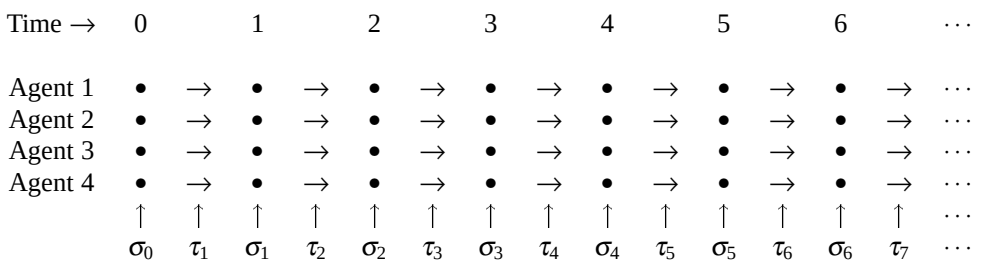
The operation of a system can thus be described as follows. A system has a defined initial state (call it σ_0). From this state, each agent picks a (legal) move, which combines with those of others to form a transition, τ_1. As a result of this transition, a new state σ_1 results. The whole process then begins again,with agents choosing moves which form transition τ_2, and so on. The result is a sequence of states. The process is illustrated in Figure 4.3.

Now to restate these definitions formally. First, the notion of *state* is formalized. The state of a system is defined as a map from agent ids to belief sets.

$$State = Agid \xrightarrow{m} Belset$$

The symbol σ is used to denote a state. The initial state of a system is defined to be the state where each agent has its initial belief set, closed under its deduction rules, and is given by the following function.

$$init\_state : System \to State$$
$$init\_state(sys) \triangleq$$
$$\quad \text{let } \langle Ag, \Delta_0, \rho, \beta, \iota, MR, AR \rangle = sys \text{ in}$$
$$\quad \{ i \mapsto close(\Delta_0(i), \rho(i)) \mid i \in Ag \}$$



Bullets ("•") indicate an agent completing a move and changing state; circles ("∘") would indicate non-completion of a move.

Figure 4.3: Synchronous Execution

Next, a *move* is a tuple of actions, one of each type (cognitive and communicative).

$$Move = Action \times Mess$$

The usual symbol for a move is m. Two "selector" functions are assumed for moves. The function *action* takes a move and returns the action of the move; the function *mess* takes a move and returns the message of the move. The formal definitions of these functions are trivial, and are therefore omitted. Both functions are used extensively in the sequel.

The following boolean valued function defines when a move is legal for some agent, given some belief set.

$$mv\_legal : Move \times Agent \times Belset \to \mathbb{B}$$

$$mv\_legal(m, ag, \Delta) \triangleq \text{let } \langle \Delta_0, \rho, \beta, \iota, MR, AR \rangle = ag \text{ in } ac\_legal(action(m), AR, \Delta) \wedge ms\_legal(mess(m), MR, \Delta)$$

A *nil move* is any move which contains the nil action and a nil message.

There is usually more than one agent acting in a system. The actions of each agent combine with those of others to change the state of the system. Analogous to a move, a *transition* is defined to be map from agent ids to the moves performed by the agent.

$$Trans = Agid \xrightarrow{m} Move$$

The usual symbol for a transition is τ. A transition is *legal* just in case each of the moves it suggests is legal.

$$trans\_legal : Trans \times System \times State \to \mathbb{B}$$

$$trans\_legal(\tau, sys, \sigma) \triangleq \forall i \in \text{dom } \tau \cdot mv\_legal(\tau(i), agent(i, sys), \sigma(i))$$

The *possible transitions* of a system in some state are the legal transitions of the system. A *nil transition* is one where each agent performs a nil move. A function *nil_trans* is assumed, which takes a system and returns a nil transition for that system.

Some utility definitions will be useful for transitions. First, a function returning the set of all messages sent in a transition.

$$sent : Trans \to \text{powerset } Mess$$

$$sent(\tau) \triangleq \{mess(m) \mid m \in \text{rng } \tau\} - Mess_{\text{nil}}$$

Next, a function which returns the set of all messages sent to a particular agent in a transition.

$$recvd : Agid \times Trans \to \text{powerset } Mess$$

$$recvd(i, \tau) \triangleq \{\mu \mid \mu \in sent(\tau) \wedge (recvr(\mu) = i)\}$$

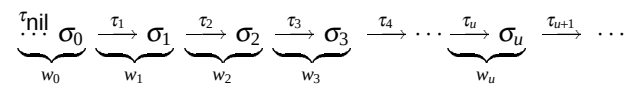


Figure 4.4: Runs, Worlds and World Sequences

A function is now defined which shows how an agent's beliefs change as a result of executing a cycle on which it performs some action and receives some messages.

$$next\_bel : Agent \times Belset \times \mathsf{powerset}\ Mess \times Action \rightarrow Belset$$
$$next\_bel(ag, \Delta, ms, \alpha) \triangleq \mathsf{let}\ \langle \Delta_0, \rho, \beta, \iota, MR, AR \rangle = ag\ \mathsf{in}\ close(\beta(\Delta, \{\alpha(\Delta)\} \cup \{\iota(\Delta, \mu) \mid \mu \in ms\}), \rho)$$

A function *next_state* is now defined, which for some system returns the successor to a state under a transition.

$$next\_state : System \times State \times Trans \rightarrow State$$
$$next\_state(sys, \sigma, \tau) \triangleq \{i \mapsto next\_bel(agent(i, sys), \sigma(i), recvd(i, \tau), action(\tau(i))) \mid i \in \mathsf{dom}\,\sigma\}$$

Almost all the components of the theory of multi-agent systems are now in place. All that remains is to define *execution sequences*, or *runs* of a system. The idea of a run has been illustrated above. A representation for runs must now be developed. One possibility would be simply to define a run as a sequence of states. However, states only contain information about the beliefs of agents. For the purposes of the next chapter, it is useful to record in an execution sequence the cognitive actions performed and messages sent by agents. This is achieved by defining a run as a sequence of *worlds*, where each world contains a state and the transition that caused the state[3]. The idea of associating each state with the transition that gave rise to it is illustrated in Figure 4.4.

The type for worlds is as follows.

$$World = State \times Trans$$

The usual symbol for a world is w. Two selector functions are assumed for worlds, and are used extensively in the sequel. The first is called *state*, which takes a world and returns the state associated with it. The second is called *trans*, which takes a world and returns the transition associated with it. The formal definitions of these functions are trivial, and are therefore omitted. Another useful function is *next_world*, which takes three arguments, (two worlds and a system), and says whether the second world represents a valid transition of the system from the first world.

$$next\_world : World \times World \times System \rightarrow \mathbb{B}$$
$$next\_world(w, w', sys) \triangleq trans\_legal(trans(w'), sys, state(w)) \wedge (state(w') = next\_state(sys, state(w), trans(w')))$$

A system has an initial state (see above), but what is its initial *world*? For convenience, the initial world of a system is defined to be its initial state together with a nil transition.

$$init\_world : System \rightarrow World$$
$$init\_world(sys) \triangleq \langle init\_state(sys), nil\_trans(sys) \rangle$$

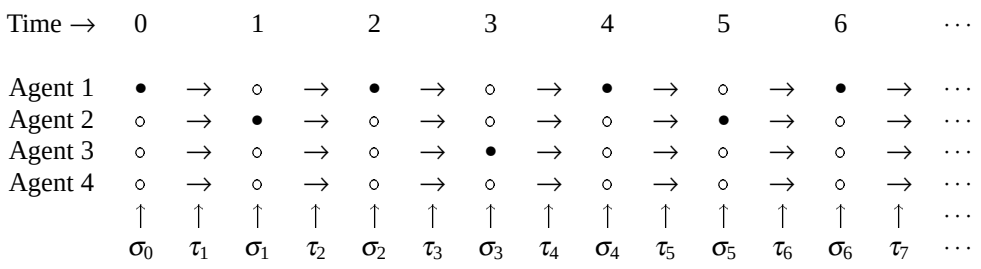
World-sequences are now defined to be countably infinite sequences of worlds. World sequences are the basic representation mechanism for runs.

$$Worldseq = World^*$$

World sequences extend infinitely into the future, and must therefore satisfy the following invariant (recall that ω is the first infinite ordinal):

$$\forall W \in Worldseq \cdot (\mathsf{len}\, W = \omega).$$

[3] *World* may seem a peculiar choice of words; it arises for historical reasons. The formal semantics of temporal logics were developed from modal logics with possible worlds semantics, (described in preceding chapters).



Bullets ("•") indicate an agent completing a move and changing state; circles ("○") indicates non-completion of a move.

Figure 4.5: Interleaved Execution

The next step is to define a function which says of any world sequence whether it is a run of a given system or not.

$$run\_of : Worldseq \times System \to \mathbb{B}$$

$$run\_of(W, sys) \triangleq (W(0) = init\_world(sys)) \wedge \forall u \in \mathbb{N}_1 \cdot next\_world(W(u-1), W(u), sys)$$

This concludes the definition of the synchronous execution model.

4.5.2 Interleaved Execution

Although synchronous execution models have the advantage of simplicity, they do not paint a realistic picture of how agents in a real multi-agent system act and interact, as the presence of a global clock seems to be implied. As Agha points out:

> "The concept of a unique global clock is not meaningful in the context of a distributed system of self-contained parallel agents". [2, p9]

A more realistic model of execution — and one that is more in tune with the standard model of reactive systems, (see, for example, [135], and Appendix C), — is an *interleaved* execution model, where at most one agent is allowed to act at any one time. Such a model is presented below. Before moving on to the details of the interleaved execution model, consider Figure 4.5, which illustrates an imaginary run of a four-agent multi-agent system. As before, a bullet ("•") indicates that an agent executes one internal cycle of receiving messages, updating beliefs, and acting, whereas a circle ("○") indicates that an agent has not completed a cycle.

During the first seven execution steps of this system, agent 1 completes four execution cycles (at times 0, 2, 4, and 6), whereas agent 2 completes two (at times 1 and 5), agent 3 completes one (at time 3), and agent 4 does not complete any. It is not difficult to see that at most one agent is considered to be acting at any one time. Also, it is easy to see that agent 1 is in some sense running faster than all the other agents: seemingly twice as fast as agent 2, and even faster than this for the rest. Suppose this run were continued indefinitely, and agent 4 *never* completed a cycle. Would this be "fair" to agent 4? Surely, no matter how slow the processor on which agent 4 was based it would ultimately complete at least one cycle, perhaps in some enormous — but nonetheless finite — time. For this reason, it is usual to demand that interleaved execution runs satisfy some "fairness" requirement. Unfortunately, fairness is a deep research issue in its own right, and a detailed treatment is quite beyond the scope of this thesis. The reader is referred to [63] for details.

Let us return to the details of the interleaved execution model. The main difference between this and the synchronous model is that only one agent may act in a transition at any one time; this gives an invariant condition for transitions. However, it also means that not all messages sent on cycle u are received on cycle $u+1$. This necessitates the introduction of some method for keeping track of all those messages that have not yet been received; this is called a *message pool*. (Note that it *is* still assumed that when an agent executes a cycle it receives *all* its outstanding messages.)

A type is introduced for pools.

$$Pool = \text{powerset } Mess$$

A state is redefined to be a map from agent ids to belief sets, as before, with the addition of a current message pool.

$$State' = (Agid \xrightarrow{m} Belset) \times Pool$$

Some selector functions are assumed, *bel* and *pool*, each of which takes a state and extracts the belief state and pool from that state, respectively. The formal definitions are trivial, and are therefore omitted. The initial state of a system is as before, with an empty message pool.

$$init\_state' : System \to State'$$
$$init\_state'(sys) \triangleq$$
$$\quad \text{let } \langle Ag, \Delta_0, \rho, \beta, \iota, MR, AR \rangle = sys \text{ in}$$
$$\quad \langle \{ i \mapsto close(\Delta_0(i), \rho(i)) \mid i \in Ag \}, \{\,\} \rangle$$

If an agent does execute a cycle, it receives all the messages sent to it that are in the pool.

$$recvd' : Agid \times State' \to \text{powerset } Mess$$
$$recvd'(i, \sigma) \triangleq \{ \mu \mid \mu \in pool(\sigma) \wedge (recvr(\mu) = i) \}$$

A transition is as before, except that only one agent performs a move; all other agents do nothing. Doing nothing is indicated by a mapping to a distinguished object called nil. This gives:

$$Trans' = Agid \xrightarrow{m} (Move \cup \{\text{nil}\})$$

with the invariant:

$$\forall \tau \in Trans' \cdot \exists!\, i \in \text{dom } \tau \cdot (\tau(i) \neq \text{nil}).$$

(Recall that $\exists!$ means: "there exists precisely one…".)

The legality of a transition is defined as before, with the obvious changes required to deal with transitions as they now appear; the definition of *mv_legal* is as before.

$$trans\_legal' : Trans' \times System \times State' \to \mathbb{B}$$
$$trans\_legal'(\tau, sys, \sigma) \triangleq$$
$$\quad \forall i \in \text{dom } \tau \cdot$$
$$\quad\quad (\tau(i) \neq \text{nil}) \Rightarrow mv\_legal(\tau(i), agent(i, sys), bel(\sigma)(i))$$

The definition of the function *sent*, which returns all the messages sent in a transition, needs to be restated.

$$sent' : Trans' \to Pool$$
$$sent'(\tau) \triangleq \{ mess(m) \mid (i \in \text{dom } \tau) \wedge (\tau(i) \neq \text{nil}) \wedge (m = \tau(i)) \}$$

The function *next_bel* requires no modifications. However a function is required to give the next belief state of a system.

$$next\_bel\_state : System \times State' \times Trans' \to (Agid \xrightarrow{m} Belset)$$
$$next\_bel\_state(sys, \sigma, \tau) \triangleq$$
$$\quad bel(\sigma) \dagger$$
$$\quad \{ i \mapsto next\_bel(agent(i, sys), bel(\sigma)(i), recvd'(i, \sigma), action(\tau(i)))$$
$$\quad \mid (i \in \text{dom } \sigma) \wedge (\tau(i) \neq \text{nil}) \}$$

The message pool resulting from a transition occurring to a state is given by the following.

$$next\_pool\_state : State' \times Trans' \to Pool$$
$$next\_pool\_state(\sigma, \tau) \triangleq sent'(\tau) \cup (pool(\sigma) - \bigcup \{ recvd'(i, \sigma) \mid (i \in \text{dom } \tau) \wedge (\tau(i) \neq \text{nil}) \})$$

Finally, the next state function for interleaved execution is as follows.

$$next\_state' : System \times State' \times Trans' \to State'$$
$$next\_state'(sys, \sigma, \tau) \triangleq \langle next\_bel\_state(sys, \sigma, \tau), next\_pool\_state(\sigma, \tau) \rangle$$

Runs and world sequences for interleaved execution can be developed in exactly the same way as for synchronous execution; the definitions are therefore omitted.

Although interleaved execution is more realistic than synchronous execution, it still does not paint a completely accurate picture of how real systems operate. Moreover, the added complexity of interleaved execution make it much more awkward to manage formally. For these reasons, and for the reasons associated with fairness, discussed earlier, the remainder of the thesis will deal solely with synchronous execution. In particular, the world sequences underlying the semantics of the various logics to be developed in subsequent chapters are assumed to be those of synchronously executing systems.

4.6 Some Extensions to the Basic Model

The basic model of agents and systems can be extended in a number of ways. Some possible extensions are presented below.

4.6.1 Alternative Definitions of Epistemic Inputs

As defined above, an epistemic input is essentially unstructured. It is simply a set of formulae which are crudely regarded as "new evidence". While this definition will probably suffice for most simple cases, sophisticated applications will almost certainly require more information *about* the input. In this brief section, we outline a couple of possible ways that epistemic inputs might be redefined to include such information.

The management of uncertainty is a central issue in AI. An obvious possibility would be to incorporate MYCIN-style certainty factors or probabilities in epistemic inputs, perhaps to indicate the certainty of the new information being true.

$$Epin' = \mathbb{R} \times \mathsf{powerset}\ Form(L)$$

Another possibility would be to give an indication of the *source* of the input, by associating an agent id with each input.

$$Epin'' = Agid \times \mathsf{powerset}\ Form(L)$$

4.6.2 Broadcast Messages

The communication paradigm modelled by the theory above is point-to-point message passing: a message originates from some agent, and is routed through a network to its intended recipient. A simplifying assumption is that an agent can only send one message at a time. While common, this style of communication is by no means the only method available. An alternative is *broadcast* messages, where an agent sends messages to a group of agents — possibly every agent — simultaneously. Broadcast message passing is not supported by the theory above, but is also quite common. For example, it is the basic communication method in Concurrent METATEM [62], and is at the heart of the Contract Net protocol [161], [160].

The modifications required for broadcast messages are simple. First, the definition of a message rule is altered, so that the second component is a set of messages, rather than a single message as before.

$$Mrule' = Cond \times \mathsf{powerset}\ Mess$$

The idea is that if a message rule is "chosen", then the set of messages will be sent, rather than the single message as before. Next, we must alter the definition of a move, to take into account this change.

$$Move' = Action \times \mathsf{powerset}\ Mess$$

And finally, we alter the definition of *sent*, which, you will recall, extracts the set of all messages sent in a transition.

$$sent' : Trans \to \mathsf{powerset}\ Mess$$
$$sent'(\tau) \triangleq \bigcup \{mess(m) \mid m \in \mathsf{rng}\ \tau\} - Mess_{\mathsf{nil}}$$

No other changes are necessary.

4.6.3 Multiple Preconditions for Actions and Messages

Another obvious extension to the basic theory is to allow multiple preconditions for actions and messages. These might be accommodated as follows. First, the types for action and message rules are redefined.

$$Arule' = \mathsf{powerset}\ Cond \times Action$$

$$Mrule' = \mathsf{powerset}\ Cond \times Mess$$

Next, the definitions of applicability need changing.

$$ar\_applic : Arule' \times Belset \to \mathbb{B}$$
$$ar\_applic(\langle \Gamma, \alpha \rangle, \Delta) \quad \triangleq \quad \forall \phi \in \Gamma \cdot \phi \in (\Delta \cup \{\mathsf{true}\})$$

The function *mr_applic*, for message rules, is essentially identical to this function, and is therefore omitted. No other changes are necessary.

4.7 A List of Types

For reference, the types which appear in the theory of multi-agent systems presented above are summarized below. Primed types are alternative definitions of existing types.

Belset A *belief set* is a set of formulae in the internal language, L. Usual symbol: Δ.

Drule A *deduction rule* is a rule of inference which must have a fixed, finite number of premises, and be an effectively computable function of those premises. Agents possess a set of deduction rules, for which the usual symbol is ρ.

Epin An *epistemic input* results in an agent either as the result of performing a cognitive action, or as the result of receiving a message. It represents a "new piece of evidence" that an agent may subsequently believe.

Brf A *belief revision function* maps a belief set and set of epistemic inputs to a new belief set. Usual symbol: β.

Agid An *agent identifier* is assigned to each agent in order to uniquely identify it. Usual symbols: i, j, k, l.

Mess A *message* is sent between agents. Usual symbol: μ.

Messint A *message interpretation function* maps a belief set and message to an epistemic input. Usual symbol: ι.

Action A *cognitive action* is a function which takes as its argument a belief set, and returns an epistemic input. Such actions represent the utilization of cognitive (i.e., computational) resources. Usual symbol: α.

Cond A *condition* is associated with all actions and messages in action/message rules — see below. Usual symbol: ϕ.

Arule A (cognitive) *action rule* is a condition/action pair, the condition determining when the action is applicable, being satisfied if it is believed. Usual symbol: *ar*. An agent possesses a set of such rules, for which the usual symbol is: *AR*.

Mrule Analogous to an action rule, a *message rule* is a condition/message pair, the condition determining when the message may be sent. Usual symbol: *mr*. An agent possesses a set of such rules, for which the usual symbol is: *MR*.

Agent An *agent* has an initial belief set, some deduction rules, an interpretation function for messages, a belief revision function, and some action and message rules. Usual symbol: *ag*.

System A *system* is a collection of named interacting agents. Usual symbol: *sys*.

Move A *move* is a pair, containing an action of each type (cognitive and communicative). It defines the actions an agent has performed on one "round". Usual symbol: *m*.

Trans A *transition* defines the moves each agent has performed on one "round" of a system; it maps each agent id in the system to a move. Usual symbol: τ.

State The *state* of a system defines the belief set possessed by each agent on one given "round" of the system. It maps each agent id in the system to a belief set. Usual symbol: σ.

World A *world* is a state together with the transition that caused the state. Usual symbol: w.

Worldseq A *world sequence* is a countably infinite sequence of worlds, representing a run of a system. The first world in the sequence contains the nil transition and the initial state of the system. Usual symbol: W.

4.8 Summary

This chapter has introduced a theory of computational multi-agent systems, in which each agent is considered to possess explicitly represented beliefs, a limited capacity for reasoning about its beliefs, and the ability to act, either by utilizing its own private, cognitive resources in the performance of a cognitive act, or by sending a message to another agent. The chapter culminated in the definition of two execution models, which define how agents act and interact. In subsequent chapters, a number of logics will be developed which can be used to reason about multi-agent systems of the type described by the theory.

Chapter 5

Linear Time Temporal Logics for Multi-Agent Systems

THE preceding chapter introduced a theory of computational multi-agent systems. The aim of this chapter is to describe a family of logics that can be shown, in a precise way, to *correspond* to this theory, and may therefore be used to reason about systems modelled by the theory. The common feature of all the logics described in this chapter is that they are based on a *linear* model of time; some alternative logics, based on a *branching* model of time are described in Chapter 7.

The remainder of this chapter is structured as follows. The next section introduces a logic called AL (for "Agent Logic"), the simplest in the family of logics developed in this thesis. AL is propositional (in that it does not allow quantification), and is based on a linear, discrete model of time. The correspondence with the theory of multi-agent systems, and the syntax, semantics, and proof theory of AL are all discussed at length. The section closes with a discussion of how AL might be made more expressive.

Section 5.2 examines the role of the internal languages used by agents, and how the choice of internal language affects AL. The possibility of using AL as an internal language is discussed at this point, and found to be (practically speaking) unworkable, due to an important syntactic restriction on the language. To remedy this, a hybrid language is developed, by combining AL with a standard first-order temporal logic. Various possible theorems of this hybrid logic are discussed, with respect to the properties of agents that they correspond to.

Section 5.3 introduces a quantified version of AL called QAL ("Quantified AL"). This language is more expressive than AL, in that it allows quantification over agents, actions, and individuals in the domain of the internal language.

5.1 The Logic AL

This section introduces the first, and simplest of the logics developed in the thesis. The logic is called AL, which stands for "Agent Logic". It is propositional, in that it does not allow quantification. It contains three atomic operators: Bel, for describing the beliefs of agents, Send, for describing the messages that agents send, and Do, for describing the cognitive actions that agents perform. AL also contains a set of modal temporal operators, which allow the description of the dynamic properties of agents. Finally, as indicated above, AL is based on a model of time that is linear.

5.1.1 Syntax

AL is intended to allow reasoning about agents: their beliefs, their actions, and the messages they send. Since it isn't possible to actually put an action or agent directly into formulae of the language (which after all, are just strings of symbols), there must be a way of referring to these objects in the language. This is achieved by putting symbols into the language whose denotation is an agent identifier or action. Since quantification isn't allowed in AL, these symbols will be *constants* (or more properly, *individual constants*). Also, to express the beliefs of agents, the internal language must appear in AL somewhere.

Definition 6 *The alphabet of AL (based on L) contains the following symbols:*

(Bel i ϕ)	Agent i believes ϕ
(Send i j ϕ)	Agent i sent j message ϕ
(Do i α)	Agent i performs action α
$\bigcirc \phi$	Next ϕ
$\bullet \phi$	Last ϕ
$\phi \mathcal{U} \psi$	ϕ Until ψ
$\phi \mathcal{S} \psi$	ϕ Since ψ

Table 5.1: Non-standard Operators in AL

1. *The symbols* $\{\mathsf{true}, \mathsf{Bel}, \mathsf{Send}, \mathsf{Do}\}$*;*
2. *A set of constant symbols Const made up of the disjoint sets* $Const_{Ag}$ *(agent constants) and* $Const_{Ac}$ *(action constants);*
3. *All closed formulae of the internal language L;*
4. *The unary propositional connective "* $\neg$ *" and the binary propositional connective "* $\vee$*";*
5. *The unary temporal connectives* $\{\bigcirc, \bullet\}$ *and the binary temporal connectives* $\{\mathcal{U}, \mathcal{S}\}$*;*
6. *The punctuation symbols* $\{), (\}$.

An instance of AL (which might be called the "external language") is parameterized by the internal language, L. Different internal languages result in different external languages. To identify the language of AL based on internal language L we write AL(L). For the most part, we are concerned with general properties of AL, and reference to the internal language will therefore be suppressed. The relationship between AL and the language upon which it is based is examined in more detail in section 5.2.

Definition 7 *The syntax of AL (based on L) is defined by the following rules:*

1. *If i, j are agent id constants,* ϕ *is a closed formula of L, and* α *is an action constant, then the following are formulae of AL:*
   true $\quad$ (Bel i ϕ) $\quad$ (Send i j ϕ) $\quad$ (Do i α)
2. *If* ϕ*,* ψ *are formulae of AL, then the following are formulae of AL:*
   $\neg \phi \quad \phi \vee \psi$
3. *If* ϕ*,* ψ *are formulae of AL, then the following are formulae of AL:*
   $\bigcirc \phi \quad \bullet \phi \quad \phi \mathcal{U} \psi \quad \phi \mathcal{S} \psi$

Following standard conventions, parentheses and square brackets will be used to disambiguate formulae where necessary (the same is true of all the languages defined in the thesis). Table 5.1 summarizes the meaning of the non-standard operators in AL.

5.1.2 Semantics

The purpose of a language's semantics is to assign some formal meaning to syntactic objects of the language. This section introduces the formal semantics of AL.

The semantics are presented in three parts: the first defines model structures for AL (and shows how the execution of a system corresponds to these structures), the second gives the semantic rules for AL, and the third presents the notions of *satisfiability* and *validity* for formulae of AL.

Model Structures

The idea which underlies the semantics of all the linear time logics for multi-agent systems developed in this thesis is that of allowing a run of a system to act as a model for a linear discrete temporal logic. The first component of a model for AL is therefore a world sequence, representing a run of a system. Unfortunately, to make the machinery of the logic work, some extra technical apparatus is required.

An *interpretation for constants* is a bijective map from *Const* to the objects the constants denote (the map will be bijective as every object is assumed to have precisely one constant associated with it, which is a standard name for the object). Note that this map must *preserve sorts*, in that it must only map agent id constants to agent ids, etc. The symbol I is generally used for an interpretation. The type for AL models is called $Model_{AL}$, and is defined as follows.

Definition 8 *A model for AL is a structure:*

$$\langle W, Ag, Ac, I \rangle$$

where

- $W \in Worldseq$ *is a world sequence;*
- $Ag \subseteq Agid$ *is a set of agent ids;*
- $Ac \subseteq Action$ *is a set of actions;*
- $I : Const \xleftrightarrow{m} (Ag \cup Ac)$ *interprets constants.*

The following boolean-valued function defines under what circumstances a model is a model of some system.

$$model\_of : Model_{AL} \times System \to \mathbb{B}$$
$$model\_of(M, sys) \triangleq$$
$$\quad \text{let } \langle W, Ag', Ac, I \rangle = M \text{ in}$$
$$\quad \text{let } \langle Ag, \Delta_0, \rho, \beta, \iota, MR, AR \rangle = sys \text{ in}$$
$$\quad run\_of(W, sys) \wedge$$
$$\quad (Ag = Ag') \wedge$$
$$\quad (Ac = \bigcup \{ \{ \alpha \mid \langle \phi, \alpha \rangle \in AR(i) \} \mid i \in Ag \})$$

The first conjunct of the condition simply requires that the model represents a valid run of the system. The second and third conjuncts require that the agents and actions in the model and system correspond. We will say an AL model M is *ordinary* iff there is some system of which M is a model.

$$ordinary : Model_{AL} \to \mathbb{B}$$
$$ordinary(M) \triangleq \exists sys \in System \cdot model\_of(M, sys)$$

For the most part, we shall be concerned solely with ordinary models, for obvious reasons: if a model is *not* ordinary, then it describes a system which cannot exist! Ordinary models are also useful because they shift the burden of proof: to show that a theorem is sound for a class of models, it is merely necessary to show that the property expressed by the theorem holds for all systems modelled by the class. This establishes a kind of "correspondence theory" for AL.

This correspondence property is so natural and useful that we will frequently abuse terminology by writing that a theorem is "sound for the class of systems/agents x" rather than the more technically correct, but long winded, "sound for the class of models which are models of the system/agent class x".

Semantic Rules

The semantics of AL are given via the satisfaction relation, "$\models$", which holds between pairs of the form:

$$\langle M, u \rangle$$

(where M is a model for AL, and $u \in \mathbb{N}$ is a temporal index into M), and formulae of AL. The semantic rules for AL are given in Figure 5.1.

The first four rules deal with atomic formulae, (or *atoms*), of AL. Atoms are the primitive components of the language. To understand the semantics, it may be helpful to recall that a world-sequence is a structured object that represents a run of a system. It contains all the information about a system that can be expressed in a formula. The semantic rules are largely concerned simply with "taking this structure apart" to get out the required information. So, for example, if W is a world sequence, $u \in \mathbb{N}$ is a time, and i is an agent id, then $state(W(u))(i)$ returns the belief set of agent i at time u in world sequence W.

$\langle M, u\rangle$	$\models$	true	
$\langle M, u\rangle$	$\models$	(Bel i ϕ)	iff $\phi \in$ *state*$(W(u))(I(i))$
$\langle M, u\rangle$	$\models$	(Do i α)	iff *action*$(trans(W(u))(I(i))) = I(\alpha)$
$\langle M, u\rangle$	$\models$	(Send i j ϕ)	iff $\langle I(i), I(j), \phi\rangle \in$ *sent*$(trans(W(u)))$
$\langle M, u\rangle$	$\models$	$\neg\,\phi$	iff $\langle M, u\rangle \not\models \phi$
$\langle M, u\rangle$	$\models$	$\phi \vee \psi$	iff $\langle M, u\rangle \models \phi$ or $\langle M, u\rangle \models \psi$
$\langle M, u\rangle$	$\models$	$\bigcirc\,\phi$	iff $\langle M, u+1\rangle \models \phi$
$\langle M, u\rangle$	$\models$	$\bullet\!\!\bigcirc\,\phi$	iff $u > 0$ and $\langle M, u-1\rangle \models \phi$
$\langle M, u\rangle$	$\models$	$\phi\,\mathcal{U}\,\psi$	iff there is some $v \in \mathbb{N} \cdot v \geq u$ such that $\langle M, v\rangle \models \psi$ and $\langle M, w\rangle \models \phi$ for all $w \in \mathbb{N} \cdot u \leq w < v$
$\langle M, u\rangle$	$\models$	$\phi\,\mathcal{S}\,\psi$	iff there is some $v \in \{0, \ldots, u-1\}$ such that $\langle M, v\rangle \models \psi$ and $\langle M, w\rangle \models \phi$ for all $w \in \mathbb{N} \cdot v < w < u$

Figure 5.1: Semantics of AL

Note that *state*, *trans*, *action* and *sent* are selector functions, fixed for all models: *state* takes a world and extracts its state, *trans* takes a world and extracts its transition, *action* takes a move and extracts the cognitive action of the move, and *sent* takes a transition and returns the set of all messages sent in the transition.

The formula true is a logical constant that has the interpretation "true" at all times; it is always satisfied. The logical constant false is defined as $\neg$ true; it is never satisfied.

The belief operator (Bel i ϕ) is read: "agent i believes ϕ"[1]. This operator is essentially the belief operator from Konolige's logic L^B, and obeys the same rules [100]. Where $\Delta = \{\phi_1, \ldots, \phi_n\}$ we let (Bel i Δ) abbreviate (Bel i ϕ_1), ..., (Bel i ϕ_n).

The operator (Do i α) describes the performance of cognitive actions. It is read: "agent i performs the action α".

The operator (Send i j ϕ) describes the sending of messages, and will be satisfied if agent i has sent j the message with content ϕ at the appropriate time. It is useful to let (Rcv i j ϕ) abbreviate (Send j i ϕ)[2].

These next two rules define the standard propositional connectives $\neg$ (not) and $\vee$ (or), which have the usual semantics. The remaining propositional connectives ($\Rightarrow$ (implies), $\wedge$ (and), and $\Leftrightarrow$ (logically equivalent)), are defined as abbreviations in the usual way:

$$\phi \Rightarrow \psi \triangleq (\neg\,\phi \vee \psi)$$
$$\phi \wedge \psi \triangleq \neg\,(\neg\,\phi \vee \neg\,\psi)$$
$$\phi \Leftrightarrow \psi \triangleq ((\phi \Rightarrow \psi) \wedge (\psi \Rightarrow \phi))$$

The final four rules define the semantics of the temporal operators. This basic stock of temporal operators turns out to be surprisingly expressive; an early result in the study of temporal logic, due to Kamp, is that temporal logics with (strict) since ("$\mathcal{S}$") and until ("$\mathcal{U}$") operators are *expressively complete* over continuous linear orders; that is, any imaginable (propositional) connective that expresses a property of such orders can be defined in terms of them [95].

For convenience, the abbreviations in Table 5.2 are assumed. "$\bigcirc$" and "$\bullet\!\!\bigcirc$" are the *next* and (strong) *last* operators respectively: $\bigcirc\,\phi$ is satisfied if ϕ is satisfied in the next world; "$\bullet\!\!\bigcirc$" is the past time version of this operator, so that $\bullet\!\!\bigcirc\,\phi$ is satisfied if ϕ was satisfied in the last state of the sequence. "$\bullet\!\!\bigcirc$" is said to be *strong* because it is always false at the beginning of time. The "$\bullet$" operator is the weak version of "$\bullet\!\!\bigcirc$": its argument is alway satisfied at the beginning of time[3]. The special symbol "*init*" is *only* satisfied at the beginning of time.

"$\Box$" and "$\blacksquare$" are the *always* and *heretofore* operators respectively: $\Box\,\phi$ will be satisfied iff ϕ is satisfied now and in all future moments; "$\blacksquare$" is the past time version. (Note, however, that "$\blacksquare$" is a *strict past* operator).

[1] In the discussion that follows, a reference time is assumed.

[2] Note that we can only do this because we assume that all messages are received the cycle after they are sent. If we did not make this assumption, (as is the case, for instance, in the interleaved execution model described in the previous chapter), then it would be necessary to introduce Rcv as an enirely new operator, with its own semantic rule.

[3] Note that there would be no need for two different kinds of last operator if time was not finite in the past.

$\bullet\,\phi$	$\triangleq$	$\neg\,\bigcirc\neg\,\phi$	(Weak) last ϕ
init	$\triangleq$	$\neg\,\bigcirc$true	Initially …
$\Diamond\phi$	$\triangleq$	true$\,\mathcal{U}\,\phi$	Sometime ϕ
$\Box\phi$	$\triangleq$	$\neg\,\Diamond\neg\,\phi$	Always ϕ
$\phi\,\mathcal{W}\,\psi$	$\triangleq$	$\Box\phi\ \vee\ \phi\,\mathcal{U}\,\psi$	ϕ Unless ψ
$\blacklozenge\ \phi$	$\triangleq$	true$\,\mathcal{S}\,\phi$	Was ϕ
$\blacksquare\phi$	$\triangleq$	$\neg\ \blacklozenge\ \neg\,\phi$	Heretofore ϕ
$\phi\,\mathcal{Z}\,\psi$	$\triangleq$	$\blacksquare\phi\ \vee\ \phi\,\mathcal{S}\,\psi$	ϕ Zince ψ
$\phi\,\mathcal{B}\,\psi$	$\triangleq$	$\neg\,((\neg\,\phi)\,\mathcal{U}\,\psi)$	ϕ Before (or precedes) ψ

Table 5.2: Derived Temporal Operators

"$\Diamond$" and "$\blacklozenge$" are the *sometime* and *was* operators respectively: $\Diamond\phi$ will be satisfied if ϕ is satisfied now, or becomes satisfied at least once in the future; "$\blacklozenge$" is the strict past version, so $\blacklozenge\ \phi$ will be satisfied if ϕ was satisfied at least once in the past.

"$\mathcal{U}$" and "$\mathcal{W}$' are the *until* and *unless* operators respectively ("$\mathcal{W}$" is sometimes called the *weak until* operator): $\phi\,\mathcal{U}\,\psi$ will be satisfied if ϕ is satisfied until ψ becomes satisfied — ψ must eventually be satisfied. "$\mathcal{W}$" is similar to "$\mathcal{U}$", but allows for the possibility that the second argument never becomes satisfied. The "$\mathcal{S}$" (since) and "$\mathcal{Z}$" (zince) operators are the strict past time versions of "$\mathcal{U}$" and "$\mathcal{W}$" respectively.

Finally, the "before" operator, ("$\mathcal{B}$") describes temporal precedence: if $\phi\,\mathcal{B}\,\psi$, then ϕ strictly precedes ψ.

Satisfiability and Validity

Satisfiability and *validity* are defined in the usual way. Let ϕ be a formula of AL. Then ϕ is satisfiable in a model M if there is some $u \in \mathbb{N}$ such that $\langle M, u\rangle \models \phi$, and satisfiable *simpliciter* iff it is satisfiable in some model. It is *true in a model M* iff $\langle M, u\rangle \models \phi$ for all $u \in \mathbb{N}$, *valid in a class of models* C (written $\models_C \phi$) iff ϕ is true in all models in the class, and valid *simpliciter* iff it is true in the class of all ordinary models. A convenient way of expressing that ϕ is valid is to write $\models \phi$.

There is an important relationship between the satisfiability of belief atoms in AL and the deductive capabilities of the agents being described. This relationship is captured in the following lemma, which is later used to prove the soundness of an axiom for belief atoms.

Lemma 1 ([100]) *The (denumerable) set:*

$$\{(\mathsf{Bel}\ i\ \Delta),\ \neg\,(\mathsf{Bel}\ i\ \Gamma)\}$$

is unsatisfiable iff there is some $\phi \in \Gamma$ such that $\Delta \vdash_{\rho(i)} \phi$.
Proof *See [100].* ∎

5.1.3 Proof Theory

In this section, a proof theory for AL is developed, framed in terms of an axiom system. It is usual practice, when axiomatizing a logic, to pick the smallest complete set of axioms and inference rules as an axiomatic base. Other axioms and inference rules are then introduced as derived constructs. However, completeness proofs tend to be rather involved, and the issue of completeness is not, therefore, addressed in this thesis. See further comments in the conclusions chapter.

Axioms

Begin by noting that the ordinary connectives $\wedge, \vee, \neg, \Rightarrow, \Leftrightarrow$ obey classical semantics, and thus:

$$\vdash\ \ \phi \quad \text{where } \phi \text{ is a propositional tautology.} \tag{5.1}$$

The next axiom captures the fundamental property of belief systems; it is called the *attachment* axiom (cf. [100]).

$$\vdash\ \ ((\mathsf{Bel}\ i\ \phi_1) \wedge \cdots \wedge (\mathsf{Bel}\ i\ \phi_n)) \ \Rightarrow\ (\mathsf{Bel}\ i\ \phi) \quad \text{where } \{\phi_1, \cdots, \phi_n\} \vdash_{\rho(i)} \phi \tag{5.2}$$

The soundness of this latter axiom is an immediate corollary of the attachment lemma (above).

The logic of the temporal operators has been studied in great detail; two good references are [116, pp228–234] and [47, pp1003–1005], both of which list a great many temporal axioms. Another good reference is [79] where axiom systems for a number of temporal logics are presented, and the soundness of a number of derived axioms is demonstrated through formal proof. Rather than present a detailed list of axioms here, we simply list some representative axioms; a fuller list is given in Appendix D.

$$\vdash \quad \bigcirc \neg \phi \Leftrightarrow \neg \bigcirc \phi \tag{5.3}$$
$$\vdash \quad \phi \Rightarrow \Diamond \phi \tag{5.4}$$
$$\vdash \quad \bigcirc \phi \Rightarrow \Diamond \phi \tag{5.5}$$
$$\vdash \quad \Box \phi \Rightarrow \Diamond \phi \tag{5.6}$$
$$\vdash \quad \phi \mathcal{U} \psi \Rightarrow \Diamond \psi \tag{5.7}$$
$$\vdash \quad \CIRCLE \phi \Rightarrow \blacklozenge \phi \tag{5.8}$$
$$\vdash \quad \blacksquare \phi \Rightarrow \blacklozenge \phi \tag{5.9}$$
$$\vdash \quad \phi \mathcal{S} \psi \Rightarrow \blacklozenge \psi \tag{5.10}$$
$$\vdash \quad \Diamond(\phi \vee \psi) \Leftrightarrow (\Diamond \phi \vee \Diamond \psi) \tag{5.11}$$
$$\vdash \quad \Box(\phi \wedge \psi) \Leftrightarrow (\Box \phi \wedge \Box \psi) \tag{5.12}$$
$$\vdash \quad \Box(\phi \Rightarrow \psi) \Rightarrow (\Box \phi \Rightarrow \Box \psi) \tag{5.13}$$
$$\vdash \quad \Box(\phi \Rightarrow \psi) \Rightarrow (\Diamond \phi \Rightarrow \Diamond \psi) \tag{5.14}$$
$$\vdash \quad \Box(\phi \Rightarrow \psi) \Rightarrow (\bigcirc \phi \Rightarrow \bigcirc \psi) \tag{5.15}$$
$$\vdash \quad \Diamond \phi \Leftrightarrow \phi \vee \bigcirc \Diamond \phi \tag{5.16}$$
$$\vdash \quad \Box \phi \Leftrightarrow \phi \wedge \bigcirc \Box \phi \tag{5.17}$$

Inference Rules

The propositional connectives in AL have classical semantics, so Modus Ponens (MP) is a rule of inference.

$$\textit{From} \vdash \phi \Rightarrow \psi \textit{ and } \vdash \phi \textit{ infer } \vdash \psi \tag{5.18}$$

It is not difficult to see that the following temporal rules are sound.

$$\text{From} \vdash \phi \text{ infer} \vdash \bigcirc \phi \tag{5.19}$$
$$\text{From} \vdash \phi \text{ infer} \vdash \Diamond \phi \tag{5.20}$$
$$\text{From} \vdash \phi \text{ infer} \vdash \Box \phi \tag{5.21}$$
$$\text{From} \vdash \phi \text{ infer} \vdash \CIRCLE \phi \tag{5.22}$$
$$\text{From} \vdash \phi \text{ infer} \vdash \blacklozenge \phi \tag{5.23}$$
$$\text{From} \vdash \phi \text{ infer} \vdash \blacksquare \phi \tag{5.24}$$

5.1.4 Discussion

As with any logical language, it is possible to classify AL in terms of two independent attributes: the *language* of formulation, and the underlying *model* [100, p88]. Each of these parameters may be further decomposed into those attributes of agents (their beliefs, actions, and messages), and those attributes of time. The main features of AL are summarized in Table 5.3. Each entry in this table reflects a design decision made during the development of AL. The aim of this section is to discuss, and, where necessary, to justify these decisions.

First, consider the properties of agents, and the language used to express them. For belief, the linguistic tool chosen was the sentential modal operator ($\mathsf{Bel}\ i\ \phi$): this operator is sentential because its key argument is a sentence, (or formula), of the internal language L, and modal as it is not a standard truth-functional operator (i.e., it "changes the mode" of a formula). What are the alternatives to such an operator?

The main alternative is to use a first-order meta-language, which contains terms denoting formulae of some other object-language. The principle advantage of meta-language approaches is that they are more expressive than their modal language counterparts. It is also often claimed that since meta-languages are just many-sorted first-order languages, ordinary first-order theorem provers may be used for them (whereas theorem provers for

	AGENTS		TIME
LANGUAGE	Belief	sentential modal operator (Bel i ϕ)	Modal (tense) logic with future/past operators: $\bigcirc$ $\bigodot$ $\bullet$ $\Diamond$ $\blacklozenge$ $\Box$ $\blacksquare$ $\mathcal{U}$ $\mathcal{S}$ $\mathcal{W}$ $\mathcal{Z}$ $\mathcal{B}$ Expressively complete: [95].
	Action	operator (Do i α)	
	Comms	sentential modal operator (Send i j ϕ)	
MODEL	Belief	deduction model	Linear Discrete Finite past Infinite future
	Action	primitive actions	
	Comms	directed messages	

Table 5.3: The Properties of AL

modal logics are very much an ongoing research area). This latter claim has yet to be satisfactorily demonstrated, however.

There are several disadvantages to the meta-language approach. Most importantly, any non-trivial meta-language reasoning involves manipulating formulae of extraordinary complexity, (this negative attribute has been reported by several researchers [98], [138]). Also, it seems that modal operators are a more natural language of expression than their meta-language counterparts [138, §3.3, p128]. For these reasons, a meta-language approach was rejected in favour of a sentential modal operator for belief.

Now to the actions and messages of agents. These are described by two operators: the sentential modal operator Send, for messages, and the operator Do, for cognitive acts. Before discussing the semantics of these operators in any detail, it is worth making some general comments about the suitability of having operators for describing the cognitive and communicative acts. Two questions arise: 1) Is it *necessary* to have operators for describing these actions? 2) If so, could not the two operators be replaced by a single, unified notation?

The first question is posed because much research in AI planning and philosophy has rejected the notion of actions as primitives — with good reason. Nevertheless, the notion of actions is a very appealing one. Shoham expressed similar sentiments: when developing the theory which underlies his AOP paradigm, he was at pains to reject *any* notion of action. Nevertheless, he found it worth introducing actions into the AGENT0 system as "syntactic sugar" [151, p15, pp24–27]. So while there may be something deeply suspect about any notion of primitive action on strictly philosophical grounds, the concept seems too useful to reject solely on the basis of dogma.

Given that some representation of action in the language *is* desirable, the second question arises. Is it necessary to distinguish between cognitive and communicative acts? Could the two operators not be replaced by a single, unified notation? The question might seem strange to those not familiar with speech act theory, where it is generally taken as axiomatic that communication is just another form of action, which may be planned for — or may fail — in precisely the same way that "normal" actions, (such as picking something up from a table), may fail [9], [145]. Cohen and Levesque, in their highly influential theory of speech acts, model a "request" act as *any* type of committed attempt to bring about a certain state of affairs (see Chapter 3 for a discussion of Cohen and Levesque's approach, and [29], [30]). In their theory, no distinction is made between essentially linguistic acts, (such as saying "please make me a cup of tea"), and other types of action (such as nodding in the direction of the kettle). The result is a satisfying theory which does much to clear up the confusion surrounding earlier attempts to formalize speech acts. However, Cohen and Levesque are only able to achieve their results by abstracting away a good deal of information — for example, the distinction between linguistic and other actions. It is not at all

certain that such a degree of abstraction is appropriate for describing computational multi-agent systems. There is a qualitative distinction between sending a message and employing computational resources, and it is surely useful to retain this distinction for our purposes. For these reasons, a distinction is made in AL.

The models that underly belief, communication and cognitive action are those which appear in the theory of multi-agent systems. These have been discussed extensively in the preceding chapter, and will therefore be mentioned only in passing here. The deduction model of belief is adopted because it seems the simplest, and at the current time, the best developed model of agents with incomplete reasoning abilities. It does have its critics: Reichgelt, for example, argues that it is still too coarse grained to really capture incomplete reasoning [138]. However, for the reasons just mentioned, the deduction model is adopted within this work.

Reasoning about time in AL is achieved through the use of the language's temporal modal operators (e.g., "$\bigcirc$"). This approach is, however, by no means the only approach to reasoning about time in general use. The choice must therefore be justified.

The simplest method for reasoning about time is to use a standard (perhaps many-sorted) first-order logic, which contains terms denoting times. For example, every n-ary predicate in the logic might be extended to an $(n+1)$-ary predicate, with the $(n+1)^{th}$ term denoting a time at which the predicate is true. The English sentence "Jones is never ill" might then be translated into the following first-order formula.

$$\forall t \cdot Time(t) \Rightarrow \neg Ill\_At(Jones, t)$$

The very obvious advantage of this approach is that no extra logical apparatus need be introduced to deal with time: the entire corpus of work on standard first-order logic can be brought to bear directly. The disadvantages are that the approach is unnatural and awkward for humans to use. Formulae representing quite trivial temporal properties become large, complicated, and hard to understand. The approach has been called the method of temporal arguments [137], or the first-order approach [68].

The second technique is to employ a first-order meta-language which contains terms denoting times, and terms denoting formulae of an object-language[4]. In such meta-language approaches, one typically finds a predicate such as $Holds(p, t)$ or $True(p, t)$ to express the fact that the object-language formula denoted by p is true at the time (or holds over the interval, etc.) denoted by t. This approach is usually called the reified approach to reasoning about time. The main advantage claimed by proponents of the reified approach is the ability to stay in a (sorted) first-order language while reasoning about time. The main disadvantage is that even simple temporal properties are difficult to express, as the logical apparatus is quite cumbersome.

Reichgelt comes to the following conclusion:

> "If one restricts oneself to purely logical considerations, then modal treatments of temporal logics are superior to [others] …From the computational point of view, [reified or first-order] treatments seem preferable". [137, p176]

Since computational tractability is *not* a key consideration of this thesis, a modal approach is adopted.

The model of time used in AL may be characterized by four properties: 1) linearity, 2) discreteness, 3) finite past, 4) infinite future (cf. [68]). The basic alternative to a linear model of time is one that branches. In fact, it is quite straightforward to construct a branching model of time from the theory of multi-agent systems; see the comments that close this section, below.

The discreteness property is not disposed of so easily! There are two basic alternatives to discrete models: the first is a dense model, (where the temporally ordered time points may be put in a one-to-one correspondence with the reals); the second is an interval model, (where time is not viewed as points at all, but as intervals). There is no doubt that the simplest of the three is the discrete model: however, it has been suggested, quite reasonably, that this is not a good model of "real world" time (are there really "time atoms"?). However, some applications are inherently based on a discrete model of time, and for these applications, discrete models are satisfactory. One such application is the Pnuelian method for reasoning about reactive systems, where time points correspond to the state of an executing computer program; transitions between states are atomic, and correspond to the execution of primitive program instructions. Discreteness in the theory of multi-agent systems arises because of the use of belief sets. If there are *not* discrete moments when an agent's beliefs can be said to be fixed, then it is difficult to see how an agent could ever be said to believe something. How could an agent rationally choose an action on the basis of *continuously* changing beliefs? Gärdenfors finds the notion of epistemic state similarly fundamental. He comments that there must be times where an agent's beliefs are — conceptually at least — at "equilibrium under all forces of internal criticism" [69, pp9–10]. These "forces of internal criticism" are modelled in the theory of

[4]The principles are exactly the same as those for using a meta-language approach for representing beliefs — see the discussion above.

$(\mathsf{Bel}\ i\ \forall x \cdot P(x) \Rightarrow Q(x))$	$\surd$	Agent i believes $\forall x \cdot P(x) \Rightarrow Q(x)$.
$(\mathsf{Bel}\ i\ P(x))$	$\times$	$P(x)$ is not a sentence of L_0.
$(\mathsf{Bel}\ i\ P(a))$	$\surd$	Agent i believes $P(a)$.
$(\mathsf{Bel}\ i\ (\mathsf{Bel}\ j\ P(a)))$	$\times$	$(\mathsf{Bel}\ j\ P(a))$ is not a sentence of L_0.
$\bigcirc\!\!\!\!\bullet\ (\mathsf{Bel}\ i\ P(a))$	$\surd$	Agent i previously believed $P(a)$.
$\blacklozenge\ (\mathsf{Bel}\ i\ P(a))$	$\surd$	At sometime past, i believed $P(a)$.
$\forall x \cdot\ \bigcirc\!\!\!\!\bullet\ (\mathsf{Bel}\ i\ P(x))$	$\times$	Bad Quantifier!

Table 5.4: Some Syntactically Correct and Incorrect Formulae of AL(L_0)

multi-agent systems by the belief revision function. For these reasons, it is felt that a discrete model of time is appropriate for AL.

Having a finite past means that some time point can be recognized as being the "beginning" of time. Finite past temporal models seem to be reasonable for describing computer systems, whose execution can be said to have some definite starting point [134].

Finally, having an infinite future (i.e., every time point has a successor), is quite a weak condition. It seems reasonable for reactive systems, which, as pointed out earlier, are generally intended to execute without terminating. However, terminating models can be mapped to non-terminating models by the simple expedient of iterating the final time-point in the sequence infinitely (this is called (infinite) stuttering).

The above discussion has justified the main design decisions made during the development of AL. There are, however, at least three areas where AL might easily be extended:

1. **Internal Languages.** Little has been said about the internal language L, or how the choice of L affects AL. Recall that L is used to express the beliefs of agents. These beliefs generally constitute, for the agent, a model of the world in which the agent exists, (i.e., a multi-agent system of the type AL is used to describe). An interesting question then arises: Could AL be used as an internal language? This issue is discussed in the next section.

2. **Quantification.** A natural extension to AL would be to allow quantification (over at least agents, actions, and individuals in the internal language). A language called QAL ("Quantified AL"), based on AL but allowing quantification, is developed in section 5.3.

3. **Branching Time.** The model of time which underlies AL is *linear*, in that each time point has a unique successor. An alternative is to view time as branching into the future. In a branching system, each time point has a number of possible successors, corresponding to the futures that might arise if various choices are made. Chapter 7 is entirely devoted to the development of logics, with a similar set of atoms to AL, that view time as a branching structure.

5.2 The Internal Languages of AL

In this section, we examine the expressiveness of AL when it is parameterized by various internal languages. We assume the existence of an ordinary first-order language, L_0. This language contains a set of predicate symbols $\{P, Q, \ldots\}$, a set of constants $\{a, b, \ldots\}$, a set of variables, $\{x, y, \ldots\}$, the usual connectives $\{\wedge, \vee, \Rightarrow, \ldots\}$, and the quantifiers $\{\forall, \exists\}$. For simplicity, we assume the language has no function symbols other than constants. The normal formation rules for first-order predicate logic are assumed. Formulae of L_0 are said to be *ordinary*. A *sentence* of L_0 is a closed formula (one with no free variables). See, e.g., [163] for a presentation of L_0.

We begin our analysis by considering the language AL(L_0), (i.e., the language formed by allowing agents to use an ordinary first-order predicate logic as their internal language). AL(L_0) might be called a *first-order intentional language*, since while it allows agents to be described in terms of intentional constructs (i.e., belief), agents themselves are not capable of supporting beliefs about other agents [39]. Some examples of syntactically correct and incorrect formulae of this language are given in Table 5.4.

The internal language is used to express the beliefs of agents. These beliefs are generally taken to constitute — for the agent — a description of the environment in which it exists, i.e., a multi-agent system of the type AL is used to describe. An intriguing possibility arises. Could AL itself be used as an internal language by agents? If we ignore for a moment the computational issues associated with reasoning using a complex modal language

such as AL, the suggestion is quite appealing. The semantics of AL are based on a formal model of multi-agent systems, that described in the preceding chapter. It would therefore seem to be an ideal language for agents to express beliefs *about* such a system. Unfortunately, a naive attempt to apply the idea immediately runs into difficulties. Consider some examples of formulae of the language AL(AL(L_0)).

$$(\mathsf{Bel}\ i\ (\mathsf{Bel}\ j\ \forall x \cdot P(x) \ \Rightarrow\ Q(x)))$$

This example is straightforward: i believes j believes $\forall x \cdot P(x) \ \Rightarrow\ Q(x)$. In contrast, the following formula is *not* syntactically acceptable.

$$(\mathsf{Bel}\ i\ \forall x \cdot P(x) \ \Rightarrow\ Q(x))$$

Yet the argument to the belief atom is a sentence of L_0. The problem lies in the way that AL is defined: formulae of the internal language cannot be formulae of AL. This means that although AL *could* be used in its present form for representing the beliefs of agents, its would be extremely unwieldy.

One solution to this problem is to develop a hybrid language, which combines (say) a first-order temporal logic (FOTL) with AL, and allows either formulae of FOTL or formulae of AL to be atoms. Such a language is developed below.

5.2.1 An Expressive Internal Language

As suggested above, an expressive internal language may be derived by "adding" FOTL to AL, and allowing formulae of FOTL to be atoms of the resulting language, which we will call IAL ("Internal AL"). The FOTL we use is more or less standard (see e.g., the language FML in [12]); a full (but terse) description is given below.

A First-Order Temporal Logic (FOTL)

For the sake of simplicity in exposition, we present a language with equality but no functions.

Definition 9 *The alphabet of FOTL contains the following symbols:*

1. *A countable set of predicate symbols* $\{P, Q, \ldots\}$*, each associated with a non negative integer, its arity;*
2. *A countable set of constant symbols* $\{a, b, \ldots\}$*;*
3. *A countable set of variable symbols* $\{x, y, \ldots\}$*;*
4. *The equality symbol " = ";*
5. *The quantifier " ∀ ", and punctuation symbol " · ";*
6. *The punctuation symbols, propositional and temporal connectives of AL.*

The syntax of FOTL is that of L_0 augmented by the temporal connectives of AL.

Definition 10 *The syntax of FOTL is defined by the following rules:*

1. *A term is either a variable or a constant;*
2. *An atom is an application of a predicate symbol P to terms* $\theta_1, \cdots, \theta_n$*, written* $P(\theta_1, \ldots, \theta_n)$*, where n is the arity of P. A ground atom is one containing no variables;*
3. *An equality is an expression of the form* $(\theta_1 = \theta_2)$ *where* θ_1, θ_2 *are terms;*
4. *A primitive is either an atom or an equality. All primitives are formulae of FOTL;*
5. *If* ϕ *is a formula of FOTL and x is a variable then* $\forall x \cdot \phi$ *is a formula of FOTL;*
6. *FOTL contains the propositional and temporal formation rules of AL.*

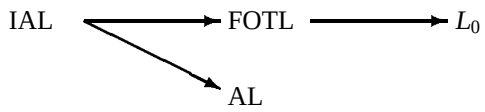
An interpretation π for FOTL is a map from times $\mathbb{N}$ to the powerset of ground atoms: $\pi(u)$ is the set of ground atoms true at time $u \in \mathbb{N}$.

A variable assignment V is a map from variables to constants. A function is defined that applies a variable assignment to an arbitrary term. (Note that this function does *not* return the denotation of a term, but rather a constant.)

$$[\![\theta]\!]_V \ \triangleq\ \begin{array}{l} \textit{if } \theta \in \textit{Var} \\ \textit{then } V(\theta) \\ \textit{else } \theta \end{array}$$

$$
\begin{array}{lcll}
\langle \pi, V, u \rangle & \models & P(\theta_1, \ldots, \theta_n) & \text{iff } P([\![\theta_1]\!], \cdots, [\![\theta_n]\!]) \in \pi(u) \\
\langle \pi, V, u \rangle & \models & (\theta_1 = \theta_2) & \text{iff } [\![\theta_1]\!] = [\![\theta_2]\!] \\
\langle \pi, V, u \rangle & \models & \forall x \cdot \phi & \text{iff } \langle \pi, V \dagger \{x \mapsto a\}, u \rangle \models \phi \\
 & & & \quad \text{for all constants } a
\end{array}
$$

Figure 5.2: Semantics of FOTL



An arc from A to B means $Form(B) \subset Form(A)$.

Figure 5.3: Relationships Between Languages

Where V is understood, $[\![\theta]\!]_V$ is abbreviated to $[\![\theta]\!]$. The semantics of FOTL are presented via the satisfaction relation "$\models$" in the usual way; the relation holds between triples of the form:

$$\langle \pi, V, u \rangle$$

(where π is an interpretation for FOTL, V is a variable assignment, and $u \in \mathbb{N}$ is a temporal index), and formulae of FOTL. See Figure 5.2 for the semantic rules. Since the propositional and temporal connectives are essentially standard, their semantic rules are omitted: semantics are just given for atoms and quantified formulae (the existential quantifier "$\exists$" is assumed to abbreviate $\neg \forall \neg$, as usual).

This completes the survey of FOTL.

Adding FOTL to AL

In this sub-section, we show how FOTL may be added to AL(FOTL), to yield a language IAL ("Internal AL").

How is FOTL "added" to AL(FOTL), syntactically and semantically? Syntactically, the atoms of the resultant language IAL are defined to be either sentences of FOTL or formulae of AL. (Note that AL(FOTL) is abbreviated to AL for the remainder of this section.) Compound formulae are built up from these atoms using the temporal and propositional connectives which are common to both AL and FOTL.

For example, the following is a closed formula of FOTL, and is therefore an atom of IAL.

$$\forall x \cdot \bigcirc P(x) \Rightarrow Q(x)$$

Also, the following is a formula of AL, and is therefore an atom of IAL.

$$(\mathsf{Bel}\ i\ P(a))$$

Compound formulae of IAL may be built up from formulae of FOTL and AL. This means the following is a syntactically acceptable formula of IAL.

$$(\mathsf{Bel}\ i\ P(a)) \wedge \forall x \cdot \bigcirc P(x) \Rightarrow Q(x)$$

Semantically, a model for IAL is constructed by adding an interpretation for FOTL to a model for AL. IAL contains the temporal and propositional connectives of FOTL/AL, and compound formulae are evaluated in the usual way. Now to make these definitions formal.

Definition 11 *The syntax of IAL is defined by the following rules:*

1. *If ϕ is a sentence of FOTL then ϕ is a formula of IAL.*
2. *If ϕ is a formula of AL then ϕ is a formula of IAL.*
3. *IAL contains the propositional and temporal formation rules of AL.*

The syntactic relationships between IAL, AL, FOTL and L_0 are summarized in Figure 5.3. The semantics of IAL are as suggested in the informal discussion above.

Definition 12 *A model for IAL is a structure:*

$$\langle \pi, W, Ag, Ac, I \rangle$$

where π is an interpretation for FOTL and the remainder form a model for AL.

The satisfaction relation "$\models$" for IAL holds between triples of the form:

$$\langle M, V, u \rangle$$

(where M is a model for IAL, V is a variable assignment for FOTL, and $u \in \mathbb{N}$ is a temporal index into M), and formulae of IAL. Let $\models_{AL}$ be the satisfaction relation for AL, and $\models_{FOTL}$ be the satisfaction relation for FOTL. The semantic rules for IAL are then as follows (semantic rules are only given for atoms of IAL; the remainder are essentially unchanged, and are therefore omitted).

$\langle M, V, u \rangle \models \phi$ iff $\langle \pi, V, u \rangle \models_{FOTL} \phi$ where ϕ is a sentence of FOTL

$\langle M, V, u \rangle \models \phi$ iff $\langle \langle W, Ag, Ac, I \rangle, u \rangle \models_{AL} \phi$ where ϕ is a formula of AL

5.2.2 Some Possible Theorems and Their Interpretation

The following section considers the significance of various formulae of AL(IAL).

Saturation, Consistency, and Introspection

Consider the class of agents whose deduction rules are complete and sound with respect to the internal language. A belief set with this property is closed under logical consequence, and is said to be *saturated*. The axiom corresponding to K from classical modal logic is sound for this class (see [100, pp35–36]).

$$(\mathsf{Bel}\ i\ \phi \Rightarrow \psi) \Rightarrow ((\mathsf{Bel}\ i\ \phi) \Rightarrow (\mathsf{Bel}\ i\ \psi)) \tag{5.25}$$

Consistency is an interesting property of agents. We say an agent is *locally consistent* at some time if it does not believe both a formula and its negation at that time, and *globally consistent* if it is always locally consistent. In contrast, an agent is *locally inconsistent* if it is not globally consistent, and *globally inconsistent* if it is never locally consistent. The following schema (corresponding to D in classical modal logic) is sound for the class of globally consistent agents (cf. [100, p37]).

$$(\mathsf{Bel}\ i\ \phi) \Rightarrow \neg(\mathsf{Bel}\ i\ \neg\phi) \tag{5.26}$$

Theorem 1 *Any agent with consistent initial belief set, sound deduction rules, and consistent belief revision function is globally consistent.*
Proof *We need to show that any agent with these properties will be locally consistent at all times $u \in \mathbb{N}$. The case for $u = 0$ is given by the initial set of beliefs being consistent and deduction rules being sound. For $u > 0$, the base beliefs will be the result of processing epistemic inputs through the belief revision function. If the belief revision function is consistent then by definition it will never produce an inconsistent set of base beliefs, so the belief set that results from deductive closure will be consistent.* ∎

The process of reasoning about one's own beliefs is called *introspection*, and is examined in detail in [100, Chapter 5]. The class of *positively introspective belief systems* is characterized by the following schema (corresponding to axiom *4* in classical modal logic).

$$(\mathsf{Bel}\ i\ \phi) \Rightarrow (\mathsf{Bel}\ i\ (\mathsf{Bel}\ i\ \phi)) \tag{5.27}$$

The following schema (corresponding to axiom *5* in classical modal logic) is characteristic of *negatively introspective* belief systems.

$$\neg(\mathsf{Bel}\ i\ \phi) \Rightarrow (\mathsf{Bel}\ i\ \neg(\mathsf{Bel}\ i\ \phi)) \tag{5.28}$$

1.	$(\mathsf{Bel}\ i\ \phi) \Rightarrow \Box(\mathsf{Bel}\ i\ \phi)$	Given
2.	$(\mathsf{Bel}\ i\ \phi) \Rightarrow \neg(\mathsf{Bel}\ i \neg\phi)$	Given
3.	$\Box((\mathsf{Bel}\ i\ \phi) \Rightarrow \neg(\mathsf{Bel}\ i \neg\phi))$	2, TL
4.	$\Box((\mathsf{Bel}\ i\ \phi) \Rightarrow \neg(\mathsf{Bel}\ i \neg\phi)) \Rightarrow (\Box(\mathsf{Bel}\ i\ \phi) \Rightarrow \Box\neg(\mathsf{Bel}\ i \neg\phi))$	TAX
5.	$\Box(\mathsf{Bel}\ i\ \phi) \Rightarrow \Box\neg(\mathsf{Bel}\ i \neg\phi)$	3, 4, PL
6.	$(\mathsf{Bel}\ i\ \phi) \Rightarrow \Box\neg(\mathsf{Bel}\ i\ \phi)$	1, 5, PL

Figure 5.4: Proof of Theorem 2

Temporal Awareness and the Persistence of Belief

Temporal awareness and the persistence of belief are two distinct but related issues. *Temporal awareness* means being in some sense aware of the passage of time: being able to distinguish one's present cognitive state from previous cognitive states. The *persistence* of belief concerns how long beliefs are held for.

We deal with temporal awareness first, and distinguish two types. *Strong* temporal awareness means being explicitly aware of the distinction between *all* one's previous cognitive states. This is most simply described by the following schema.

$$(\mathsf{Bel}\ i\ \phi) \Rightarrow \bigcirc(\mathsf{Bel}\ i\ \bullet\phi) \tag{5.29}$$

It is easy to see that a strongly temporally aware agent (by this definition) believing ϕ at time u will believe $\bullet^v\phi$ at time $u+v$ (where $\bullet^v$ stands for the operator "$\bullet$" iterated v times). *Weak* temporal awareness, as its name suggests, is a weaker condition. It means being able to distinguish what is believed *now* from at least one other, previous cognitive state. This condition is most easily described by the following schema.

$$((\mathsf{Bel}\ i\ \blacklozenge\ \phi) \vee (\mathsf{Bel}\ i\ \phi)) \Rightarrow \bigcirc(\mathsf{Bel}\ i\ \blacklozenge\ \phi) \tag{5.30}$$

That is, a weakly temporally aware agent (by this definition) believing ϕ at time u will believe $\blacklozenge\ \phi$ for all times $v \in \mathbb{N}$ such that $v > u$. (Note that the schemata for temporal awareness are not intended to strictly characterize the associated conditions, but give the simplest and most obvious interpretation to them.)

A related issue is *persistence*: how *long* beliefs are held for. We define an agent as being *persistent* if once believing something, it always believes it. The following schema is characteristic.

$$(\mathsf{Bel}\ i\ \phi) \Rightarrow \Box(\mathsf{Bel}\ i\ \phi) \tag{5.31}$$

Persistence is a strong limiting condition, as the following theorem shows.

Theorem 2 *The following schema is valid for the class of persistent, globally consistent agents.*

$$(\mathsf{Bel}\ i\ \phi) \Rightarrow \Box\neg(\mathsf{Bel}\ i \neg\phi) \tag{5.32}$$

Proof *See Figure 5.4.* ∎

So consistent, persistent agents must remain largely ignorant about the passage of time: once they come to believe something they can never believe its negation. It follows that such agents are severely limited in their scope for reasoning about the world. A more reasonable form of persistence is *default persistence* (cf. [151, pp18–19]). Default persistence means carrying on believing everything that used to be believed unless there is evidence to the contrary. The appropriate schema is

$$((\bullet(\mathsf{Bel}\ i\ \phi)) \wedge \neg(\mathsf{Bel}\ i \neg\phi)) \Rightarrow (\mathsf{Bel}\ i\ \phi). \tag{5.33}$$

This schema has appeared in many guises in AI. For example, it is similar to the so called *STRIPS assumption* for reasoning about a changing world (i.e., that everything not known to have changed is assumed unchanged [57]).

Anticipation and Retention

Suppose an agent is always aware of what messages it sends: such an agent is said to *anticipate* its messages. The following schema is characteristic of agents that anticipate all their messages.

$$(\mathsf{Send}\ i\ j\ \phi) \Rightarrow (\mathsf{Bel}\ i\ (\mathsf{Send}\ i\ j\ \phi)) \tag{5.34}$$

Similarly, consider agents that are aware of the messages they receive. If an agent receives a message and believes that it has received that message, then it is said to be *message retentive*. The following schema is characteristic of message retentive agents.

$$(\mathsf{Rcv}\ i\ j\ \phi) \Rightarrow (\mathsf{Bel}\ i\ (\mathsf{Rcv}\ i\ j\ \phi)) \tag{5.35}$$

The following schema is characteristic of agents that anticipate their actions:

$$(\mathsf{Do}\ i\ \alpha) \Rightarrow (\mathsf{Bel}\ i\ (\mathsf{Do}\ i\ \alpha)) \tag{5.36}$$

Some similar properties were examined by Genesereth and Nilsson in Chapter 13 of [73].

Future Directed Beliefs

Future directed beliefs are simply beliefs about the future. This section explores the possibility that future directed beliefs affect action. To motivate the discussion, consider the following axioms.

$$(\mathsf{Bel}\ i\ \bigcirc(\mathsf{Do}\ i\ \alpha)) \Rightarrow \bigcirc(\mathsf{Do}\ i\ \alpha) \tag{5.37}$$

$$(\mathsf{Bel}\ i\ \bigcirc(\mathsf{Send}\ i\ j\ \phi)) \Rightarrow \bigcirc(\mathsf{Send}\ i\ j\ \phi) \tag{5.38}$$

By these axioms, whenever an agent believes it will perform an action or send a message, then it actually *does* perform the action/send the message. The axioms describe a direct relationship between the beliefs of an agent and the actions it performs. An interesting possibility suggests itself: what if an agent has beliefs about what it will do in the future, and these beliefs have a causal effect on what the agent actually does? The possibility of axiomatizing such a relationship is explored in this section. (The idea of future directed beliefs having a causative effect on subsequent actions is at the heart of an entire programming paradigm called METATEM [12], [62] — much of the discussion in this section was motivated by the METATEM approach.)

We begin by more formally defining what we mean by a future directed belief. We say an agent's belief is future directed if it satisfies the following conditions:

- the belief refers *only* to the actions of the believer;
- the belief contains *only* propositional and future time temporal operators.

The first condition arises because we are only concerned with how an agent's beliefs about its own future actions affect its behaviour. An obvious axiom to examine is a restricted form of T from classical modal logic.

$$(\mathsf{Bel}\ i\ \phi) \Rightarrow \phi \quad \text{where } \phi \text{ satisfies the conditions above} \tag{5.39}$$

Clearly, the two axioms given at the start of this section are instances of this schema. The axiom T is generally taken to be the characteristic axiom of *knowledge*, i.e., true belief. However, it can be interpreted as expressing *constraints* on an agent's behaviour. If an agent is to satisfy the modified T, then it must pick its actions so as to satisfy the constraints expressed in its beliefs. Conceptually, we might describe an agent as trying to find a model for those of its beliefs that satisfy the above conditions. If no such model exists, then there is no way the agent can act so as to satisfy them. One possibility would then be to relax the constraints — for example by dropping a belief about future action. In practice, trying to use such a method to choose actions is unworkable, due to the complexity of the decision problem for even simple temporal logics.

5.3 A First-Order Linear Time Logic for Multi-Agent Systems

It should be clear from the discussions presented above that the expressiveness of AL is limited in a number of important ways. Among these limitations is the inability to write AL formulae which correspond to the following English sentences.

Ralph performed an action. (5.40)

Ralph knows who the murderer is. (5.41)

Everyone believes Ralph is the murderer. (5.42)

Ralph sent Freda a request for something. (5.43)

Everyone sent someone a request for something. (5.44)

The problem is that AL is not a first-order language. It is not possible to quantify over terms in AL. The aim of this section is to develop a language QAL ("Quantified AL"), based on AL, which rectifies this omission.

5.3.1 Syntax

The alphabet of QAL contains the following symbols in addition to those of AL:

- some extra constants, $Const_U$, for individuals in the domain of the internal language;
- a set *Var* of *individual variables*, made up of the disjoint sets Var_{Ag} (agent variables), Var_{Ac} (action variables), and Var_U, (variables denoting individuals in the domain U of L);
- the equality symbol "=";
- the quantifier symbol "$\forall$";
- the punctuation symbol "$\cdot$".

Definition 13 *A term is either a variable or constant. The symbol θ is used to denote a term. The sort of a term is either Ag, Ac or U. To indicate that a term θ is of sort s we write θ_s.*

The syntax of QAL is given by the following definition.

Definition 14 *The syntax of QAL (based on L) is defined by the following rules:*

1. *Let $\theta_s, \theta'_s, \ldots$ be terms of named sorts, and ϕ be a formula of L. Then the following are atomic formulae of QAL:*
   true $\quad$ (Bel θ_{Ag} ϕ) $\quad$ (Send θ_{Ag} θ'_{Ag} ϕ) $\quad$ (Do θ_{Ag} θ_{Ac}) $\quad$ ($\theta_s = \theta'_s$)
2. *If ϕ is a formula of QAL and x is a variable then $\forall x \cdot \phi$ is a formula of QAL.*
3. *QAL contains the propositional and temporal formation rules of AL.*

Note that arbitrary formulae of L are now allowed to be arguments to the Bel operator, rather than the closed formulae required in AL. The propositional and temporal abbreviations made for AL are assumed. It is easy to see that the following formulae correspond to the English sentences (5.40) to (5.44), above.

$\exists x \cdot$ (Do *Ralph* x) (5.45)

$\exists x \cdot$ (Bel *Ralph Murderer*(x)) (5.46)

$\forall x \cdot$ (Bel x *Murderer*(*Ralph*)) (5.47)

$\exists x \cdot$ (Send *Ralph Freda Request*(x)) (5.48)

$\forall x \cdot \exists y \cdot \exists z \cdot$ (Send x y *Request*(z)) (5.49)

5.3.2 Semantics

The semantics of QAL are presented in two parts: first, model structures are defined, and then the semantics of the language are presented via the satisfaction relation "$\models$", in the usual way. The definitions of satisfiability and validity are essentially standard, and are therefore omitted.

Model Structures

The language of QAL contains three non-empty sets of constants which are standard names: $Const_{Ag}$ (denoting agents), $Const_{Ac}$ (denoting actions), and $Const_U$ (constants for individuals in the domain of L). *Const* is the union of these sets. Each of these sets corresponds to a set that occurs in QAL model structures:

- Ag — a set of agents;
- Ac — a set of actions;
- U — a universe of individuals, the domain of L.

The *domain of quantification* is the union of these sets; so quantification is allowed over agents, actions, and individuals in the domain of L. A bijective map between constants and the domain of interpretation that preserves sorts is an *interpretation for constants* (as in AL). The symbol I is usually used for an interpretation, as in AL.

The inverse of an interpretation I is a *naming function* N_I. (Where I is understood, N_I is abbreviated to N.) Thus a naming function assigns a unique standard name to each individual in the domain of quantification[5]. In addition to constants, QAL contains a set of variables *Var*. A mapping from *Var* to the domain of quantification that preserves sorts is a *variable assignment.* The symbol V is usually used for a variable assignment. A transformation is now defined on formulae of L.

Definition 15 *Let V be a variable assignment, N be a naming function, and ϕ be an arbitrary formula of L. By $\phi^{N,V}$ we mean the formula obtained from ϕ by replacing every variable x which occurs free in ϕ by $N(V(x))$.*

For example, let $P(x, a_2, y)$ be a formula of L, $V(x) = d_1$, $V(y) = d_3$, $N(d_1) = a_1$, and $N(d_2) = a_2$, then $P(x, a_2, y)^{N,V} = P(a_1, a_2, a_3)$.

So every free variable is replaced by the standard name associated with the object the variable denotes. Finally, a function $[\![\_]\!]_{I,V}$ is defined, which returns the denotation of a term relative to an interpretation for constants and variable assignment.

$$[\![\theta]\!]_{I,V} \triangleq \begin{array}{l} \textit{if } \theta \in \textit{Const} \\ \textit{then } I(\theta) \\ \textit{else } V(\theta) \end{array}$$

Where I, V are understood, $[\![\theta]\!]_{I,V}$ is abbreviated to $[\![\theta]\!]$. Model structures for QAL can now be defined.

Definition 16 *A model for QAL is a structure:*

$$\langle W, Ag, Ac, U, I \rangle$$

where

- *U is a non-empty universe of individuals (the domain of L),*
- *I: $Const \xleftrightarrow{m} (Ag \cup Ac \cup U)$ interprets constants*

and the remainder are as in AL.

(*Aside.* The semantics of QAL require that we know who the individuals are that agents have names for, in order that they can be assigned names. However, it is *not* necessary to know anything about these individuals. So if the semantics of QAL say that Ralph believes Freda is the murderer, it is necessary to know who the individual is that the standard name "Freda" denotes. However, it is not necessary to have an opinion on the properties that Freda has, or how Freda stands in relation to other individuals in the domain.)

Semantic Rules

The satisfaction relation, "$\models$", for QAL, holds between triples of the form:

$$\langle M, V, u \rangle$$

(where M is a model for QAL, V is a variable assignment, and $u \in \mathbb{N}$ is a temporal index into M), and formulae of QAL. The semantic rules for QAL are presented in Figure 5.5. (The rules for propositional and temporal connectives remain essentially unchanged, and are therefore omitted).

As usual, the existential quantifier "$\exists$" is assumed to abbreviate $\neg \forall \neg$.

[5] The function N plays a similar role to the naming function η in Konolige's logic L^{Bq} [100, pp39–40]. It is made simpler by the standard names assumption.

$\langle M, V, u\rangle$	$\models$	(Bel θ_{Ag} ϕ)	iff $\phi^{V,N} \in state(W(u))([\![\theta_{Ag}]\!])$
$\langle M, V, u\rangle$	$\models$	(Do θ_{Ag} θ_{Ac})	iff $action(trans(W(u))([\![\theta_{Ag}]\!])) = [\![\theta_{Ac}]\!]$
$\langle M, V, u\rangle$	$\models$	(Send θ_{Ag} θ'_{Ag} ϕ)	iff $\langle [\![\theta_{Ag}]\!], [\![\theta'_{Ag}]\!], \phi^{V,N}\rangle \in sent(trans(W(u)))$
$\langle M, V, u\rangle$	$\models$	$(\theta_s = \theta'_s)$	iff $[\![\theta_s]\!] = [\![\theta'_s]\!]$
$\langle M, V, u\rangle$	$\models$	$\forall x \cdot \phi$	iff $\langle M, V \dagger \{x \mapsto d\}, u\rangle \models \phi$ for all d in the domain of quantification such that d is the same sort as x

Figure 5.5: Semantics of QAL

5.3.3 Proof Theory

QAL contains all the axioms and inference rules of AL, and in addition the rules of a many-sorted first-order logic with equality but no functions (the restrictions on the proof theory of a many-sorted first-order logic are largely a result of the need to substitute objects of the correct sort). The axiomatization below is based on [104, pp45–46].

Throughout the axiomatization, we use the normal form $\phi(x)$ to mean that x is *free* in ϕ; the meaning of "free" is standard. Additionally, $\phi[x/y]$ is used to denote the formula obtained from ϕ by systematically replacing every occurrence of y in ϕ by x; i.e., normal substitution.

$$\vdash \quad \forall x \cdot \phi(x) \Rightarrow \phi[\theta/x] \left\{ \begin{array}{l} \text{where } \theta \text{ is a term that} \\ \text{does not occur free in } \phi \\ \text{and } x \text{ and } \theta \text{ are the same sort} \end{array} \right. \tag{5.50}$$

$$\vdash \quad (\theta = \theta) \tag{5.51}$$

$$\vdash \quad (\theta = \theta') \Rightarrow (\phi \Rightarrow \phi[\theta'/\theta]) \tag{5.52}$$

Axiom (5.50) is the universal instantiation axiom. Axioms (5.51) and (5.52) deal with equality.

Note that the Barcan formulae for temporal operators are sound ([104, p47]). We give a fuller list of axioms in Appendix D; below, we just list the axioms for the "$\bigcirc$" operator.

$$\vdash \quad \forall x \cdot \bigcirc \phi(x) \Leftrightarrow \bigcirc \forall x \cdot \phi(x) \tag{5.53}$$

$$\vdash \quad \exists x \cdot \bigcirc \phi(x) \Leftrightarrow \bigcirc \exists x \cdot \phi(x) \tag{5.54}$$

The only basic inference rule is generalization.

$$\begin{array}{l} \text{From} \quad \vdash \quad \phi \Rightarrow \psi \\ \text{infer} \quad \vdash \quad \phi \Rightarrow \forall x \cdot \psi \end{array} \left\{ \begin{array}{l} \text{where there is no free} \\ \text{occurrence of } x \text{ in } \phi \end{array} \right. \tag{5.55}$$

Some Possible Theorems and Their Interpretation

There are some other interesting theorems which we might consider [100, pp45–47]. The first is the *Barcan formula* for belief (see Chapter 2 and [88, pp142–143]).

$$\forall x \cdot (\text{Bel } i\ \phi(x)) \Rightarrow (\text{Bel } i\ \forall x \cdot \phi(x)) \tag{5.56}$$

Let D be the domain of quantification: the universally quantified variable x ranges over D. This theorem thus means that "if i believes $\phi(d)$ of every element $d \in D$ then i believes every $d \in D$ has the property ϕ". The antecedent and consequent of this implication convey quite different information, as the following example illustrates.

Suppose there were just two elements in D (an unlikely occurrence, but bear with us!) Let the names of these elements be a_1 and a_2. Suppose also that (Bel i $\phi(a_1)$) and (Bel i $\phi(a_2)$). Then i believes of every element $d \in D$ that $\phi(d)$, i.e., the antecedent of the Barcan formula is satisfied. So if the Barcan formula correctly describes i then i will also believe $\forall x \cdot \phi(x)$.

The Barcan formula is a fairly extreme property, as it requires that an agent is somehow aware of all the elements in the domain D. Another interesting property is the reverse implication of the Barcan formula, the so called *converse Barcan formula*. This schema characterizes agents whose deduction rules include universal instantiation.

$$(\text{Bel } i\ \forall x \cdot \phi(x)) \Rightarrow \forall x \cdot (\text{Bel } i\ \phi(x)) \tag{5.57}$$

Finally, suppose an agent can perform existential generalization: from believing that a specific agent (named by a standard name) has a property, it can deduce that *some* agent has the property. The following schema is characteristic:

$$\exists x \cdot (\mathsf{Bel}\ i\ \phi(x)) \ \Rightarrow\ (\mathsf{Bel}\ i\ \exists x \cdot \phi(x)) \tag{5.58}$$

5.4 Summary

This chapter has described in detail a number of linear time temporal logics for reasoning about multi-agent systems. The first, simplest, of these logics was the propositional logic AL. The syntax, semantics and proof theory of this logic were discussed at length. A correspondence was established between systems in the theory of multi-agent systems presented in the preceding chapter, and models of AL; this correspondence was that of a model representing a "run" of a system. Some problems were identified with the possibility of using AL as an internal language; in order to overcome these problem, a "hybrid" language called IAL was developed, based on a first-order temporal logic and AL. Some possible theorems of IAL were presented, and their meaning discussed. Finally, a quantified version of AL called QAL was developed.

The next chapter will use these logics to develop specifications and axiomatizations of various systems.

Chapter 6

Five Examples

THE preceding chapters have laid down a theory of multi-agent systems, and some linear time temporal logics which may be used to reason about systems modelled by the theory. This chapter presents five case studies, which demonstrate how these logics may be used for modelling and reasoning about multi-agent systems. The case studies are divided into two groups:

- axiomatizations of frameworks for building multi-agent systems;
- specifications of cooperative problem solving paradigms.

The two frameworks for building multi-agent systems axiomatized are: Shoham's agent oriented programming paradigm, and particularly the AGENT0 system [151], [171], and Fisher's concurrent METATEM processes [62], [60].

The cooperative problem solving paradigms specified are: a simple "master/slave" framework, along the lines of that described in [140], the cooperative inference technique employed in the FELINE cooperating expert system [189], and the contract net protocol [159], [161], [160], [162].

Before presenting these case studies, a brief discussion is presented on reasoning about multi-agent systems using temporal logics.

6.1 Reasoning About Multi-Agent Systems

The literature on reasoning about reactive systems using temporal logic is, quite literally, enormous. A detailed review of this literature is well beyond the scope of this thesis (however, some more details appear in Appendix C). The reader is therefore referred to Emerson's excellent introduction to the area [47], and Pnueli's detailed technical reviews [135], [134]. All of these articles have useful bibliographies. A volume devoted to the theoretical aspects of the logics used in this work is [78]. An article by Barringer [11] contains a good example of how a formal, temporal logic specification may be derived from an informal specification, and serves to outline most of the key techniques.

How are our logics to be used for specifying, verifying, and more generally, reasoning about multi-agent systems? Consider the specification case.

Very crudely, a specification is a description of the desired behaviour of a system. During implementation, one aims for a program that will *satisfy* the specification. To show that a program satisfies its specification is to *verify* it.

Temporal logics have been widely used in the specification of reactive systems, of which multi-agent systems are an example (see Pnueli's comments quoted in Chapter 1). This is because, when giving a specification for a reactive system, it is natural to want to express such statements as: "if a request is sent, then a response should eventually be sent". Statements such as this can be elegantly expressed in temporal logic, whereas they are not so easy to express in other formalisms. A specification for a multi-agent system can therefore be expressed as a set of temporal logic formulae; call a specification $\mathcal{SPEC}$. The idea is that each formula in $\mathcal{SPEC}$ is valid in the class of models representing runs of systems which satisfy the specification.

One useful feature of such specifications is that it is possible to explore their properties mathematically. For example, to show that $\mathcal{SPEC}$ entails some property $\mathcal{PROP}$, one can try to establish the relation

$$\mathcal{SPEC} \vdash \mathcal{PROP}$$

using the proof systems established for the logic of specification. An examination of the particular properties of a system that may be specified in temporal logic is presented in Appendix C.

Now consider the verification case. The verification of a system $\mathcal{SYS}$ involves showing that $\mathcal{SYS}$ satisfies its specification $\mathcal{SPEC}$. Verification depends on the idea of the *temporal theory* of a system. The temporal theory $\mathcal{TH}(\mathcal{SYS})$ of $\mathcal{SYS}$ is a set of formulae, true of every run of $\mathcal{SYS}$. The verification process for $\mathcal{SYS}$ with specification $\mathcal{SPEC}$ involves demonstrating the relation

$$\mathcal{TH}(\mathcal{SYS}) \vdash \mathcal{SPEC}$$

using the proof systems established for the logic of specification. Typically, the temporal theory of a system will include "system specific" formulae, as well as formulae capturing more general properties of systems.

An obvious question is, given a particular system $\mathcal{SYS}$, implemented using some established framework for building multi-agent systems, how is $\mathcal{TH}(\mathcal{SYS})$ to be obtained?

Suppose we were to use AL as the language of specification. The first step in developing $\mathcal{TH}(\mathcal{SYS})$ would be to model $\mathcal{SYS}$ using the theory of multi-agent systems developed in Chapter 4. The result would be some system; call it *sys*. Corresponding to this system would be a set of models for AL:

$$\mathcal{M}(\mathcal{SYS}) \triangleq \{M \mid M \in \mathit{Model}_{AL} \wedge \mathit{model\_of}(M, \mathit{sys})\}$$

$\mathcal{TH}(\mathcal{SYS})$ is then the set of formulae valid in $\mathcal{M}(\mathcal{SYS})$:

$$\mathcal{TH}(\mathcal{SYS}) \triangleq \{\phi \mid \phi \in \mathit{Form}(AL) \wedge \models_{\mathcal{M}(\mathcal{SYS})} \phi\}$$

Moving on to the more general case, it may be possible to *axiomatize* the properties of a specific system, which may help in the development of the temporal theory of a system.

A Note on the Specifications

All specifications and axiomatizations in this chapter are given using QAL, the quantified version of AL. Free variables in formulae are assumed to be universally quantified: in practice, only agent identifiers are left free.

It is worth observing that the specifications/axiomatizations given are not intended to be definitive in any sense. Specification is a personal thing. One has choices about what to specify at every step, and different people will specify different things in different ways. There is also the question of how much detail to go into. It is a sad fact that formal specifications tend to require exhaustive, tedious amounts of detail in order to be useful. One therefore has to make choices about exactly what is specified, and what is left unsaid. So the specifications given herein should be taken as a pointer to what is possible, rather than as definitive statements.

6.2 Axiomatizing Two Frameworks for Multi-Agent Systems

This section attempts to describe two existing frameworks for multi-agent systems in terms of the theory of multi-agent systems described earlier, and thereby justify the claim that the theory does indeed reflect how multi-agent systems are actually built. The frameworks are then axiomatized using the logics developed earlier. The frameworks chosen are Shoham's agent oriented programming paradigm, [151], and the AGENT0 system in particular [171], and Fisher's Concurrent METATEM paradigm [62], [60].

6.2.1 Agent Oriented Programming and AGENT0

Agent Oriented Programming (AOP) is "a new programming paradigm, based on a societal view of computation" [151, p4]. The idea which informs AOP is that of using mentalistic notions, (such as belief and choice), in a declarative programming regime. AOP is unusual — perhaps unique — in that it is based, (or, at least, claims to be based), on a general theory of agency, outlined in [150]. At the time of writing, only a prototype (called AGENT0) of the AOP paradigm has been implemented [171]. This section shows that AGENT0 can be modelled within the theory of multi-agent systems described in Chapter 4, and axiomatizes AGENT0 using the logics developed in Chapter 5.

Shoham identifies three components that will make up a complete AOP system [151, p12]:

- a formal (logical) system for defining mental state, (presumably based on the partial theory of agency reported in [150]);
- an interpreted programming language for programming agents;

$\mathtt{B}_a^t\ \phi$	Agent a believes ϕ at time t
$\mathtt{CMT}_{a,b}^t\ \phi$	Agent a is committed to b about ϕ at time t
$\mathtt{CAN}_a^t\ \phi$	Agent a can bring about ϕ at time t

Table 6.1: Modalities in AOP

- an "agentification" process, for converting agent programs into low-level executable objects.

AOP therefore admits three levels of description for agents: first, in terms of a logical language; second, in terms of a programming language, and third, in terms of a low-level device. At the time of writing, only the first two components have received any attention. (Shoham writes that "the third is still somewhat mysterious to me" [151, p12], but later indicates that he is thinking of Rosenschein and Kaelbling's "situated automata" approach [143].) The review below will therefore ignore the "agentification" process.

The first component, the language for defining mental state, is informally described in [151, §4]. No semantics are given for the language, although it appears to be based on the theory of agency outlined in [150]. The language is a modal predicate logic with three modalities: B, for belief, CMT, for commitment, and CAN for capability (see Table 6.1 for details).

Woven into this language is a method for reasoning about time, based on the "method of temporal arguments", briefly described in the preceding chapter (and in more detail in [137, pp155–164]). Thus rather than augment the language with modalities for describing time, each modal operator and predicate is given an extra argument to describe the time at which it is true. The following is a syntactically acceptable formula of Shoham's language:

$$\mathtt{CAN}_a^5\ \mathit{open}(\mathit{door})^8\ \Rightarrow\ \mathtt{B}_b^8\ \mathtt{CAN}_a^5\ \mathit{open}(\mathit{door})^8$$

This formula is read: "if at time 5 agent a can ensure that the door is open at time 8, then at time 8 agent b believes that at time 5 agent a can ensure that the door is open at time 8".

Of the three operators, it seems clear what B and CAN mean. The CMT operator is, however, unusual, and seems worthy of closer study. Sadly, too few details are given for its role to be precisely understood.

No proof system is presented for the language, and no restrictions are placed on beliefs and commitments, other than: 1) beliefs and commitments are assumed to be internally consistent, 2) agents are not committed to anything they do not believe they are capable of, and 3) agents are aware of what they are committed to, and what other agents have committed to on their behalf. Finally, some persistence of mental state is assumed (see the comments on persistence in the previous chapter).

The next component in the makeup of an AOP system is the interpreted programming language. The concrete example given by Shoham is the AGENT0 system (see [151, §6] and [171]). This system will now be described.

When developing the theory which underlies AOP, Shoham avoids introducing the notion of actions as primitives. However, "since actions are such a natural concept [they are introduced] as syntactic sugar" [151, p15]. In AGENT0, two types of action are available to agents:

- private, or internal (cf. cognitive) actions — it is clear from [171] that private actions correspond to the invocation of a procedure in a programming language;
- communicative actions — only three types of communicative act are allowed in AGENT0: INFORM, (which causes the recipient to believe the content of the message), REQUEST, and UNREQUEST, (which attempt to get the recipient to commit to and un-commit to something, respectively).

An AGENT0 agent definition, or program, consists of three components:

- a set of *capabilities*, each of which is a condition/action pair, the condition determining under what circumstances the action may be performed — the condition is a formula of the modal language described above;
- a set of *initial beliefs*, which are formulae of the modal language described above;
- a set of *commitment rules*, defining under what circumstances the agent will adopt commitments.

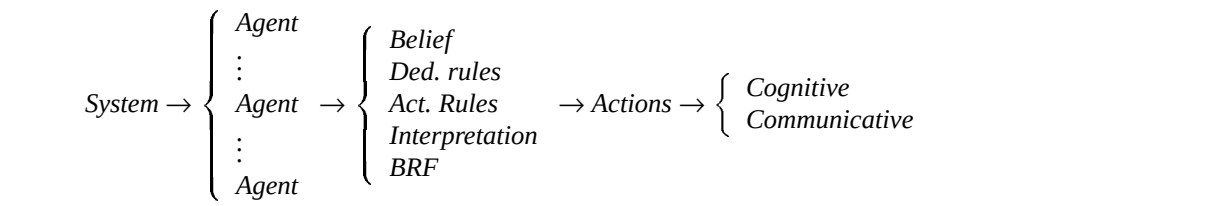


Figure 6.1: Components of the Theory of Multi-Agent Systems

Of these three components, the first and last will remain fixed throughout the execution life of an agent; the beliefs of the agent will change with time[1]. The agent program proper is thus just the set of commitment rules; it is worth looking in detail at what these rules are.

A commitment rule is simply a quad:

$$\langle msgcond, mntlcond, agent, act \rangle$$

where *msgcond* is a condition on the messages that have been received by the agent at the current time; *mntlcond* is a "mental condition" on the beliefs that the agent currently has; *agent* is an agent name, and *act* is an action expression, dictating an action to be performed. The precise nature of the conditions is not important. Suffice to say that they may contain variables, which may be quantified universally or existentially. Suppose the above rule was possessed by agent *a*. Then on every "cycle" in *a*'s life, it would check to see whether the rule fired; it would do this by matching the messages received against *msgcond*, and beliefs held against *mntlcond*. If the rule matched, and thus fired, the following belief would be added to *a*'s belief set:

$$\mathtt{CMT}^{now}_{a,agent}\ act.$$

This would mean that *a* had become committed to performing the action *act* for agent *agent* at time "now" (i.e., the current time). Thus throughout an agent's life, its commitment rules continually generate commitments which the agent subsequently tries to discharge. The operation of an agent can therefore be described by the following loop:

1. Read all current messages, updating beliefs — and hence commitments — where necessary;
2. Execute all commitments for the current cycle where the capability condition of the associated action is satisfied;
3. Goto (1).

This completes the review of AGENT0. The task is now to see to what extent AGENT0 can be modelled by the theory of multi-agent systems. To do this, each component of the theory is taken in turn, and discussed with respect to AGENT0. Recall the components of the theory, illustrated in Figure 6.1.

Belief and reasoning. In AGENT0, agents maintain belief sets in precisely the way described by the theory of multi-agent systems. These beliefs are formulae of the modal language described above, which is therefore identified as the internal language, L. In the AGENT0 implementation described in [171], agents have no deductive capabilities, and therefore each agent is assigned an empty set of deduction rules.

Rules and actions — cognitive and communicative. Clearly, the private (cf. cognitive) and communicative act in AOP map directly onto their equivalents in the theory of multi-agent systems. Moreover, each private action in AGENT0 is associated with a condition determining its applicability — these condition/action pairs, held in the capabilities database, map directly onto rules. There is, however, one important caveat: mental conditions may contain quantifiers.

In AGENT0, private acts are allowed to cause epistemic inputs (this is clear from the user manual [171, §5, p3]). There is no interpretation process in AGENT0 — the interpretation of a message may be viewed as the message itself.

Belief revision. The belief revision process in AOP is rather complex. It involves the following steps on every cycle:

[1] Shoham actually suggests that agents possess a "commitments database", defining the commitments an agent has. However, since agents are aware of their commitments, (i.e., if they are committed to something, then they believe they are committed to it), a separate database seems redundant — at least conceptually. Indeed, the authors of the AGENT0 programming manual write that an agent can gain a commitment by "asserting the commitment into [its] beliefs" [171, p5].

1. Generating a new belief set by:
   - for each received INFORM message, remove any old beliefs that are inconsistent with INFORM'd information, and add the new information;
   - for each UNREQUEST message, remove from belief set any UNREQUEST'd commitments;
2. Generating new commitments, (by comparing rules against beliefs and epistemic inputs to see which commitment rules fire, as described above), and adding them to the belief set that resulted from the first step.

Axiomatizing AGENT0

What aspects of the behaviour of AGENT0 systems may be axiomatized in the languages developed earlier? The first, and most useful technique is to write axioms for every commitment rule possessed by an agent. Recall that a commitment rule is a quad:

$$\langle msgcond, mntlcond, agent, act \rangle$$

where *mntlcond* and *msgcond* are message conditions and mental conditions, respectively. Such commitment rules may be rewritten as formulae of QAL, with the following general form:

$$\left[\begin{array}{c} \overbrace{((\mathsf{Send}\ j_1\ i\ \phi_1) \wedge \cdots \wedge (\mathsf{Send}\ j_m\ i\ \phi_m))}^{msgcond} \\ \wedge \\ \bigcirc \underbrace{((\mathsf{Bel}\ i\ \phi_n) \wedge \cdots \wedge (\mathsf{Bel}\ i\ \phi_o))}_{mntlcond} \end{array} \right] \Rightarrow (\mathsf{Bel}\ i\ \mathtt{CMT}_{i,agent} act) \tag{6.1}$$

Shoham gives an example of a commitment rule, which might be expressed in English: "if I've been asked to perform action α by agent j [the message condition], and I believe j is my friend [the mental condition], then commit to α". This rule might be expressed as follows:

$$\left[\begin{array}{l} (\mathsf{Send}\ j\ i\ \mathtt{REQUEST}(j, \alpha)) \quad \wedge \\ \bigcirc (\mathsf{Bel}\ i\ \mathit{MyFriend}(j)) \end{array} \right] \Rightarrow (\mathsf{Bel}\ i\ \mathtt{CMT}_{i,j}\ \alpha)$$

The only real failing of this rule with respect to Shoham's original is that the explicit reference to time that appeared in the original cannot be expressed in any of the logics developed in this thesis, and has therefore been dropped. Note that other properties of commitments could be specified, if desired: for example, it would be possible to specify that commitments were eventually discharged.

The properties of INFORM acts are given by the following formula.

$$(\mathsf{Send}\ j\ i\ \mathtt{INFORM}(j, fact)) \Rightarrow \left[\begin{array}{ll} (\mathsf{Bel}\ i\ fact) & \wedge \\ (\mathsf{Bel}\ i\ (\mathsf{B}_j\ fact)) & \end{array} \right] \tag{6.2}$$

Similarly, the following property of UNREQUEST acts holds.

$$(\mathsf{Send}\ j\ i\ \mathtt{UNREQUEST}(j, \alpha)) \Rightarrow \neg (\mathsf{Bel}\ i\ \mathtt{CMT}_{i,j}\ \alpha) \tag{6.3}$$

For entries in the capabilities database, a similar technique to that used for commitment rules is used. Entries in the capabilities database are pairs, of the form

$$\langle mntlcond, act \rangle$$

the meaning being that *act* can only be performed if *mntlcond* is believed. This gives axioms of the following form:

$$(\mathsf{Do}\ i\ act) \Rightarrow \bigcirc \underbrace{((\mathsf{Bel}\ i\ \phi_1) \wedge \cdots \wedge (\mathsf{Bel}\ i\ \phi_n))}_{mntlcond} \tag{6.4}$$

This concludes the axiomatization of AGENT0.

6.2.2 Concurrent METATEM Processes

Concurrent METATEM Processes (CMP) [62], [60] is a framework for building multi-agent systems that is based on the METATEM temporal logic programming paradigm [12]. A CMP system contains a set of concurrently executing *objects*, which are able to communicate through asynchronous broadcast message passing. Each object has two main components:

- an *interface*, which defines how the object may interact with its environment (i.e., other objects);
- a *computational engine*, which defines how the object acts.

The interface itself consists of three components:

- a unique *object identifier* (or just object id), which names the object;
- a set of symbols defining what messages may be received by the object — these are called *environment predicates*;
- a set of symbols defining messages that the object may send — these are called *component predicates*.

For example, the interface definition of a "stack" object might be [60]:

`stack(pop,push)[popped,stackfull]`

Here, "`stack`" is the name of the object, {`pop`, `push`} are the environment predicates, and {`popped`, `stackfull`} are the component predicates.

The computational engine of an object is based on the METATEM temporal logic programming paradigm. The idea which informs this approach is that of directly executing a declarative object specification, where this specification is given as a set of *rules*, which are temporal logic formulae of the form:

$$\begin{array}{c}\text{Antecedent}\\ \text{about past}\end{array} \Rightarrow \begin{array}{c}\text{Consequent}\\ \text{about future.}\end{array}$$

The past-time antecedent is a temporal logic formula referring strictly to the past, whereas the future time consequent is a temporal logic formula referring either to the present or future. The intuitive interpretation of such a rule is "on the basis of the past, do the future", which gives rise to the name of the paradigm: declarative past and imperative future [64]. The actual execution of an object is, superficially at least, very simple to understand. Each object obeys a cycle of trying to match the past time antecedents of its rules against a recorded *history*, and executing the consequents of those rules that "fire"[2].

The language used for object rules is the first-order temporal logic, FOTL, described in the preceding chapter (in the METATEM literature, the language is called FML — for "First-order METATEM Logic"). To recap, this language is the ordinary first-order predicate logic L_0, augmented by the set of temporal modal operators that appear in AL.

The "past $\Rightarrow$ future" form of rules are actually restricted to be in *Separated Normal Form* (SNF) [61]; however, the details of SNF are not important here. For our purposes, we can assume that an SNF formula is simply one in the "past $\Rightarrow$ future" form.

It is useful to identify various subsets of formulae of FOTL. The set $Form_{SNF}(FOTL)$ is the set of SNF formulae of FOTL; the set $Form_{<}(FOTL)$ is the set of FOTL formulae referring strictly to the past; $Form_{\geq}(FOTL)$ is the set of FOTL formulae referring to the present or future. Any element of $Form_{SNF}(FOTL)$ is called a *rule*; any element of $Form_{<}(FOTL)$ is called a *history* formula, and any element of $Form_{\geq}(FOTL)$ is called a *commitment*.

A more precise definition of object execution will now be given. Objects continually execute the following cycle:

1. Update the *history* of the object by receiving messages from other objects and adding them to the history. (This process is described in more detail below.)
2. Find out which rules *fire*, by "matching" the past time antecedents of the rules against the history. Note that the matching process is a deductive one, not a literal one. For example, suppose an object has a history formula $\bullet\phi$ and a rule $\blacklozenge\ \phi \Rightarrow \psi$ then the rule would fire, since $\bullet\phi \vdash \blacklozenge\ \phi$.

[2] There are obvious similarities between the execution cycle of an object and production systems — but there are also significant differences. The reader is cautioned against taking the analogy too seriously.

$rc1(give1)[ask1]$:
$\bullet\, ask1 \;\Rightarrow\; ask1;$

$rc2(give2, ask1)[ask2]$:
$\circ\!\!\!\bullet\, (ask1 \wedge \neg\, ask2) \;\Rightarrow\; ask2;$

$rp(ask1, ask2)[give1, give2]$:
$\circ\!\!\!\bullet\, ask1 \;\Rightarrow\; \Diamond give1;$
$\circ\!\!\!\bullet\, ask2 \;\Rightarrow\; \Diamond give2;$
$\bullet\, \mathsf{true} \;\Rightarrow\; \neg\, (give1 \wedge give2);$
$(\neg\, ask1)\; \mathcal{Z}\; (give1 \wedge \neg\, ask1) \;\Rightarrow\; \neg\, give1;$
$(\neg\, ask2)\; \mathcal{Z}\; (give2 \wedge \neg\, ask2) \;\Rightarrow\; \neg\, give2;$

Figure 6.2: A Simple CMP System

3. *Jointly execute* the fired rules, together with any commitments carried over from previous cycles. This is done by first collecting the consequents of newly fired rules and old commitments, which become *new commitments*. It may not be possible to satisfy *all* commitments on the current cycle, in which case unsatisfied commitments are carried over to the next cycle. An object will then have to choose between a number of execution possibilities. Those commitments which are satisfied are considered to have become "true".

4. Goto (1).

Clearly, step (3) is the heart of the execution process. Making a bad choice at this step may mean that the object specification cannot subsequently be satisfied. Unfortunately, the process of finding the best choice is difficult (see [12] for details).

A natural question to ask is, how do objects do things — perform actions and send messages, that is? The idea is that an action is performed when a commitment becomes true. If the commitment is a component predicate, as defined in that objects interface (see above), then that predicate is broadcast as a message to all other objects. On receipt of a message, each object attempts to match the predicate against the environment predicates in their interface. If there is a match then they add the predicate to their history, prefixed by a "$\circ\!\!\!\bullet$" operator. Finally, if the commitment corresponds to an action, then the action associated with the commitment is executed.

To illustrate the CMP paradigm in more detail, Figure 6.2 gives a definition of a CMP system (adapted from [60]). To simplify things a little, this system is a propositional one. The system contains three objects: *rp* (resource producer), *rc*1 (resource consumer one) and *rc*2 (resource consumer two). The object *rc*1 will send an *ask*1 message on every cycle. The object *rc*2 will send an *ask*2 message on every cycle such that: 1) it did not "ask" on the previous cycle, and 2) *rc*1 *did* ask on the previous cycle.

The resource producer object, *rp* is defined by five rules. The first two rules say that if either object asks, it will eventually be given to. The third rule says that the producer will never "give" to both objects at the same time. The fourth and fifth rules guarantee that an object will not be "given" unless it has asked previously. We will later prove some properties of this system.

Abstractly, a CMP system can be described as a quintuple:

$\langle obj, env, act, comp, rules \rangle$

where

- *obj* is a set of object ids;
- *env* maps each element of *obj* to a set of predicate symbols which head messages the object will accept;
- *act* maps each element of *obj* to a set of predicate symbols which correspond to actions the object is capable of performing;
- *comp* maps each element of *obj* to a set of predicate symbols which head messages the object may send;
- *rules* maps each element of *obj* to a subset of $Form_{SNF}(FOTL)$ representing the object's program rules.

At each step during its execution, an object will be characterized by a *state*, which will be a set of formulae containing the rules, history, and commitments of the object, and will therefore be a subset of:

$Form_{SNF}(FOTL) \cup Form_{<}(FOTL) \cup Form_{\geq}(FOTL)$.

This concludes the brief review of Concurrent METATEM processes. The next step is to see how CMP is described by the theory of multi-agent systems. As before, each component of the theory is discussed in turn.

Belief and reasoning. The beliefs of an agent correspond well to the state of an object. The language FOTL is thus identified as the internal language, L. Steps (2) and (3) in the execution cycle of a CMP object (above) involve matching the history of the object against the specification rules to determine the new commitments. This can be viewed as a deductive process.

Rules and actions — cognitive and communicative. Although the CMP framework does not explicitly identify the notions of cognitive and communicative act, these ideas are implicit within it. Taking communicative acts first, objects communicate by exchanging messages containing ground atoms of the language FOTL. This maps well into the theory of multi-agent systems[3]. Cognitive acts appear in CMP as system calls, in the style of the built-in predicates of the PROLOG language: the epistemic input that results from the performance of a cognitive act is simply the belief that the action has been performed. This input is then added to the history of the object. For example, suppose an object performed the action "$write(a)$". Then on the next cycle, the object would have $\bullet\, write(a)$ in its history[4].

The notion of conditions for actions is also implicit in CMPs: in order to perform any action, communicative or cognitive, the associated predicate must be currently true. So the condition of any action is simply its associated predicate.

Interpretations. Each object has an interface which "filters out" those messages that the object is "interested" in receiving. This interface consists of a set of predicate symbols: only those messages headed by one of these predicate symbols will be accepted by the object. This process can be modelled in an interpretation function.

Belief revision. At every step in its life, an object must compare its history with its rules, in order to determine new commitments. This means that objects must be (strongly) temporally aware in the following sense: whenever ϕ appears in the history of the object on some cycle u, at cycle $u+v$, $\bullet^v \phi$ must appear in the history. This effectively means rewriting the history on every cycle, a task that is handled by the belief revision function. The actual management of epistemic inputs is trivial: they are simply added to the belief set/state of the object. The belief revision function also handles the updating of commitments, removing those that have become satisfied at the current time.

Now to the axiomatization of CMPs.

Axiomatizing Concurrent METATEM Processes

Whenever a CMP object believes the predicate associated with an action, it performs the action. This gives the following axiom (note that in what follows, P is an arbitrary predicate symbol, and $\overline{a}$ is an arbitrary sequence of constants):

$$(\mathsf{Bel}\ i\ P(\overline{a})) \Rightarrow \bigcirc(\mathsf{Do}\ i\ P(\overline{a})) \quad \text{where } P \in act(i) \tag{6.5}$$

An associated safety property ensures that actions are *only* performed when the condition of the action is true.

$$(\mathsf{Do}\ i\ P(\overline{a})) \Rightarrow \bullet(\mathsf{Bel}\ i\ P(\overline{a})) \quad \text{where } P \in act(i) \tag{6.6}$$

Similarly, whenever a CMP object believes a component predicate, it sends the associated message. This gives the following axiom.

$$(\mathsf{Bel}\ i\ P(\overline{a})) \Rightarrow \bigcirc(\mathsf{Send}\ i\ j\ P(\overline{a})) \quad \text{where } P \in comp(i),\ (i \neq j) \tag{6.7}$$

An associated safety condition ensures that messages are *only* sent when the condition of the message is true.

$$(\mathsf{Send}\ i\ j\ P(\overline{a})) \Rightarrow \bullet(\mathsf{Bel}\ i\ P(\overline{a})) \quad \text{where } P \in comp(i) \tag{6.8}$$

(Recall that in CMP, messages are broadcast to all objects except the sender.) CMP objects believe all the messages they receive, if those messages pass through the "message filter", which gives the following axiom.

$$(\mathsf{Send}\ i\ j\ P(\overline{a})) \Rightarrow (\mathsf{Bel}\ j\ \bullet P(\overline{a})) \quad \text{where } P \in env(j) \tag{6.9}$$

Objects maintain accurate histories, and do not lose beliefs. This gives the following.

$$(\mathsf{Bel}\ i\ \phi) \Rightarrow \bigcirc^u(\mathsf{Bel}\ i\ \bullet^u \phi) \quad \text{where } \phi \text{ is a history formula} \tag{6.10}$$

[3] Note, however, that messages in CMP are broadcast, rather than sent point-to-point.

[4] Similarly, when an object sends a message, it may generate an epistemic input for the *sender* of the message, a concept not directly supported within the theory of multi-agent systems. For example, when an object sends a message such as "$ask(rc1)$", its history on the next cycle will contain $\bullet\, ask(rc1)$. It is not difficult to extend the theory of multi-agent systems to support this, however.

At least conceptually, an object's rules are part of its state.

$$(\text{Bel } i\ P\ \Rightarrow\ F) \quad \text{where } P\ \Rightarrow\ F \in \mathit{rules}(i) \tag{6.11}$$

The general rule for adopting commitments is, "on the basis of the past, do the future". This gives the following axiom, an instance of the attachment axiom, capturing the basic deductive capabilities of CMP objects.

$$\left[\begin{array}{l}(\text{Bel } i\ P\ \Rightarrow\ F)\quad \wedge \\ (\text{Bel } i\ \Gamma)\end{array}\right]\ \Rightarrow\ (\text{Bel } i\ F)\left\{\begin{array}{l}\text{where each } \phi_i \in \Gamma \\ \text{is a history formula,} \\ P\ \Rightarrow\ F \in \mathit{rules}(i), \\ \text{and } \Gamma \vdash P\end{array}\right. \tag{6.12}$$

Note that the above two axioms could be simplified to the following.

$$(\text{Bel } i\ \Gamma)\ \Rightarrow\ (\text{Bel } i\ F)\quad\left\{\begin{array}{l}\text{where each } \phi_i \in \Gamma \text{ is a history formula} \\ \text{and there is some } P\ \Rightarrow\ F \in \mathit{rules}(i) \\ \text{such that } \Gamma \vdash P\end{array}\right. \tag{6.13}$$

Note that the use of these last two axioms may require the introduction of a subsidiary proof, namely: $\Gamma \vdash P$.

Finally, an object will eventually discharge all its commitments. To understand the axiom associated with discharging commitments, it is necessary to understand the notion of an *eventuality*. Any formula of the form $\phi\,\mathcal{U}\,\psi$ or $\Diamond\psi$ contains eventuality $\Diamond\psi$; the formula $\bigcirc\psi$ contains eventuality $\bigcirc\psi$. If ϕ is a present time formula, then ϕ contains eventuality ϕ.

$$(\text{Bel } i\ F)\ \Rightarrow\ \Diamond(\text{Bel } i\ \phi) \quad \text{where } E\phi \text{ is the eventuality of } F \tag{6.14}$$

Note the use of the "$\Diamond$" operator in this example, used instead of the eventuality operator. This means if an object commits to, say, $\bigcirc \mathit{give}1$, it will only be guaranteed to do $\Diamond \mathit{give}1$. The reason for this is rather subtle and complex; the reader is referred to the cited references for details.

A Sample Proof

In this section, we show how the temporal theory of a simple CMP system can be derived, and prove some properties of the system. The system we take is that described in Figure 6.2. Call the system $S1$. Using the axiomatization of CMPs described above, the temporal theory $\mathcal{TH}(S1)$ of the system can be systematically derived. The properties of the system can then be examined through formal proof. The following theorem illustrates this.

Theorem 3 *The system $S1$ has the following property:*

$$\mathcal{TH}(S1) \vdash (\text{Send } \mathit{rc}1\ \mathit{rp}\ \mathit{ask}1)\ \Rightarrow\ \Diamond(\text{Send } \mathit{rp}\ \mathit{rc}1\ \mathit{give}1)$$

Proof *See Figure 6.3.* ∎

This completes the axiomatization of Concurrent METATEM.

6.3 Specifying Three Paradigms for Cooperative Problem Solving

This section presents specifications of three existing cooperative problem solving paradigms. These are: a simple "master/slave" framework, along the lines of that described in [140], the cooperative inference technique employed in the FELINE cooperating expert system [189], and the contract net protocol [159], [161], [160].

6.3.1 Master/Slave Problem Solving Systems

In a 1982 paper, J. Rosenschein outlined a technique for coordinating problem solving activities in simple, master/slave style problem solving systems [140]. The technique outlined in the paper was not (to the best of the author's knowledge) ever implemented, but it nevertheless quite clearly demonstrates many of the principles of such systems. The section that follows does *not* model Rosenschein's framework faithfully; it is simply an attempt to use it as the basis from which to describe such systems.

In the specification, the internal language of agents is assumed to be a first-order meta-language (ML), for reasoning about an ordinary first-order object-language (OL). The usual quoting convention is assumed for this language; if ϕ is an object-language formula, then $\ulcorner\phi\urcorner$ is the ML term denoting ϕ.

1.	(Bel *rp* ⧀ *ask*1) ⇒ (Bel *rp* ◇*give*1)	(6.13)
2.	(Bel *rp* ◇*give*1) ⇒ ◇(Bel *rp give*1)	(6.14)
3.	(Bel *rp* ⧀ *ask*1) ⇒ ◇(Bel *rp give*1)	1, 2, PL
4.	(Send *rc*1 *rp ask*1) ⇒ (Bel *rp* ⧀ *ask*1)	(6.9)
5.	(Send *rc*1 *rp ask*1) ⇒ ◇(Bel *rp give*1)	4, 3, PL
6.	(Bel *rp give*1) ⇒ ○(Send *rp j give*1)	(6.7)
7.	○(Send *rp j give*1) ⇒ ◇(Send *rp j give*1)	TAX
8.	(Bel *rp give*1) ⇒ ◇(Send *rp j give*1)	6, 7, PL
9.	(Bel *rp give*1) ⇒ ◇(Send *rp rc*1 *give*1)	8
10.	□((Bel *rp give*1) ⇒ ◇(Send *rp rc*1 *give*1))	9, TL
11.	□((Bel *rp give*1) ⇒ ◇(Send *rp rc*1 *give*1)) ⇒ (◇(Bel *rp give*1) ⇒ ◇◇(Send *rp rc*1 *give*1))	TAX
12.	◇(Bel *rp give*1) ⇒ ◇◇(Send *rp rc*1 *give*1)	10, 11, PL
13.	◇(Bel *rp give*1) ⇒ ◇(Send *rp rc*1 *give*1)	12, TL, PL
14.	(Send *rc*1 *rp ask*1) ⇒ ◇(Send *rp rc*1 *give*1)	5, 13, PL

Figure 6.3: Proof of Theorem 3

MESSAGE TYPES	
REQUEST(i,j,α)	Agent i requests agent j to perform goal α
INFORM$(i,j,\ulcorner\phi\urcorner)$	Agent i informs agent j of fact ϕ
DOMAIN PREDICATES	
GOAL(i,α)	Agent i has goal α
FACT$(i,\ulcorner\phi\urcorner)$	Agent i believes fact ϕ
HASCAP(i,α)	Agent i has capability of goal α
WILLPERFORM(i,α)	Agent i will perform goal α

Table 6.2: Message Types and Domain Predicates for the Master/Slave System

The ML predicate FACT is used to record facts; "self belief" for agent i is recorded via FACT$(i,\ulcorner\phi\urcorner)$. The ML predicate GOAL is used to record goals. Goals are of two sorts; communicative or cognitive. For example, an agent may have a goal to REQUEST or INFORM something, or it may have a goal to perform action α.

Agents may send two types of message: REQUEST, and INFORM. Rosenschein took his inspiration for messages from the plan-based theory of speech acts described in [31].

In general, no action — cognitive or communicative — will be performed unless the actor WILLPERFORM it. This is the basic pre-condition for action, and leads to the following axioms, which are safety properties.

$$\begin{array}{l}(\text{Send } i\ j\ \texttt{INFORM}(i,j,\ulcorner\phi\urcorner)) \Rightarrow \\ \quad \text{⧀}(\text{Bel } i\ \texttt{FACT}(i,\ulcorner\texttt{WILLPERFORM}(i,\texttt{INFORM}(i,j,\ulcorner\phi\urcorner))\urcorner))\end{array} \tag{6.15}$$

$$\begin{array}{l}(\text{Send } i\ j\ \texttt{REQUEST}(i,j,\alpha)) \Rightarrow \\ \quad \text{⧀}(\text{Bel } i\ \texttt{FACT}(i,\ulcorner\texttt{WILLPERFORM}(i,\texttt{REQUEST}(i,j,\alpha))\urcorner))\end{array} \tag{6.16}$$

$$\begin{array}{l}(\text{Do } i\ \alpha) \Rightarrow \\ \quad \text{⧀}(\text{Bel } i\ \texttt{FACT}(i,\ulcorner\texttt{WILLPERFORM}(i,\alpha)\urcorner))\end{array} \tag{6.17}$$

The INFORM act will not be performed unless its argument is believed beforehand: this is a safety property.

$$(\text{Send } i\ j\ \texttt{INFORM}(i,j,\ulcorner\phi\urcorner)) \Rightarrow \text{⧀}(\text{Bel } i\ \texttt{FACT}(i,\ulcorner\phi\urcorner)) \tag{6.18}$$

Similarly, a REQUEST act will not be performed unless the sender has a goal of its argument: a safety property.

$$(\text{Send } i\ j\ \texttt{REQUEST}(i, j, \alpha)) \Rightarrow \bigcirc(\text{Bel } i\ \texttt{GOAL}(i, \alpha)) \tag{6.19}$$

The above axioms rule out "insincere" REQUESTs or INFORMs. The immediate effects of the communicative acts are given by the following liveness properties.

$$(\text{Send } j\ i\ \texttt{REQUEST}(j, i, \alpha)) \Rightarrow \Diamond(\text{Bel } i\ \texttt{GOAL}(j, \alpha)) \tag{6.20}$$

$$(\text{Send } j\ i\ \texttt{INFORM}(j, i, \ulcorner \phi \urcorner)) \Rightarrow \Diamond(\text{Bel } i\ \texttt{FACT}(j, \ulcorner \phi \urcorner)) \tag{6.21}$$

The perlocutionary force of the REQUEST act is given by the following axiom, called CAUSE-TO-WANT: a liveness property.

$$\left[\begin{array}{ll} (\text{Bel } i\ \texttt{GOAL}(j, \alpha)) & \wedge \\ (\text{Bel } i\ \texttt{ACCEPT}(j, i, \alpha)) & \end{array}\right] \Rightarrow \Diamond(\text{Bel } i\ \texttt{GOAL}(i, \alpha)) \tag{6.22}$$

The ability of j to "persuade" i to adopt α as a goal thus depends on the ACCEPT predicate. The use of this predicate allows the performance of actions to be dependent on social relationships, and gives agents some measure of autonomy over what they adopt as a goal. For example, suppose that j was the "master" of i, i.e., it had some "authority" over i. This might be expressed by the following axioms.

$$(\text{Bel } i\ \texttt{FACT}(i, \ulcorner \texttt{MASTER}(j, i) \urcorner))$$
$$(\text{Bel } i\ \texttt{FACT}(i, \ulcorner \forall x \cdot \texttt{MASTER}(x, i) \Rightarrow \forall y \cdot \texttt{ACCEPT}(x, i, y) \urcorner))$$

Then, if i is endowed with suitable reasoning abilities, it will adopt as a goal anything that j requests it to. Obviously, more sophisticated social relationships may be built up in a similar fashion.

The perlocutionary force of the INFORM act is given by the following axiom, called CONVINCE: a liveness property.

$$\left[\begin{array}{ll} (\text{Bel } i\ \texttt{FACT}(j, \ulcorner \phi \urcorner)) & \wedge \\ (\text{Bel } i\ \textit{BESWAYED}(j, i, \ulcorner \phi \urcorner)) & \end{array}\right] \Rightarrow \Diamond \left[\begin{array}{ll} (\text{Bel } i\ \texttt{FACT}(i, \ulcorner \phi \urcorner)) & \wedge \\ \neg(\text{Bel } i\ \texttt{FACT}(i, \ulcorner \neg \phi \urcorner)) & \end{array}\right] \tag{6.23}$$

The BESWAYED predicate in this axiom has an analogous role to the ACCEPT predicate in the CAUSE-TO-WANT axiom, above, and thus provides agents with a measure of autonomy over what they believe. Note that some measure of consistency, (or at least non-contradiction), is assured by this definition.

Commitment to action — given by the WILLPERFORM predicate — depends on having the appropriate capability to achieve a goal. Agents commit to all goals they are capable of: a liveness property.

$$\left[\begin{array}{ll} (\text{Bel } i\ \texttt{GOAL}(i, \alpha)) & \wedge \\ (\text{Bel } i\ \texttt{HASCAP}(i, \alpha)) & \end{array}\right] \Rightarrow \Diamond(\text{Bel } i\ \texttt{FACT}(i, \ulcorner \texttt{WILLPERFORM}(i, \alpha) \urcorner)) \tag{6.24}$$

Another property to specify is that goals are eventually achieved: liveness properties.

$$(\text{Bel } i\ \texttt{FACT}(i, \ulcorner \texttt{WILLPERFORM}(i, \alpha) \urcorner)) \Rightarrow \Diamond(\text{Do } i\ \alpha) \quad \alpha \text{ is a cognitive action} \tag{6.25}$$

$$(\text{Bel } i\ \texttt{FACT}(i, \ulcorner \texttt{WILLPERFORM}(i, \texttt{INFORM}(i, j, \ulcorner \phi \urcorner)) \urcorner)) \Rightarrow \Diamond(\text{Send } i\ j\ \texttt{INFORM}(i, j, \ulcorner \phi \urcorner)) \tag{6.26}$$

$$(\text{Bel } i\ \texttt{FACT}(i, \ulcorner \texttt{WILLPERFORM}(i, \texttt{REQUEST}(i, j, \alpha)) \urcorner)) \Rightarrow \Diamond(\text{Send } i\ j\ \texttt{REQUEST}(i, j, \alpha)) \tag{6.27}$$

This completes the specification of the master/slave system.

6.3.2 A Cooperative Inference Technique

A cooperating expert system is an expert system constructed as a group of agents, each of which contains a "classical" expert system with its own domain knowledge, inference engine, etc. Through a process of cooperative interaction, the agents pool their resources and solve problems requiring their collective domain expertise. A basic issue in the development of such a system is to devise what might be called a *cooperative inference* technique: a method for extending classical, intra-agent reasoning/inference techniques to multiple agent scenarios. This section gives a specification of one such technique, that employed in the FELINE system [189].

The following description of the FELINE cooperative inference technique is taken directly from [189]:

"Each agent in FELINE maintains a data structure representing its beliefs about itself and its environment. This data structure is called the *environment model*. It contains an entry for the modelling agent and each agent that the modelling agent might communicate with (its *acquaintances*). Each entry contains two important attributes:

- **Skills.** This attribute is a set of identifiers denoting hypotheses which the agent has the expertise to establish or deny. The skills of an agent will correspond roughly to root nodes of the inference networks representing the agent's domain expertise.
- **Interests.** This attribute is a set of identifiers denoting hypotheses for which the agent requires the truth value. It may be that an agent actually has the expertise to establish the truth value of its interests, but is never the less 'interested' in them. The interests of an agent will correspond roughly to leaf nodes of the inference networks representing the agent's domain expertise.

An agent's environment model serves to delimit the cooperative problem solving process. Before describing inter-agent problem solving in detail, we describe the types of message that agents may send.

First, we define a message as a triple of *sender*, *receiver*, and *contents*. In FELINE, the contents field is also a triple, containing *message type*, *attribute*, and *value*. Agents communicate using three message types:

- **Request**. If an agent sends a request, then the attribute field will contain an identifier denoting a hypothesis. It is assumed that the hypothesis is one which lies within the domain of the intended recipient. A request is assumed to mean that the sender wants the receiver to derive a truth value for the hypothesis.
- **Response**. If an agent receives a request and manages to successfully derive a truth value for the hypothesis, then it will send a response to the originator of the request. The attribute field will contain the identifier denoting the hypothesis; the value field will contain the associated truth value.
- **Inform**. The attribute field of an inform message will contain an identifier denoting a hypothesis. The value field will contain an associated truth value. An inform message will be unsolicited; an agent sends one if it thinks the recipient will be 'interested' in the hypothesis.

We now look in detail at the cooperative problem solving technique used. First, consider goal-driven problem solving in a normal rule-based system. Typically, goal driven reasoning proceeds by attempting to establish the truth value of some hypothesis. If the truth value is not known, then a recursive descent of the inference network associated with the hypothesis is performed. Leaf nodes in the inference network typically correspond to questions which are asked of the user. Within FELINE, this well known scheme is augmented by the following principle. When evaluating a leaf-node, if it is not a question, then check the environment model to see if any other agent has the node as a 'skill'. If there is some agent which lists the node as a skill, then send a request to that agent requesting the hypothesis. Wait until a response is received; the response will indicate the truth value of the node. This technique is an instance of the general problem solving technique called *task sharing* [161], since evaluation tasks are explicitly delegated on the basis of the skills entry in the environment model.

We now turn to data-driven problem solving. Typically, data-driven problem solving proceeds by taking a database of facts (hypotheses and associated truth values), and a set of rules, and repeatedly generating a conflict set of new facts. These new facts are then added to the database, and the process begins again. If a hypothesis follows from a set of facts and a set of rules, then this style of problem solving will eventually generate a result. In FELINE, the basic scheme is augmented as follows. Whenever a new fact is generated, the environment model is consulted to see if any agent has the hypothesis as an 'interest'. If it does, then an 'inform' message is sent to the appropriate agent, containing the hypothesis and truth value. Upon receipt of an 'inform' message, an agent adds the fact to its database and enters the forward chaining cycle. This technique is an instance of *result sharing*, [161] since agents share results that they believe may be of interest to other agents". [189]

The specification is straightforward. It is assumed that an agent's "environment model" is a set of formulae in some restricted first-order language. Table 6.3 summarizes the message types and domain predicates used in the specification.

MESSAGE TYPES	
REQUEST(h)	A request for the value of h
RESPONSE(h, v)	A response to request: value of h is v
INFORM(h, v)	Unsolicited data: h has value v

DOMAIN PREDICATES	
MYSKILL(h)	Modelling agent has domain expertise h
SKILL(i, h)	Agent i has domain expertise h
INTEREST(i, h)	Agent i has interest h
GOAL(h)	Modelling agent requires value for h
VALUE(h, v)	The value of h is v

Table 6.3: Message Types and Domain Predicates for the Cooperative Inference Technique

$$
\begin{aligned}
\mathsf{Request}(i,j,h) &\triangleq (\mathsf{Send}\ i\ j\ \mathtt{REQUEST}(h)) \\
\mathsf{Response}(i,j,h,v) &\triangleq (\mathsf{Send}\ i\ j\ \mathtt{RESPONSE}(h,v)) \\
\mathsf{Inform}(i,j,h,v) &\triangleq (\mathsf{Send}\ i\ j\ \mathtt{INFORM}(h,v)) \\
\mathsf{Can}(i,h) &\triangleq (\mathsf{Bel}\ i\ \mathtt{MYSKILL}(h)) \\
\mathsf{Skill}(i,j,h) &\triangleq (\mathsf{Bel}\ i\ \mathtt{SKILL}(j,h)) \\
\mathsf{Interest}(i,j,h) &\triangleq (\mathsf{Bel}\ i\ \mathtt{INTEREST}(j,h)) \\
\mathsf{Goal}(i,h) &\triangleq (\mathsf{Bel}\ i\ \mathtt{GOAL}(h)) \\
\mathsf{Value}(i,h,v) &\triangleq (\mathsf{Bel}\ i\ \mathtt{VALUE}(h,v))
\end{aligned}
$$

Table 6.4: Abbreviations for the Cooperative Inference Technique

Thus an agent with belief MYSKILL(h) would have the domain expertise to evaluate hypothesis h; with belief SKILL(i, h) would believe that i had expertise to evaluate h; with belief INTEREST(i, h) would believe that i was "interested" in the value of h, and with belief GOAL(h) would be currently trying to find a value for h. The message types correspond exactly to the informal description given above. In the interests of readability, some abbreviations are defined in Table 6.4.

Agents may perform two types of (cognitive) action. First, they may forward-chain, in an attempt to derive new information. Second, they may *evaluate* a specific hypothesis in order to determine its value. The following abbreviations are assumed[5].

$$
\begin{aligned}
\mathsf{FwdChain}(i) &\triangleq (\mathsf{Do}\ i\ \mathtt{FWDCHAIN}) \\
\mathsf{Eval}(i,h) &\triangleq (\mathsf{Do}\ i\ \mathtt{EVAL}(h))
\end{aligned}
$$

The first property for specification is that evaluation actions will ultimately result in a value being found for the hypothesis being evaluated: a liveness property.

$$\forall h \cdot \mathsf{Eval}(i,h) \Rightarrow \Diamond \exists v \cdot \mathsf{Value}(i,h,v) \tag{6.28}$$

Agents can only evaluate things they have the skill for: this is a safety property.

$$\forall h \cdot \mathsf{Eval}(i,h) \Rightarrow \bullet\, \mathsf{Can}(i,h) \tag{6.29}$$

Next, the various properties of Request acts are given. First, agents will always work toward a hypothesis that they have appropriate domain expertise for.

$$\forall h \cdot \left[\begin{array}{ll} \mathsf{Request}(j,i,h) & \wedge \\ \mathsf{Can}(i,h) & \end{array} \right] \Rightarrow \Diamond \mathsf{Goal}(i,h) \tag{6.30}$$

If possible, goals will eventually be acted on: a liveness property.

$$\forall h \cdot \left[\begin{array}{ll} \mathsf{Goal}(i,h) & \wedge \\ \mathsf{Can}(i,h) & \end{array} \right] \Rightarrow \Diamond \mathsf{Eval}(i,h) \tag{6.31}$$

[5]Notation is being abused here in the interests of readability: the second argument to a Do operator must be a constant or variable term denoting an action — not a function. The meaning is quite clear, however, and all specifications written in this way can be expanded out into the strictly correct form if necessary.

Goals that cannot be evaluated locally, (because of lack of expertise), are "farmed out"; a liveness property. Requests are also honest; a safety property. Also, we can specify the assumption that an agent with a goal that it does not have the expertise to evaluate knows of at least one other agent that *does* have the expertise.

$$\forall h \cdot \left[\begin{array}{ll} \mathsf{Goal}(i,h) & \wedge \\ \neg\,\mathsf{Can}(i,h) & \wedge \\ \mathsf{Skill}(i,j,h) & \end{array}\right] \Rightarrow \Diamond \mathsf{Request}(i,j,h) \tag{6.32}$$

$$\forall h \cdot \mathsf{Request}(i,j,h) \Rightarrow \bullet \left[\begin{array}{ll} \mathsf{Goal}(i,h) & \wedge \\ \neg\,\mathsf{Can}(i,h) & \wedge \\ \mathsf{Skill}(i,j,h) & \end{array}\right] \tag{6.33}$$

$$\forall h \cdot \left[\begin{array}{ll} \mathsf{Goal}(i,h) & \wedge \\ \neg\,\mathsf{Can}(i,h) & \end{array}\right] \Rightarrow \exists j \cdot \mathsf{Skill}(i,j,h) \tag{6.34}$$

Now to the properties of Response actions. First, responses are guaranteed: a classic liveness property.

$$\forall h \cdot \mathsf{Request}(i,j,h) \Rightarrow \Diamond \exists v \cdot \mathsf{Response}(j,i,h,v) \tag{6.35}$$

Responses are solicited: a safety property.

$$\forall h \cdot \forall v \cdot \mathsf{Response}(j,i,h,v) \Rightarrow \blacklozenge\, \mathsf{Request}(i,j,h) \tag{6.36}$$

As the result of receiving a response, an agent will come to believe the data it has been given: a liveness property.

$$\forall h \cdot \forall v \cdot \mathsf{Response}(j,i,h,v) \Rightarrow \Diamond \mathsf{Value}(i,h,v) \tag{6.37}$$

Agents do not "lie" about responses, and must, moreover, have a value before they respond. These are safety properties.

$$\forall h \cdot \forall v \cdot \mathsf{Response}(i,j,h,v) \Rightarrow \bullet\, \mathsf{Value}(i,h,v) \tag{6.38}$$

Now to the properties of Inform acts. Informs will always be sent to agents with the appropriate interests: a liveness property.

$$\forall h \cdot \forall v \cdot \left[\begin{array}{ll} \mathsf{Interest}(i,j,h) & \wedge \\ \mathsf{Value}(i,h,v) & \end{array}\right] \Rightarrow \Diamond \mathsf{Inform}(i,j,h,v) \tag{6.39}$$

The associated safety property ensures that Informs will not be sent to agents that are not interested, and that agents are honest about Informs.

$$\forall h \cdot \forall v \cdot \mathsf{Inform}(i,j,h,v) \Rightarrow \bullet \left[\begin{array}{ll} \mathsf{Interest}(i,j,h) & \wedge \\ \mathsf{Value}(i,h,v) & \end{array}\right] \tag{6.40}$$

The effect of an Inform is captured by the following liveness property.

$$\forall h \cdot \forall v \cdot \mathsf{Inform}(j,i,h,v) \Rightarrow \Diamond(\mathsf{Value}(i,h,v) \wedge \bigcirc \Diamond \mathsf{FwdChain}(i)) \tag{6.41}$$

Finally, some miscellaneous properties. Agents drop goals when a value is found, and never re-adopt them.

$$\forall h \cdot \forall v \cdot \mathsf{Value}(i,h,v) \Rightarrow \Diamond \Box \neg\, \mathsf{Goal}(i,h) \tag{6.42}$$

It is also possible to specify that agent's beliefs about other's skills are correct.

$$\forall h \cdot \mathsf{Skill}(j,i,h) \Rightarrow \mathsf{Can}(i,h) \tag{6.43}$$

This completes the specification of the cooperative inference technique.

6.3.3 The Contract Net Protocol

The Contract Net (CNET) protocol is a high-level protocol for achieving efficient cooperative action in networks of communicating problem solvers. It was developed by R. Smith while at Stanford University in the USA, and was initially described as early as 1977 [159]. It formed the basis of Smith's doctoral thesis, (published as [161]), and was further described in [160], [162]. The basic protocol has subsequently been adapted and extended by several other researchers [173], [170]. The aim of this section is to develop a formal specification of a subset of the CNET protocol — the description of the protocol given below is therefore a distillation of the cited references, and is not intended to be definitive.

The CNET protocol makes an interesting case study for several reasons. The first is historical. The protocol was the first major attempt to address the issue of cooperative problem solving by *autonomous* problem solvers, where each participating agent is able to explicitly make choices about what to do. (Earlier work focussed mainly on blackboard architectures, where autonomy is not an issue.) Secondly, the CNET protocol remains one of the more completely specified cooperative problem solving paradigms, which makes it well suited to formalization. Finally, a simple formalization of the CNET protocol has already been achieved, providing a useful point of comparison [179, pp27–30][6].

The key idea which motivated the CNET work was that *negotiation* is a central component of cooperative problem solving[7]: in essence, the CNET protocol is a framework for managing negotiation. Each agent, (or *node*, in Smith's parlance), in a CNET system is an autonomous problem solver, capable of communicating with its peers via message passing. During execution, an agent may generate a task to be expedited. This may be done locally, if the agent has the ability, or the agent may try to get some other agent to expedite it:

> "In brief, a node that generates a task advertises existence of that task to other nodes in the net with a *task announcement*, then acts as the *manager* of that task for its duration. In the absence of any information about the specific capabilities of the other nodes in the net, the manager is forced to issue a *general broadcast* to all other nodes. If, however, the manager possesses some knowledge about which of the other nodes in the net are likely candidates, then it can issue a *limited broadcast* to just those candidates. Finally, if the manager knows exactly which of the other nodes in the net is appropriate, then it can issue a *point-to-point* announcement. As work on the problem progresses, many such task announcements will be made by various managers.
>
> Nodes in the net listen to the task announcements and evaluate them with respect to their own specialized hardware and software resources. When a task to which a node is suited is found, it submits a *bid*. A bid indicates the capabilities of the bidder that are relevant to the execution of the announced task. A manager may receive several such bids in response to a single task announcement; based on the information in the bids, it selects the most appropriate nodes to execute the task. The selection is communicated to the successful bidders through an *award* message. These selected nodes assume responsibility for execution of the task, and each is called a *contractor* for that task.
>
> ...
>
> After the task has been completed, the contractor sends a *report* to the manager. [161, pp60–61]
>
> [This] normal contract negotiation process can be simplified in some instances, with a resulting enhancement in the efficiency of the protocol. If a manager knows exactly which node is appropriate for the execution of a task, a *directed contract* can be awarded. This differs from the *announced contract* in that no announcement is made and no bids are submitted. Instead, an award is made directly. In such cases, nodes awarded contracts must acknowledge receipt, and have the option of refusal.
>
> ...
>
> Finally, for tasks that amount to simple requests for information, a contract may not be appropriate. In such cases, a request-response sequence can be used without further embellishment. Such messages (that aid in the distribution of data as opposed to control) are implemented as *request* and *information* messages. The request message is used to encode straightforward requests for information when contracting is unnecessary. The information message is used both as a response to a request message and a general data transfer message". [161, pp62–63]

[6] In fact, Werner only formalized a fragment of the CNET protocol. The formalization presented here is also incomplete: those elements of the original description not described here may be assumed to have been dropped in the interests of simplicity.

[7] In retrospect, this idea is probably the key contribution of the CNET work. Several researchers — (notably Zlotkin and Rosenschein [190]) — have subsequently investigated the properties of negotiation strategies using game-theoretic techniques.

MESSAGE TYPES	
TASKANN(t, e)	Sender announces task with id t and e-spec e
BID(t)	Sender bids for task with id t
AWARD(i, t)	Sender awards task with id t to agent i
REQUEST(t)	Sender requests information associated with t
INFORM(t, v)	Sender informs recipient that value of t is v

DOMAIN PREDICATES	
ELIGIBLE(e)	Agent is eligible for tasks with e-spec e
NEWTASK(t, e)	Task t is a newly generated sub-task with e-spec e
ANNOUNCED(t, i)	Task with id t has been announced by agent i
AWARDED(t, i)	Agent has been awarded task t by agent i
TASK(t)	Agent is committed to (eventually) doing task t
GOAL(t)	The task currently being expedited is t
RESULT(t, v)	The result of expediting t is v
BIDDED(i, t)	Agent i has bid for t

Table 6.5: Message Types and Domain Predicates for the CNET

In addition to describing the various messages that agents may send, Smith describes the procedures to be carried out on receipt of a message. Briefly, these procedures are as follows (see [161, pp96–102] for more details):

1. **Task announcement processing.** On receipt of a task announcement, an agent decides if it is *eligible* for the task. It does this by looking at the *eligibility specification* contained in the announcement. If it is eligible, then details of the task are stored, and the agent will subsequently bid for the task.

2. **Bid processing.** Details of bids from would-be contractors are stored by (would-be) managers until some deadline is reached. The manager then awards the task to a single bidder.

3. **Award processing.** Agents that bid for a task, but fail to be awarded it, simply delete details of the task. The successful bidder must attempt to expedite the task (which may mean generating new sub-tasks).

4. **Request and inform processing.** These messages are the simplest to handle. A request simply causes an inform message to be sent to the requestor, containing the required information, but only if that information is immediately available. (Otherwise, the requestee informs the requestor that the information is unknown.) An inform message causes its content to be added to the recipient's database. It is assumed that at the conclusion of a task, a contractor will send an information message to the manager, detailing the results of the expedited task[8].

So to the protocol specification. Once again, it is assumed that agents use an internal language which is some subset of the first-order language L_0, though it is not necessary for them to have any reasoning capabilities. The domain predicates and message types used in the specification are given in Table 6.5.

Messages and domain predicates contain terms of four sorts: t is a task id(entifier), (a standard name for a task); e is an eligibility specification, (or e-spec), which allows agents to determine whether they are eligible for a task; i, as usual, is an agent id, and v is a value, the result of expediting a task (the domain of values is assumed to contain one distinguished element, unknown). The meaning of the message types is fairly obvious; domain predicates require some explanation.

Agents maintain beliefs about what tasks they are eligible for. This is achieved via ELIGIBLE predicates. A record of all task announcements the agent has received is kept via ANNOUNCED predicates. If an agent has been awarded a task, it records the fact via AWARDED predicates — the second argument denotes the manager of the task. Commitment to (eventually) performing a task is recorded via TASK predicates. Details of tasks currently being expedited are recorded via GOAL predicates; the result of expediting a task is recorded via RESULT predicates, which relate a task id to a value. A manager records details of all the bids it receives via BIDDED predicates, which relate a task id to a bidder. Finally, details of new tasks are recorded via NEWTASK predicates. Note that

[8]This is done via a special *report* message type in the original CNET framework.

$$
\begin{array}{rcl}
\mathsf{TaskAnn}(i,j,t,e) & \triangleq & (\mathsf{Send}\ i\ j\ \mathtt{TASKANN}(t,e)) \\
\mathsf{Bid}(i,j,t) & \triangleq & (\mathsf{Send}\ i\ j\ \mathtt{BID}(t)) \\
\mathsf{Award}(i,j,k,t) & \triangleq & (\mathsf{Send}\ i\ j\ \mathtt{AWARD}(k,t)) \\
\mathsf{Request}(i,j,t) & \triangleq & (\mathsf{Send}\ i\ j\ \mathtt{REQUEST}(t)) \\
\mathsf{Inform}(i,j,t,v) & \triangleq & (\mathsf{Send}\ i\ j\ \mathtt{INFORM}(t,v)) \\
\mathsf{Eligible}(i,e) & \triangleq & (\mathsf{Bel}\ i\ \mathtt{ELIGIBLE}(e)) \\
\mathsf{NewTask}(i,t,e) & \triangleq & (\mathsf{Bel}\ i\ \mathtt{NEWTASK}(t,e)) \\
\mathsf{Announced}(i,t,j) & \triangleq & (\mathsf{Bel}\ i\ \mathtt{ANNOUNCED}(t,j)) \\
\mathsf{Awarded}(i,t,j) & \triangleq & (\mathsf{Bel}\ i\ \mathtt{AWARDED}(t,j)) \\
\mathsf{Task}(i,t) & \triangleq & (\mathsf{Bel}\ i\ \mathtt{TASK}(t)) \\
\mathsf{Goal}(i,t) & \triangleq & (\mathsf{Bel}\ i\ \mathtt{GOAL}(t)) \\
\mathsf{Result}(i,t,v) & \triangleq & (\mathsf{Bel}\ i\ \mathtt{RESULT}(t,v)) \\
\mathsf{Bidded}(i,j,t) & \triangleq & (\mathsf{Bel}\ i\ \mathtt{BIDDED}(j,t))
\end{array}
$$

Table 6.6: Abbreviations for the CNET

NEWTASK means an agent has a task which it wants expediting; TASK means it has committed to the task; GOAL means it is currently working on the task.

Some definitions are given in order to improve readability; see Table 6.6.

Four further definitions are illuminating. The first two capture the dynamic nature of the roles "contractor" and "manager".

$$\mathsf{Contractor}(i,j,t) \triangleq \left[\begin{array}{c} (\neg\, \exists v \cdot \mathsf{Inform}(i,j,t,v)) \\ \mathcal{S} \\ \mathsf{Award}(j,i,i,t) \end{array}\right] \tag{6.44}$$

$$\mathsf{Manager}(i,j,t) \triangleq \mathsf{Contractor}(j,i,t) \tag{6.45}$$

An agent is thus a contractor between the time it receives an award and sends the final result. Similarly, an agent is a manager between the time it sends the award and receives the final result — note the symmetry in these roles. These definitions can be used to define what it means for a contract to exist between two agents.

$$\mathsf{Contract}(i,j) \triangleq \exists t \cdot \left[\begin{array}{ll} \mathsf{Contractor}(i,j,t) & \vee \\ \mathsf{Manager}(i,j,t) & \end{array}\right] \tag{6.46}$$

The fourth definition gives what it means for a contract to be successfully expedited.

$$\mathsf{Expedited}(i,t,j) \triangleq \left[\begin{array}{ll} (\ \bullet\!\!\!\circ\ \mathsf{Contractor}(i,j,t)) & \wedge \\ \exists v \cdot \mathsf{Inform}(i,j,t,v) & \end{array}\right] \tag{6.47}$$

Now for the actual properties to be specified. First, task announcements. These are broadcast, giving the following.

$$\forall t \cdot \forall e \cdot \mathsf{TaskAnn}(i,j,t,e) \ \Rightarrow\ \mathsf{TaskAnn}(i,k,t,e) \quad (i \neq k) \tag{6.48}$$

The immediate effect of a task announcement is given by the following liveness property.

$$\forall t \cdot \forall e \cdot \left[\begin{array}{ll} \mathsf{TaskAnn}(j,i,t,e) & \wedge \\ \mathsf{Eligible}(i,e) & \end{array}\right] \ \Rightarrow\ \Diamond \mathsf{Announced}(i,t,j) \tag{6.49}$$

Agents have correct beliefs about announcements: a safety property.

$$\forall t \cdot \mathsf{Announced}(i,t,j) \ \Rightarrow\ \blacklozenge\ \exists e \cdot \left[\begin{array}{ll} \mathsf{TaskAnn}(j,i,t,e) & \wedge \\ \mathsf{Eligible}(i,e) & \end{array}\right] \tag{6.50}$$

Agents only announce tasks which they want expediting, and cannot expedite themselves: a safety property.

$$\forall t \cdot \forall e \cdot \mathsf{TaskAnn}(i,j,t,e) \ \Rightarrow\ \bullet\!\!\!\circ \left[\begin{array}{ll} \mathsf{NewTask}(i,t,e) & \wedge \\ \neg\, \mathsf{Eligible}(i,e) & \end{array}\right] \tag{6.51}$$

Now to the creation of task announcements. An agent with a locally generated task that cannot be expedited locally will announce the task: a liveness property.

$$\forall t \cdot \forall e \cdot \left[\begin{array}{ll} \mathsf{NewTask}(i,t,e) & \wedge \\ \neg\, \mathsf{Eligible}(i,e) & \end{array} \right] \Rightarrow \Diamond \mathsf{TaskAnn}(i,j,t,e) \tag{6.52}$$

However, tasks which *can* be expedited locally *are*: a liveness property.

$$\forall t \cdot \forall e \cdot \left[\begin{array}{ll} \mathsf{NewTask}(i,t,e) & \wedge \\ \mathsf{Eligible}(i,e) & \end{array} \right] \Rightarrow \Diamond \mathsf{Task}(i,t) \tag{6.53}$$

Agents always bid for tasks they did not originate, but are eligible for: a liveness property.

$$\forall t \cdot \mathsf{Announced}(i,t,j) \Rightarrow \Diamond \mathsf{Bid}(i,j,t) \tag{6.54}$$

Bids are always solicited: a safety property.

$$\forall t \cdot \mathsf{Bid}(i,j,t) \Rightarrow \blacklozenge\, \mathsf{Announced}(i,t,j) \tag{6.55}$$

The effect of receiving a bid is given by the following liveness property.

$$\forall t \cdot \mathsf{Bid}(i,j,t) \Rightarrow \Diamond \mathsf{Bidded}(j,i,t) \tag{6.56}$$

A manager has correct beliefs about bids: a safety property.

$$\forall t \cdot \mathsf{Bidded}(i,j,t) \Rightarrow \blacklozenge\, \mathsf{Bid}(j,i,t) \tag{6.57}$$

Eventually, a would-be manager assesses bids, and sends an award: a liveness property.

$$\forall t \cdot \forall e \cdot \mathsf{TaskAnn}(i,j,t,e) \Rightarrow \Diamond \exists k \cdot \mathsf{Award}(i,j,k,t) \tag{6.58}$$

Managers only award to agents that bidded: a safety property.

$$\forall t \cdot \mathsf{Award}(i,j,k,t) \Rightarrow \bullet\, \mathsf{Bidded}(i,k,t) \tag{6.59}$$

If desired, it can be specified that managers only award to one agent: a safety property.

$$\forall t \cdot \left[\begin{array}{ll} \mathsf{Award}(i,j,k,t) & \wedge \\ \mathsf{Award}(i,j,l,t) & \end{array} \right] \Rightarrow (k = l) \tag{6.60}$$

As with task announcements, awards are broadcast: this gives the following.

$$\forall t \cdot \mathsf{Award}(i,j,k,t) \Rightarrow \mathsf{Award}(i,l,k,t) \quad (i \neq l) \tag{6.61}$$

The successful bidder (now a contractor) will believe it has been awarded the task: a liveness property.

$$\forall t \cdot \mathsf{Award}(i,j,k,t) \Rightarrow \Diamond \mathsf{Awarded}(k,t,i) \tag{6.62}$$

Agents have correct beliefs about what they have been awarded: a safety property.

$$\forall t \cdot \mathsf{Awarded}(i,t,j) \Rightarrow \blacklozenge\, \mathsf{Award}(j,i,i,t) \tag{6.63}$$

Contractors eventually become committed to their tasks: a liveness property.

$$\forall t \cdot \mathsf{Awarded}(i,t,j) \Rightarrow \Diamond \mathsf{Task}(i,t) \tag{6.64}$$

Unexpedited tasks are eventually adopted as goals: a liveness property.

$$\forall t \cdot \mathsf{Task}(i,t) \Rightarrow \Diamond \mathsf{Goal}(i,t) \tag{6.65}$$

Goals must have been tasks: a safety property.

$$\forall t \cdot \mathsf{Goal}(i,t) \Rightarrow \blacklozenge\, \mathsf{Task}(i,t) \tag{6.66}$$

The next axiom states that goals are successfully expedited: a liveness property.

$$\forall t \cdot \mathsf{Goal}(i,t) \Rightarrow \Diamond \exists v \cdot \mathsf{Result}(i,t,v) \tag{6.67}$$

An agent contracted with a task, who has a result for that task, will eventually send the result: a liveness property.

$$\forall t \cdot \forall v \cdot \left[\begin{array}{ll} \mathsf{Result}(i,t,v) & \wedge \\ \mathsf{Contractor}(i,j,t) & \end{array} \right] \Rightarrow \Diamond \mathsf{Inform}(i,j,t,v) \tag{6.68}$$

A simple "first come, first served" ordering is imposed on tasks.

$$\forall t_1 \cdot \forall t_2 \cdot \left[\begin{array}{c} \mathsf{Contractor}(i,j,t_1) \\ \mathcal{B} \\ \mathsf{Contractor}(i,k,t_2) \end{array} \right] \Rightarrow \left[\begin{array}{c} \mathsf{Expedited}(i,t_1,j) \\ \mathcal{B} \\ \mathsf{Expedited}(i,t_2,k) \end{array} \right] \tag{6.69}$$

Inform and request acts have comparatively simple properties. The recipient of an Inform message will add the appropriate information to its local database: a liveness property.

$$\forall t \cdot \forall v \cdot \mathsf{Inform}(i,j,t,v) \Rightarrow \Diamond \mathsf{Result}(j,t,v) \tag{6.70}$$

Informs are honest: a safety property.

$$\forall t \cdot \forall v \cdot \mathsf{Inform}(i,j,t,v) \Rightarrow \bullet \mathsf{Result}(i,t,v) \tag{6.71}$$

A request will not be sent if the required value is known by the sender: a safety property.

$$\forall t \cdot \mathsf{Request}(i,j,t) \Rightarrow \bullet \neg \exists v \cdot \mathsf{Result}(i,t,v) \tag{6.72}$$

If the recipient of a request has the required information, then it will inform the sender of this: a liveness property.

$$\forall t \cdot \mathsf{Request}(j,i,t) \wedge \exists v \cdot \mathsf{Result}(i,t,v) \Rightarrow \Diamond \mathsf{Inform}(i,j,t,v) \tag{6.73}$$

If the required value is not known by the recipient of the Request, then the recipient tells the sender that the value is unknown.

$$\forall t \cdot \left[\begin{array}{ll} \mathsf{Request}(j,i,t) & \wedge \\ \neg \exists v \cdot \mathsf{Result}(i,t,v) & \end{array} \right] \Rightarrow \Diamond \mathsf{Inform}(i,j,t,\mathit{unknown}) \tag{6.74}$$

Other properties may be specified: for example, the property that there will always be at least one eligible agent for every task is given as follows.

$$\forall e \cdot \exists i \cdot \mathsf{Eligible}(i,e) \tag{6.75}$$

This completes the specification of the CNET protocol. Note that although the specification is quite long and fairly tedious, a great deal has been left unsaid. For example, the specification has very little to say on the subject of how *long* beliefs should be held for. The reason for these omissions is lack of space. The reader should be aware that the specification could be expanded with considerably more detail.

A Sample Proof

One of the advantages of presenting a specification in this manner is that its properties may be examined through formal proof. In this section, we present a sample proof. Call the specification given above *SP*1. The following theorem states the property to be proven.

Theorem 4 *The specification SP*1 *has the following property:*

$$SP1 \vdash \left[\begin{array}{ll} \mathsf{NewTask}(a1,t1,e1) & \wedge \\ \neg\, \mathsf{Eligible}(a1,e1) & \wedge \\ \mathsf{Eligible}(a2,e1) & \end{array} \right] \Rightarrow \Diamond \mathsf{Bid}(a2,a1,t1)$$

Proof *See Figure 6.4.* ∎

1.	$\forall t \cdot \forall e \cdot \mathsf{NewTask}(i,t,e) \wedge \neg \mathsf{Eligible}(i,e)$ $\Rightarrow \Diamond \mathsf{TaskAnn}(i,j,t,e)$	(6.52)
2.	$\mathsf{NewTask}(a1,t1,e1) \wedge \neg \mathsf{Eligible}(a1,e1)$ $\Rightarrow \Diamond \mathsf{TaskAnn}(a1,a2,t1,e1)$	1, PRED, PL
3.	$\forall t \cdot \forall e \cdot \mathsf{TaskAnn}(j,i,t,e) \wedge \mathsf{Eligible}(i,e)$ $\Rightarrow \Diamond \mathsf{Announced}(i,t,j)$	(6.49)
4.	$\mathsf{TaskAnn}(a1,a2,t1,e1) \wedge \mathsf{Eligible}(a2,e1)$ $\Rightarrow \Diamond \mathsf{Announced}(a2,t1,a1)$	3, PRED, PL
5.	$\mathsf{NewTask}(a1,t1,e1) \wedge \neg \mathsf{Eligible}(a1,e1) \wedge \mathsf{Eligible}(a2,e1)$ $\Rightarrow \Diamond \mathsf{Announced}(a2,t1,a1)$	2, 4, TL, PL
6.	$\forall t \cdot \mathsf{Announced}(i,t,j) \Rightarrow \Diamond \mathsf{Bid}(i,j,t)$	(6.54)
7.	$\mathsf{Announced}(a2,t1,a1) \Rightarrow \Diamond \mathsf{Bid}(a2,a1,t1)$	6, PRED
8.	$\mathsf{NewTask}(a1,t1,e1) \wedge \neg \mathsf{Eligible}(a1,e1) \wedge \mathsf{Eligible}(a2,e1)$ $\Rightarrow \Diamond \mathsf{Bid}(a2,a1,t1)$	5, 7, TL, PL

Figure 6.4: Proof of Theorem 4

A Comparison with Werner's Formalization of the Contract Net

In Werner's formalization of the CNET, [179, pp27–30], a contract net is defined to be a social group[9]:

$$\Sigma\Gamma_{CNET} = \langle L_{CNET}, G, \Sigma T_{CNET}, Roles, \Omega_{CNET} \rangle$$

where

- L_{CNET} is the language for communication in the CNET;
- G is a set of agents;
- ΣT_{CNET} is a social structure defining the abstract cognitive states associated with the roles contractor and manager;
- *Roles* maps a role in ΣT_{CNET} to each agent in G at each possible time;
- Ω_{CNET} is the "space of all possible histories of the system".

The two most significant parts of this description are the language L_{CNET} and the social structure ΣT_{CNET}.

The language L_{CNET} parallels the message types defined in our formalization (see above). Associated with the language is a *pragmatic interpretation function*, *Prag*, which defines how the cognitive states of agents are affected by messages. Werner only defines this function for one message type: awards. The effect of being awarded a contract is to make the awardee reject all possible courses of action that do not lead to the task being expedited; additionally, the manager "expects that the contractor intends to do the awarded task" [179, p29].

The next item of interest is the social structure, ΣT_{CNET}, which consists of a set of *roles*: a role is an "abstract agent …that defines the state, information, permission, responsibilities and values of that …role" [179]. However, Werner does no more than state that there are two roles in the CNET: contractor and manager.

In what ways are Werner's and our formalizations alike, and in what ways do they differ? Perhaps the main similarity is that they both recognize the dynamic nature of the roles manager and contractor. Additionally, both formalisms are in broad agreement on the perlocutionary force of the "award" message type. Apart from these points of comparison, the formalizations differ markedly. The most obvious difference is that our formalization is axiomatic, with properties that may readily be demonstrated via formal proof, whereas Werner's is not. This makes comparison difficult. Werner's formalization is, in one sense, very much more general than our own: whereas we assume the existence of specific domain predicates in an agent's belief set, Werner does not make any assumptions at all.

[9] A more detailed description and critique of Werner's formalism may be found in Appendix B.

6.4 Summary

Previous chapters have laid out a theory of multi-agent systems, and a number of linear time temporal logics that may be used to reason about systems modelled by the theory. This chapter was devoted to five case studies, which demonstrated how these logics might be used for describing, reasoning about, and specifying the properties of multi-agent systems.

After a brief introduction to the subject of reasoning about systems using temporal logic, two frameworks for building multi-agent systems were examined and modelled in terms of the theory of multi-agent systems. These frameworks were then axiomatized using QAL. The two frameworks were AGENT0 and Concurrent METATEM. Subsequently, three existing cooperative problem solving techniques were reviewed, and specified using QAL. The problem solving techniques were a simple master/slave framework, the cooperative inference technique used in the cooperating expert system FELINE, and the well known Contract Net.

The next chapter — the final chapter in this main part of the thesis — shows how a number of branching time temporal logics may be developed, and shown to correspond to the theory of multi-agent systems developed earlier. In contrast to the linear models of time that underpin the logics developed in Chapter 5, these branching time logics are based on a model of time that branches infinitely into the future from any point in time. These logics are more expressive than their linear time counterparts, and allow the consideration of attributes of agents (such as cooperative goals), that cannot be described using linear models.

Chapter 7

Branching Time Temporal Logics for Multi-Agent Systems

ALL of the logics that have been described and used in Chapters 5 and 6 have been based on a *linear* model of time: one in which each time point has at most one successor. The idea upon which their semantics are based is that of allowing a run of a multi-agent system to act as a model for a linear time temporal logic. However, linear temporal logics are by no means the only type of temporal logic available. A basic alternative is to view time as *branching* into the future from each time point. In a branching time structure, each time point may have a number of successors, depending on what actions are taken to change the state of the system. To see how branching time structures are related to multi-agent systems, consider the set of all runs of a system. One way of representing this set is to collect the runs together into a single branching structure. To fully express the properties of such a structure, one requires not only the ability to talk about things happening in the future, but of things being true in *some possible future* or *all possible futures*. This is the language of branching time temporal logic. This chapter develops two branching time logics for reasoning about multi-agent systems: BAL ("Branching AL") is a propositional branching time logic based on the expressive branching time system CTL^* [48]; the logic QBAL ("Quantified BAL") extends BAL with quantification, and with the addition of operators for describing the goals and abilities of agents and groups of agents.

There are several good reasons for supposing that branching time logics are suitable for reasoning about multi-agent systems:

1. Consider the semantic structures which underly branching time logics. These structures closely resemble *game trees*, the extensive representation formalism developed by game theorists for describing game-like multi-agent interactions. The link between multi-agent systems and game theory is by now well established (see e.g., [141]). So it seems that the semantic structure of branching time logic is appropriate for representing multi-agent interactions (Ladner and Reif came to similar conclusions [105]). Branching time logic therefore seems an appropriate linguistic tool for describing multi-agent interactions.

2. Branching time logics have certain inherent advantages over their linear counterparts: they allow the consideration of what *might* happen, rather than just what *actually* happens. This in turn allows the development of theories of ability, cooperative ability, goals, and cooperative goals, that are literally meaningless in a linear structure: this point is examined in detail later in the chapter.

3. Multi-agent systems are inherently reactive (in the Pnuelian sense [133]), and a proven tool for describing and reasoning about reactive systems is temporal logic (this issue was discussed in Chapter 1).

The remainder of this chapter is structured as follows. The next section develops objects called BAL frames. These objects are the foundation upon which a branching time semantic structure is built. The subsequent section develops the basic propositional branching time logic BAL. Section 7.3 then develops theories which show how goals and abilities can be attributed to agents in a branching time framework. Section 7.4 then incorporates these theories as operators in a quantified version of BAL called QBAL (see also [188]). Some short examples of the use of BAL are then given. The chapter concludes with some comments and a summary.

For a good introduction to the technical framework of branching time logics, and to CTL^* in particular (the system on which all the logics in this chapter are based), the reader is referred to [49]. CTL^* was originally described in [48]. A complete axiomatization of CTL^* was described in [166].

7.1 BAL Frames

The first step in the development of the branching time logics is to define BAL frames, the precursors to model structures of the language BAL.

$$Balfrm = \text{powerset } World \times \text{powerset } (World \times World)$$

In other words, a BAL frame is a set of worlds together with a binary relation on worlds: this is what is meant by a frame in classical modal logic [28]. The idea is that a BAL frame will represent the set of all runs of a system, in which case it is a *model of* that system (this follows the terminology of Chapters 4 and 5: the reader might find it useful to glance back at these chapters in order to refresh their memory on the various function and type definitions given therein).

The basic idea is as follows. Suppose $\langle W, R \rangle$ is a BAL frame. Then if $\langle w, w' \rangle \in R$, then w' represents one world that could arise from the world w in the system being modelled. The following function defines what it means for a BAL frame to be a model of a system.

$$\begin{array}{l}
model_of' : Balfrm \times System \rightarrow \mathbb{B} \\
model_of'(bfr, sys) \triangleq \\
\quad \text{let } \langle W, R \rangle = bfr \text{ in} \\
\quad init_world(sys) \in W \wedge \\
\quad \forall w \in W \cdot \forall w' \in World \cdot \\
\quad\quad next_world(w, w', sys) \Rightarrow (w' \in W) \wedge (\langle w, w' \rangle \in R) \wedge \\
\quad \forall w \in W \cdot \\
\quad\quad (w \neq init_world(sys)) \Rightarrow \langle init_world(sys), w \rangle \in tc(R) \wedge \\
\quad \forall w \in W \cdot \forall w' \in W \cdot \\
\quad\quad (\langle w, w' \rangle \in R) \Rightarrow next_world(w, w', sys)
\end{array}$$

This function requires some explanation. The first conjunct simply demands that the initial world of the system is in W. The second conjunct demands that all the legal ways the system could evolve are in W, R. The third conjunct demands that the only worlds in W are ones that could have arisen through a legal set of transitions of the system from the initial world. This demand is made via a function *tc*, which takes a relation and returns the *transitive closure* of the relation; the definition of transitive closure is essentially standard, and is therefore omitted[1]. The final conjunct ensures that the relation R contains only legal transitions.

A BAL frame that is the model of some system is said to be ordinary. Ordinary BAL frames have the property that the relation in the frame is *total*, (i.e., that every world has at least one successor). A definition of totality is given by the following function.

$$\begin{array}{l}
total : Balfrm \rightarrow \mathbb{B} \\
total(bfr) \triangleq \\
\quad \text{let } \langle W, R \rangle = bfr \text{ in} \\
\quad \forall w \in W \cdot \exists w' \in W \cdot \langle w, w' \rangle \in R
\end{array}$$

The result is stated formally in the following theorem.

Theorem 5 *All ordinary BAL frames are total.*

Proof *By the weak completness properties of agents, every agent will always have at least one possible move. So every state will have at least one possible transition associated with it. If there are any possible transitions from a world w then there will be a successor world w′, and* $\langle w, w' \rangle \in R$ *by the definition of model_of′. Hence R must be total.* ∎

Now some utility definitions. First, a *path* is defined to be a countably infinite sequence of worlds. A type for this already exists: *Worldseq*. Paths can thus be defined to be the same type. The reason for the change in terminology is to retain compatibility with the branching time logic literature.

$$Path = Worldseq$$

[1] Suppose $\langle w, w' \rangle \in R$ and $\langle w', w'' \rangle \in R$, then $\langle w, w'' \rangle$ would be in the transitive closure of R. The transitive closure of a relation is sometimes called the *ancestral* of the relation.

In addition, the following notational convention will be used: if p is a path, then the path obtained from p by omitting the first u elements is denoted $p^{(u)}$. The following function defines what it means for a path to be "on" a BAL frame.

$$path\_on : Path \times Balfrm \to \mathbb{B}$$

$$path\_on(p, bfr) \triangleq$$
$$\quad \text{let } \langle W, R \rangle = bfr \text{ in}$$
$$\quad (\text{hd}\, p \in W) \wedge$$
$$\quad \forall u \in \mathbb{N} \cdot (\langle p(u), p(u+1) \rangle \in R)$$

The following function takes a frame and returns the set of paths on it.

$$paths : Balfrm \to \text{powerset } Path$$

$$paths(bfr) \triangleq \{p \mid (p \in Path) \wedge path\_on(p, bfr)\}$$

7.2 The Logic BAL

This section introduces the logic BAL (for "Branching AL"). This logic is essentially AL augmented by two *path quantifiers*, "*E*" and "*A*", to describe things true in just one possible future, or true in all possible futures. We begin by defining the syntax of BAL. We subsequently develop model structures, and show how a model corresponds to a system. The semantics of BAL are then defined in the usual way. Finally, an axiomatization of BAL is presented.

7.2.1 Syntax

As with AL, the language BAL is parameterized by an internal language L. Thus an instance of BAL based on L is denoted $BAL(L)$. The alphabet of BAL is essentially that of AL augmented by the two path quantifiers mentioned above (note that BAL, like CTL^*, contains no past-time operators). The path quantifiers turn out to be duals of each other, so only one is introduced into the language as basic. The other is introduced as a derived operator.

Definition 17 *The alphabet of BAL (based on L) contains the following symbols:*

1. *The symbols* $\{\text{true}, \text{Bel}, \text{Send}, \text{Do}\}$*;*
2. *A set of constant symbols Const made up of the disjoint sets* $Const_{Ag}$ *(agent constants) and* $Const_{Ac}$ *(action constants);*
3. *All closed formulae of the internal language L;*
4. *The unary propositional connective "* $\neg$ *" and the binary propositional connective "* $\vee$ *";*
5. *The unary temporal connective "* $\bigcirc$ *", binary temporal connective "* $\mathcal{U}$ *", and path quantifier "A";*
6. *The punctuation symbols* $\{), (\}$.

The syntax of BAL is given by the following definition.

Definition 18 *The syntax of BAL (based on L) is defined by the following rules.*

1. *If* ϕ *is a closed formula of L,* α *is an action constant, and i, j are agent constants, then the following are formulae of BAL:*
   true $\quad$ (Bel i ϕ) $\quad$ (Send i j ϕ) $\quad$ (Do i α)
2. *If* ϕ, ψ *are formulae of BAL, then the following are formulae of BAL:*
   $\neg \phi \quad \phi \vee \psi$
3. *If* ϕ, ψ *are formulae of BAL, then the following are formulae of BAL:*
   $A\phi \quad \phi \mathcal{U} \psi \quad \bigcirc \phi$

(Bel i ϕ)	Agent i believes ϕ
(Do i α)	Agent i performs action α
(Send i j ϕ)	Agent i sent j message ϕ
$A\phi$	On all paths ϕ
$\bigcirc\phi$	Next ϕ (X)
$\phi\,\mathcal{U}\,\psi$	ϕ Until ψ (U)

Table 7.1: Non-Standard Operators in BAL

As observed earlier, the temporal component of BAL is based on the logic CTL^* [48]. The notation used in the original work has been altered slightly, in the interests of compatibility with AL. Table 7.1 summarizes the meaning of the non-standard operators in BAL. Where appropriate, Emerson and Halpern's original notation is given in parentheses.

7.2.2 Semantics

The next step is to define the semantics of BAL. This is carried out in three parts. First, model structures are defined, and a correspondence between model structures and systems is established. The formal semantics of the language are then given, as a set of semantic rules in terms of the satisfaction relation, in the usual way. Finally, satisfiability and validity for BAL are defined.

Model Structures

The language of BAL contains two non-empty sets of constants, ($Const_{Ag}$ — agent constants, and $Const_{Ac}$ — action constants), which are put in a bijective correspondence with the sets Ag (a set of agents) and Ac (a set of actions) in model structures by an *interpretation*, I. As usual, I must preserve sorts.

As might be expected, a model structure also contains a BAL frame. The type for BAL models is $Model_{BAL}$: it is defined as follows.

Definition 19 *A model for BAL is a structure:*

$$\langle W, R, Ag, Ac, I\rangle$$

where

- $\langle W, R\rangle$ *is a BAL frame;*
- $Ag \subseteq Agid$ *is a set of agent ids;*
- $Ac \subseteq Action$ *is a set of actions;*
- $I : Const \xleftrightarrow{m} (Ac \cup Ag)$ *interprets constants.*

As with AL, there is a close relationship between models and systems. This relationship is formalized in the following function, which defines the conditions under which a model is considered to be a model of a system.

$$model\_of'' : Model_{BAL} \times System \to \mathbb{B}$$

$$\begin{array}{l} model\_of''(M, sys) \triangleq \\ \quad \text{let } \langle W, R, Ag', Ac, I\rangle = M \text{ in} \\ \quad model\_of'(\langle W, R\rangle, sys) \wedge \\ \quad \text{let } \langle Ag, \Delta_0, \rho, \beta, \iota, MR, AR\rangle = sys \text{ in} \\ \quad (Ag = Ag') \wedge \\ \quad (Ac = \bigcup\{\{\alpha \mid \langle \phi, \alpha\rangle \in AR(i)\} \mid i \in Ag\}) \end{array}$$

$\langle M, p\rangle$	$\models$	true	
$\langle M, p\rangle$	$\models$	(Bel i ϕ)	iff $\phi \in$ *state*(hd p)($I(i)$)
$\langle M, p\rangle$	$\models$	(Send i j ϕ)	iff $\langle I(i), I(j), \phi\rangle \in$ *sent*(*trans*(hd p))
$\langle M, p\rangle$	$\models$	(Do i α)	iff $I(\alpha)$ = *action*(*trans*(hd p)($I(i)$))
$\langle M, p\rangle$	$\models$	$\neg\,\phi$	iff $\langle M, p\rangle \not\models \phi$
$\langle M, p\rangle$	$\models$	$\phi \vee \psi$	iff $\langle M, p\rangle \models \phi$ or $\langle M, p\rangle \models \psi$
$\langle M, p\rangle$	$\models$	$A\phi$	iff $\langle M, p'\rangle \models \phi$ for all $p' \in$ *paths*($\langle W, R\rangle$) such that (hd p' = hd p)
$\langle M, p\rangle$	$\models$	$\bigcirc\phi$	iff $\langle M, p^{(1)}\rangle \models \phi$
$\langle M, p\rangle$	$\models$	$\phi\,\mathcal{U}\,\psi$	iff $\langle M, p^{(u)}\rangle \models \psi$ for some $u \in \mathbb{N}$ and $\langle M, p^{(v)}\rangle \models \phi$ for all $v \in \mathbb{N} \cdot 0 \leq v < u$.

Figure 7.1: Semantics of BAL

$E\phi$	$\triangleq$	$\neg A \neg\,\phi$	On some path, ϕ
$\Diamond\phi$	$\triangleq$	true $\mathcal{U}\,\phi$	Sometime ϕ
$\Box\phi$	$\triangleq$	$\neg \Diamond \neg\,\phi$	Always ϕ
$\phi\,\mathcal{W}\,\psi$	$\triangleq$	$\Box\phi \vee \phi\,\mathcal{U}\,\psi$	ϕ Unless ψ

Table 7.2: Derived Operators for BAL

So a model for BAL is a model of a system if its frame defines all execution sequences of the system, and the set of actions and agents in the model have the same members as the corresponding sets in the system. As with AL, a model for BAL is *ordinary* iff it is the model of some system.

Semantic Rules

The semantics of BAL are presented via the satisfaction relation "$\models$" in the usual way. The relation holds between pairs of the form:

$$\langle M, p\rangle$$

(where M is a model for BAL, and p is a path), and formulae of the language. The semantic rules for BAL are given in Figure 7.1.

The first four rules define the semantics of atomic formulae of BAL. These primitives have the same reading as their AL counterparts, and have the same properties (e.g., the attachment lemma for belief still holds). The next two rules define the usual propositional connectives $\neg$ (not) and $\vee$ (or). These operators have standard semantics: the remaining propositional connectives ($\wedge$, $\Rightarrow$, $\Leftrightarrow$ and false) are defined as abbreviations in the usual way.

The A operator is read: "on all paths". Thus $A\phi$ will be satisfied if ϕ is satisfied on all paths that have the same head as the reference path. The E operator is defined as an abbreviation in Table 7.2. It is read: "on some path". Thus $E\phi$ will be satisfied if ϕ is satisfied on at least *one* path that has the same head as the reference path.

The final rules define the temporal operators. These are essentially the future time operators from AL (see Table 7.2 for derived operators). Thus $\bigcirc\phi$ will be true if ϕ is satisfied in the next world of the reference path, $\Box\phi$ will be satisfied if ϕ is satisfied now and at all points along the reference path, and $\phi\,\mathcal{U}\,\psi$ will be satisfied if ψ is true at some future point on the path, and ϕ is true until that point.

Satisfiability and Validity

Satisfiability and validity are defined in the normal way. If ϕ is a formula of BAL, then ϕ is *satisfiable* in a model M if for some path p, $\langle M, p\rangle \models \phi$, satisfiable *simpliciter* if it is satisfiable in some normal model, *true in a model* if it is satisfied on all paths in the model, and *valid* in a non-empty class of models if it is true in each member of the class. Finally, it is valid *simpliciter* if it is valid in the class of all models, this latter property being indicated by $\models \phi$.

7.2.3 Proof Theory

Axioms

Begin by noting that all instances of propositional tautologies are theorems.

$$\vdash \quad \phi \quad \text{where } \phi \text{ is a propositional tautology} \tag{7.1}$$

The following axioms deal with the temporal/path component of BAL (they are adapted from Stirling's axiomatization of *CTL** [166]). One other axiom from Stirling's axiomatization has been omitted, due to its complexity; see [128] for a discussion of the axiomatization.

$$\vdash \quad \Box(\phi \Rightarrow \psi) \Rightarrow (\Box\phi \Rightarrow \Box\psi) \tag{7.2}$$
$$\vdash \quad \neg \bigcirc \phi \Leftrightarrow \bigcirc \neg \phi \tag{7.3}$$
$$\vdash \quad \bigcirc(\phi \Rightarrow \psi) \Rightarrow (\bigcirc\phi \Rightarrow \bigcirc\psi) \tag{7.4}$$
$$\vdash \quad \Box\phi \Leftrightarrow \phi \wedge \bigcirc \Box\phi \tag{7.5}$$
$$\vdash \quad \Box(\phi \Rightarrow \bigcirc\phi) \Rightarrow (\phi \Rightarrow \Box\phi) \tag{7.6}$$
$$\vdash \quad \phi \mathcal{U} \psi \Rightarrow \Diamond\psi \tag{7.7}$$
$$\vdash \quad \phi \mathcal{U} \psi \Leftrightarrow \psi \vee (\phi \wedge \bigcirc(\phi \mathcal{U} \psi)) \tag{7.8}$$
$$\vdash \quad \phi \Rightarrow A\phi \text{ if } \phi \text{ is atomic} \tag{7.9}$$
$$\vdash \quad E\phi \Rightarrow \phi \text{ if } \phi \text{ is atomic} \tag{7.10}$$
$$\vdash \quad A\phi \Rightarrow \phi \tag{7.11}$$
$$\vdash \quad A(\phi \Rightarrow \psi) \Rightarrow (A\phi \Rightarrow A\psi) \tag{7.12}$$
$$\vdash \quad A\phi \Rightarrow AA\phi \tag{7.13}$$
$$\vdash \quad E\phi \Rightarrow AE\phi \tag{7.14}$$
$$\vdash \quad A\bigcirc\phi \Rightarrow \bigcirc A\phi \tag{7.15}$$

Axioms (7.2) to (7.8) deal with the future time operators, and require no explanation. To see that (7.9) and (7.10) are sound, recall that an atomic formula can only refer to the present time.

Axiom (7.11) is simply a version of axiom *T*, from classical modal logic; axiom (7.12) is a version of classical modal axiom *K*; (7.13) is a version of classical modal axiom *4*, and (7.14) is a version of classical modal axiom *5*. Axiom (7.15) captures the interaction of the next-time operator and the path quantifier "*A*".

Finally, there is the attachment axiom for belief operators: the soundness of this axiom follows from the attachment lemma (see Chapter 5).

$$\vdash \quad (\mathsf{Bel}\ i\ \phi_1) \wedge \cdots \wedge (\mathsf{Bel}\ i\ \phi_n) \Rightarrow (\mathsf{Bel}\ i\ \phi) \tag{7.16}$$
$$\text{where } \{\phi_1, \ldots, \phi_n\} \vdash_{\rho(i)} \phi$$

Inference Rules

Begin by observing that propositional reasoning is sound, and so modus ponens (MP) is a rule of inference.

$$\text{From } \vdash \phi \text{ and } \vdash \phi \Rightarrow \psi \text{ infer} \vdash \psi \tag{7.17}$$

The following inference rules are from Stirling's axiomatization of *CTL** [166].

$$\text{From } \vdash \phi \text{ infer } \vdash \Box\phi \tag{7.18}$$

$$\text{From } \vdash \phi \text{ infer } \vdash A\phi \tag{7.19}$$

7.3 Cooperative Ability and Cooperative Goals

This section shows how the semantic structures developed in previous sections can be used to examine some important issues in the development of cooperative systems. First, we develop a theory of *cooperative ability*. (Ability is here assumed to mean having the power to bring about some state of events — i.e., "know that" rather than "know how".) In addition, we develop a theory of cooperative goals. This theory does not posit the existence of a distinguished cognitive "goal" state (cf. [29]), but rather allows us to *attribute* goals to agents (and groups of agents). As we shall see, when these theories are incorporated into a first-order language QBAL, they provide a powerful tool for reasoning about cooperative systems.

7.3.1 Cooperative Ability

What does it mean to have the ability to achieve a goal? It doesn't mean having a single message or cognitive action available which will bring about the desired goal, as many goals depend on the successful execution of complex sequences of actions. We say that an agent (or group of agents) is able to achieve a goal if there is a plan for the agent (group of agents) telling the agent (group of agents) what moves to make such that if the agent (group of agents) follows the plan then the goal is *guaranteed* to be achieved. Two problems: What do we mean by "plan", and what do we mean by "guaranteed"? The AI planning community generally views a plan as a partially ordered sequence of actions. Rather than choose a direct, literal representation of plans, we fix on an abstraction called *strategies*, a concept originally developed by game theorists. A strategy can be thought of as a strong kind of abstract conditional plan. We model a strategy as a function from belief sets to moves.

Now what do we mean by "guaranteed"? Take some world in a BAL frame, some agent and some strategy. From that world, a set of *futures* emerge, the paths rooted in the world. On some (but not necessarily all) of these futures, the moves performed by the agent will correspond to those "suggested" by the strategy. Call these paths the futures of the strategy. If the goal is achieved in *every* future of the strategy then the goal is a necessary consequence of following the strategy. This is what we mean by a strategy guaranteeing a goal.

This definition rests on a rather subtle property of BAL frames; that they actually do contain all the possible legal ways a system might evolve. If they did not then there might be some future of a strategy which did not appear in the frame, on which the goal was not achieved. In this case the strategy could not be said to guarantee the goal. Note that this definition of ability can be applied just as easily to groups of agents. The discussion above is formalized below. First, the type for strategies.

$$Strat = Belset \to Move$$

A strategy will be *sound* for an agent if it never dictates an illegal move for the agent.

$$st\_sound : Strat \times Agent \to \mathbb{B}$$
$$st\_sound(st, ag) \triangleq \forall \Delta \in Belset \cdot mv\_legal(st(\Delta), ag, \Delta)$$

The next function defines what it means for a strategy for some agent to *hold* on a path. The idea is that the strategy will hold on a path if the agent's moves on that path correspond to those dictated by the strategy.

$$s\_holds\_on : Strat \times Agid \times Path \to \mathbb{B}$$
$$s\_holds\_on(st, i, p) \triangleq$$
$$\quad \forall u \in \mathbb{N}_1 \cdot$$
$$\quad\quad \text{let } \Delta = state(\mathsf{hd}\, p^{(u-1)})(i) \text{ in}$$
$$\quad\quad \text{let } m = trans(\mathsf{hd}\, p^{(u)})(i) \text{ in}$$
$$\quad\quad st(\Delta) = m$$

The idea of a strategy can be generalized to a group of agents via a *joint strategy*. It is convenient to define a joint strategy as a map.

$$Jstrat = Agid \xrightarrow{m} Strat$$

It seems sensible to demand that joint strategies are non-empty, which leads to the following invariant.

$$\forall js \in Jstrat \cdot (\mathsf{dom}\, js \neq \{\})$$

A joint strategy will be sound for a group of agents just in case each of its "member" strategies is sound.

$$js\_sound : Jstrat \times System \to \mathbb{B}$$
$$js\_sound(js, sys) \triangleq \forall i \in \mathsf{dom}\, js \cdot st\_sound(js(i), agent(i, sys))$$

For convenience, a function is defined returning the set of sound joint strategies for a group of agents in a system.

$$sjs : \mathsf{powerset}\, Agid \times System \to \mathsf{powerset}\, Jstrat$$
$$sjs(g, sys) \triangleq \{js \mid (js \in Jstrat) \wedge (\mathsf{dom}\, js = g) \wedge js\_sound(js, sys)\}$$

The definition of a single strategy holding on a path can easily be extended to encompass joint strategies.

$$
\begin{array}{l}
js\_holds\_on : Jstrat \times Path \to \mathbb{B} \\
js\_holds\_on(js, p) \triangleq \\
\quad \forall i \in \mathrm{dom}\, js \cdot s\_holds\_on(js(i), i, p)
\end{array}
$$

Now it is possible to define for some joint strategy, BAL frame, and world in the BAL frame, the set of paths emerging from the world such that the joint strategy holds on the paths: these are called the *futures* of the strategy. The idea is somewhat similar to what Werner calls *potential* [180].

$$
\begin{array}{l}
futures : Jstrat \times Balfrm \times World \to \mathrm{powerset}\, Path \\
futures(js, bfr, w) \triangleq \\
\quad \{p \mid p \in paths(bfr) \wedge js\_holds\_on(js, p) \wedge (\mathrm{hd}\, p = w)\}
\end{array}
$$

A joint strategy will then *guarantee* a goal from some world in a BAL frame if the goal is satisfied on each future of the strategy from the world. This leads to the following, semi-formal definition of ability: it is semi-formal because it does not define what it means for a goal to be satisfied on some path. This is a logical concept, which is defined in the semantics of a language.

$$
\begin{array}{l}
can : \mathrm{powerset}\, Agid \times goal \times Balfrm \times World \times System \to \mathbb{B} \\
can(g, \phi, bfr, w, sys) \triangleq \\
\quad \exists js \in sjs(g, sys) \cdot \\
\quad\quad \forall p \in futures(js, bfr, w) \cdot \\
\quad\quad\quad \phi \text{ is satisfied on } p
\end{array}
$$

Note that the definition of ability presented in this section is loosely based on that developed by Werner [182]; see also Appendix B. Note also that the definition is quite different to that of Moore, who presented an essentially mentalistic definition of ability (see [124], and Chapter 2).

7.3.2 Cooperative Goals

This section uses the semantic structures developed earlier to develop a theory of (joint) goals. This theory allows the observer of a system to *attribute* goals to agents or groups of agents. The theory does not posit the existence of distinguished cognitive "goal states", either in individual agents or groups of agents. It is not possible to identify any direct representation of a goal in the *structure* of an agent: there is no internalized collection of possible worlds with a goal relation holding between them (cf. [29]), nor is there a "goal stack", or "intention box" (cf. [76]). An agent — or group of agents — only has goals by virtue of our attributing them.

How are we to go about attributing goals to agents? The idea we adopt was inspired by Seel [148]. The semantics of goals are developed via possible worlds, but rather than "gratuitously inflict" possible worlds onto the semantic structures we have developed, we instead look to our model to see where such worlds might lie latent. If we can find suitable candidates, then they can be used to give a semantics to goals.

The worlds that we pick out for the semantics of "goal" are paths in BAL structures as before. The idea is that to attribute a goal to an agent you must first look at what it has actually done. Doing involves choosing. Choices mean expressing a preference for the consequences of one move over another. Preferences are themselves dictated by goals. In making a choice, say choosing move m_1 to move m_2, an agent is preferring the consequences of m_1 to m_2. Crudely, an agent has a goal of the necessary consequences of its actions. Since the consequences of all moves are contained in a BAL frame, it ought to be possible to somehow "decode" a path and deduce what goals an agent had, relative to that path.

The first step is to define what it means for two paths to *agree* on the actions of a group of agents.

$$
\begin{array}{l}
agree : Path \times Path \times \mathrm{powerset}\, Agid \to \mathbb{B} \\
agree(p, p', g) \triangleq \\
\quad \forall i \in g \cdot \forall u \in \mathbb{N} \cdot (trans(\mathrm{hd}\, p^{(u)})(i) = trans(\mathrm{hd}\, p'^{(u)})(i))
\end{array}
$$

FOR ϕ	1^{st} MOVE	2^{nd} MOVE
R	m_1	m_2
F	$\times$	m_4

FOR ψ	1^{st} MOVE	2^{nd} MOVE
R	m_1	m_1
F	$\times$	$\times$

Table 7.3: Ability and Goals: An Example Scenario

This leads immediately to the following semi-formal definition of joint goals.

$$goal : \mathsf{powerset}\ Agid \times goal \times Balfrm \times Path \to \mathbb{B}$$

$$goal(g, \phi, bfr, p) \triangleq$$
$$\forall p' \in paths(bfr) \cdot$$
$$(\mathsf{hd}\, p = \mathsf{hd}\, p') \wedge agree(p, p', g) \Rightarrow$$
$$\phi \text{ is satisfied on } p'$$

7.3.3 Ability and Goals: An Example

To illustrate the ideas of the previous sections, an extended example is presented.

Suppose a system contains two agents, Ralph (R) and Freda (F). At every point in time, each agent has two possible moves. R can perform any of $\{m_1, m_2\}$, and F can perform any of $\{m_3, m_4\}$. To achieve a goal ϕ, it is necessary for R to perform m_1 on the first move, and m_2 on the second, and for F to perform m_4 on the second move. (It doesn't matter what F does on the first move.) To achieve the goal ψ, it is necessary for R to perform m_1 on both moves — it doesn't matter what F does.

This scenario is summarized in Table 7.3; a cross at the intersection of a row and column indicates that it does not matter what the agent does.

At each point in time there will be four possible transitions of the system, so a sequence of two moves generates a total of sixteen distinct futures, which we call p_1 to p_{16}. The possible futures are described in Table 7.4. In the final two columns, a tick ($\surd$) indicates that the associated goal is achieved on that path, whereas a cross ($\times$) indicates that it is not.

On only two of the sixteen futures — those labeled p_4 and p_{12} in the Table — will ϕ be achieved. By the definition above, the coalition $\{R, F\}$ can achieve ϕ. The strategy simply requires that the agents perform the moves dictated by the table above. However, neither of the agents can achieve ϕ individually.

If one asks "do $\{R, F\}$ have a goal of ϕ on path p_4 (or p_{12})", the answer is yes, because on each path that agrees with p_4/p_{12} on the moves of $\{R, F\}$, ϕ is achieved. However, neither agent individually has a goal of ϕ on any path in the scenario.

The goal ψ will be achieved on four of the futures — p_1, p_2, p_9 and p_{10}. R will have a goal of ψ on any of these paths; F will not have a goal of ψ on any path. By the definition above, R can achieve ψ, whereas F cannot.

To paraphrase Halpern, ([80]), the definitions of ability and goals are *external* concepts. We don't imagine an agent scratching its head, wondering if it can achieve ϕ, or wondering if it has a goal of ϕ. Instead, we, as observers and designers may use these notions to describe a system, and perhaps to design it. It is not necessary for an agent to represent goals or ability in order to *have* goals and ability.

7.4 A First-Order Branching Time Logic for Multi-Agent Systems

This section draws together the results of the previous chapters and sections by developing a quantified version of BAL called QBAL ("Quantified BAL"). The logic QBAL allows quantification over agents, actions, and individuals in the domain of the internal language. It extends BAL by the addition of modal operators for describing the abilities and goals of agents and groups of agents, and by allowing the relationship between an agent and the groups of which it is a member to be described. It thus becomes possible to express such statements as "the goal ϕ cannot be achieved without the agent i", and so on.

PATH	1^{st} MOVE		2^{nd} MOVE		ϕ ?	ψ ?
	R	F	R	F		
p_1	m_1	m_3	m_1	m_3	×	√
p_2	m_1	m_3	m_1	m_4	×	√
p_3	m_1	m_3	m_2	m_3	×	×
p_4	m_1	m_3	m_2	m_4	√	×
p_5	m_2	m_3	m_1	m_3	×	×
p_6	m_2	m_3	m_1	m_4	×	×
p_7	m_2	m_3	m_2	m_3	×	×
p_8	m_2	m_3	m_2	m_4	×	×
p_9	m_1	m_4	m_1	m_3	×	√
p_{10}	m_1	m_4	m_1	m_4	×	√
p_{11}	m_1	m_4	m_2	m_3	×	×
p_{12}	m_1	m_4	m_2	m_4	√	×
p_{13}	m_2	m_4	m_1	m_3	×	×
p_{14}	m_2	m_4	m_1	m_4	×	×
p_{15}	m_2	m_4	m_2	m_3	×	×
p_{16}	m_2	m_4	m_2	m_4	×	×

Table 7.4: Paths, Moves, and Goals

7.4.1 Syntax

The alphabet of the QBAL language extends that of BAL by the addition of the following symbols:

- the symbols $\{=, \in, \mathsf{Can}, \mathsf{Goal}\}$;
- extra constants $Const_U$ for individuals in the domain of the internal language;
- a countable set of individual variables *Var*, made up of the sets Var_{Ag} (agent variables), Var_{Ac} (action variables), Var_G (variables denoting groups of agents) and Var_U, (variables for individuals in the domain of L);
- the quantifier symbol "$\forall$";
- the punctuation symbol "$\cdot$".

As usual, a term is defined to be either a variable or a constant. The symbol θ will be used to denote a term. Each term has an associated *sort*, which will be one of: *Ag*, *Ac*, *U* or *G*. To indicate that a term θ is of sort s we write θ_s. The syntax of QBAL is defined below.

Definition 20 *The syntax of QBAL (based on L) is defined by the following rules:*

1. *If $\theta_s, \ldots$ are terms of the named sorts, and ϕ is a formula of L, then the following are formulae of QBAL:*
   true $(\mathsf{Bel}\ \theta_{Ag}\ \phi)$ $(\mathsf{Send}\ \theta_{Ag}\ \theta_{Ag}'\ \phi)$ $(\mathsf{Do}\ \theta_{Ag}\ \theta_{Ac})$
   $(\theta_s = \theta'_s)$ $(\theta_{Ag} \in \theta_G)$
2. *If θ_G is a term of sort G, and ϕ is a formula of QBAL then the following are formulae of QBAL:*
   $(\mathsf{Can}\ \theta_G\ \phi)$ $(\mathsf{Goal}\ \theta_G\ \phi)$
3. *If ϕ is a formula of QBAL and x is a variable then $\forall x \cdot \phi$ is a formula of QBAL.*
4. *QBAL contains the propositional, path, and temporal formation rules of BAL.*

QBAL thus contains four primitive operators in addition to those provided by BAL. One of these is standard first-order equality; the remainder are summarized in Table 7.5.

7.4.2 Semantics

The semantics of the language are presented in two parts: first model structures, then semantic rules. The definitions of satisfiability and validity are essentially standard, and are therefore omitted.

(Can g ϕ)	The group/coalition g can make ϕ true
(Goal g ϕ)	The group/coalition g have a goal of ϕ
($i \in g$)	Agent i is a member of the group/coalition g

Table 7.5: Non-standard Operators in QBAL

Model Structures

QBAL contains terms of four sorts: agent, action, group, and individual in the domain of L. Associated with each is a non-empty set of individuals, which will appear in model structures:

- Ag — agents;
- Ac — actions;
- U — the domain of L;
- G — the set of groups of agents (i.e., the set of non-empty subsets of Ag).

The *domain of quantification* is defined as the union of these sets. The next few definitions deal with the technical apparatus of quantification, which is essentially the same as the logic QAL in Chapter 5; the reader familiar with this treatment can safely skip these definitions.

As in BAL, a bijective mapping I is defined between elements of *Const* and elements of the domain of quantification, called an interpretation. Interpretations must preserve sorts. The inverse of I, written N_I, is a naming function; where I is understood, N_I is abbreviated to N. Note that constants are assumed to be standard names, so N assigns each element of the domain of quantification a unique standard name. A mapping between elements of *Var* and the domain of quantification that preserves sorts is called a *variable assignment*.

A transformation is now defined on formulae of L.

Definition 21 *Let V be a variable assignment, N be a naming function, and ϕ be an arbitrary formula of L. By $\phi^{N,V}$ we mean the formula obtained from ϕ by replacing every variable x which occurs free in ϕ by $N(V(x))$.*

So, for example, let $P(x, a_2, y)$ be a formula of L, $V(x) = d_1$, $V(y) = d_3$, $N(d_1) = a_1$, and $N(d_2) = a_2$, then $P(x, a_2, y)^{N,V} = P(a_1, a_2, a_3)$. So every free variable is replaced by the standard name associated with the object the variable denotes.

A function $[\![\_]\!]_{I,V}$ is defined, which returns the denotation of a term relative to a variable assignment and interpretation.

$$[\![\theta]\!]_{I,V} \triangleq \begin{array}{l} \textit{if } \theta \in \textit{Const} \\ \textit{then } I(\theta) \\ \textit{else } V(\theta) \end{array}$$

Where I, V are understood, $[\![\theta]\!]_{I,V}$ is abbreviated to $[\![\theta]\!]$.

Finally, to give the semantics for ability, it will be necessary to include in model structures the function *sjs*, which returns the set of sound joint strategies for each group. However, as it was defined earlier, one of the arguments to this function was a system, which is not available in a model structure. So the function *sjs*, when it appears in model structures, is assumed to be relativized to the system the model structure is a model of. We can now define the form of model structures for QBAL.

Definition 22 *A model for QBAL is a structure:*

$$\langle W, R, Ag, Ac, G, U, I, sjs \rangle$$

where

- *G is the set of non-empty subsets of Ag;*
- *U is a non-empty set of individuals (the domain of L);*
- *I: $Const \overset{m}{\longleftrightarrow} (Ag \cup Ac \cup G \cup U)$ interprets constants;*
- *sjs is a function that returns, for each group of agents $g \in G$, the set of sound joint strategies for that group*

and the remaining components are as in BAL.

$$
\begin{array}{lll}
\langle M, V, p\rangle & \models & (\mathsf{Bel}\ \theta_{Ag}\ \phi) \quad \text{iff } \phi^{V,N} \in \mathit{state}(\mathsf{hd}\, p)([\![\theta_{Ag}]\!]) \\
\langle M, V, p\rangle & \models & (\mathsf{Do}\ \theta_{Ag}\ \theta_{Ac}) \quad \text{iff } \mathit{action}(\mathit{trans}(\mathsf{hd}\, p)([\![\theta_{Ag}]\!])) = [\![\theta_{Ac}]\!] \\
\langle M, V, p\rangle & \models & (\mathsf{Send}\ \theta_{Ag}\ \theta'_{Ag}\ \phi) \quad \text{iff } \langle [\![\theta_{Ag}]\!], [\![\theta'_{Ag}]\!], \phi^{V,N}\rangle \in \mathit{sent}(\mathit{trans}(\mathsf{hd}\, p)) \\
\langle M, V, p\rangle & \models & (\mathsf{Can}\ \theta_G\ \phi) \quad \text{iff } \exists js \in \mathit{sjs}([\![\theta_G]\!]) \cdot \\
 & & \qquad \forall p' \in \mathit{futures}(js, \langle W, R\rangle, \mathsf{hd}\, p) \cdot \\
 & & \qquad\quad \langle M, V, p'\rangle \models \phi \\
\langle M, V, p\rangle & \models & (\mathsf{Goal}\ \theta_G\ \phi) \quad \text{iff } \forall p' \in \mathit{paths}(\langle W, R\rangle) \cdot \\
 & & \qquad (\mathsf{hd}\, p = \mathsf{hd}\, p') \wedge \\
 & & \qquad\quad \mathit{agree}(p, p', [\![\theta_G]\!]) \Rightarrow \\
 & & \qquad\quad\quad \langle M, V, p'\rangle \models \phi \\
\langle M, V, p\rangle & \models & (\theta_s = \theta'_s) \quad \text{iff } [\![\theta_s]\!] = [\![\theta'_s]\!] \\
\langle M, V, p\rangle & \models & (\theta_{Ag} \in \theta_G) \quad \text{iff } [\![\theta_{Ag}]\!] \in [\![\theta_G]\!] \\
\langle M, V, p\rangle & \models & \forall x \cdot \phi \quad \text{iff } \langle M, V \dagger \{x \mapsto d\}, p\rangle \models \phi \text{ for all} \\
 & & \qquad d \text{ in the domain of quantification} \\
 & & \qquad \text{such that } d \text{ is the same sort as } x
\end{array}
$$

Figure 7.2: Semantics of QBAL

Semantic Rules

The semantics of QBAL are defined via the satisfaction relation "$\models$", which holds between structures of the form:

$$\langle M, V, p\rangle$$

(where M is a model for QBAL, V is a variable assignment, and p is a path), and formulae of QBAL. The semantic rules for atomic formulae are given in Figure 7.2. The remaining rules (for propositional connectives, temporal operators and path quantifiers) are all essentially unchanged from BAL, and are therefore omitted.

The first three operators have the same intended reading as their BAL counterparts. The remainder are new. The atomic formula ($\mathsf{Can}\ g\ \phi$) is read: "the group g can bring about a world where ϕ is true". Note that ϕ is a formula of QBAL. The semantics of Can are based on the theory of cooperative ability developed earlier. Thus being able to achieve something implies the existence of a strategy which guarantees the goal.

The atomic formula ($\mathsf{Goal}\ g\ \phi$) is read: "the group g have a goal of ϕ". Its semantics are based on the theory of cooperative goals outlined above. This operator is perhaps slightly more difficult to understand than Can, as it expresses a property of paths. It is an impartial assessment of what a group of agent's goals are if they follow a particular course of action. It says that the agents acted in such a way as to make the goal *inevitable*. Note that the goal operator has an interesting dual.

$$(\mathsf{WeakGoal}\ g\ \phi) \triangleq \neg (\mathsf{Goal}\ g \neg \phi)$$

The dual is called *weak goal* because it suggests that an agent is not entirely antagonistic towards the goal, i.e., it doesn't want the possibility of the goal absolutely excluded, but neither does it want it guaranteed.

The "=" operator is usual first-order equality. Note that its arguments must be of the same sort. The "$\in$" operator allows reasoning about the members of a coalition in a simple way: the atomic formula ($i \in g$) is read: "i is a member of the group g". Using this operator and equality it is possible to build some other useful operators for reasoning about coalitions. For example:

$$
\begin{array}{lll}
(g \subseteq g') & \triangleq & \forall x \cdot (x \in g) \Rightarrow (x \in g') \\
(g \subset g') & \triangleq & (g \subseteq g') \wedge \neg (g = g')
\end{array}
$$

It is occasionally useful to reason about coalitions which contain just a single agent. An operator Grp can be defined which "makes a group" out of its first argument.

$$(\mathit{Grp}\ i\ g) \triangleq \forall x \cdot (x \in g) \Rightarrow (x = i).$$

Using this operator it would be possible to define single agent versions of Can and Goal.

The final rule defines the semantics of universally quantified formulae: this is standard for many-sorted first-order logics. As usual, the existential quantifier "$\exists$" is defined as the dual of the universal quantifier.

7.4.3 Proof Theory

QBAL contains the proof rules and axioms of a many-sorted first-order logic (which largely amounts to ensuring that substitutions are of the correct sort). Since many-sorted first-order proof systems are standard fare, this section will focus on the logic of the two new modal operators: Can and Goal.

We begin by looking at the Can operator.

$$\vdash \quad \forall x \cdot A(\phi \Rightarrow \psi) \Rightarrow ((\mathsf{Can}\ x\ \phi) \Rightarrow (\mathsf{Can}\ x\ \psi)) \tag{7.20}$$

$$\vdash \quad E\phi \Leftrightarrow \exists x \cdot (\mathsf{Can}\ x\ \phi) \tag{7.21}$$

$$\vdash \quad \forall x \cdot \neg (\mathsf{Can}\ x\ \phi) \Rightarrow (\mathsf{Can}\ x \neg (\mathsf{Can}\ x\ \phi)) \tag{7.22}$$

$$\vdash \quad \forall x \cdot (\mathsf{Can}\ x\ \phi) \Rightarrow \forall y \cdot (x \subseteq y) \Rightarrow (\mathsf{Can}\ y\ \phi) \tag{7.23}$$

Axiom (7.20) says that if $\phi \Rightarrow \psi$ holds on all paths, then if any group can bring about ϕ, they can also bring about ψ.

Axiom (7.21) might be called a *principle of achievability*: if something is possible, then there is a group of agents that can achieve it (if something is true on just one path then it requires the "grand coalition" of all agents to achieve it). The reverse implication says that anything which is achievable is also possible. The reverse implication is actually a version of the axiom *T* from classical modal logic.

Axiom (7.22) is a version of axiom *5* from classical modal logic: it says that if a group cannot do something, then they can bring about a state where they cannot do it.

Axiom (7.23) might be called a *principle of added value*: any group to which an agent adds its efforts can achieve anything the agent can achieve on its own. This axiom is similar to the "superadditivity" principle in game theory (which states that the utility of a coalition is equal to the sum of the utilities of the member agents working in isolation).

The logic of Goal is similar.

$$\vdash \quad \forall x \cdot (\mathsf{Goal}\ x\ \phi \Rightarrow \psi) \Rightarrow ((\mathsf{Goal}\ x\ \phi) \Rightarrow (\mathsf{Goal}\ x\ \psi)) \tag{7.24}$$

$$\vdash \quad \forall x \cdot (\mathsf{Goal}\ x\ \phi) \Rightarrow E\phi \tag{7.25}$$

$$\vdash \quad \forall x \cdot \neg (\mathsf{Goal}\ x\ \phi) \Rightarrow (\mathsf{Goal}\ x \neg (\mathsf{Goal}\ x\ \phi)) \tag{7.26}$$

$$\vdash \quad \forall x \cdot (\mathsf{Goal}\ x\ \phi) \Rightarrow \forall y \cdot (x \subseteq y) \Rightarrow (\mathsf{Goal}\ y\ \phi) \tag{7.27}$$

Axiom (7.24) is just *K*, from classical modal logic.

Axiom (7.25) might be called a *realism* axiom: agents only have goals of things that are possible; it is a version of *T*.

Axiom (7.26) is a version of axiom *5* from classical modal logic, and says that if a group don't have a goal of something, then they have a goal of not having a goal of it. One might say they are "determined" in their attitudes.

Finally, axiom (7.27) says that a group inherits all the goals of its members.

Before leaving the axioms, note that the following are *not* theorems.

$$\nvdash \quad \forall x \cdot (\mathsf{Can}\ x\ \phi) \Rightarrow \forall y \cdot (y \subseteq x) \Rightarrow (\mathsf{Can}\ y\ \phi)$$
$$\nvdash \quad \forall x \cdot (\mathsf{Goal}\ x\ \phi) \Rightarrow \forall y \cdot (y \subseteq x) \Rightarrow (\mathsf{Goal}\ y\ \phi)$$

In other words, the members of a group do not inherit all the abilities and goals of the group.

There are two new inference rules, one each for Can and Goal. These rules are generalization rules, which say that groups can vacuously achieve necessary truths, and vacuously have goals of necessary truths.

$$\text{From } \vdash \phi \text{ infer } \vdash \forall x \cdot (\mathsf{Can}\ x\ \phi) \tag{7.28}$$

$$\text{From } \vdash \phi \text{ infer } \vdash \forall x \cdot (\mathsf{Goal}\ x\ \phi) \tag{7.29}$$

Soundness proofs for all the above axioms and inference rules are straightforward, and are therefore omitted. In particular, the soundness proofs for axioms *K*, *T*, and *5*, and the necessitation rule from classical modal logic may be readily adapted to prove the soundness of most of the above [28].

7.4.4 Examples

The aim of this section is to present some short examples, showing how QBAL might be used to express desirable properties of cooperative systems. No larger examples are given as the principles of specification using branching time logics are essentially the same as those using linear time logics; see Chapter 6.

The first example shows how the logic can be used to capture the idea of an agent being *required* in order to achieve a goal. Some goals cannot be achieved without the help of some agents. For example, suppose an agent manages a database containing information that is not replicated elsewhere. Then any goal which involves the use of that information *must* require the helpful interaction of the database manager agent. The property of an agent being *required* to achieve a goal is captured by the following definition.

$$(\mathit{Reqd}\ i\ \phi) \triangleq \forall x \cdot (\mathsf{Can}\ x\ \phi) \ \Rightarrow\ (i \in x) \tag{7.30}$$

Analogous to the above, we can pick out the set of agents that are required to achieve a goal. Any coalition which forms to achieve a goal must contain all required members if it is to succeed.

$$(\mathit{GReqd}\ g\ \phi) \triangleq \forall x \cdot (\mathsf{Can}\ x\ \phi) \ \Rightarrow\ (g \subseteq x) \tag{7.31}$$

We present a derived operator intended to capture the idea of a coalition committing to some goal. The idea is that committing to a goal means restricting your actions to just those that have the goal as a consequence (this is superficially similar to the definition of commitment in [150], though we are here using the term commitment in a very loose sense; we certainly don't intend it to capture the nuances of commitment that Cohen and Levesque focus on in [29]).

$$(\mathsf{Commit}\ g\ \phi) \triangleq A(\mathsf{Goal}\ g\ \phi) \tag{7.32}$$

The following examples will make use of Commit.

Suppose that a specification has been given as a set of formulae $\{\phi_1, \ldots, \phi_n\}$ of QBAL. These formulae are assumed to be *common goals* of the system. That is, each agent works towards achieving each of the goals. Suppose the formulae are *liveness* requirements, so that each ϕ_i should be satisfied infinitely often. The simplest requirement is that this should actually happen.

$$A\Box(\Diamond\phi_1 \wedge \cdots \wedge \Diamond\phi_n) \tag{7.33}$$

Note that joint satisfaction of the goals is not required. This is a very general requirement to place on a system. It says nothing of how or when the goals are to be achieved. What more specific constraints might we place on system behaviour? One possibility is to specify that no agent should act to make any goal unachievable.

$$A\Box \bigwedge_{i \in \{1,\ldots,n\}} \forall x \cdot (\mathsf{WeakGoal}\ x\ \phi_i) \tag{7.34}$$

This formula is in one sense a *safety* property, since on one reading it says that each agent should not do something bad (i.e., make a goal unachievable). How do we specify that agents should act *positively* to achieve goals? One possibility is to specify that coalitions should form wherever possible in order to achieve goals.

$$A\Box \bigwedge_{i \in \{1,\ldots,n\}} \forall x \cdot (\mathsf{Can}\ x\ \phi_i) \ \Rightarrow\ (\mathsf{Commit}\ x\ \phi_i) \tag{7.35}$$

This specification can be extended to express the idea that agents should commit to goals everywhere that such commitment does not make any other goal unachievable.

$$A\Box \bigwedge_{i \in \{1,\ldots,n\}} \forall x \cdot \left[\begin{array}{l} (\mathsf{Can}\ x\ \phi_i) \wedge \\ ((\mathsf{Commit}\ x\ \phi_i) \ \Rightarrow \\ \quad \bigwedge_{j \in \{1,\ldots,n\}} E\Diamond\phi_j) \end{array} \right] \ \Rightarrow\ (\mathsf{Commit}\ x\ \phi_i) \tag{7.36}$$

7.5 Summary

Whereas earlier chapters have focussed exclusively on linear time temporal logics, this chapter has investigated the use of branching time temporal logics for multi-agent systems. The idea behind the semantics of such a logic is to collect all the runs of a system together into a "computation tree", which can be incorporated into a model for the logic, in much the same way as a run was used in a model for a linear time logic.

Two branching time logics were developed. The first, BAL, is the simpler of the two, and is effectively AL augmented by path quantifiers. The second, QBAL, is based on QAL, but includes operators for describing the abilities, goals, and structure of agents and groups of agents.

This concludes the main text of the thesis. The final chapter presents some conclusions.

Part IV

Conclusions

Chapter 8

Conclusions

VLADIMIR: That passed the time.
ESTRAGON: It would have passed in any case.

Samuel Beckett (Waiting for Godot)

THIS chapter concludes the main text of the thesis by first reviewing the work that has been carried out, then attempting to put this work into context, and finally providing some pointers to possible future lines of research.

8.1 Review

The first part of the thesis contained a detailed literature survey. This survey began, in Chapter 2, by posing the question: What is an agent? It was shown that several naive attempts to define an agent, either in terms of action or in terms of high-level cognitive function, were problematic. This discussion led to the recognition of an agent as an intentional system: one that is most appropriately described in terms of the intentional notions (belief, desire, …). This prompted a detailed analysis of various logic-based formalisms developed by researchers in AI, computer science, philosophy etc. for reasoning about intentional notions. The dominant approach was found to be the use of normal modal logics with possible worlds semantics. However, some objections to the basic possible worlds model were found and discussed. A number of variations on the possible worlds theme were then described. However, for several reasons, this model was not found to be a suitable one for the purposes of this thesis. One alternative formalism, the deduction model of belief developed by Konolige, was found to be a good model of the beliefs of AI systems.

Chapter 2 concluded with an examination of the various approaches taken within the AI community to the problem of building intelligent agents.

Chapter 3 examined the wider issues of social agency. It began by looking at the role of communication and the various ways that communication has been treated in DAI. This led to the recognition of speech act theories as being the dominant paradigm for reasoning about communication in DAI. A detailed review of various speech act theories was then presented. Chapter 3 concluded with a review of the various ways that DAI has addressed the issue of building intelligent social agents.

The main contribution of the thesis was presented in Chapters 4 through 7. Chapter 4 presented a formal theory of computational multi-agent systems. The basic components of this theory, agents, were modelled as having three important attributes:

- agents maintain beliefs, in the style of the deduction model proposed by Konolige [100];
- agents have internal, computational resources upon which they may draw, by performing internal, cognitive actions;
- agents can affect each other's cognitive state by sending messages.

Two execution models for multi-agent systems were then presented, which represent models of how the agents in a system may act and interact. The first, simpler of the two models assumes synchronous action, where every agent is assumed to act simultaneously. The second, more realistic model allows for interleaved execution, where at

most one agent is considered to act at any one time. Some difficulties with the interleaved model were identified, and the remainder of the thesis dealt with synchronous action only.

Chapter 5 then developed various linear time temporal logics for reasoning about systems modelled by the theory developed in Chapter 4. The simplest of these logics was called AL (for "Agent Logic"). AL is a propositional temporal logic containing operators for describing the beliefs, actions, and messages of agents. Its properties were discussed in detail. A version of AL called IAL (for "Internal AL") was then developed, which was more suited to use as an internal language (i.e., a knowledge representation language). Finally, a quantified version of AL called QAL (for "Quantified AL") was developed.

Chapter 6 presented five detailed case studies, showing how the logics developed in Chapter 5 could be used for reasoning about systems. First, it was shown how some properties of systems implemented using AGENT0 or Concurrent METATEM could be captured using the logics. A number of specification examples were then presented, culminating in a formalization of the contract net protocol.

Chapter 7 developed a number of branching time temporal logics which could be used to reason about multi-agent systems. The simplest of these, called BAL (for "Branching AL") is based on AL, but contains the operators of an expressive branching time logic called CTL^*. It was then shown how the semantic structures that underly branching time logics could be exploited to allow the attribution of goals and ability to agents and groups of agents. A quantified version of BAL (called QBAL) was then developed, which includes operators for describing abilities and goals.

8.2 The Work in Context

It is never easy to assess how a piece of academic research stands in relation to other work, even with the benefit of hindsight. Such assessments are particularly hard for the author of the work, without the benefit of hindsight. Nevertheless, this section will attempt to consider how the work described in this thesis relates to other work in DAI and related areas.

First, consider the model of agents and multi-agent systems. Very little work in AI/DAI has addressed the question of developing formal architectural models of agents, (though as Chapters 2 and 3 demonstrated, there has been much work on agent architectures generally, and on modelling aspects of agency; for example belief and desire). The author is aware of no similar work on modelling explicitly computational multi-agent systems.

Now consider the logics developed in Chapters 5 and 7. These logics take as their centrepiece the deduction model of belief developed by Konolige [100]. In his closing comments, Konolige observes that one line of future research for the deduction model would be its integration with other aspects of an agent's cognitive makeup [100, p157]: this work can therefore be seen as extending the deduction model in this way. The work also extends the deduction model into the temporal dimension.

Linguistically, the logics are superficially similar to the logic developed by Cohen and Levesque [29]. However, there are actually many differences. The most obvious of these is that Cohen and Levesque adopt an ungrounded possible worlds semantics for beliefs and goals; we have been at pains not to do this here. The motivation for Cohen and Levesque's work was quite different to ours. They were concerned with developing theories of mentalistic aspects of human behaviour (i.e., intention). Our work was not concerned with humans at all, but with modelling computational systems.

Finally, the general technique of developing a model of agents and multi-agent systems, and then letting a run of such a system act as a model for a temporal logic, was adapted from the Pnuelian tradition in mainstream computer science [135]. The approach is, so far as the author is aware, novel in DAI circles, and seems both appropriate and useful.

8.3 Future Work

The scope of the work described in this thesis has necessarily been broad. The work has raised a number of issues in a wide range of areas. This section will attempt to highlight some of these issues.

Begin with the theory of multi-agent systems. As it stands, this theory is quite *coarse grained*, in two senses. First, the model of agents developed in Chapter 4 is coarse grained. For example, agents are assumed to have a belief revision function, but the properties of this function are hardly examined at all. A fruitful area for future work would be in developing finer-grained formal models of real computational agents. It is often claimed that AI is becoming an unnecessarily formal field, and yet surprisingly few attempts have been made to develop formal

models of real AI systems[1].

Second, the models of concurrency employed in the theory of multi-agent systems are poor. If one is ever to realistically claim that one has proved a property of a real multi-agent system, then the issue of concurrency, (and the whole can of worms called *fairness* [63]), must be addressed.

Moving on to the logics developed in the thesis, there are a number of obvious areas for future work. The first of these is completeness. The axiom systems presented for the various logics in the AL family are all sound, but are not shown to be complete. This is an important issue, which must be addressed if the logics are ever to be used by a wider audience[2].

A related issue is automation of the logics. Of course, automation of classical logics is very much an ongoing research area, but this issue must ultimately be addressed if the logics are to be put in general use. Even if the logics are never used as knowledge representation formalisms themselves, there is no reason to suppose that automated versions of the logics could not be used for reasoning *about* systems, by designers.

Finally, another area for future work would be attempting to unify the logics developed in Chapter 7 with, for example, more abstract models of cooperation and cooperative problem solving systems developed using Cohen and Levesque's formalism [66], [113], [91].

[1] Note that even if the alternative models of agency discussed at the close of Chapter 2 do finally come to usurp the classical AI model of agency, it seems likely that some formal tools will be required to reason about them.

[2] There is a limit to what can be done here, however: first-order temporal logic is not finitely axiomatizable [1].

Part V

Appendices and Bibliography

Appendix A

Notation

THE techniques used to present the mathematical parts of the theory of multi-agent systems, etc., are loosely based on the VDM specification language: the first seven chapters of [93] cover all the necessary material. However, anyone familiar with basic set notation and logic should have no difficulty.

A central idea is that of the *type*: for the purposes of this thesis, a type can be thought of as a (possible infinite) set. Five "standard" types are assumed:

$\mathbb{N} = \{0, 1, 2, \ldots\}$	the natural numbers
$\mathbb{N}_1 = \{1, 2, 3, \ldots\}$	the natural numbers greater than 0
$\mathbb{Z} = \{\ldots, -1, 0, 1, 2, \ldots\}$	the integers
$\mathbb{B} = \{\mathsf{true}, \mathsf{false}\}$	the truth values
$\mathbb{R}$	the real numbers

The set of subsets of a type T (i.e., the powerset of T) is given by $\mathsf{powerset}\ T$.

The type T of all ordered n-tuples whose first element is of type T_1, second element is of type T_2, …, n^{th} element is of type T_n, may be specified in two ways. The simplest method is to define it as the cross product of its component types:

$$T = T_1 \times \cdots \times T_n.$$

The alternative is to name the elements:

$$\langle t_1, \cdots, t_n \rangle$$

where $t_1 \in T_1$, $t_2 \in T_2$, …, $t_n \in T_n$. In general, the latter method is used for large or complex types.

A restriction on a type is called an *invariant*, and is generally specified by a first-order formula that the type must satisfy. The general form of an invariant for type T is:

$$\forall t \in T \cdot P(t)$$

where P is the property that each element of type T must satisfy. For example, the type T of all natural numbers divisible by 3 may be specified by

$$T = \mathbb{N}$$

where T must satisfy the invariant

$$\forall t \in T \cdot (t \ \mathit{mod}\ 3) = 0.$$

Following the VDM convention, type names are chosen to be mnemonic, and begin with a capital letter, followed by a sequence of lowercase letters.

Functions are specified by giving their *signature* and a direct definition. The general form of function definitions is:

$$f : D_1 \times \cdots \times D_n \to R$$
$$f(d_1, \ldots, d_n) \quad \triangleq \quad \ldots$$

This says that the function f takes n arguments, the first of type, (or *from domain*) D_1, …, the n^{th} from domain D_n, and returns a value of type (or *from the range*) R. A direct definition describes how the value is to be obtained from the arguments. Direct definitions of boolean functions are given as a first-order formula: if the arguments of the function satisfy the formula, the result is true, otherwise it is false.

For example, here is the definition of a function that takes a set of natural numbers as its sole argument, and returns true if every element of its argument is even, false otherwise.

$$all\_even : \mathsf{powerset}\,\mathbb{N} \to \mathbb{B}$$
$$all\_even(ns) \quad \triangleq \quad \forall u \in ns \cdot ((u \; mod \; 2) = 0)$$

The symbol

$$\triangleq$$

can be read “is defined as”.

Function names begin with a lowercase letter, to distinguish them from types, and may contain the underscore character (“_”).

A *map* can be thought of as a partial many-to-one function, or, extensionally, as a set of *maplets*, written:

$$\{d_1 \mapsto r_1, \cdots, d_n \mapsto r_n\}$$

Maps are specified in the same way as functions, except that the map arrow (“ $\xrightarrow{m}$ ”) is used instead of the function arrow (“$\to$”). So, the following specifies a type which maps a natural number to a set of integers.

$$T = \mathbb{N} \xrightarrow{m} \mathsf{powerset}\,\mathbb{Z}$$

A *bijective* map is one that is both one-to-one and onto. Bijective maps are specified using the bijective map arrow, “ $\xleftrightarrow{m}$ ”.

A useful operator is the *map overwrite*, “†”. Its use is best illustrated by example. Suppose m_1 and m_2 are maps:

$$m_1 = \{1 \mapsto 3, 2 \mapsto 4\}$$
$$m_2 = \{1 \mapsto 5\}$$

Then

$$m_1 \dagger m_2 = \{1 \mapsto 5, 2 \mapsto 4\}$$

Map application is denoted in the same way as function application: for the previous example, $m_1(1) = 3$ and $m_2(1) = 5$.

Two standard functions are assumed: dom takes a map or function and returns the set of elements in its domain for which it is defined; rng takes a map or function and returns the set of elements in its range for which the corresponding elements in the domain is defined. For the above example:

$$\mathsf{dom}\, m_1 = \{1, 2\} \quad \mathsf{dom}\, m_2 = \{1\}$$
$$\mathsf{rng}\, m_1 = \{3, 4\} \quad \mathsf{rng}\, m_2 = \{5\}$$

Map names follow the convention of function names.

A *sequence* can be thought of simply as a list of objects, each of which is of the same type. The type T, of sequences of type T_1 is specified by:

$$T = T_1^*.$$

A sequence may be listed in square brackets (“[”, “]”). The n^{th} element of a sequence is picked out by indexing. So for some sequence s, the k^{th} element is $s(k)$. Sequences may be infinite in length, (unlike in VDM, where sequences must generally be finite), in which case they have a length of ω (where ω is the first infinite ordinal). The length of a sequence s is given by $\mathsf{len}\, s$.

For example, here is a sequence:

$$s = [1, 2, 4, 8]$$

Then

$$\mathsf{hd}\, s = 1 \quad s(0) = 1$$
$$\mathsf{len}\, s = 4 \quad s(2) = 4$$

This thesis deals with several languages: the symbols φ, ϕ, and ψ are used as meta-variables, ranging over the formulae of these languages. The symbols Δ and Γ are used as meta-variables ranging over sets of formulae of these languages. $Form(L)$ denotes the set of all formulae of a language L. Where L is a first-order language, a *sentence* of L is assumed to be a closed formula, i.e., one with no free variables (where "free" has its usual meaning). So, for some language L, formula ϕ of L, and set of formulae Δ of L, we have: $\phi \in Form(L)$, $\Delta \subset Form(L)$, and $\Delta \in \text{powerset } Form(L)$.

Formal logic proofs are presented in the usual way: the abbreviation PL is used to denote simple but tedious steps of propositional reasoning, PRED is used to indicate predicate logic reasoning, TL is used to indicate temporal logic reasoning, and TAX is used to denote the introduction of a temporal axiom. All formal proofs are closed by the symbol "$\dashv$".

Appendix B

Technical Reviews

THIS appendix presents detailed technical reviews of various formalisms, that were held over from the main text of the thesis due to their complexity. Three formalisms are described: the situated automata paradigm [142], Cohen and Levesque's logic [29], and Werner's formalism [179], [182].

B.1 Situated Automata

It was pointed out in Chapter 2 that many people regard the ontological status of possible worlds and epistemic alternatives as problematic. Do worlds exist? If so, where are they? Few researchers in AI would take an *extreme realist* view, that possible worlds actually do exist, and are worlds like our own; but even the *moderate realist* view, that worlds don't have to exist, but are a useful abstraction for theorizing, is a contentious one. A solution is to provide some method for *grounding* epistemic alternatives: giving them a precise meaning in real-world terms. One method for doing this in distributed systems was discussed in Chapter 2, where a node was said to know ϕ if ϕ was true in all indistinguishable runs of a system. An alternative, but essentially equivalent grounding has been described by Rosenschein, in his *situated automata* paradigm [142], [143]. This is basically an analysis of the information implicit in the state of an automaton. The situated automata paradigm is the object of study in this section.

First, a definition of what an automaton is. An automaton, or machine, M, is a structure:

$$\langle S, \Sigma, A, \delta, \lambda, s_0 \rangle$$

where

- S is a set of *states*;
- Σ is a set of *stimuli*, or *inputs*;
- A is a set of *output actions*;
- $\delta: S \times \Sigma \rightarrow S$ is a *next state function*;
- $\lambda: S \rightarrow A$ is an *output function*;
- $s_0 \in S$ is an *initial state*.

The automaton will interact with its environment by sending outputs, (which intuitively change the state of the environment), and the environment then providing some stimulus $\sigma \in \Sigma$, which, taken together with the machine's internal state $s \in S$, causes a state transition through δ, defining the next output, and so on.

Let Φ be the set of *world conditions*; things that can be true of the world. A *world state* is a $\langle$*environment-state, machine-state*$\rangle$ pair. If w is a world state, then denote by $\phi(w)$ the fact that condition ϕ holds of w. Assume that there is some distinguished world condition ϕ_0, which corresponds to the *strongest condition the world must satisfy* when the machine is in its initial state s_0. The idea is that this property is guaranteed to hold at the "start" of a run. Note the distinction between a world, w, which is an environment state/machine state pair, the fact that $\phi(w)$ holds of world w, and the state of a machine. These are distinct notions.

Next, a function is defined for every state $\sigma \in \Sigma$, which maps Φ to Φ. This function is called the *strongest post-condition* function. The idea is that ϕ/σ is the most specific world condition that can be guaranteed to hold at time t' given that ϕ holds at t and the stimulus σ is sensed over the time interval $[t, t']$.

The "/" operator can be extended to *sequences* of stimuli. As usual, let Σ^* be the set of finite sequences of elements of Σ, (i.e., the set of strings over the alphabet Σ). Let Λ be the null sequence, let $\overline{\sigma}$ be some element of Σ^*, and ";" be the concatenation operator. Then extend the function as follows:

$$\begin{aligned} \phi/\Lambda &\triangleq \phi \\ \phi/(\sigma;\overline{\sigma}) &\triangleq (\phi/\sigma)/\overline{\sigma} \end{aligned}$$

Now an automaton will not necessarily be able to distinguish between *all* world conditions. It therefore makes sense to define what is *knowable* for the automaton. These are the things that, given some sequence of inputs, the automaton will be able to distinguish; unknowable things never have any impact on the state of the automaton.

$$\hat{\Phi} \triangleq \{\phi_0/\overline{\sigma} \mid \overline{\sigma} \in \Sigma^*\}$$

The important point is that the states of an automaton divide the set Σ^* into equivalence classes. In general, automata with coarse equivalence classes will not be able to distinguish as much about the world as those with fine equivalence classes.

Next, for each state s of the automaton, there is a set of input sequences, or language, L_s, each element of which is guaranteed to leave the automaton in state s when it started in state s_0. L_s is defined in two parts; first, δ is extended to sequences.

$$\begin{aligned} \overline{\delta}(s,\Lambda) &\triangleq s \\ \overline{\delta}(s,\overline{\sigma};\sigma) &\triangleq \delta(\overline{\delta}(s,\overline{\sigma}),\sigma) \end{aligned}$$

Then L_s is defined as follows.

$$L_s = \{\overline{\sigma} \in \Sigma^* \mid \overline{\delta}(s_0,\overline{\sigma}) = s\}$$

Now the *condition of a language* is the strongest condition guaranteed to hold after any input sequence contained in the language (when the automaton starts in its initial state). The condition of a language L is denoted $\phi(L)$. So after any sequence in the language L is "fed" to the automaton, the condition $\phi(L)$ can be guaranteed to hold, and nothing else can be guaranteed. The condition of a language is defined as follows.

$$\phi(L) \triangleq \bigvee_{\overline{\sigma} \in L} (\phi_0/\overline{\sigma})$$

From this definition, it is possible to characterize the *information content* of a state as those things which can be guaranteed to hold in the world, given that the automaton is in the state.

$$\mathit{info}(s) \triangleq \phi(L_s)$$

It is then possible to define an epistemic accessibility relation R. Let w and w' be worlds. Then let $\mathit{state}(w)$ denote the state of the automaton in world w. Then the relation R can be defined as follows:

$$(w,w') \in R \text{ iff } \mathit{state}(w) = \mathit{state}(w')$$

Note that R will be an equivalence relation.

Now let us write $w \models \phi$ if ϕ follows from $\mathit{info}(\mathit{state}(w))$. The semantics of a knowledge operator K can then be given as follows

$$w \models \mathsf{K}\phi \quad \text{iff } w' \models \phi \text{ for all } w' \text{ such that } (w,w') \in R$$

Note that since R is an equivalence relation, the logic of K will be S5 (see [28]).

This concludes the review of situated automata.

B.2 Cohen and Levesque's Formalism

In Chapter 2, Cohen and Levesque's formalism for reasoning about rational agents was briefly described. The syntax and semantics of the formalism are, unfortunately, rather complex: for this reason they have been held over to this appendix. The following two sections define the syntax and semantics of the logic, and some of its properties. The material in this section is adapted and summarized from [29]. It was originally intended to also review the work of [66], [113], and [91], but this review has been omitted due to space restrictions.

⟨*action-var*⟩	: : =	$\alpha, \alpha_1, \ldots$	
⟨*agent-var*⟩	: : =	$x, x_1, \ldots$	
⟨*regular-var*⟩	: : =	$i, i_1, \ldots$	
⟨*variable*⟩	: : =	⟨*action-var*⟩	|
		⟨*agent-var*⟩	|
		⟨*regular-var*⟩	
⟨*pred*⟩	: : =	(⟨*pred-symbol*⟩ ⟨*variable*⟩$_1, \ldots,$ ⟨*variable*⟩$_n$)	
⟨*wff*⟩	: : =	⟨*pred*⟩	|
		$\neg$ ⟨*wff*⟩	|
		⟨*wff*⟩ $\vee$ ⟨*wff*⟩	|
		$\exists$⟨*variable*⟩ $\cdot$ ⟨*wff*⟩	|
		(⟨*variable*⟩ = ⟨*variable*⟩)	|
		(HAPPENS ⟨*action-exp*⟩)	|
		(DONE ⟨*action-exp*⟩)	|
		(AGT ⟨*agent-var*⟩ ⟨*action-var*⟩)	|
		(BEL ⟨*agent-var*⟩ ⟨*wff*⟩)	|
		(GOAL ⟨*agent-var*⟩ ⟨*wff*⟩)	|
		⟨*time-prop*⟩	|
		⟨*action-var*⟩ $\leq$ ⟨*action-var*⟩	
⟨*time-prop*⟩	: : =	⟨*numeral*⟩	
⟨*action-exp*⟩	: : =	⟨*action-var*⟩	|
		⟨*action-exp*⟩; ⟨*action-exp*⟩	|
		⟨*action-exp*⟩" | "⟨*action-exp*⟩	|
		⟨*wff*⟩?	|
		⟨*action-exp*⟩*	

Figure B.1: Syntax of Cohen and Levesque's Formalism

B.2.1 Syntax

The logic of rational agency is a many-sorted first-order multi-modal logic, with four key modalities: BEL (for belief), GOAL (for goals), HAPPENS (for what happens next), and DONE (for what has just happened). The logic allows direct reference to time, (by including times as terms in the language), and the more usual relative reference to time found in normal temporal modal logics. The syntax of the logic is defined in BNF form in Figure B.1.

Reference to time is assumed through the use of numerals; however, Cohen and Levesque allow for times such as "2.30PM, 3rd July 1992"; such time references have an obvious interpretation. The ordering relation "$\leq$" is for actions. The action constructors, defined by the final production, have the following meaning:

$\alpha; \alpha'$	α followed by a α'
$\alpha \mid \alpha'$	α or α', non deterministically
$\alpha?$	test action
α^*	iterative action

These pseudo-dynamic logic constructs provide an effective method for talking about action sequences.

B.2.2 Semantics

The basic semantic components of the language are worlds, which are sequences of events, stretching infinitely into both past and future. Time is discrete; the sequences are therefore indexed by the integers, $\mathbb{Z}$. Belief and goal accessibility relations are defined, indexed by each agent, on the set of worlds.

Assume E is a set of primitive event types. The type for worlds is then as follows:

$$World = \mathbb{Z} \rightarrow E$$

A model for the logic is then a structure:

$$\langle \Theta, P, E, Agt, T, B, G, \Phi \rangle$$

$\langle M, \sigma, v, n \rangle$	$\models$	$P(x_1, \ldots, x_k)$	iff $\langle v(x_1), \ldots, v(x_k) \rangle \in \Phi(P, \sigma, n)$
$\langle M, \sigma, v, n \rangle$	$\models$	$\neg \phi$	iff $\langle M, \sigma, v, n \rangle \not\models \phi$
$\langle M, \sigma, v, n \rangle$	$\models$	$\phi \vee \psi$	iff $\langle M, \sigma, v, n \rangle \models \phi$ or $\langle M, \sigma, v, n \rangle \models \psi$
$\langle M, \sigma, v, n \rangle$	$\models$	$\exists x \cdot \phi$	iff $\langle M, \sigma, v \dagger \{x \mapsto d\}, n \rangle \models \phi$ for some $d \in D$
$\langle M, \sigma, v, n \rangle$	$\models$	$(x_1 = x_2)$	iff $v(x_1) = v(x_2)$
$\langle M, \sigma, v, n \rangle$	$\models$	$\langle \textit{time-prop} \rangle$	iff $v(\langle \textit{time-prop} \rangle) = n$
$\langle M, \sigma, v, n \rangle$	$\models$	$(e_1 \leq e_2)$	iff $v(e_1)$ is an initial sub-sequence of $v(e_2)$
$\langle M, \sigma, v, n \rangle$	$\models$	(AGT x e)	iff $AGT(v(e)) = \{v(x)\}$
$\langle M, \sigma, v, n \rangle$	$\models$	(HAPPENS α)	iff $\exists m \in \mathbb{Z} \cdot (m \geq n) \wedge \langle M, \sigma, v, n [\![\alpha]\!] m \rangle$
$\langle M, \sigma, v, n \rangle$	$\models$	(DONE α)	iff $\exists m \in \mathbb{Z} \cdot (m \leq n) \wedge \langle M, \sigma, v, m [\![\alpha]\!] n \rangle$
$\langle M, \sigma, v, n \rangle$	$\models$	(BEL x ϕ)	iff $\forall \sigma' \in \textit{World} \cdot \langle \sigma, v(x), n, \sigma' \rangle \in B \Rightarrow \langle M, \sigma', v, n \rangle \models \phi$
$\langle M, \sigma, v, n \rangle$	$\models$	(GOAL x ϕ)	iff $\forall \sigma' \in \textit{World} \cdot \langle \sigma, v(x), n, \sigma' \rangle \in G \Rightarrow \langle M, \sigma', v, n \rangle \models \phi$

Figure B.2: Semantics of Cohen and Levesque's Formalism — Part 1

where

- Θ is a set of things, (i.e., a universe of discourse);
- P is a set of people/agents;
- E is a set of primitive event/action types;
- $Agt: E \rightarrow P$ gives the agent of an event;
- $T \subseteq \textit{World}$ is a set of worlds;
- $B \subseteq T \times P \times \mathbb{Z} \times T$ is the belief accessibility relation, and is assumed to be euclidean, transitive, and serial;
- $G \subseteq T \times P \times \mathbb{Z} \times T$ is the goal accessibility relation: it is assumed that G is serial, and $G \subseteq B$;
- Φ interprets predicates.

The domain of quantification, D, is defined:

$$D = \Theta \cup P \cup E^*.$$

So it is possible to quantify over people, things and sequences of primitive event types. Also, the function *AGT* is assumed to give the agents of a sequence of events; this function can be derived from *Agt*.

The semantics of the language are defined via the satisfaction relation "$\models$", as usual. The relation holds between structures of the form:

$$\langle M, \sigma, v, n \rangle$$

(where M is a model, σ is a sequence of events, (i.e., a world), v is a variable assignment, and $n \in \mathbb{Z}$ is a temporal index), and formulae of the language. The semantics are given in two parts: Figure B.2, and Figure B.3.

The rules in Part 1 define the semantics of the modal operators, etc. The rules in Part 2 are somewhat unusual, and define the semantics of action constructors. Note the use of the $[\![\_]\!]$ construct: $n [\![\alpha]\!] m$ means that α occurs *between* time points n and m. This construct is itself defined in terms of satisfaction.

$$
\begin{array}{ll}
\langle M, \sigma, v, n\,[\![e]\!]\,n+m\rangle & \text{iff } v(e) = e_1, e_2, \ldots, e_m \wedge \sigma(n+i) = e_i, 1 \le i \le m \\
\langle M, \sigma, v, n\,[\![\alpha \mid \alpha']\!]\,m\rangle & \text{iff } \langle M, \sigma, v, n\,[\![\alpha]\!]\,m\rangle \text{ or } \langle M, \sigma, v, n\,[\![\alpha']\!]\,m\rangle \\
\langle M, \sigma, v, n\,[\![\alpha; \alpha']\!]\,m\rangle & \text{iff } \exists k \in \mathbb{Z} \cdot (n \le k \le m) \wedge \langle M, \sigma, v, n\,[\![\alpha]\!]\,k\rangle \wedge \\
 & \qquad \langle M, \sigma, v, k\,[\![\alpha']\!]\,m\rangle \\
\langle M, \sigma, v, n\,[\![\alpha?]\!]\,n\rangle & \text{iff } \langle M, \sigma, v, n\rangle \models \alpha \\
\langle M, \sigma, v, n\,[\![\alpha^*]\!]\,m\rangle & \text{iff } \exists n_1, \ldots, n_k \in \mathbb{Z} \cdot (n_1 = n) \wedge (n_k = m) \wedge \\
 & \qquad \forall i \in \mathbb{Z} \cdot (1 \le i \le m) \;\Rightarrow\; \langle M, \sigma, v, n_i\,[\![\alpha]\!]\,n_{i+1}\rangle
\end{array}
$$

Figure B.3: Semantics of Cohen and Levesque's Formalism — Part 2

$$
\begin{array}{lcl}
\text{IF } \phi \text{ THEN } \alpha \text{ ELSE } \alpha' & \triangleq & \phi?; \alpha \mid \neg\phi?; \alpha' \\
\text{WHILE } \phi \text{ DO } \alpha & \triangleq & (\phi?; \alpha)^*; \neg\phi? \\
\Diamond\phi & \triangleq & \exists x \cdot (\text{HAPPENS } x; \phi?) \\
\Box\phi & \triangleq & \neg\Diamond\neg\phi \\
(\text{DONE } x\ \alpha) & \triangleq & (\text{DONE } \alpha) \wedge (\text{AGT } x\ \alpha) \\
(\text{HAPPENS } x\ \alpha) & \triangleq & (\text{HAPPENS } \alpha) \wedge (\text{AGT } x\ \alpha) \\
(\text{LATER } \phi) & \triangleq & \neg\phi \wedge \Diamond\phi \\
(\text{BEFORE } \phi\ \psi) & \triangleq & \forall c \cdot (\text{HAPPENS } c; \psi?) \;\Rightarrow\; \exists\alpha \cdot (\alpha \le c) \wedge \\
 & & \qquad (\text{HAPPENS } \alpha; \phi?)
\end{array}
$$

Figure B.4: Derived Operators for Cohen and Levesque's Formalism

B.2.3 Derived Operators and Properties of the Logic

A number of derived constructs are possible for the logic; for example, the empty action nil and empty sequence NIL; the empty event sequence is a subsequence of every other sequence. Some other derived constructs are defined in Figure B.4. These derived operators have fairly obvious interpretations; the derived versions of DONE and HAPPENS simply associate the last/next action with an agent. LATER is a strict future "sometime" operator. BEFORE allows the temporal order of events to be described.

Cohen and Levesque do not define an axiom system in the normal way, but study the validities of their logic instead. Some of these validities are obvious: for example, axioms *K45D* for belief, and axiom *D* for goals. Necessitation works for beliefs and goals. Another important property is the following:

$$\models (\text{BEL } x\ \phi) \;\Rightarrow\; (\text{GOAL } x\ \phi)$$

This, perhaps unusual looking, axiom is a consequence of an agent's goal accessibility relation being a subset of its belief accessibility relation.

This concludes the review of Cohen and Levesque's formalism.

B.3 Werner's Formalism

In an extensive sequence of papers published since 1988, Eric Werner has described a formalism for reasoning about multi-agent systems which draws upon work in mathematical logic, game theory, and linguistics [178], [176], [177], [179], [180], [181], [182], [183]. This section will describe the basic components of the formalism, including his model of social groups and social structure, and his language LT $\Box$CAN (later renamed CANPLAN). The material described here is mainly taken from [179] and [182].

> "Let $P, P_1, \ldots$ be *properties*, and $R^n, R^n_1, \ldots$ be n-ary *relations* for $n \ge 2$. Let O be the set of *individuals*. Let T be the set of *time instants* ordered by a relation $<$. The elements of T will be denoted by t, t', t_1, etc. Let TP be the set of *time periods*. The elements of TP will be denoted by τ, τ', etc.
>
> A *situation* s is a set of objects in O that have properties P and that stand in various relations R^n to one another. For example,
>
> $s = \{Pa; yes.\ Qa; no.\ R(a, b); yes.\}$

is a situation where property P holds of a, a does not have the property Q, nothing is said about b's properties, and a is related to b by the relationship R. Let *Sits* be the set of all possible situations.

A state σ is a complete situation where all properties and relations are specified. Let Σ be the set of all states. *Histories* or *worlds* are series of states. More formally, a *history* H is a function from times T to the set of states Σ. Let H_t be the value of the function H at t. …Let Ω be the set of all possible *histories* or *worlds* H. Let H^t be the *partial history* of H up to and including the time $t \in T$. Let $Hist(\Omega)$ be the set of possible *partial histories* H^t for all times $t \in T$. Partial histories H^t are then partial functions where $H^t_{t_0}$ is the state of the partial history at time t_0. *Events* are situations over a time period τ, i.e., events are functions taking times $t \in \tau$ to situations $s \in Sits$. *Actions* a are events with agents given by the function $Agent(a)$. …An event e is *realized* in H at τ if e is contained in H and the domain of e is time period τ. …

An *information state* I is a set of partial histories that are possible given the available information. Each information set I has an associated set of alternatives or choices $Alt(I)$. Each *alternative* is a set of possible histories leaving I. …The possible information states of an agent A in the environment Ω form an *information partition* Ξ_A. A *strategy* [or plan] is a function from information states I to the alternatives at I. [The potential] π^* [of strategy π] is the set of possible histories consistent with the strategy π. …

The *cognitive* or *representational state* R of an agent is described by three components

$$R = \langle I, S, V \rangle.$$

I is the *information state* of the agent. S is the *intentional state* of the agent. S is a set of possible strategies that guide the actions of the agent. V is the *evaluative state* of the agent. V represents the agent's evaluation and focus on situations. The representational state R_A may include the agent A's representation of B's representation, R^B_A". [179, pp11–12]

Before moving on to social groups and social structures, some comments on the formalism so far. First, information states: these seem to be epistemic alternatives in much the standard sense. However, the use of situations is unusual: some comments are made on this later.

Next, to Werner's definition of social groups and social structures. First, a *role* is defined to be an "abstract agent" defined by a particular representational structure…

"that defines the state information, permissions, responsibilities and values of that agent role. When an agent A assumes a role *rol*, he internalizes that role by constraining his representational state R_A to $R_A + R_{rol}$. Positively expressed, when an agent A assumes a role *rol*, he changes his representational state so that R_{rol} becomes a part of R_A". [179, p20]

A *social structure* ΣT is a set of roles $\{rol_1, \ldots, rol_n\}$. A *social group* is a social structure in a particular setting. A social group $\Sigma\Gamma$ is a structure:

$$\langle L, G, \Sigma T, Roles, \Omega \rangle$$

where

- L is a language;
- G is a group of agents;
- ΣT is a social structure;
- $Roles: G \xrightarrow{m} \Sigma T$ maps each member of the group to a role;
- Ω is the set of all possible histories of the environment.

An example social group (Werner's formalization of the contract net [179, pp27–29]) is described in Chapter 6, and contrasted with the formalization of the contract net developed therein.

The next step is to develop a language for describing and reasoning about the systems described by the model of multi-agent systems. Werner defines such a language: in [180] it as called LT $\Box$PLAN, and in [182] it is called CANPLAN. The treatment here is adapted from the latter source.

$P\phi$	Was, or sometime past ϕ (strict)
$F\phi$	Sometime ϕ (strict)
$\Box_A\phi$	It is necessary for agent A that ϕ
$\rhd_A\ \phi$	Agent A plans that ϕ
$\rhd_G\ \phi$	The group G plans that ϕ
$CAN_A\phi$	Agent A can bring about ϕ
$COCAN_A\phi$	Agent A can coordinate with others to achieve ϕ
$COOPCAN_G\phi$	Group G can cooperatively achieve ϕ

Table B.1: Operators in Werner's Language CANPLAN

B.3.1 Syntax of CANPLAN

CANPLAN is a propositional multi-modal logic containing eight basic modalities (see Table B.1). Formulae of CANPLAN are then built up in the obvious way over some set of primitive propositions *Prop*.

B.3.2 Semantics of CANPLAN

The semantics of CANPLAN require some preliminary definitions. Suppose π is a strategy, then the *potential* of that strategy, π^*, is the set of all histories compatible with that strategy. Also, if I is an information state, then the potential I^* of I is the set of all histories compatible with the state. Finally, define $\pi^*[I]$ as follows:

$$\pi^*[I] \triangleq \pi^* \cap I^*.$$

Thus $\pi^*[I]$ is: "the set of worlds allowed by the strategy π given the information I" [182, p8].

Next, it is necessary to introduce the idea of an *information ensemble*. If Ω is the set of all histories of a system, then an information ensemble Ξ for Ω is a set of information sets such that the following conditions hold:

1. For any partial history $H^t \in Hist(\Omega)$, there is an information set $I \in \Xi$ such that $H^t \in I$.
2. For any $I, J \in \Xi$, if $I \neq J$ then $I \cap J = \{\}$.

Clearly, any partial history $H^t \in Hist(\Omega)$ is a member of just one information set in Ξ; call this information set $I(H^t)$. However, it does not follow that all information sets in an information ensemble are singletons; only that partial histories are not duplicated between different information sets in the ensemble.

If H^t is a partial history, then denote by $I(H^t)$ the information set that H^t is a member of, relative to some information ensemble. Each agent is associated with an ensemble, which implicitly defines the "information conditions" ([182]) for the agent at all times: it is possible to define for each information ensemble a time dependent epistemic accessibility relation, which will be used to give the semantics of "informational necessity".

$$(H, K) \in I_t^{\Xi} \text{ iff } K \in I(H^t)$$

Next, let *STRAT* be a function taking an agent and returning the set of all strategies for that agent. Finally, if S is an intentional state, then $S^*(I)$ is the potential of that state, given information I (see [182, pp8-9]).

Models for CANPLAN are structures of the form:

$$\langle \Sigma, T, <, \Omega, Ag, \Xi_i, S_i, \Phi \rangle$$

where

- Σ is the set of all states;
- T is a set of time instants;
- $<$ is an ordering on T;
- Ω is the set of all histories of the system;
- $Ag = \{1, \ldots, n\}$ is a set of agents;
- $\Xi_i, S_i = \Xi_1, S_1, \ldots, \Xi_n, S_n$ are the information and plan ensembles for agents $i \in Ag$;

$\langle M, \sigma \rangle$	$\models$	p	where $p \in Prop$ iff $p \in \Phi(\sigma)$
$\langle M, H, t \rangle$	$\models$	p	where $p \in Prop$ iff $\langle M, H_t \rangle \models p$
$\langle M, H, t \rangle$	$\models$	$P\phi$	iff $\exists t' \in T \cdot (t' < t) \wedge \langle M, H, t' \rangle \models \phi$
$\langle M, H, t \rangle$	$\models$	$F\phi$	iff $\exists t' \in T \cdot (t < t') \wedge \langle M, H, t' \rangle \models \phi$
$\langle M, H, t \rangle$	$\models$	$\Box_A \phi$	iff $\forall K \in \Omega \cdot (H, K) \in I_t^{\Xi_A} \Rightarrow \langle M, K, t \rangle \models \phi$
$\langle M, H, t \rangle$	$\models$	$\rhd_A \phi$	iff $\forall K \in S_A(I_A(H^t))^* \cdot \langle M, K, t \rangle \models \phi$
$\langle M, H, t \rangle$	$\models$	$\rhd_G \phi$	iff $\forall K \in S_G(I_G(H^t))^* \cdot \langle M, K, t \rangle \models \phi$
$\langle M, H, t \rangle$	$\models$	$CAN_A\phi$	iff $\exists \pi \in STRAT_A \cdot \forall K \in \pi^*[I_A(H^t)] \cdot \langle M, K, t \rangle \models \phi$
$\langle M, H, t \rangle$	$\models$	$COCAN_A\phi$	iff $\exists \pi \in STRAT_A \cdot \pi^*[I_A(H^t)] \cap S_A^{Ag}(I_A(H^t))^* \neq \{\} \wedge \forall K \in \pi^*[I_A(H^t)] \cap S_A^{Ag}(I_A(H^t))^* \cdot \langle M, K, t \rangle \models \phi$
$\langle M, H, t \rangle$	$\models$	$COOPCAN_G\phi$	iff $\forall i \in G \cdot \exists \pi_i \in STRAT_i \cdot \cap_{i \in G} \pi_i^*[I_i(H^t)] \neq \{\} \wedge \forall K \in \cap_{i \in G} \pi_i^*[I_i(H^t)] \cdot \langle M, K, t \rangle \models \phi$

Figure B.5: Semantics of Werner's Language CANPLAN

- $\Phi: \Sigma \rightarrow$ powerset *Prop* interprets propositions in states.

The semantic rules for the language are presented in Figure B.5. Semantics are defined via the usual satisfaction relation "$\models$", which holds between structures of the form: $\langle M, \sigma \rangle$ or $\langle M, H, t \rangle$, (where M is a model, $\sigma \in \Sigma$, $H \in \Omega$ and $t \in T$), and formulae of CANPLAN. The first rule defines the satisfaction of propositions with respect to a model and state. The second defines the satisfaction of propositions with respect to a model, time and history. The second rule is defined in terms of the first; recall that if H is a history and t is a time, H_t returns the state of H at time t. The semantic rules for propositional connectives are omitted.

Let us now examine these semantics. The rules for propositions, propositional connectives, and the two temporal operators seem clear enough, with one caveat. Werner uses the notion of a situation extensively in his semantic structures, but then defines an ordinary propositional valuation for states: in other words, states act as possible worlds in the standard modal logic sense. Thus situations turn out to be redundant for the language, since they are not used in any way.

The "informational necessity" operator seems to be a standard modal epistemic knowledge operator. The two "plan" operators are also straightforward: the single agent plan operator captures the idea of an agent's intentional state "forcing" a goal to be true. The group plan operator simply generalizes this definition to a multi-agent case.

The final operators — *COCAN* and *COOPCAN* — are more complex. $COCAN_A\phi$ says that if there is a strategy for A such that there is at least one history consistent with the strategy which coincides with A's representation of the other agent's intentional states, and on all the histories which do coincide, ϕ is achieved.

$COOPCAN_G\phi$ says that the group G can cooperate to achieve ϕ. Semantically, this will be true if the group have "nonconflicting strategies" which ensure the outcome of ϕ.

This concludes the review of Werner's formalism.

Appendix C

Temporal Logic and Reactive Systems

SINCE Amir Pnueli published his landmark 1977 paper on the use of temporal logic for reasoning about programs ([133]), an enormous amount of literature has been produced on the subject: a whole thesis in itself could not do justice to the extraordinary wealth of material. In any case, much of the work is not relevant to this thesis. The aim of this appendix is not, therefore, to present a detailed review of the area, but to outline how programs are modelled in the Pnuelian framework, and to point to some major developments in the area. The material in this appendix has been adapted from [47], [78], [135] and [168, Chapter 4]. The reader is assumed to be familiar with the elements of modal logic (see, e.g., [88], [28], [89] or [78]).

C.1 A Plethora of Temporal Logics

There seem to be almost as many different temporal logics as there are people using temporal logics to reason about programs. In this section, we briefly describe some important developments in temporal logics.

The earliest temporal logic studied in detail was the modal logic K_t, corresponding to a modal system with two basic modal operators, (often written "[P]" or "H" — "heretofore", or "always past", and "[F]" or "G" — "henceforth", or "always") [136]. The semantics of these operators are given via two ordering relations R_P and R_F on the set of worlds, such that the two relations are the inverse of each other. The usual "was" and "sometime" operators, (written "$\langle P \rangle$" or "P" and "$\langle F \rangle$" or "F") are defined as the duals of these operators. This logic is discussed at length in [78, Chapter 6].

Although such a system *can* be used to crudely reason about programs, it cannot describe the "fine structure" of the state sequences generated by executing programs (see below). For this reason, a "next time" operator was used by Pnueli in his original proposal [133]. Also, the standard heretofore/always combination of operators was discovered to be inadequate for expressing many properties, and so the basic temporal logic was augmented by the *since* and *until* operators (written "$\mathcal{S}$" and "$\mathcal{U}$"). In his 1968 doctoral thesis, Kamp demonstrated that the logic with since and until operators was *expressively complete* over continuous linear orders [95][1].

The "standard" linear discrete temporal logic — and the one employed for the most part in this thesis — is that based on until/since and next/last. In the propositional case, this logic is decidable, (though the problem is PSPACE complete — see, e.g., [157] for discussions on the complexity of such logics), and is generally regarded as expressive enough to capture many interesting properties of reactive systems. Gough has developed tableaux-based decision procedures for this logic [79]; Fisher has developed resolution methods [59].

In the first-order case, temporal logics containing these operators are not decidable, but this should come as no surprise! Perhaps more worrying is that first-order temporal logic is not finitely axiomatizable [1].

A temporal logic based solely on these operators does have its limitations, however. Sistla *et al* showed that it could not be used to describe an unbounded FIFO buffer [158]. Another disadvantage is that it cannot express the fact that a proposition is true on every even moment.

Many variations on this basic temporal logic theme exist. For example, the use of *fixpoint* operators has been discussed. Banieqbal and Barringer describe a logic which keeps the next operator as basic, but then defines two fixpoint operators from which the other standard operators of linear discrete temporal logic can be derived [10]. Wolper has described a logic ETL, which extends the standard temporal logic with a set of *grammar operators* [186]. Yet another variation is to add a *chop* operator, $\mathcal{C}$, which "composes" two finite sequences [13].

[1] The since/until operators are assumed to be *strict*: the since operator refers strictly to the past, the until operator refers strictly to the future.

All of the logics mentioned above have assumed the model of time is *discrete* (i.e., that for any time point, there exists a time such that no other time point occurs between them) and *linear* (i.e., that each point in time has at most one successor). But these assumptions are not essential: a temporal logic of reals (where the worlds are isomorphic with the real numbers) has been developed; however, such logics are complex and uncommon.

Much more common are *branching time* logics. Briefly, a branching time structure is one where each time point may have a number of "successor" times, each one intuitively corresponding to one way things *could* turn out. The repertoire of operators in linear temporal logic is not sufficient to express all the properties of such structures (despite Lamport's claims in his famous 1980 paper [108]).

The earliest branching time logic to be studied at length was UB (unified system of branching time) [15]. This logic extended linear time temporal logic by the addition of two operators "*A*" ("on all paths …") and "*E*" ("on some path …"), called *path quantifiers*, which could be prefixed to any formula containing at most one occurrence of the usual linear time temporal operators. This introduces an obvious restriction on the expressibility of UB.

UB does not contain an "until" operator; this omission was rectified in a logic CTL (computation tree logic). However, CTL is still not expressive enough to capture many interesting properties of branching time structures. The most expressive branching time logic so far studied in any detail is called CTL^* [48]. This logic allows path quantifiers to be prefixed to arbitrary formulae of the language: intermixing of path quantifiers and standard temporal logic operators is freely allowed, resulting in a highly expressive logic. In the propositional case, all of these branching time logics are known to be decidable: a complete axiomatization of CTL^* is given in [166]. A good general review of branching time logics is provided in [49].

This concludes the overview of temporal logics.

C.2 Program Modelling

To show how temporal logic can be used to reason about a program, we define a simple model of parallel programs; this model is analogous to the model of multi-agent systems developed in Chapter 4. We then show how a first-order linear discrete temporal logic can be used to reason about such a program.

Consider a parallel *program*:

$$P_1 \parallel \cdots \parallel P_k$$

consisting of a finite, fixed, non-zero number of concurrently executing processes $P_i \in \{P_1, \ldots, P_n\}$. These processes are assumed to share the same variables, so that one process may communicate with another by changing the value of a variable. The "memory state" of a program may thus be defined as a tuple of variable values:

$$\langle x_1, \ldots, x_n \rangle.$$

These variables will be called *program variables* to distinguish them from logical variables. Let *mst* be the set of all possible memory states. The "internals" of *mst* depend on what values the variables may take; this is not considered here. Each of the program variables is assumed to have an initial value, which is set at the start of computation.

Each process can be represented as a finite, connected, directed graph. Each node in the graph is associated with a *label*. Each arc $\langle \ell, \ell' \rangle$ in the graph of a process is associated with an *instruction* which may be executed by the process whenever the process is selected for execution and the current state of the process is ℓ. Let *loc* be a function which takes a process and returns the set of all locations for that process. Each process P_i has a distinguished *initial* node, ℓ_i^0, where it "starts execution" from.

An instruction is defined to be an ordered pair, $\langle C, A \rangle$, where C is a condition and A is an assignment operation of the form:

$$\langle x_1, \ldots, x_n \rangle := \langle e_1, \ldots, e_n \rangle$$

where e_i are expressions.

The "execution" of an instruction thus implicitly defines a change in the state of the program variables, and a process moving from one label to another. The purpose of the condition, or "guard" is to define when the instruction can be executed. An instruction with a satisfied condition is said to be enabled.

The state of a program is therefore defined as follows.

$$State = (loc(P_1) \times \cdots \times loc(P_k)) \times mst$$

Now consider an execution of a program to be a (possibly infinite) sequence of states:

$$s_0, s_1, s_2, \ldots$$

where

- in s_0, each process is in its "initial" state, and the variables have their initial values;
- if s_{k+1} exists, for $k \in \mathbb{N}$, then s_{k+1} represents the state resulting from the execution of one processes enabled instruction from state s_k. (Note that this will mean a change to the state of the program variables, dictated by the assignment operation, and a change to the state of one processes label.)

Note that the execution of program instructions is *interleaved*: only one process ever changes the state of the program at any one time. It is widely accepted that the set of all such executions of a program represent all the possible ways execution of the corresponding "real" program could evolve: if something can be shown never to occur in any execution of the program, then it can reasonably be said to have been proved not to occur in the "real" program. Similarly, something true in *all* executions of the program must be true in all runs of the corresponding "real" program.

This simple model of programs can capture many interesting aspects of imperative programs. It is not difficult to derive similar models of systems based on, for example, message passing. Note that there are, however, problems in trying to represent the properties of sophisticated, high-level programming languages using such frameworks.

How can a temporal logic be used to reason about a program? A run of a program is a sequence of discrete states: it is possible to incorporate such a run into the model structure of a first-order temporal logic, and to write formulae of the logic to express properties of the run. In this way, the logic can be used to reason about properties of the program. We do not go into details of first-order temporal logics here; see the cited references above. We simply give an indication of what properties can be described and reasoned about. It is worth pointing out that the set of all computations of a program can be collected together into a branching structure, called a computation tree: a branching time logic can then be used to express properties of such a structure, and hence the program.

So suppose we want to use a linear temporal logic to reason about a program. A predicate $at_i(\ell_j)$ is generally used to represent the fact that the execution of process i has reached label ℓ_j. The value of variables can be described using standard first-order equality.

Suppose it was desired to specify that execution of a process P_i would always proceed from label ℓ to ℓ'. This could be done using the formula:

$$\Box(at_i(\ell) \;\Rightarrow\; \Diamond at_i(\ell')).$$

Similarly, suppose that two processes P_i and P_j have critical regions ℓ_{ci} and ℓ_{cj} respectively. Mutual exclusion could be specified by the following:

$$\Box\neg\,(at_i(\ell_{ci}) \wedge at_j(\ell_{cj})).$$

Many other properties of a program can be specified in a similar way.

Suppose a specification $\mathcal{SPEC}$ has been developed, and a program $\mathcal{PROG}$ has been developed. How might it be possible to *verify* $\mathcal{PROG}$, i.e., to show that it satisfies its specification? The idea is to develop the *temporal theory* $\mathcal{TH}(\mathcal{PROG})$ of $\mathcal{PROG}$, and show that:

$$\mathcal{TH}(\mathcal{PROG}) \vdash \mathcal{SPEC}$$

The temporal theory of the program is usually derived from the text of a program by a transformation process. For example, for each arc, or transition in each processes graph, there will be an axiom in the temporal theory describing the effect of that transition. The effect will be to change the state of the variables and change the state of the program so that at the next time instant, the executed process will be at a new label.

In addition to the program specific axioms in a temporal theory, there will be other axioms capturing more general properties of acceptable execution sequences. For example, there will typically be an axiom of the following form:

$$\Box \bigwedge_{i\in\{1,\ldots,n\}} at_i(\ell) \wedge at_i(\ell') \;\Rightarrow\; (\ell = \ell')$$

which states that a process can only ever be at one label. Other axioms might specify that if a transition occurs, then its guard condition must have held, and so on.

This concludes the review of program modelling.

C.3 Specification and Temporal Logic

This section reviews the properties of a reactive system that may be specified using temporal logic. It is generally accepted that such properties fall into two categories: the so-called *safety* properties, and *liveness* properties[2].

Informally, a safety property can be interpreted as saying that "something bad won't happen". More formally, a safety property states that every finite prefix of a run satisfies some requirement. For obvious reasons, safety properties are sometimes called invariance properties. The simplest kind of safety property is *global invariance*, expressed by a formula of the form:

$$\Box\phi.$$

A *local invariance*, stating that whenever ϕ holds, ψ must hold also, is given by the following formula:

$$\Box(\phi \Rightarrow \psi).$$

Where a system terminates, *partial correctness* may be specified in terms of a precondition ϕ, which must hold initially, a postcondition ψ, which must hold on termination, and a condition φ, which indicates when termination has been reached.

$$init \wedge \phi \Rightarrow \Box(\varphi \Rightarrow \psi)$$

A *mutual exclusion* property is a global invariance of the form:

$$\Box\left(\sum_{i=1}^{n} \phi_i = 1\right).$$

This formula states that at most one of the properties $\phi_i \in \{\phi_1, \cdots, \phi_n\}$ should hold at any one time. (The Σ notation arises if one imagines that truth is valued at 1, falsity at 0; the above formula is read "at most one of $\phi_i \in \{\phi_1, \ldots, \phi_n\}$ is true". Any formula written in Σ-form, (where n is finite), can be expanded into an ordinary formula if required; the Σ-notation may therefore be regarded as an abbreviation.)

A *liveness* property is one that states that "something good will eventually happen". The simplest liveness properties have the form:

$$\Diamond\phi$$

Termination is an example of liveness. The basic termination property is:

$$init \wedge \phi \Rightarrow \Diamond\varphi$$

which states that every run which initially satisfied the property ϕ eventually satisfied the property φ. Here φ is the property which holds when a run has terminated.

A more useful liveness property is *temporal implication*:

$$\Box(\phi \Rightarrow \Diamond\psi)$$

which states that "every ϕ is followed by a ψ".

Responsiveness is a classic example of temporal implication: suppose ϕ represents a "request", and ψ a "response". The above temporal implication would then state that every request is followed by a response. Another example of temporal implication is *total correctness*, which states that a system initially satisfying property ϕ will eventually terminate (given by φ), and on termination will satisfy property ψ.

$$\Box(init \wedge \phi \Rightarrow \Diamond(\varphi \wedge \psi))$$

There are a class of properties which deal with the temporal ordering of events in a system. The simplest example of such a property is the "absence of unsolicited response". If ϕ is a request, and ψ a response, then this property can be given as:

$$\Box(\psi \Rightarrow \blacklozenge\, \phi).$$

A crude way of specifying that requests should be honoured on a first-in, first-out basis is:

$$\Box(\phi_i \mathcal{B}\, \phi_j) \Rightarrow (\psi_i \mathcal{B}\, \psi_j).$$

This concludes the review of properties that may be specified using temporal logic.

[2]The material in this section has been adapted from [47, p1049–1054].

Appendix D

Axioms from Chapter 5

THE axioms listed below are from the logics AL and QAL, developed in Chapter 5.

D.1 Axioms from Section 5.1

The first set are from AL. The list below has been adapted and extended from [116, pp228–234] and [47, pp1003–1005].

The following three rules axiomatize the interaction of the next/last temporal operators.

$$\vdash \quad \bigcirc \neg \phi \Leftrightarrow \neg \bigcirc \phi \tag{D.1}$$

$$\vdash \quad \neg \, \boxdot \phi \Leftrightarrow \bullet \neg \phi \tag{D.2}$$

$$\vdash \quad \bigcirc \bullet \phi \Leftrightarrow \phi \tag{D.3}$$

Some useful forward implications, dealing with future time operators, are:

$$\vdash \quad \phi \Rightarrow \Diamond \phi \tag{D.4}$$

$$\vdash \quad \bigcirc \phi \Rightarrow \Diamond \phi \tag{D.5}$$

$$\vdash \quad \Box \phi \Rightarrow \Diamond \phi \tag{D.6}$$

$$\vdash \quad \phi \, \mathcal{U} \, \psi \Rightarrow \Diamond \psi \tag{D.7}$$

$$\vdash \quad \Box \phi \Rightarrow \phi \tag{D.8}$$

$$\vdash \quad \Box \phi \Rightarrow \bigcirc \phi \tag{D.9}$$

$$\vdash \quad \Box \phi \Rightarrow \bigcirc \Box \phi \tag{D.10}$$

These axioms have fairly obvious past time "mirrors" — there is not a *precise* correspondence, as 1) time is bounded in the past, and 2) past time operators are strict, whereas future time operators are not.

$$\vdash \quad \boxdot \phi \Rightarrow \blacklozenge \phi \tag{D.11}$$

$$\vdash \quad \phi \, \mathcal{S} \, \psi \Rightarrow \blacklozenge \psi \tag{D.12}$$

$$\vdash \quad \blacksquare \phi \Rightarrow \bullet \phi \tag{D.13}$$

$$\vdash \quad \blacksquare \phi \Rightarrow \bullet \blacksquare \phi \tag{D.14}$$

The following axioms show that iterating any of $\{\Diamond, \Box\}$ makes no difference:

$$\vdash \quad \Diamond \Diamond \phi \Leftrightarrow \Diamond \phi \tag{D.15}$$

$$\vdash \quad \Box \Box \phi \Leftrightarrow \Box \phi \tag{D.16}$$

$$\vdash \quad \blacklozenge \blacklozenge \phi \Rightarrow \blacklozenge \phi \tag{D.17}$$

$$\vdash \quad \blacksquare \blacksquare \phi \Rightarrow \blacksquare \phi \tag{D.18}$$

The following axioms capture the interaction of the next/last operators with the sometime-always-until operators (and past time equivalents).

$$\vdash \quad \bigcirc \Diamond \phi \Leftrightarrow \Diamond \bigcirc \phi \tag{D.19}$$

$$\vdash \quad \bigcirc \Box \phi \Leftrightarrow \Box \bigcirc \phi \tag{D.20}$$

$$\vdash \quad ((\bigcirc \phi) \mathcal{U} (\bigcirc \psi)) \Leftrightarrow \bigcirc (\phi \mathcal{U} \psi) \tag{D.21}$$

(D.22)

The distributivity properties of the operators $\{\Diamond, \Box, \blacklozenge, \blacksquare, \mathcal{U}, \mathcal{S}\}$ are captured by the following axioms.

$$\vdash \quad \Diamond(\phi \vee \psi) \Leftrightarrow (\Diamond\phi \vee \Diamond\psi) \tag{D.23}$$

$$\vdash \quad \Box(\phi \wedge \psi) \Leftrightarrow (\Box\phi \wedge \Box\psi) \tag{D.24}$$

$$\vdash \quad ((\varphi \wedge \phi) \mathcal{U} \psi) \Leftrightarrow ((\varphi \mathcal{U} \psi) \wedge (\phi \mathcal{U} \psi)) \tag{D.25}$$

$$\vdash \quad (\varphi \mathcal{U} (\phi \vee \psi)) \Leftrightarrow ((\varphi \mathcal{U} \phi) \vee (\varphi \mathcal{U} \psi)) \tag{D.26}$$

$$\vdash \quad \blacklozenge (\phi \vee \psi) \Leftrightarrow (\blacklozenge \phi \vee \blacklozenge \psi) \tag{D.27}$$

$$\vdash \quad \blacksquare(\phi \wedge \psi) \Leftrightarrow (\blacksquare\phi \wedge \blacksquare\psi) \tag{D.28}$$

$$\vdash \quad ((\varphi \wedge \phi) \mathcal{S} \psi) \Leftrightarrow ((\varphi \mathcal{S} \psi) \wedge (\phi \mathcal{S} \psi)) \tag{D.29}$$

$$\vdash \quad (\varphi \mathcal{S} (\phi \vee \psi)) \Leftrightarrow ((\varphi \mathcal{S} \phi) \vee (\varphi \mathcal{S} \psi)) \tag{D.30}$$

The following are similar implications that do not hold in the reverse direction.

$$\vdash \quad (\Box\phi \vee \Box\psi) \;\Rightarrow\; \Box(\phi \vee \psi) \tag{D.31}$$

$$\vdash \quad \Diamond(\phi \wedge \psi) \;\Rightarrow\; ((\Diamond\phi) \wedge (\Diamond\psi)) \tag{D.32}$$

$$\vdash \quad ((\varphi \mathcal{U} \phi) \vee (\psi \mathcal{U} \phi)) \;\Rightarrow\; ((\varphi \vee \psi) \mathcal{U} \phi) \tag{D.33}$$

$$\vdash \quad (\varphi \mathcal{U} (\phi \wedge \psi)) \;\Rightarrow\; ((\varphi \mathcal{U} \phi) \wedge (\varphi \mathcal{U} \psi)) \tag{D.34}$$

$$\vdash \quad (\blacksquare\phi \vee \blacksquare\psi) \;\Rightarrow\; \blacksquare(\phi \vee \psi) \tag{D.35}$$

$$\vdash \quad \blacklozenge (\phi \wedge \psi) \;\Rightarrow\; (\blacklozenge \phi \wedge \blacklozenge \psi) \tag{D.36}$$

$$\vdash \quad ((\varphi \mathcal{S} \phi) \vee (\psi \mathcal{S} \phi)) \;\Rightarrow\; ((\varphi \vee \psi) \mathcal{S} \phi) \tag{D.37}$$

$$\vdash \quad (\varphi \mathcal{S} (\phi \wedge \psi)) \;\Rightarrow\; ((\varphi \mathcal{S} \phi) \wedge (\varphi \mathcal{S} \psi)) \tag{D.38}$$

The following axioms show that temporal operators are “monotonic in each argument” [47, p1005].

$$\vdash \quad \Box(\phi \;\Rightarrow\; \psi) \;\Rightarrow\; (\Box\phi \;\Rightarrow\; \Box\psi) \tag{D.39}$$

$$\vdash \quad \Box(\phi \;\Rightarrow\; \psi) \;\Rightarrow\; (\Diamond\phi \;\Rightarrow\; \Diamond\psi) \tag{D.40}$$

$$\vdash \quad \Box(\phi \;\Rightarrow\; \psi) \;\Rightarrow\; (\bigcirc\phi \;\Rightarrow\; \bigcirc\psi) \tag{D.41}$$

$$\vdash \quad \Box(\phi \;\Rightarrow\; \psi) \;\Rightarrow\; ((\phi \mathcal{U} \varphi) \;\Rightarrow\; (\psi \mathcal{U} \varphi)) \tag{D.42}$$

$$\vdash \quad \Box(\phi \;\Rightarrow\; \psi) \;\Rightarrow\; ((\varphi \mathcal{U} \phi) \;\Rightarrow\; (\varphi \mathcal{U} \psi)) \tag{D.43}$$

$$\vdash \quad \blacksquare(\phi \;\Rightarrow\; \psi) \;\Rightarrow\; (\blacksquare\phi \;\Rightarrow\; \blacksquare\psi) \tag{D.44}$$

$$\vdash \quad \blacksquare(\phi \;\Rightarrow\; \psi) \;\Rightarrow\; (\blacklozenge \phi \;\Rightarrow\; \blacklozenge \psi) \tag{D.45}$$

$$\vdash \quad \blacksquare(\phi \;\Rightarrow\; \psi) \;\Rightarrow\; (\CIRCLE\phi \;\Rightarrow\; \CIRCLE\psi) \tag{D.46}$$

$$\vdash \quad \blacksquare(\phi \;\Rightarrow\; \psi) \;\Rightarrow\; ((\phi \mathcal{S} \varphi) \;\Rightarrow\; (\psi \mathcal{S} \varphi)) \tag{D.47}$$

$$\vdash \quad \blacksquare(\phi \;\Rightarrow\; \psi) \;\Rightarrow\; ((\varphi \mathcal{S} \phi) \;\Rightarrow\; (\varphi \mathcal{S} \psi)) \tag{D.48}$$

The *fixpoint* characteristics of the temporal operators are given by the following axioms.

$$\vdash \quad \Diamond\phi \Leftrightarrow \phi \vee \bigcirc\Diamond\phi \tag{D.49}$$

$$\vdash \quad \Box\phi \Leftrightarrow \phi \wedge \bigcirc\Box\phi \tag{D.50}$$

$$\vdash \quad \phi \mathcal{U} \psi \Leftrightarrow \psi \vee (\phi \wedge \bigcirc(\phi \mathcal{U} \psi)) \tag{D.51}$$

$$\vdash \quad \phi \mathcal{B} \psi \Leftrightarrow \neg\psi \wedge (\phi \vee \bigcirc(\phi \mathcal{B} \psi)) \tag{D.52}$$

Finally, there is an *induction* axiom.

$$\vdash \quad \Box(\phi \;\Rightarrow\; \bigcirc\phi) \;\Rightarrow\; (\phi \;\Rightarrow\; \Box\phi) \tag{D.53}$$

D.2 Axioms from Section 5.3

The following list includes the Barcan formulae for temporal operators ([104, p47]).

$$\vdash \quad \forall x \cdot \bigcirc \phi(x) \Leftrightarrow \bigcirc \forall x \cdot \phi(x) \tag{D.54}$$

$\vdash \quad \exists x \cdot \bigcirc \phi(x) \Leftrightarrow \bigcirc \exists x \cdot \phi(x)$ (D.55)

$\vdash \quad \forall x \cdot \circledcirc \phi(x) \Leftrightarrow \circledcirc \forall x \cdot \phi(x)$ (D.56)

$\vdash \quad \exists x \cdot \circledcirc \phi(x) \Leftrightarrow \circledcirc \exists x \cdot \phi(x)$ (D.57)

$\vdash \quad \forall x \cdot \square \phi(x) \Leftrightarrow \square \forall x \cdot \phi(x)$ (D.58)

$\vdash \quad \forall x \cdot \blacksquare \phi(x) \Leftrightarrow \blacksquare \forall x \cdot \phi(x)$ (D.59)

$\vdash \quad \exists x \cdot \Diamond \phi(x) \Leftrightarrow \Diamond \exists x \cdot \phi(x)$ (D.60)

$\vdash \quad \exists x \cdot \blacklozenge \phi(x) \Leftrightarrow \blacklozenge \exists x \cdot \phi(x)$ (D.61)

Bibliography

[1] M. Abadi. *Temporal Logic Theorem Proving*. PhD thesis, Computer Science Department, Stanford University, Stanford, CA 94305, 1987.

[2] G. Agha. *ACTORS: A Model of Concurrent Computation in Distributed Systems.* The MIT Press: Cambridge, MA, 1986.

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

[4] J. F. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23(2):123–154, 1984.

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

[6] D. E. Appelt. Planning natural language utterances. In *Proceedings of the Second National Conference on Artificial Intelligence (AAAI-82)*, pages 59–62, Pittsburgh, PA, 1982.

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

[8] D. E. Appelt and K. Konolige. A nonmonotonic logic for reasoning about speech acts and belief revision. In M. Reinfrank, J. de Kleer, M. L. Ginsberg, and E. Sandewall, editors, *Non-Monotonic Reasoning — Proceedings of the Second International Workshop (LNAI Volume 346)*, pages 164–175. Springer-Verlag: Heidelberg, Germany, 1988.

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

[10] B. Banieqbal and H. Barringer. A study of an extended temporal language and a temporal fixed point calculus. Technical Report UMCS–86–10–2, Department of Computer Science, Manchester University, Oxford Rd., Manchester M13 9PL, UK, 1986.

[11] H. Barringer. Up and down the temporal way. Technical Report UMCS–85–9–3, Department of Computer Science, Manchester University, Oxford Rd., Manchester M13 9PL, UK, 1985.

[12] H. Barringer, M. Fisher, D. Gabbay, G. Gough, and R. Owens. METATEM: A framework for programming in temporal logic. In *REX Workshop on Stepwise Refinement of Distributed Systems: Models, Formalisms, Correctness (LNCS Volume 430)*, pages 94–129. Springer-Verlag: Heidelberg, Germany, June 1989.

[13] H. Barringer, R. Kuiper, and A. Pnueli. Now you may compose temporal logic specifications. In *Proceedings of the Sixteenth ACM Symposium on the Theory of Computing*, pages 51–63, 1984.

[14] J. Barwise and J. Perry. *Situations and Attitudes*. The MIT Press: Cambridge, MA, 1983.

[15] M. Ben-Ari, Z. Manna, and A. Pnueli. The temporal logic of branching time. In *Proceedings of the Eighth ACM Symposium on the Principles of Programming Languages (POPL)*, 1981.

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

[17] M. E. Bratman. *Intentions, Plans, and Practical Reason.* Harvard University Press: Cambridge, MA, 1987.

[18] M. E. Bratman. What is intention? In P. R. Cohen, J. L. Morgan, and M. E. Pollack, editors, *Intentions in Communication*, pages 15–32. The MIT Press: Cambridge, MA, 1990.

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

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

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

[22] H. J. Burckert and J. Muller. RATMAN: Rational agents testbed for multi-agent networks. In Y. Demazeau and J.-P. Müller, editors, *Decentralized AI 2 — Proceedings of the Second European Workshop on Modelling Autonomous Agents and Multi-Agent Worlds (MAAMAW-90)*, pages 217–230. Elsevier Science Publishers B.V.: Amsterdam, The Netherlands, 1991.

[23] B. Burmeister and K. Sundermeyer. Cooperative problem solving guided by intentions and perception. In E. Werner and Y. Demazeau, editors, *Decentralized AI 3 — Proceedings of the Third European Workshop on Modelling Autonomous Agents and Multi-Agent Worlds (MAAMAW-91)*, pages 77–92. Elsevier Science Publishers B.V.: Amsterdam, The Netherlands, 1992.

[24] S. Cammarata, D. McArthur, and R. Steeb. Strategies of cooperation in distributed problem solving. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence (IJCAI-83)*, Karlsruhe, Federal Republic of Germany, 1983.

[25] A. Cawsey, J.R. Galliers, S. Reece, and K. Sparck Jones. Automating the librarian: Belief revision as a base for system action and communication with the user. *The Computer Journal*, 25(3):221–232, June 1992.

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

[27] D. Chapman and P. Agre. Abstract reasoning as emergent from concrete activity. In M. P. Georgeff and A. L. Lansky, editors, *Reasoning About Actions & Plans — Proceedings of the 1986 Workshop*, pages 411–424. Morgan Kaufmann Publishers: San Mateo, CA, 1986.

[28] B. Chellas. *Modal Logic: An Introduction*. Cambridge University Press: Cambridge, England, 1980.

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

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

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

[32] D. Connah and P. Wavish. An experiment in cooperation. In Y. Demazeau and J.-P. Müller, editors, *Decentralized AI — Proceedings of the First European Workshop on Modelling Autonomous Agents in Multi-Agent Worlds (MAAMAW-89)*, pages 197–214. Elsevier Science Publishers B.V.: Amsterdam, The Netherlands, 1990.

[33] I. Craig. *Formal Specification of Advanced AI Architectures.* Ellis Horwood: Chichester, England, 1991.

[34] N. J. Davies. A first-order theory of reasoning agents. Technical Report CSM-130, Department of Computer Science, University of Essex, Colchester, UK, 1989.

[35] N. J. Davies. A first-order theory of truth, knowledge and belief. In *Logics in AI — Proceedings of the European Workshop JELIA-90 (LNAI Volume 478)*, pages 170–179. Springer-Verlag: Heidelberg, Germany, 1991.

[36] Y. Demazeau and J.-P. Müller, editors. *Decentralized AI — Proceedings of the First European Workshop on Modelling Autonomous Agents in Multi-Agent Worlds (MAAMAW-89)*. Elsevier Science Publishers B.V.: Amsterdam, The Netherlands, 1990.

[37] Y. Demazeau and J.-P. Müller, editors. *Decentralized AI 2 — Proceedings of the Second European Workshop on Modelling Autonomous Agents and Multi-Agent Worlds (MAAMAW-90)*. Elsevier Science Publishers B.V.: Amsterdam, The Netherlands, 1991.

[38] D. C. Dennett. *Brainstorms*. The MIT Press: Cambridge, MA, 1978.

[39] D. C. Dennett. *The Intentional Stance*. The MIT Press: Cambridge, MA, 1987.

[40] J. des Rivieres and H. J. Levesque. The consistency of syntactical treatments of knowledge. In J. Y. Halpern, editor, *Proceedings of the 1986 Conference on Theoretical Aspects of Reasoning About Knowledge*, pages 115–130. Morgan Kaufmann Publishers: San Mateo, CA, 1986.

[41] K. Devlin. *Logic and Information*. Cambridge University Press: Cambridge, England, 1991.

[42] J. Doran, H. Carvajal, Y. J. Choo, and Y. Li. The MCS multi-agent testbed: Developments and experiments. In S. M. Deen, editor, *CKBS-90 — Proceedings of the International Working Conference on Cooperating Knowledge Based Systems*, pages 240–254. Springer-Verlag: Heidelberg, Germany, 1991.

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

[44] E. H. Durfee and V. Lesser. Negotiating task decomposition and allocation using partial global planning. In L. Gasser and M. Huhns, editors, *Distributed Artificial Intelligence Volume II*, pages 229–244. Pitman Publishing: London and Morgan Kaufmann: San Mateo, CA, 1989.

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

[46] C. Dwork and Y. Moses. Knowledge and common knowledge in a byzantine environment I: Crash failures (extended abstract). In J. Y. Halpern, editor, *Proceedings of the 1986 Conference on Theoretical Aspects of Reasoning About Knowledge*, pages 149–170. Morgan Kaufmann Publishers: San Mateo, CA, 1986.

[47] E. A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, pages 996–1072. Elsevier Science Publishers B.V.: Amsterdam, The Netherlands, 1990.

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

[49] E. A. Emerson and J. Srinivasan. Branching time logic. In J. W. de Bakker, W.-P. de Roever, and G. Rozenberg, editors, *REX School-Workshop on Linear Time, Branching Time and Parial Order in Logics and Models for Concurrency (LNCS Volume 354)*, pages 123–172. Springer-Verlag: Heidelberg, Germany, 1988.

[50] H. B. Enderton. *A Mathematical Introduction to Logic*. Academic Press, 1972.

[51] R. Engelmore and T. Morgan, editors. *Blackboard Systems*. Addison-Wesley: Reading, MA, 1988.

[52] R. Fagin and J. Y. Halpern. Belief, awareness, and limited reasoning. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence (IJCAI-85)*, pages 480–490, Los Angeles, CA, 1985.

[53] R. Fagin, J. Y. Halpern, and M. Y. Vardi. What can machines know? on the properties of knowledge in distributed systems. *Journal of the ACM*, 39(2):328–376, 1992.

[54] R. Fagin and M. Y. Vardi. Knowledge and implicit knowledge in a distributed environment: Preliminary report. In J. Y. Halpern, editor, *Proceedings of the 1986 Conference on Theoretical Aspects of Reasoning About Knowledge*, pages 187–206. Morgan Kaufmann Publishers: San Mateo, CA, 1986.

[55] R. D. Fennell and V. R. Lesser. Parallelism in artificial intelligence problem solving: A case study of Hearsay II. *IEEE Transactions on Computers*, 26(2), 1977.

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

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

[58] M. Fischer and N. Immerman. Foundations of knowledge for distributed systems. In J. Y. Halpern, editor, *Proceedings of the 1986 Conference on Theoretical Aspects of Reasoning About Knowledge*, pages 171–186. Morgan Kaufmann Publishers: San Mateo, CA, 1986.

[59] M. Fisher. A resolution method for temporal logic. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91)*, Sydney, Australia, August 1991.

[60] M. Fisher. Concurrent METATEM Processes — the language and its implementation. (In preparation), 1992.

[61] M. Fisher. A normal form for first-order temporal formulae. In *Proceedings of Eleventh International Conference on Automated Deduction (CADE-92)*. Springer-Verlag: Heidelberg, Germany, June 1992.

[62] M. Fisher and H. Barringer. Concurrent METATEM Processes — a language for distributed AI. In *European Simulation Multiconference*, Copenhagen, Denmark, June 1991.

[63] N. Francez. *Fairness.* Springer-Verlag: Heidelberg, Germany, 1986.

[64] D. Gabbay. Declarative past and imperative future. In B. Banieqbal, H. Barringer, and A. Pnueli, editors, *Proceedings of the Colloquium on Temporal Logic in Specification (LNCS Volume 398)*, pages 402–450. Springer-Verlag: Heidelberg, Germany, 1989.

[65] J. R. Galliers. A strategic framework for multi-agent cooperative dialogue. In *Proceedings of the Eighth European Conference on Artificial Intelligence (ECAI-88)*, pages 415–420, Munich, Federal Republic of Germany, 1988.

[66] J. R. Galliers. *A Theoretical Framework for Computer Models of Cooperative Dialogue, Acknowledging Multi-Agent Conflict*. PhD thesis, Open University, UK, 1988.

[67] J. R. Galliers. Cooperative interaction as strategic belief revision. In S. M. Deen, editor, *CKBS-90 — Proceedings of the International Working Conference on Cooperating Knowledge Based Systems*, pages 148–163. Springer-Verlag: Heidelberg, Germany, 1991.

[68] A. Galton. Temporal logic and computer science: An overview. In A. Galton, editor, *Temporal Logics and their Applications*, pages 1–52. Academic Press, 1987.

[69] P. Gärdenfors. *Knowledge in Flux.* The MIT Press: Cambridge, MA, 1988.

[70] L. Gasser, C. Braganza, and N. Hermann. MACE: A flexible testbed for distributed AI research. In M. Huhns, editor, *Distributed Artificial Intelligence*, pages 119–152. Pitman Publishing: London and Morgan Kaufmann: San Mateo, CA, 1987.

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

[72] L. Gasser, N. Rouqette, R. W. Hill, and J. Lieb. Representing and using organizational knowledge in DAI systems. In L. Gasser and M. Huhns, editors, *Distributed Artificial Intelligence Volume II*, pages 55–78. Pitman Publishing: London and Morgan Kaufmann: San Mateo, CA, 1989.

[73] M. R. Genesereth and N. Nilsson. *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann Publishers: San Mateo, CA, 1987.

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

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

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

[77] M. L. Ginsberg. Decision procedures. In M. Huhns, editor, *Distributed Artificial Intelligence*, pages 3–28. Pitman Publishing: London and Morgan Kaufmann: San Mateo, CA, 1987.

[78] R. Goldblatt. *Logics of Time and Computation*. Centre for the Study of Language and Information — Lecture Notes Series, 1987. (Distributed by Chicago University Press).

[79] G. D. Gough. Decision procedures for temporal logic. Master's thesis, Department of Computer Science, Manchester University, Oxford Rd., Manchester M13 9PL, UK, October 1984.

[80] J. Y. Halpern. Reasoning about knowledge: An overview. In J. Y. Halpern, editor, *Proceedings of the 1986 Conference on Theoretical Aspects of Reasoning About Knowledge*, pages 1–18. Morgan Kaufmann Publishers: San Mateo, CA, 1986.

[81] J. Y. Halpern. Using reasoning about knowledge to analyze distributed systems. *Annual Review of Computer Science*, 2:37–68, 1987.

[82] J. Y. Halpern. Knowledge and common knowledge in a distributed environment. *Journal of the ACM*, 37(3), 1990.

[83] J. Y. Halpern and Y. Moses. A guide to completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence*, 54:319–379, 1992.

[84] D. Harel. *First-Order Dynamic Logic (LNCS Volume 68)*. Springer-Verlag: Heidelberg, Germany, 1979.

[85] C. Hewitt. A universal modular ACTOR formalism for AI. In *Proceedings of the Third International Joint Conference on Artificial Intelligence (IJCAI-73)*, pages 235–245, 1973.

[86] C. Hewitt. Viewing control structures as patterns of passing messages. *Artificial Intelligence*, 8(3):323–364, 1977.

[87] J. Hintikka. *Knowledge and Belief.* Cornell University Press: Ithaca, NY, 1962.

[88] G. E. Hughes and M. J. Cresswell. *Introduction to Modal Logic*. Methuen and Co., Ltd., 1968.

[89] G. E. Hughes and M. J. Cresswell. *Companion to Modal Logic*. Methuen and Co., Ltd., 1984.

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

[91] N. R. Jennings. On being responsible. In E. Werner and Y. Demazeau, editors, *Decentralized AI 3 — Proceedings of the Third European Workshop on Modelling Autonomous Agents and Multi-Agent Worlds (MAAMAW-91)*, pages 93–102. Elsevier Science Publishers B.V.: Amsterdam, The Netherlands, 1992.

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

[93] C. B. Jones. *Systematic Software Development using VDM (second edition)*. Prentice Hall, 1990.

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

[95] J. A. W. Kamp. *Tense Logic and the Theory of Linear Order*. PhD thesis, University of California, 1968.

[96] G. Kiss. Variable coupling of agents to their environment: Combining situated and symbolic automata. In E. Werner and Y. Demazeau, editors, *Decentralized AI 3 — Proceedings of the Third European Workshop on Modelling Autonomous Agents and Multi-Agent Worlds (MAAMAW-91)*, pages 231–248. Elsevier Science Publishers B.V.: Amsterdam, The Netherlands, 1992.

[97] G. Kiss and H. Reichgelt. Towards a semantics of desires. In E. Werner and Y. Demazeau, editors, *Decentralized AI 3 — Proceedings of the Third European Workshop on Modelling Autonomous Agents and Multi-Agent Worlds (MAAMAW-91)*, pages 115–128. Elsevier Science Publishers B.V.: Amsterdam, The Netherlands, 1992.

[98] K. Konolige. A first-order formalization of knowledge and action for a multi-agent planning system. In J. E. Hayes, D. Michie, and Y. Pao, editors, *Machine Intelligence 10*, pages 41–72. Ellis Horwood: Chichester, England, 1982.

[99] K. Konolige. *A Deduction Model of Belief and its Logics*. PhD thesis, Computer Science Department, Stanford University, Stanford, CA 94305, 1984.

[100] K. Konolige. *A Deduction Model of Belief*. Pitman Publishing: London and Morgan Kaufmann: San Mateo, CA, 1986.

[101] K. Konolige. What awareness isn't: A sentential view of implicit and explicit belief (position paper). In J. Y. Halpern, editor, *Proceedings of the 1986 Conference on Theoretical Aspects of Reasoning About Knowledge*, pages 241–250. Morgan Kaufmann Publishers: San Mateo, CA, 1986.

[102] K. Konolige. Hierarchic autoepistemic theories for nonmonotonic reasoning: Preliminary report. In M. Reinfrank, J. de Kleer, M. L. Ginsberg, and E. Sandewall, editors, *Nonmonotonic Reasoning — Proceedings of the Second International Workshop (LNAI Volume 346)*, pages 42–59. Springer-Verlag: Heidelberg, Germany, 1988.

[103] S. Kripke. Semantical analysis of modal logic. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 9:67–96, 1963.

[104] F. Kröger. *Temporal Logic of Programs (EATCS Monographs on Theoretical Computer Science Vol 8)*. Springer-Verlag: Heidelberg, Germany, 1987.

[105] R. E. Ladner and J. H. Reif. The logic of distributed protocols: preliminary report. In *Proceedings of the 1986 Conference on Theoretical Aspects of Reasoning About Knowledge*, pages 207–222. Morgan Kaufmann Publishers: San Mateo, CA, 1986.

[106] G. Lakemeyer. Steps towards a first-order theory of explicit and implicit belief. In J. Y. Halpern, editor, *Proceedings of the 1986 Conference on Theoretical Aspects of Reasoning About Knowledge*, pages 325–340. Morgan Kaufmann Publishers: San Mateo, CA, 1986.

[107] G. Lakemeyer. A computationally attractive first-order logic of belief. In *JELIA-90: Proceedings of the European Workshop on Logics in AI (LNAI Volume 478)*, pages 333–347. Springer-Verlag: Heidelberg, Germany, 1991.

[108] L. Lamport. Sometimes is sometimes not never — but not always. In *Proceedings of the Seventh ACM Symposium on the Principles of Programming Languages (POPL)*, 1980.

[109] A. L. Lansky. A representation of parallel activity based on events, structure, and causality. In M. P. Georgeff and A. L. Lansky, editors, *Reasoning About Actions & Plans — Proceedings of the 1986 Workshop*, pages 123–160. Morgan Kaufmann Publishers: San Mateo, CA, 1986.

[110] D. B. Lenat. BEINGS: Knowledge as interacting experts. In *Proceedings of the Fourth International Joint Conference on Artificial Intelligence (IJCAI-75)*, Stanford, CA, 1975.

[111] Y. Lespérance. A formal account of self knowledge and action. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89)*, pages 868–874, Detroit, MI, 1989.

[112] H. J. Levesque. A logic of implicit and explicit belief. In *Proceedings of the Fourth National Conference on Artificial Intelligence (AAAI-84)*, pages 198–202, Austin, TX, 1984.

[113] H. J. Levesque, P. R. Cohen, and J. H. T. Nunes. On acting together. In *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-90)*, pages 94–99, Boston, MA, 1990.

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

[115] V. Lifschitz. On the semantics of STRIPS. In M. P. Georgeff and A. L. Lansky, editors, *Reasoning About Actions & Plans — Proceedings of the 1986 Workshop*, pages 1–10. Morgan Kaufmann Publishers: San Mateo, CA, 1986.

[116] Z. Manna and A. Pnueli. Verification of concurrent programs: The temporal framework. In R. S. Boyer and J. S. Moore, editors, *The Correctness Problem in Computer Science*. Academic Press, 1981.

[117] J. McCarthy. Ascribing mental qualities to machines. Technical report, Stanford University AI Lab., Stanford, CA 94305, 1978.

[118] J. McCarthy and P. J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence 4*. Edinburgh University Press, 1969.

[119] J. McClelland and D. Rumelhart. *Parallel Distributed Processing*. The MIT Press: Cambridge, MA, 1987.

[120] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.

[121] R. Montague. Syntactical treatments of modality, with corollaries on reflexion principles and finite axiomatizations. *Acta Philosophica Fennica*, 16:153–167, 1963.

[122] R. C. Moore. Reasoning about knowledge and action. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence (IJCAI-77)*, Cambridge, MA, 1977.

[123] R. C. Moore. Semantical considerations in non-monotonic reasoning. *Artificial Intelligence*, 25(1), 1985.

[124] R. C. Moore. A formal theory of knowledge and action. In J. F. Allen, J. Hendler, and A. Tate, editors, *Readings in Planning*, pages 480–519. Morgan Kaufmann Publishers: San Mateo, CA, 1990.

[125] L. Morgenstern. A first-order theory of planning, knowledge, and action. In J. Y. Halpern, editor, *Proceedings of the 1986 Conference on Theoretical Aspects of Reasoning About Knowledge*, pages 99–114. Morgan Kaufmann Publishers: San Mateo, CA, 1986.

[126] L. Morgenstern. Knowledge preconditions for actions and plans. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence (IJCAI-87)*, pages 867–874, Milan, Italy, 1987.

[127] Y. Moses and M. Tennenholtz. On formal aspects of artificial social systems. Technical Report CS91-01, Weizmann Institute of Science, Rehovot: Israel, 1991.

[128] W. Penczek. Branching time and partial order in temporal logic. Technical Report UMCS–91–3–3, Department of Computer Science, Manchester University, Oxford Rd., Manchester M13 9PL, UK, 1991.

[129] D. Perlis. Languages with self reference I: Foundations. *Artificial Intelligence*, 25:301–322, 1985.

[130] D. Perlis. Languages with self reference II: Knowledge, belief, and modality. *Artificial Intelligence*, 34:179–212, 1988.

[131] D. Perlis. Meta in logic. In P. Maes and D. Nardi, editors, *Meta-Level Architectures and Reflection*, pages 37–49. Elsevier Science Publishers B.V.: Amsterdam, The Netherlands, 1988.

[132] C. R. Perrault. An application of default logic to speech acts theory. In P. R. Cohen, J. Morgan, and M. E. Pollack, editors, *Intentions in Communication*, pages 161–186. The MIT Press: Cambridge, MA, 1990.

[133] A. Pnueli. The temporal logic of programs. In *Proceedings of the Eighteenth Symposium on the Foundations of Computer Science*, 1977.

[134] A. Pnueli. Specification and development of reactive systems. In *Information Processing 86*. Elsevier Science Publishers B.V.: Amsterdam, The Netherlands, 1986.

[135] A. Pnueli. Applications of temporal logic to the specification and verification of reactive systems. In *REX School-Workshop on Linear Time, Branching Time, and Partial Order in Logics and Models for Concurrency*, Noordwijkerhout, Netherlands, 1988.

[136] A. Prior. *Past, Present and Future*. Oxford University Press: Oxford, England, 1967.

[137] H. Reichgelt. A comparison of first-order and modal logics of time. In P. Jackson, H. Reichgelt, and F. van Harmelen, editors, *Logic Based Knowledge Representation*, pages 143–176. The MIT Press: Cambridge, MA, 1989.

[138] H. Reichgelt. Logics for reasoning about knowledge and belief. *Knowledge Engineering Review*, 4(2):119–139, 1989.

[139] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.

[140] J. S. Rosenschein. Synchronisation of multi-agent plans. In *Proceedings of the Second National Conference on Artificial Intelligence (AAAI-82)*, Pittsburgh, PA, 1982.

[141] J. S. Rosenschein. *Rational Interaction: Cooperation Among Intelligent Agents.* PhD thesis, Computer Science Department, Stanford University, Stanford, CA 94305, 1985.

[142] S. Rosenschein. Formal theories of knowledge in AI and robotics. *New Generation Computing*, pages 345–357, 1985.

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

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

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

[146] J. R. Searle. *Expression and Meaning*. Cambridge University Press: Cambridge, England, 1979.

[147] N. Seel. *Agent Theories and Architectures.* PhD thesis, Surrey University, Guildford, UK, 1989.

[148] N. Seel. Intentional descriptions of reactive systems. In Y. Demazeau and J.-P. Müller, editors, *Decentralized AI 2 — Proceedings of the Second European Workshop on Modelling Autonomous Agents and Multi-Agent Worlds (MAAMAW-90)*, pages 15–34. Elsevier Science Publishers B.V.: Amsterdam, The Netherlands, 1991.

[149] N. Shardlow. Action and agency in cognitive science. Master's thesis, Department of Psychlogy, University of Manchester, Oxford Rd., Manchester M13 9PL, UK, 1990.

[150] Y. Shoham. Time for action: on the relation between time, knowledge and action. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89)*, pages 954–959, Detroit, MI, 1989.

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

[152] M. P. Singh. Towards a theory of situated know-how. In *Proceedings of the Ninth European Conference on Artificial Intelligence (ECAI-90)*, pages 604–609, Stockholm, Sweden, 1990.

[153] M. P. Singh. Group ability and structure. In Y. Demazeau and J.-P. Müller, editors, *Decentralized AI 2 — Proceedings of the Second European Workshop on Modelling Autonomous Agents and Multi-Agent Worlds (MAAMAW-90)*, pages 127–146. Elsevier Science Publishers B.V.: Amsterdam, The Netherlands, 1991.

[154] M. P. Singh. Towards a formal theory of communication for multi-agent systems. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 69–74, Sydney, Australia, 1991.

[155] M. P. Singh. A critical examination of the Cohen-Levesque theory of intention. In *Proceedings of the Tenth European Conference on Artificial Intelligence (ECAI-92)*, pages 364–368, Vienna, Austria, 1992.

[156] M. P. Singh and N. M. Asher. Towards a formal theory of intentions. In *Logics in AI — Proceedings of the European Workshop JELIA-90 (LNAI Volume 478)*, pages 472–486. Springer-Verlag: Heidelberg, Germany, 1991.

[157] A. Sistla. Theoretical issues in the design and verification of distributed systems. Technical Report CMU-CS-83-146, School of Computer Science, Carnegie-Mellon University, Pittsburgh, PA, 1983.

[158] A. P. Sistla, E. M. Clarke, N. Francez, and A. R. Meyer. Can message buffers be axiomatized in temporal logic? *Information and Control*, 63(1/2), 1985.

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

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

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

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

[163] R. M. Smullyan. *First-Order Logic*. Springer-Verlag: Heidelberg, Germany, 1968.

[164] L. Sommaruga, N. Avouris, and M. Van Liedekerke. An environment for experimentation with interactive cooperating knowledge-based systems. In N. Shadbolt, editor, *Research and Development in Expert Systems VI*. Cambridge University Press: Cambridge, England, 1989.

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

[166] C. Stirling. Completeness results for full branching time logic. In *REX School-Workshop on Linear Time, Branching Time, and Partial Order in Logics and Models for Concurrency*, Noordwijkerhout, Netherlands, 1988.

[167] A. Tate. Generating project networks. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence (IJCAI-77)*, Cambridge, MA, 1977.

[168] A. Thayse, editor. *From Modal Logic to Deductive Databases*. John Wiley & Sons: Chichester, England, 1989.

[169] R. Thomason. A note on syntactical treatments of modality. *Synthese*, 44:391–395, 1980.

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

[171] M. Torrance and P. A. Viola. The AGENT0 manual. Technical report, Program in Symbolic Systems, Stanford University, CA, 1991.

[172] R. Turner. *Truth and Modality for Knowledge Representation*. Pitman Publishing: London, 1990.

[173] H. Van Dyke Parunak. Manufacturing experience with the contract net. In M. Huhns, editor, *Distributed Artificial Intelligence*, pages 285–310. Pitman Publishing: London and Morgan Kaufmann: San Mateo, CA, 1987.

[174] H. van dyke Parunak. Distributed AI and manufacturing control: Some issues and insights. In Y. Demazeau and J.-P. Müller, editors, *Decentralized AI — Proceedings of the First European Workshop on Modelling Autonomous Agents in Multi-Agent Worlds (MAAMAW-89)*, pages 81–104. Elsevier Science Publishers B.V.: Amsterdam, The Netherlands, 1990.

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

[176] E. Werner. A formal computational semantics and pragmatics of speech acts. In *Proceedings COLING–88*, pages 744–749, 1988.

[177] E. Werner. Social intentions. In *Proceedings of the Eighth European Conference on Artificial Intelligence (ECAI-88)*, pages 719–723, Munich, Federal Republic of Germany, 1988.

[178] E. Werner. Toward a theory of communication and cooperation for multiagent planning. In M. Y. Vardi, editor, *Proceedings of the Second Conference on Theoretical Aspects of Reasoning About Knowledge*, pages 129–144. Morgan Kaufmann Publishers: San Mateo, CA, 1988.

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

[180] E. Werner. Distributed cooperation algorithms. In Y. Demazeau and J.-P. Müller, editors, *Decentralized AI — Proceedings of the First European Workshop on Modelling Autonomous Agents in Multi-Agent Worlds (MAAMAW-89)*, pages 17–32. Elsevier Science Publishers B.V.: Amsterdam, The Netherlands, 1990.

[181] E. Werner. What can agents do together: A semantics of co-operative ability. In *Proceedings of the Ninth European Conference on Artificial Intelligence (ECAI-90)*, pages 694–701, Stockholm, Sweden, 1990.

[182] E. Werner. Planning and uncertainty. In *Proceedings of the Tenth Workshop of the Planning Special Interest Group*, Cambridge, UK, 1991.

[183] E. Werner. A unified view of information, intention and ability. In Y. Demazeau and J.-P. Müller, editors, *Decentralized AI 2 — Proceedings of the Second European Workshop on Modelling Autonomous Agents and Multi-Agent Worlds (MAAMAW-90)*, pages 109–126. Elsevier Science Publishers B.V.: Amsterdam, The Netherlands, 1991.

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

[185] T. Wittig. ARCHON: Cooperation of heterogeneous on-line systems. In *Wissenbasierte Systeme — Proceedings of the Third International Congress*. Springer-Verlag: Heidelberg, Germany, 1989.

[186] P. Wolper. Temporal logic can be more expressive. *Information and Control*, 56, 1983.

[187] M. Wooldridge. An approach to reasoning about multi-agent systems. In *Proceedings of the Third UK Workshop on Belief Representation and Agent Architecture (BRAA-92)*, University of Durham, UK, June 1992.

[188] M. Wooldridge and M. Fisher. A first-order branching time logic of multi-agent systems. In *Proceedings of the Tenth European Conference on Artificial Intelligence (ECAI-92)*, pages 234–238, Vienna, Austria, 1992.

[189] M. Wooldridge, G. M. P. O'Hare, and R. Elks. FELINE — a case study in the design and implementation of a co-operating expert system. In *Proceedings of the Eleventh European Conference on Expert Systems and Their Applications*, Avignon, France, May 1991.

[190] G. Zlotkin and J. S. Rosenschein. Negotiation and task sharing among autonomous agents in cooperative domains. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89)*, pages 912–917, Detroit, MI, 1989.