Handbook of Knowledge Representation
Edited by F. van Harmelen, V. Lifschitz and B. Porter

DOI: 10.1016/S1574-6526(07)03024-6

Chapter 24

Multi-Agent Systems

Wiebe van der Hoek and Michael Wooldridge

We review the state of the art in knowledge representation formalisms for multi-agent systems. We divide work in this area into two categories. In the first category are approaches that attempt to represent the cognitive state of rational agents, and to characterize logically how such a state leads a rational agent to act. We begin by motivating this approach. We then describe four of the best-known such logical frameworks, and discuss the possible roles that such logics can play in helping us to *engineer* artificial agents. In the second category are approaches based on representing the strategic structure of a multi-agent environment, and in particular, the *powers* that agents have, either individually or in coalitions. Here, we describe Coalition Logic, Alternating-time Temporal Logic (ATL), and epistemic extensions.

24.1 Introduction

The discipline of knowledge representation focuses on how to represent and reason about environments with various different properties, usually with the goal of making decisions, for example about how best to act in this environment. But what are the things that are actually *doing* this representation and reasoning? The now-conventional terminology is to refer to these entities as *agents*. The agents may be computer programs (in which case they are called *software agents*) or they may be people like you or I. The case where there is only assumed to be one agent in the environment (for example, a single autonomous robot operating in a warehouse) is usually the simplest scenario for knowledge representation, and often does not require techniques beyond those described elsewhere in this book. However, where there are *multiple* agents in the environment, things get much more interesting—and challenging. This is because it becomes necessary for an agent to represent and reason about *the other agents in the environment*. Again there are two possibilities. The first is that all the agents in the environment can be assumed to *share a common purpose*. This might be the case, for example, if we are designing a multi-robot system to operate in a warehouse environment. Here, we can assume the robots share a common purpose because we can design them that way. However, the second case is again much more interesting, and

presents many more challenges for knowledge representation. This is where the agents comprising the system do *not* share the same purpose. This might be the case, for example, in e-commerce systems, where a software agent is attempting to buy some particular item for as low a price as possible, while a seller agent tries to sell it for as high a price as possible. While in one sense the agents share a common goal of engaging in trade, there is obviously a fundamental difference with respect to their more specific goals.

How should we go about representing and reasoning about environments containing multiple agents? That is, what aspects of them should we be attempting to represent? Within the multi-agent systems community, one can distinguish two distinct trends:

Cognitive models of rational action: The first main strand of research in representing multi-agent systems focuses on the issue of representing the *attitudes* of agents within the system: their beliefs, aspirations, intentions, and the like. The aim of such formalisms is to derive a model that predicts how a *rational agent* would go from its beliefs and desires to actions. Work in this area builds largely on research in the philosophy of mind.

Models of the strategic structure of the system: The second main strand of research focuses not on the internal states or attitudes of agents, but on the strategic structure of the environment: what agents can accomplish in the environment, either together or alone. Work in this area builds on models of effectivity from the game theory community, and the models underpinning such logics are closely related to formal games.

Inevitably, the actual divisions between these two categories are more blurred than our rather crisp categorization suggests.

24.2 Representing Rational Cognitive States

In attempting to understand the behavior of agents in the everyday world, we frequently make use of *folk psychology*:

> Many philosophers and cognitive scientists claim that our everyday or "folk" understanding of mental states constitutes a theory of mind. That theory is widely called "folk psychology" (sometimes "commonsense" psychology). The terms in which folk psychology is couched are the familiar ones of "belief" and "desire", "hunger", "pain" and so forth. According to many theorists, folk psychology plays a central role in our capacity to predict and explain the behavior of ourselves and others. However, the nature and status of folk psychology remains controversial. [117]

For example, we use statements such as *Michael intends to write a paper* in order to explain Michael's behavior. Once told this statement, we expect to find Michael shelving other commitments and developing a plan to write the paper; we would expect him to spend a lot of time at his computer; we would not be surprised to find him in a grumpy mood; but we *would* be surprised to find him at a late night party. The philosopher Dennett coined the phrase *intentional system* to refer to an entity that is

best understood in terms of folk-psychology notions such as beliefs, desires, and the like [25]. This was also what Hofstadter was referring to already in 1981, when he sketched "coffee house conversation on the Turing test to determine if a machine can think" [55], in which several students discuss AI and in which one of them states that "you AI advocates have far underestimated the human mind, and that there are things a computer will never, ever be able to do". Sandy, a philosophy student puts the following forward:

> But eventually, when you put enough feelingless calculations together in a huge coordinated organization, you'll get something that has properties on another level. You can see it—in fact you *have* to see it—not as a bunch of little calculations, but as a system of tendencies and desires and beliefs and so on. When things get complicated enough, you're forced to change your level of description. To some extend that's already happening, which is why we use words such as "want", "think", "try", and "hope", to describe chess programs and other attempts at mechanical thought.

The intentional stance is essentially nothing more than an abstraction tool. It is a convenient shorthand for talking about certain complex systems (such as people), which allows us to succinctly predict and explain their behavior without having to understand or make claims about their internal structure or operation. Note that the intentional stance has been widely discussed in the literature—let us just remark here that Sandy of the Coffeeshop Conversation claims that the really interesting things in AI will only begin to happen, 'when the program *itself* adopts the intentional stance towards itself'—and it is not our intention to add to this debate; see [112] for a discussion and references.

If we accept the usefulness of the intentional stance for characterizing the properties of rational agents, then the next step in developing a formal theory of such agents is to identify the components of an agent's state. There are many possible mental states that we might choose to characterize an agent: beliefs, goals, desires, intentions, commitments, fears, hopes, and obligations are just a few. We can identify several important categories of such attitudes, for example:

Information attitudes: those attitudes an agent has towards information about its environment. The most obvious members of this category are knowledge and belief.

Pro attitudes: those attitudes an agent has that tend to lead it to perform actions. The most obvious members of this category are goals, desires, and intentions.

Normative attitudes: including obligations, permissions and authorization.

Much of the literature on developing formal theories of agency has been taken up with the relative merits of choosing one attitude over another, and investigating the possible relationships between these attitudes. While there is no consensus on which attitudes should be chosen as primitive, most formalisms choose knowledge or belief together with at least goals or desires.

24.2.1 A Logical Toolkit

In attempting to axiomatize the properties of a rational agent in terms of (say) its beliefs and desires, we will find ourselves attempting to formalize statements such as the following

Wiebe believes Ajax are great. (24.1)

Wiebe desires that Ajax will win. (24.2)

This suggests that a logical characterization of these statements must include constructions of the form

$$i \left\{ \begin{matrix} \text{believes} \\ \text{desires} \end{matrix} \right\} \varphi$$

where i is a term denoting an agent, and φ is a sentence. We immediately encounter difficulties if we attempt to represent such statements in first-order logic. First of all, the constructs mentioned above should definitely not be *extensional*—even if "it rains in Utrecht" and "it rains in Liverpool" may accidentally both be true, one can believe one without the other, desire the second but not the first, even try to achieve one while hindering the other. Apart from this, representing such statements in first-order logic—as binary predicates of the form $Bel(i, \varphi)$ and $Desire(i, \varphi)$—will not work, because the second term is a sentence, and not a term. By fixing the domain of the first-order language to be itself a language, we can get around this problem, thereby obtaining a first-order meta-language. The meta-language approach has been successfully adopted by a number of researchers, for example, [106]. However, meta-language approaches have also been criticized for representing mental states (see, e.g., [63] for a detailed critique). Instead of choosing a meta-language approach, most researchers opt for a *modal* approach, whereby an agent's beliefs, desires, and the like are represented by an indexed collection of modal operators. The semantics of these operators are generally given in terms or Kripke structures, in the by-now familiar way [19, 86, 13]. The use of Kripke structures and their associated mathematics of correspondence theory makes it possible to quickly generate a number of soundness results for axiomatizations of these logics. However, the *combination* of many modalities into a single framework presents a significant challenge from a logical point of view. Completeness, expressivity and complexity results for logics that incorporate multiple modalities into a single framework are typically complex, and this area of research is much at the leading edge of contemporary modal logic research [34]. Moreover, reasoning in such enriched systems is typically computationally very hard [39]. Despite these problems, modal approaches dominate in the literature, and in this article, we focus exclusively on such approaches.

In addition to representing an agent's attitudes, logics of rational agency also typically incorporate some way of representing the *actions* that agents perform, and the effects of these actions. Many researchers adapt techniques from *dynamic* logic in order to represent actions and their effects [42], whereas others confine themselves to a *temporal* set-up. Although there is some work in establishing the exact relation between the two approaches, this issue still deserves a better investigation.

In the next four sections, we review some of the best-known formalisms for reasoning about the cognitive states of rational agents:

- Dynamic Epistemic Logic (DEL);
- Cohen and Levesque's seminal intention logic [22];
- Rao and Georgeff's BDI framework [91]; and
- the KARO framework of Linder et al. [70].

24.2.2 Dynamic Epistemic Logic

The first formalism we deal with, dynamic epistemic logic, is intended to capture the interaction between the actions that an agent performs and its knowledge. Elsewhere in this handbook is a full treatment of logics for knowledge, and we pre-suppose some familiarity with this subject. The idea is simply to take the logical machinery of epistemic logic [30] and augment it with a dynamic component [43], for referring to actions. The origins of such logics for knowledge representation lie in the work of Robert Moore [77]. Moore's chief concern was to study the ways that knowledge and action interact, and he identified two main issues. The first is that some actions *produce knowledge*, and therefore their effects must be formulated in terms of the epistemic states of participants. The second is that of *knowledge preconditions*: what an agent needs to know in order to be able to perform an action. A simple example is that in order to unlock a safe, one must know the combination for the lock. Using these ideas, Moore formalized a notion of *ability*. He suggested that in order for an agent to be able to achieve some state of affairs φ, the agent must either:

- know the identity of an action α (i.e., have an "executable description" of an action α) such that after α is performed, φ holds; or else
- know the identity of an action α such that after α is performed, the agent will know the identity of an action α' such that after α' is performed, φ holds.

The point about "knowing the identity" of an action is that, in order for me to be able to become rich, it is not sufficient for me simply to know that *there exists some action* I could perform which would make me rich; I must either know what that action is (the first clause above), or else to be able to perform some action which would furnish me with the information about which action to perform in order to make myself rich. This apparently subtle distinction is rather important, and it is known as the distinction between knowledge *de re* (which involves knowing the identity of a thing) and *de dicto* (which involves knowing that something exists) [30, p. 101]. We will see later, when we review more recent work on temporal logics of ability, that this distinction also plays an important role there.

Nowadays, the term *Dynamic Epistemic Logic* (DEL) [11, 108] is used to refer to formalisms that add a special class of actions—*epistemic actions*—to the standard logic S5 for knowledge. The term "epistemic action" is used to refer to an action with an epistemic component, such as learning or announcing something. Thus, in DEL, actions themselves have an epistemic flavor: they denote an announcement, a private message, or even the act of "suspecting" something.

There are several variants of dynamic epistemic logic in the literature. In the language of [108], apart from the static formulas involving knowledge, there is also the construct $[\alpha]\varphi$, meaning that after execution of the epistemic action α, statement φ is

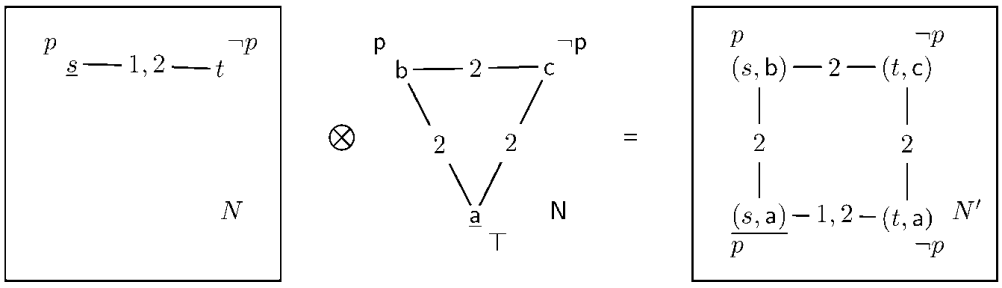

Figure 24.1: Multiplying an epistemic state N, s with the action model (N, a) representing the action $L_{12}(L_1?p \cup L_1?\neg p \cup !\top)$.

true. Actions α specify who is informed by what. To express "learning", actions of the form $L_B\beta$ are used, where β again is an action: this expresses the fact that "coalition B learns that β takes place". The expression $L_B(!\alpha \cup \beta)$, means the coalition B learns that either α or β is happening, while in fact α takes place.

To make the discussion concrete, assume we have two agents, 1 and 2, and that they commonly know that a letter on their table contains either the information p or $\neg p$ (but they do not know, at this stage, which it is). Agent 2 leaves the room for a minute, and when he returns, he is unsure whether or not 1 read the letter. This action would be described as

$$L_{12}(L_1?p \cup L_1?\neg p \cup !\top)$$

which expresses the following. First of all, in fact nothing happened (this is denoted by $!\top$). However, the knowledge of *both* agents changes: they commonly learn that 1 might have learned p, and he might have learned $\neg p$.

Although this is basically the language for DEL as used in [108], we now show how the example can be interpreted using the appealing semantics of [11]. In this semantics, both the uncertainty about the state of the world, and that of the action taking place, are represented in two independent Kripke models. The result of performing an epistemic action in an epistemic state is then computed as a "cross-product", see Fig. 24.1. Model N in this figure represents that it is common knowledge among 1 and 2 that both are ignorant about p. The triangular shaped model N is the *action model* that represents the knowledge and ignorance when $L_{12}(L_1?p \cup L_1?\neg p \cup !\top)$ is carried out. The points a, b, c of the model N are also called actions, and the formulas accompanying the name of the actions are called *pre-conditions*: the condition that has to be fulfilled in order for the action to take place. Since we are in the realm of *truthful information transfer*, in order to perform an action that reveals p, the pre-condition p must be satisfied, and we write $\mathsf{pre}(\mathsf{b}) = \mathsf{p}$. For the case of nothing happening, only the precondition $\top$ need be true. Summarizing, action b represents the action that agent 1 reads p in the letter, action c is the action when $\neg p$ is read, and a is for nothing happening. As with 'static' epistemic models, we omit reflexive arrows, so that N indeed represents that p or $\neg p$ is learned by 1, or that nothing happens: moreover, it is commonly known between 1 and 2 that 1 knows which action takes place, while for 2 they all look the same.

Now let $M, w = \langle W, R_1, R_2, \ldots, R_m, \pi\rangle$, w be a static epistemic state, and M, w an action in a finite action model. We want to describe what $M, w \oplus \mathsf{M}, \mathsf{w} = \langle W', R'_1,$

$R'_2, \ldots, R'_m, \pi' \rangle, w'$, looks like—the result of 'performing' the action represented by M, w in M, w. Every action from M, w that is executable in any state $v \in W$ gives rise to a new state in W': we let $W' = \{(v, \mathsf{v}) \mid v \in W, M, v \models \mathsf{pre(v)}\}$. Since epistemic actions do not change any objective fact in the world, we stipulate $\pi'(v, \mathsf{v}) = \pi(v)$. Finally, when are two states (v, v) and (u, u) indistinguishable for agent i? Well, he should be both unable to distinguish the originating states ($R_i uv$), and unable to know what is happening ($\mathsf{R_i uv}$). Finally, the new state w' is of course (w, w). Note that this construction indeed gives $N, s \oplus \mathsf{N}, \mathsf{a} = N', (s, \mathsf{a})$, in our example of Fig. 24.1. Finally, let the action α be represented by the action model state M, w. Then the truth definition under the action model semantics reads that $M, w \models [\alpha]\varphi$ iff $M, w \models \mathsf{pre(w)}$ implies $(M, w) \oplus (\mathsf{M}, \mathsf{w}) \models \varphi$. In our example: $N, s \models [L_{12}(L_1?p \cup L_1?\neg p \cup !\top)]\varphi$ iff $N', (s, \mathsf{a}) \models \varphi$.

Note that the accessibility relation in the resulting model is defined as

$$R_i(u, \mathsf{u})(v, \mathsf{v}) \Leftrightarrow R_i uv \,\&\, \mathsf{R_i} uv. \tag{24.3}$$

This means that an agent cannot distinguish two states after execution of an action α, if and only if he could not distinguish the 'sources' of those states, and he does not know which action exactly takes place. Put differently: if an agent knows the difference between two states s and t, then they can never look the same after performing an action, and likewise, if two indistinguishable actions α and β take place in a state s, they will give rise to new states that can be distinguished.

Dynamic epistemic logics provide us with a rich and powerful framework for reasoning about information flow in multi-agent systems, and the possible epistemic states that may arise as a consequence of actions performed by agents within a system. However, they do not address the issues of how an agent *chooses* an action, or whether an action represents a *rational choice* for an agent. For this, we need to consider pro-attitudes: desires, intentions, and the like. The frameworks we describe in the following three sections all try to bring together information-related attitudes (belief and knowledge) with attitudes such as desiring and intending, with the aim of providing a more complete account of rational action and agency.

24.2.3 Cohen and Levesque's Intention Logic

One of the best known, and most sophisticated attempts to show how the various components of an agent's cognitive makeup could be combined to form a logic of rational agency is due to Cohen and Levesque [22]. 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 [23] 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 [36, 35], as well as several studies in the theoretical foundations of cooperative problem solving [67, 60, 61]. 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, goals, and intentions 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. [22, p. 214]

Following [15, 16], 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.

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" [22, p. 221]. On top of this framework, they introduce a number of derived constructs, which constitute a "partial theory of rational action" [22, p. 221]; 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 24.1. 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

Table 24.1. Atomic modalities in Cohen and Levesque's logic

Operator	Meaning
(Bel i φ)	agent i believes φ
(Goal i φ)	agent i has goal of φ
(Happens α)	action α will happen next
(Done α)	action α has just happened

it believes will not happen. Worlds in the formalism are a discrete sequence of events, stretching infinitely into past and future. The system is only defined semantically, and Cohen and Levesque derive a number of properties from that. In the semantics, a number of assumptions are implicit, and one might vary on them. For instance, there is a fixed domain assumption, giving us properties as $\forall x(\mathsf{Bel}\ i\ \varphi(x)) \rightarrow (\mathsf{Bel}\ i\ \forall x\varphi(x))$. Also, agents 'know what time it is', we immediately obtain from the semantics the validity of formulas like $2:30\mathsf{PM}/3/6/85 \rightarrow \mathsf{Bel}\ i2:30\mathsf{PM}/3/6/85$.

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 [41]. The two most important of these constructors are "$;$" and "$?$":

$\alpha; \alpha'$ denotes α followed by α'

$\varphi?$ denotes a "test action" φ

Here, the test must be interpreted as a test by the system; it is not a so-called 'knowledge-producing action' that can be used by the agent to acquire knowledge.

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:

$$\Diamond\alpha \mathrel{\hat{=}} \exists x \cdot (\mathsf{Happens}\ x; \alpha?)$$
$$\Box\alpha \mathrel{\hat{=}} \neg\Diamond\neg\alpha$$
$$(\mathsf{Later}\ p) \mathrel{\hat{=}} \neg p \wedge \Diamond p$$

A temporal precedence operator, $(\mathsf{Before}\ pq)$ can also be derived, and holds if p holds before q. An important assumption is that *all* goals are eventually dropped:

$$\Diamond\neg(\mathsf{Goal}\ x\ (\mathsf{Later}\ p))$$

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

$$(\mathsf{P\text{-}Goal}\ i\ p) \mathrel{\hat{=}} (\mathsf{Goal}\ i\ (\mathsf{Later}\ p)) \wedge (\mathsf{Bel}\ i\ \neg p) \wedge \left[\begin{array}{l}\mathsf{Before} \\ \quad ((\mathsf{Bel}\ i\ p) \vee (\mathsf{Bel}\ i\ \Box\neg p)) \\ \quad \neg(\mathsf{Goal}\ i\ (\mathsf{Later}\ p))\end{array}\right]$$

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 [22]. An agent i intends to perform action α if it has a

persistent goal to have brought about a state where it had just believed it was about to perform α, and then did α.

$$(\text{Intend } i\ \alpha) \hat{=} (\text{P-Goal } i\ [\text{Done } i\ (\text{Bel } i\ (\text{Happens } \alpha))?; \alpha])$$

Cohen and Levesque go on to show how such a definition meets many of Bratman's criteria for a theory of intention (outlined above). In particular, by basing the definition of intention on the notion of a persistent goal, Cohen and Levesque are able to avoid overcommitment or undercommitment. An agent will only drop an intention if it believes that the intention has either been achieved, or is unachievable.

A critique of Cohen and Levesque's theory of intention is presented in [102]; space restrictions prevent a discussion here.

24.2.4 Rao and Georgeff's BDI Logics

One of the best-known (and most widely misunderstood) approaches to reasoning about rational agents is the *belief-desire-intention* (BDI) model [17]. The BDI model gets its name from the fact that it recognizes the primacy of beliefs, desires, and intentions in rational action. The BDI model is particularly interesting because it combines three distinct components:

- A *philosophical foundation*.
  The BDI model is based on a widely respected theory of rational action in humans, developed by the philosopher Michael Bratman [15].

- A *software architecture*.
  The BDI model of agency does not prescribe a specific implementation. The model may be realized in many different ways, and indeed a number of different implementations of it have been developed. However, the fact that the BDI model *has* been implemented successfully is a significant point in its favor. Moreover, the BDI model has been used to build a number of significant real-world applications, including such demanding problems as fault diagnosis on the space shuttle.

- A *logical formalization*.
  The third component of the BDI model is a family of logics. These logics capture the key aspects of the BDI model as a set of logical axioms. There are many candidates for a formal theory of rational agency, but BDI logics in various forms have proved to be among the most useful, longest-lived, and widely accepted.

Intuitively, an agent's *beliefs* correspond to information the agent has about the world. These beliefs may be incomplete or incorrect. An agent's *desires* represent states of affairs that the agent would, in an ideal world, wish to be brought about. (Implemented BDI agents require that desires be *consistent* with one another, although *human* desires often fail in this respect.) Finally, an agent's *intentions* represent desires

that it has *committed* to achieving. The intuition is that an agent will not, in general, be able to achieve *all* its desires, even if these desires *are* consistent. Ultimately, an agent must therefore fix upon some subset of its desires and commit resources to achieving them. These chosen desires, to which the agent has some commitment, are intentions [22]. The BDI theory of human rational action was originally developed by Michael Bratman [15]. It is a theory of *practical reasoning*—the process of reasoning that we all go through in our everyday lives, deciding moment by moment which action to perform next. Bratman's theory focuses in particular on the role that *intentions* play in practical reasoning. Bratman argues that intentions are important because they constrain the reasoning an agent is required to do in order to select an action to perform. For example, suppose I have an intention to write a book. Then while deciding what to do, I need not expend any effort considering actions that are incompatible with this intention (such as having a summer holiday, or enjoying a social life). This reduction in the number of possibilities I have to consider makes my decision making considerably simpler than would otherwise be the case. Since any real agent we might care to consider—and in particular, any agent that we can implement on a computer—must have resource bounds, an intention-based model of agency, which constrains decision-making in the manner described, seems attractive.

The BDI model has been implemented several times. Originally, it was realized in IRMA, the Intelligent Resource-bounded Machine Architecture [17]. IRMA was intended as a more or less direct realization of Bratman's theory of practical reasoning. However, the best-known implementation is the Procedural Reasoning System (PRS) [37] and its many descendants [32, 88, 26, 57]. In the PRS, an agent has data structures that explicitly correspond to beliefs, desires, and intentions. A PRS agent's beliefs are directly represented in the form of PROLOG-like facts [21, p. 3]. Desires and intentions in PRS are realized through the use of a *plan library*.[1] A plan library, as its name suggests, is a collection of plans. Each plan is a recipe that can be used by the agent to achieve some particular state of affairs. A plan in the PRS is characterized by a *body* and an *invocation condition*. The body of a plan is a course of action that can be used by the agent to achieve some particular state of affairs. The invocation condition of a plan defines the circumstances under which the agent should "consider" the plan. Control in the PRS proceeds by the agent continually updating its internal beliefs, and then looking to see which plans have invocation conditions that correspond to these beliefs. The set of plans made active in this way correspond to the *desires* of the agent. Each desire defines a possible course of action that the agent may follow. On each control cycle, the PRS picks one of these desires, and pushes it onto an execution stack, for subsequent execution. The execution stack contains desires that have been chosen by the agent, and thus corresponds to the agent's *intentions*.

The third and final aspect of the BDI model is the logical component, which gives us a family of tools that allow us to reason about BDI agents. There have been several versions of BDI logic, starting in 1991 and culminating in Rao and Georgeff's 1998 paper on systems of BDI logics [92, 96, 93–95, 89, 91]; a book-length survey was published as [112]. We focus on [112].

Syntactically, BDI logics are essentially branching time logics (CTL or CTL*, depending on which version you are reading about), enhanced with additional modal

[1] In this description of the PRS, we have modified the original terminology somewhat, to be more in line with contemporary usage; we have also simplified the control cycle of the PRS slightly.

operators Bel, Des, and Intend, for capturing the beliefs, desires, and intentions of agents respectively. The BDI modalities are indexed with agents, so, for example, the following is a legitimate formula of BDI logic

$$(\mathsf{Bel}\ i\ (\mathsf{Intend}\ j\ \mathsf{A} \Diamond\ p)) \rightarrow (\mathsf{Bel}\ i\ (\mathsf{Des}\ j\ \mathsf{A} \Diamond\ p))$$

This formula says that if i believes that j intends that p is inevitably true eventually, then i believes that j desires p is inevitable. Although they share much in common with Cohen–Levesque's intention logics, the first and most obvious distinction between BDI logics and the Cohen–Levesque approach is the explicit starting point of CTL-like branching time logics. However, the differences are actually much more fundamental than this. The semantics that Rao and Georgeff give to BDI modalities in their logics are based on the conventional apparatus of Kripke structures and possible worlds. However, rather than assuming that worlds are instantaneous states of the world, or even that they are linear sequences of states, it is assumed instead that worlds are themselves branching temporal structures: thus each world can be viewed as a Kripke structure for a CTL-like logic. While this tends to rather complicate the semantic machinery of the logic, it makes it possible to define an interesting array of semantic properties, as we shall see below.

Before proceeding, we summarize the key semantic structures in the logic. Instantaneous states of the world are modeled by *time points*, given by a set T; the set of all possible evolutions of the system being modeled is given by a binary relation $R \subseteq T \times T$. A *world* (over T and R) is then a pair $\langle T', R' \rangle$, where $T' \subseteq T$ is a non-empty set of time points, and $R' \subseteq R$ is a branching time structure on T'. Let W be the set of all worlds over T. A pair $\langle w, t \rangle$, where $w \in W$ and $t \in T$, is known as a *situation*. If $w \in W$, then the set of all situations in w is denoted by S_w. We have belief accessibility relations B, D, and I, modeled as functions that assign to every agent a relation over situations. Thus, for example:

$$B : \mathit{Agents} \rightarrow \wp(W \times T \times W).$$

We write $B_t^w(i)$ to denote the set of worlds accessible to agent i from situation $\langle w, t \rangle$: $B_t^w(i) = \{w' \mid \langle w, t, w' \rangle \in B(i)\}$. We define D_t^w and I_t^w in the obvious way. The semantics of belief, desire and intention modalities are then given in the conventional manner:

- $\langle w, t \rangle \models (\mathsf{Bel}\ i\ \varphi)$ iff $\langle w', t \rangle \models \varphi$ for all $w' \in B_t^w(i)$.
- $\langle w, t \rangle \models (\mathsf{Des}\ i\ \varphi)$ iff $\langle w', t \rangle \models \varphi$ for all $w' \in D_t^w(i)$.
- $\langle w, t \rangle \models (\mathsf{Intend}\ i\ \varphi)$ iff $\langle w', t \rangle \models \varphi$ for all $w' \in I_t^w(i)$.

The primary focus of Rao and Georgeff's early work was to explore the possible interrelationships between beliefs, desires, and intentions from the perspective of semantic characterization. In order to do this, they defined a number of possible interrelationships between an agent's belief, desire, and intention accessibility relations. The most obvious relationships that can exist are whether one relation is a subset of another: for example, if $D_t^w(i) \subseteq I_t^w(i)$ for all i, w, t, then we would have as an interaction axiom $(\mathsf{Intend}\ i\ \varphi) \rightarrow (\mathsf{Des}\ i\ \varphi)$. However, the fact that worlds themselves have structure in BDI logic also allows us to combine such properties with relations on

the *structure* of worlds themselves. The most obvious structural relationship that can exist between two worlds—and the most important for our purposes—is that of one world being a *subworld* of another. Intuitively, a world w is said to be a subworld of world w' if w has the same structure as w' but has fewer paths *and is otherwise identical*. Formally, if w, w' are worlds, then w is a subworld of w' (written $w \sqsubseteq w'$) iff $paths(w) \subseteq paths(w')$ but w, w' agree on the interpretation of predicates and constants in common time points.

The first property we consider is the *structural subset* relationship between accessibility relations. We say that accessibility relation R is a structural subset of accessibility relation $\bar{R}$ if for every R-accessible world w, there is an $\bar{R}$-accessible world w' such that w is a subworld of w'. Formally, if R and $\bar{R}$ are two accessibility relations then we write $R \subseteq_{sub} \bar{R}$ to indicate that if $w' \in R_t^w(i)$, then there exists some $w'' \in \bar{R}_t^w(i)$ such that $w' \sqsubseteq w''$. If $R \subseteq_{sub} \bar{R}$, then we say R is a *structural subset* of $\bar{R}$.

We write $\bar{R} \subseteq_{sup} R$ to indicate that if $w' \in R_t^w(i)$, then there exists some $w'' \in \bar{R}_t^w(i)$ such that $w'' \sqsubseteq w'$. If $R \subseteq_{sup} \bar{R}$, then we say R is a *structural superset* of $\bar{R}$. In other words, if R is a structural superset of $\bar{R}$, then for every R-accessible world w, there is an $\bar{R}$-accessible world w' such that w' is a subworld of w.

Finally, we can also consider whether the *intersection* of accessibility relations is empty or not. For example, if $B_t^w(i) \cap I_t^w(i) \neq \emptyset$, for all i, w, t, then we get the following interaction axiom:

$$(\mathsf{Intend}\ i\ \varphi) \rightarrow \neg(\mathsf{Bel}\ i\ \neg\varphi).$$

This axiom expresses an *inter-modal consistency* property. Just as we can undertake a more fine-grained analysis of the basic interactions among beliefs, desires, and intentions by considering the structure of worlds, so we are also able to undertake a more fine-grained characterization of inter-modal consistency properties by taking into account the structure of worlds. We write $R_t^w(i) \cap_{sup} \bar{R}_t^w(i)$ to denote the set of worlds $w' \in \bar{R}_t^w(i)$ for which there exists some world $w'' \in R_t^w(i)$ such that $w' \sqsubseteq w''$. We can then define $\cap_{sub}$ in the obvious way.

Putting all these relations together, we can define a range of BDI logical systems. The most obvious possible systems, and the semantic properties that they correspond to, are summarized in Table 24.2.

24.2.5 The KARO Framework

The KARO framework (for Knowledge, Actions, Results and Opportunities) is an attempt to develop and formalize the ideas of Moore [76], who realized that dynamic and epistemic logic can be perfectly combined into one modal framework. The basic framework comes with a sound and complete axiomatization [70]. Also, results on automatic verification of the theory are known, both using translations to first order logic, as well as in a clausal resolution approach. The core of KARO is a combination of epistemic (the standard knowledge operator $\mathbf{K}_i$ is an S5-operator) and dynamic logic; many extensions have also been studied.

Along with the notion of the result of events, the notions of ability and opportunity are among the most discussed and investigated in analytical philosophy. Ability plays an important part in various philosophical theories, as, for instance, the theory of free

Table 24.2. Systems of BDI logic

Name	Semantic condition	Corresponding formula schema
BDI-S1	$B \subseteq_{sup} D \subseteq_{sup} I$	$(\mathsf{Intend}\ i\ \mathsf{E}(\varphi)) \rightarrow (\mathsf{Des}\ i\ \mathsf{E}(\varphi)) \rightarrow (\mathsf{Bel}\ i\ \mathsf{E}(\varphi))$
BDI-S2	$B \subseteq_{sub} D \subseteq_{sub} I$	$(\mathsf{Intend}\ i\ \mathsf{A}(\varphi)) \rightarrow (\mathsf{Des}\ i\ \mathsf{A}(\varphi)) \rightarrow (\mathsf{Bel}\ i\ \mathsf{A}(\varphi))$
BDI-S3	$B \subseteq D \subseteq I$	$(\mathsf{Intend}\ i\ \varphi) \rightarrow (\mathsf{Des}\ i\ \varphi) \rightarrow (\mathsf{Bel}\ i\ \varphi)$
BDI-R1	$I \subseteq_{sup} D \subseteq_{sup} B$	$(\mathsf{Bel}\ i\ \mathsf{E}(\varphi)) \rightarrow (\mathsf{Des}\ i\ \mathsf{E}(\varphi)) \rightarrow (\mathsf{Intend}\ i\ \mathsf{E}(\varphi))$
BDI-R2	$I \subseteq_{sub} D \subseteq_{sub} B$	$(\mathsf{Bel}\ i\ \mathsf{A}(\varphi)) \rightarrow (\mathsf{Des}\ i\ \mathsf{A}(\varphi)) \rightarrow (\mathsf{Intend}\ i\ \mathsf{A}(\varphi))$
BDI-R3	$I \subseteq D \subseteq B$	$(\mathsf{Bel}\ i\ \varphi) \rightarrow (\mathsf{Des}\ i\ \varphi) \rightarrow (\mathsf{Intend}\ i\ \varphi)$
BDI-W1	$B \cap_{sup} D \neq \emptyset$	$(\mathsf{Bel}\ i\ \mathsf{A}(\varphi)) \rightarrow \neg(\mathsf{Des}\ i\ \neg\mathsf{A}(\varphi))$
	$D \cap_{sup} I \neq \emptyset$	$(\mathsf{Des}\ i\ \mathsf{A}(\varphi)) \rightarrow \neg(\mathsf{Intend}\ i\ \neg\mathsf{A}(\varphi))$
	$B \cap_{sup} I \neq \emptyset$	$(\mathsf{Bel}\ i\ \mathsf{A}(\varphi)) \rightarrow \neg(\mathsf{Intend}\ i\ \neg\mathsf{A}(\varphi))$
BDI-W2	$B \cap_{sub} D \neq \emptyset$	$(\mathsf{Bel}\ i\ \mathsf{E}(\varphi)) \rightarrow \neg(\mathsf{Des}\ i\ \neg\mathsf{E}(\varphi))$
	$D \cap_{sub} I \neq \emptyset$	$(\mathsf{Des}\ i\ \mathsf{E}(\varphi)) \rightarrow \neg(\mathsf{Intend}\ i\ \neg\mathsf{E}(\varphi))$
	$B \cap_{sub} I \neq \emptyset$	$(\mathsf{Bel}\ i\ \mathsf{E}(\varphi)) \rightarrow \neg(\mathsf{Intend}\ i\ \neg\mathsf{E}(\varphi))$
BDI-W3	$B \cap D \neq \emptyset$	$(\mathsf{Bel}\ i\ \varphi) \rightarrow \neg(\mathsf{Des}\ i\ \neg\varphi)$
	$D \cap I \neq \emptyset$	$(\mathsf{Des}\ i\ \varphi) \rightarrow \neg(\mathsf{Intend}\ i\ \neg\varphi)$
	$B \cap I \neq \emptyset$	$(\mathsf{Bel}\ i\ \varphi) \rightarrow \neg(\mathsf{Intend}\ i\ \neg\varphi)$

Source: [91, p. 321].

will and determinism, the theory of refraining and seeing-to-it, and deontic theories. Following Kenny [62], the authors behind KARO consider ability to be the complex of physical, mental and moral capacities, internal to an agent, and being a positive explanatory factor in accounting for the agent's performing an action. Opportunity, on the other hand, is best described as circumstantial possibility, i.e., possibility by virtue of the circumstances. The opportunity to perform some action is external to the agent and is often no more than the absence of circumstances that would prevent or interfere with the performance. Although essentially different, abilities and opportunities are interconnected in that abilities can be exercised only when opportunities for their exercise present themselves, and opportunities can be taken only by those who have the appropriate abilities. From this point of view it is important to remark that abilities are understood to be *reliable* (cf. [18]), i.e., having the ability to perform a certain action suffices to take the opportunity to perform the action every time it presents itself. The combination of ability and opportunity determines whether or not an agent has the (practical) possibility to perform an action.

Let i be a variable over a set of agents $\{1, \ldots, n\}$. Actions in the set Ac are either atomic actions ($\mathrm{Ac} = \{a, b, \ldots\}$) or composed ($\alpha, \beta, \ldots$) by means of confirmation of formulas ($\texttt{confirm}\ \varphi$), sequencing ($\alpha; \beta$), conditioning ($\texttt{if}\ \varphi\ \texttt{then}\ \alpha\ \texttt{else}\ \beta$) and repetition ($\texttt{while}\ \varphi\ \texttt{do}\ \alpha$). These actions α can then be used to build new formulas to express the possible *result* of the execution of α by agent i (the formula $[do_i(\alpha)]\varphi$ denotes that φ is a result of i's execution of α), the *opportunity* for i to perform α ($\langle do_i(\alpha)\rangle\top$) and i's *capability* of performing the action α ($\mathbf{A}_i\alpha$). The formula $\langle do_i(\alpha)\rangle\varphi$ is shorthand for $\neg[do_i(\alpha)]\neg\varphi$, thus expressing that one possible result of performance of α by i implies φ.

With these tools at hand, one has already a rich framework to reason about agent's knowledge about doing actions. For instance, a property like *perfect recall*

$$\mathbf{K}_i[do_i(\alpha)]\varphi \rightarrow [do_i(\alpha)]\mathbf{K}_i\varphi$$

can now be enforced for *particular actions* α. Also, the core KARO already guarantees a number of properties, of which we list a few:

1. $\mathbf{A}_i\ \mathtt{confirm}\,\varphi \leftrightarrow \varphi$.
2. $\mathbf{A}_i\alpha_1;\alpha_2 \leftrightarrow \mathbf{A}_i\alpha_1 \wedge [do_i(\alpha_1)]\mathbf{A}_i\alpha_2$ or $\mathbf{A}_i\alpha_1;\alpha_2 \leftrightarrow \mathbf{A}_i\alpha_1 \wedge \langle do_i(\alpha_1)\rangle\mathbf{A}_i\alpha_2$.
3. $\mathbf{A}_i\ \mathtt{if}\,\varphi\,\mathtt{then}\,\alpha_1\,\mathtt{else}\,\alpha_2\,\mathtt{fi} \leftrightarrow ((\varphi \wedge \mathbf{A}_i\alpha_1) \vee (\neg\varphi \wedge \mathbf{A}_i\alpha_2))$.
4. $\mathbf{A}_i\ \mathtt{while}\,\varphi\,\mathtt{do}\,\alpha\,\mathtt{od} \leftrightarrow (\neg\varphi \vee (\varphi \wedge \mathbf{A}_i\alpha \wedge [do_i(\alpha)]\mathbf{A}_i\ \mathtt{while}\,\varphi\,\mathtt{do}\,\alpha\,\mathtt{od}))$ or $\mathbf{A}_i\ \mathtt{while}\,\varphi\,\mathtt{do}\,\alpha\,\mathtt{od} \leftrightarrow (\neg\varphi \vee (\varphi \wedge \mathbf{A}_i\alpha \wedge \langle do_i(\alpha)\rangle\mathbf{A}_i\ \mathtt{while}\,\varphi\,\mathtt{do}\,\alpha\,\mathtt{od}))$.

For a discussion about the problems with the ability to do a sequential action (the possible behavior of the items 2 and 4 above), we refer to [70], or to a general solution to this problem that was offered in [48].

Practical possibility is considered to consist of two parts, viz. correctness and feasibility: action α is *correct* with respect to φ iff $\langle do_i(\alpha)\rangle\varphi$ holds and α is *feasible* iff $\mathbf{A}_i\alpha$ holds.

$$\mathbf{PracPoss}_i(\alpha,\varphi) \langle do_i(\alpha)\rangle\varphi \wedge \mathbf{A}_i\alpha.$$

The importance of practical possibility manifests itself particularly when ascribing —from the outside—certain qualities to an agent. It seems that for the agent itself practical possibilities are relevant in so far as the agent has knowledge of these possibilities. To formalize this kind of knowledge, KARO comes with a Can-predicate and a Cannot-predicate. The first of these predicates concerns the knowledge of agents about their practical possibilities, the latter predicate does the same for their practical impossibilities.

$$\mathbf{Can}_i(\alpha,\varphi) \stackrel{\triangle}{=} \mathbf{K}_i\mathbf{PracPoss}_i(\alpha,\varphi) \quad \text{and}$$
$$\mathbf{Cannot}_i(\alpha,\varphi) \stackrel{\triangle}{=} \mathbf{K}_i\neg\mathbf{PracPoss}_i(\alpha,\varphi).$$

The Can-predicate and the Cannot-predicate integrate knowledge, ability, opportunity and result, and seem to formalize one of the most important notions of agency. In fact it is probably not too bold to say that knowledge like that formalized through the Can-predicate, although perhaps in a weaker form by taking aspects of uncertainty into account, underlies all acts performed by rational agents. For rational agents act only if they have some information on both the possibility to perform the act, and its possible outcome. It therefore seems worthwhile to take a closer look at both the Can-predicate and the Cannot-predicate. The following properties focus on the behavior of the *means*-part of the predicates, which is the α in $\mathbf{Can}_i(\alpha,\varphi)$ and $\mathbf{Cannot}_i(\alpha,\varphi)$.

1. $\mathbf{Can}_i(\mathtt{confirm}\,\varphi,\psi) \leftrightarrow \mathbf{K}_i(\varphi \wedge \psi)$.
2. $\mathbf{Cannot}_i(\mathtt{confirm}\,\varphi,\psi) \leftrightarrow \mathbf{K}_i(\neg\varphi \vee \neg\psi)$.
3. $\mathbf{Can}_i(\alpha_1;\alpha_2,\varphi) \leftrightarrow \mathbf{Can}_i(\alpha_1,\mathbf{PracPoss}_i(\alpha_2,\varphi))$.
4. $\mathbf{Can}_i(\alpha_1;\alpha_2,\varphi) \rightarrow \langle do_i(\alpha_1)\rangle\mathbf{Can}_i(\alpha_2,\varphi)$ if i has perfect recall regarding α_1.
5. $\mathbf{Can}_i(\mathtt{if}\,\varphi\,\mathtt{then}\,\alpha_1\,\mathtt{else}\,\alpha_2\,\mathtt{fi},\psi) \wedge \mathbf{K}_i\varphi \leftrightarrow \mathbf{Can}_i(\alpha_1,\psi) \wedge \mathbf{K}_i\varphi$.

6. $\mathbf{Can}_i(\texttt{if}\ \varphi\ \texttt{then}\ \alpha_1\ \texttt{else}\ \alpha_2\ \texttt{fi}, \psi) \wedge \mathbf{K}_i\neg\varphi \leftrightarrow \mathbf{Can}_i(\alpha_2, \psi) \wedge \mathbf{K}_i\neg\varphi$.
7. $\mathbf{Can}_i(\texttt{while}\ \varphi\ \texttt{do}\ \alpha\ \texttt{od}, \psi) \wedge \mathbf{K}_i\varphi \leftrightarrow \mathbf{Can}_i(\alpha, \mathbf{PracPoss}_i(\texttt{while}\ \varphi\ \texttt{do}\ \alpha\ \texttt{od}, \psi)) \wedge \mathbf{K}_i\varphi$.

In *Actions that make you change your mind* [69], the authors of KARO look at specific atomic actions. At that the agents can perform, i.e., doxastic actions of expanding, contracting or revising its beliefs (we have now both knowledge ($\mathbf{K}_i$) and belief ($\mathbf{B}_i$)). Those actions are assumed to have the following general properties:

- $\models \langle do_i(\alpha)\rangle\top$ realizability
- $\models \langle do_i(\alpha)\rangle\chi \rightarrow [do_i(\alpha)]\chi$ determinism
- $\models \langle do_i(\alpha;\alpha)\rangle\chi \leftrightarrow \langle do_i(\alpha)\rangle\chi$ idempotence

Realizability of an action implies that agents have the opportunity to perform the action regardless of circumstances; determinism of an action means that performing the action results in a unique state of affairs, and idempotence of an action implies that performing the action an arbitrary number of times has the same effect as performing the action just once.

Then, specific definitions for the three actions are given, and related to the AGM framework of belief revision [4]. As an illustration, we list some properties, written in one object language, of the action of revising one's beliefs (here, φ is an objective formula):

- $[do_i(\texttt{revise}\ \varphi)]\mathbf{B}_i\varphi$.
- $[do_i(\texttt{revise}\ \varphi)]\mathbf{B}_i\vartheta \rightarrow [do_i(\texttt{expand}\ \varphi)]\mathbf{B}_i\vartheta$.
- $\neg\mathbf{B}_i\neg\varphi \rightarrow ([do_i(\texttt{expand}\ \varphi)]\mathbf{B}_i\vartheta \leftrightarrow [do_i(\texttt{revise}\ \varphi)]\mathbf{B}_i\vartheta)$.
- $\mathbf{K}_i\neg\varphi \leftrightarrow [do_i(\texttt{revise}\ \varphi)]\mathbf{B}_i\bot$.
- $\mathbf{K}_i(\varphi \leftrightarrow \psi) \rightarrow ([do_i(\texttt{revise}\ \varphi)]\mathbf{B}_i\vartheta \leftrightarrow [do_i(\texttt{revise}\ \psi)]\mathbf{B}_i\vartheta)$.

In [74], the KARO-authors show how motivational attitudes can be incorporated in their framework. The most primitive notion here is that agent i *wishes* φ ($\mathbf{W}_i\varphi$), from which it has to *select* some (if so, $\mathbf{C}_i\varphi$ becomes true). In order to define what a goal is, a higher order notion of *implementability* is first defined:

$$\Diamond_i\varphi \Leftrightarrow \exists k \in \mathbb{N}\,\exists a_1, \ldots, a_k \in \mathrm{At}\mathbf{PracPoss}_i(a_1; \ldots; a_k, \varphi)).$$

Now the notion of a goal in KARO is as follows:

$$\mathbf{Goal}_i\varphi \stackrel{\triangle}{=} \mathbf{W}_i\varphi \wedge \neg\varphi \wedge \Diamond_i\varphi \wedge \mathbf{C}_i\varphi.$$

It is easily seen that this definition of a goal does not suffer from effects as being closed under consequence. In [74], these motivational attitudes are also 'dynamized', in the sense that actions, like committing and decommitting are added, with which an agent can change its motivational attitudes. Semantically, this is supported by letting the agents maintain an "agenda". Space does not permit us to investigate this issue further.

24.2.6 Discussion

Undoubtedly, formalizing the informational and motivational attitudes in a context with evolving time, or where agents can do actions, have greatly helped to improve our understanding of complex systems. At the same time, admittedly, there are many weaknesses and open problems with such approaches.

To give one example of how a formalization can help us to become more clear about the interrelationship between the notions defined here, recall that Rao and Georgeff assume the notion of *belief-goal compatibility*, saying

$$\mathbf{Goal}_i \varphi \rightarrow \mathbf{B}_i \varphi$$

for formulas φ that refer to the future.

Cohen and Levesque, however, put a lot of emphasis on their notion of *realizability*, stating exactly the opposite:

$$\mathbf{B}_i \varphi \rightarrow \mathbf{Goal}_i \varphi.$$

By analyzing the framework of Cohen and Levesque more closely, it appears that they have a much weaker property in mind, which is

$$\mathbf{Goal}_i \varphi \rightarrow \neg \mathbf{B}_i \neg \varphi.$$

To mention just one aspect in which the approaches mentioned here are still far from completed, we recall that the three frameworks allow one to reason about many agents, but are in essence still one-agent systems. Where notions as distributed and common knowledge are well understood epistemic notions in multi-agent systems, their motivational analogues seem to be much harder and are yet only partially understood (see Cohen and Levesque's [24] or Tambe's [104] on teamwork).

24.2.7 Cognitive Agent Logics in Practice

Broadly speaking, logic has played a role in three aspects of software development.

- as a *specification language*;
- as a *programming language*; and
- as a *verification language*.

In the sections that follow, we will discuss the possible use of logics of rational agency in these three processes.

Specification

The software development process begins by establishing the client's requirements. When this process is complete, a *specification* is developed, which sets out the functionality of the new system. Temporal and dynamic logics have found wide applicability in the specification of systems. An obvious question is therefore whether logics of rational agency might be used as specification languages.

A specification expressed in such a logic would be a formula φ. The idea is that such a specification would express the desirable behavior of a system. To see how this

might work, consider the following formula of BDI logic (in fact from [112]), intended to form part of a specification of a process control system.

$$(\mathsf{Bel}\ i\ \mathit{Open}(\mathit{valve}32)) \rightarrow (\mathsf{Intend}\ i\ (\mathsf{Bel}\ j\ \mathit{Open}(\mathit{valve}32))).$$

This formula says that if i believes valve 32 is open, then i should intend that j believes valve 32 is open. A rational agent i with such an intention can select a speech act to perform in order to inform j of this state of affairs. It should be intuitively clear how a system specification might be constructed using such formulae, to define the intended behavior of a system.

One of the main desirable features of a software specification language is that it should not dictate *how* a specification should be satisfied by an implementation. It should be clear that the specification above has exactly these properties. It does not dictate how agent i should go about making j aware that valve 32 is open. We simply expect i to behave as a rational agent given such an intention.

There are a number of problems with the use of such logics for specification. The most worrying of these is with respect to their semantics. The semantics for the modal connectives (for beliefs, desires, and intentions) are given in the normal modal logic tradition of possible worlds [19]. So, for example, an agent's beliefs in some state are characterized by a set of different states, each of which represents one possibility for how the world could actually be, given the information available to the agent. In much the same way, an agent's desires in some state are characterized by a set of states that are consistent with the agent's desires. Intentions are represented similarly. There are several advantages to the possible worlds model: it is well studied and well understood, and the associated mathematics of correspondence theory is extremely elegant. These attractive features make possible worlds the semantics of choice for almost every researcher in formal agent theory. However, there are also a number of serious drawbacks to possible worlds semantics. First, possible worlds semantics imply that agents are logically perfect reasoners (in that their deductive capabilities are sound and complete), and they have infinite resources available for reasoning. No real agent, artificial or otherwise, has these properties.

Second, possible worlds semantics are generally *ungrounded*. That is, there is usually no precise relationship between the abstract accessibility relations that are used to characterize an agent's state, and any concrete computational model. As we shall see in later sections, this makes it difficult to go from a formal specification of a system in terms of beliefs, desires, and so on, to a concrete computational system. Similarly, given a concrete computational system, there is generally no way to determine what the beliefs, desires, and intentions of that system are. If temporal modal logics of rational agency are to be taken seriously as *specification* languages, then this is a significant problem.

Implementation

Once given a specification, we must implement a system that is correct with respect to this specification. The next issue we consider is this move from abstract specification to concrete computational system. There are at least two possibilities for achieving this transformation that we consider here:

1. somehow directly execute or animate the abstract specification; or

2. somehow translate or compile the specification into a concrete computational form using an automatic translation technique.

In the subsections that follow, we shall investigate each of these possibilities in turn.

Directly executing agent specifications. Suppose we are given a system specification, φ, which is expressed in some logical language L. One way of obtaining a concrete system from φ is to treat it as an *executable specification*, and *interpret* the specification directly in order to generate the agent's behavior. Interpreting an agent specification can be viewed as a kind of constructive proof of satisfiability, whereby we show that the specification φ is satisfiable by *building a model* (in the logical sense) for it. If models for the specification language L can be given a computational interpretation, then model building can be viewed as executing the specification. To make this discussion concrete, consider the Concurrent METATEM programming language [33]. In this language, agents are programmed by giving them a temporal logic specification of the behavior it is intended they should exhibit; this specification is directly executed to generate each agent's behavior. Models for the temporal logic in which Concurrent METATEM agents are specified are linear discrete sequences of states: executing a Concurrent METATEM agent specification is thus a process of constructing such a sequence of states. Since such state sequences can be viewed as the histories traced out by programs as they execute, the temporal logic upon which Concurrent METATEM is based has a computational interpretation; the actual execution algorithm is described in [12]. A somewhat related language is the IMPACT framework of Subrahmanian et al. [103]. IMPACT is a rich framework for programming agents, which draws upon and considerably extends some ideas from logic programming. Agents in IMPACT are programmed by using rules that incorporate deontic modalities (permitted, forbidden, obliged [75]). These rules can be interpreted to determine the actions that an agent should perform at any given moment [103, p. 171].

Note that executing Concurrent METATEM agent specifications is possible primarily because the models upon which the Concurrent METATEM temporal logic is based are comparatively simple, with an obvious and intuitive computational interpretation. However, agent specification languages in general (e.g., the BDI formalisms of Rao and Georgeff [90]) are based on considerably more complex logics. In particular, they are usually based on a semantic framework known as *possible worlds* [19]. The technical details are somewhat involved but the main point is that, *in general*, possible worlds semantics do not have a computational interpretation in the way that Concurrent METATEM semantics do. Hence it is not clear what "executing" a logic based on such semantics might mean.

In response to this issue, a number of researchers have attempted to develop executable agent specification languages with a simplified logical basis, that has a computational interpretation. An example is Rao's AgentSpeak(L) language, which although essentially a BDI system, has a simple computational semantics [88]. The 3APL project [45] is also an attempt to have a agent programming language with a well-defined semantics, based on transition systems. One advantage of having a thorough semantics is that it enables one to compare different agent programming languages, such as AgentSpeak(L) with 3APL [44] or AGENT-0 and 3APL [46]. One complication in bridging the gap between the agent programming paradigm and the formal systems of Sections 24.2.3–24.2.5, is that the former usually take goals to be procedural

(a plan), whereas goals in the latter are declarative (a desired state). A programming language that tries to bridge the gap in this respect is the language GOAL [64].

GOLOG [66, 97] and its multiagent sibling CONGOLOG [65] represent another rich seam of work on logic-oriented approaches to programming rational agents. Essentially, GOLOG is a framework for executing a fragment of the situation calculus; the situation calculus is a well known logical framework for reasoning about action [73]. Put crudely, writing a GOLOG program involves expressing a logical theory of what action an agent should perform, using the situation calculus; this theory, together with some background axioms, represents a logical expression of what it means for the agent to do the right action. Executing such a program reduces to constructively solving a deductive proof problem, broadly along the lines of showing that there is a sequence of actions representing an acceptable computation according to the theory [97, p. 121]; the witness to this proof will be a sequence of actions, which can then be executed.

Compiling agent specifications. An alternative to direct execution is *compilation*. In this scheme, we take our abstract specification, and transform it into a concrete computational model via some automatic synthesis process. The main perceived advantages of compilation over direct execution are in run-time efficiency. Direct execution of an agent specification, as in Concurrent METATEM, above, typically involves manipulating a symbolic representation of the specification at run time. This manipulation generally corresponds to reasoning of some form, which is computationally costly. Compilation approaches aim to reduce abstract symbolic specifications to a much simpler computational model, which requires no symbolic representation. The 'reasoning' work is thus done off-line, at compile-time; execution of the compiled system can then be done with little or no run-time symbolic reasoning.

Compilation approaches usually depend upon the close relationship between models for temporal/modal logic (which are typically labeled graphs of some kind), and automata-like finite state machines. For example, Pnueli and Rosner [85] synthesize reactive systems from branching temporal logic specifications. Similar techniques have also been used to develop concurrent system skeletons from temporal logic specifications. Perhaps the best-known example of this approach to agent development is the *situated automata* paradigm of Rosenschein and Kaelbling [99]. They use an epistemic logic to specify the perception component of intelligent agent systems. They then used a technique based on constructive proof to directly synthesize automata from these specifications [98].

The general approach of automatic synthesis, although theoretically appealing, is limited in a number of important respects. First, as the agent specification language becomes more expressive, then even offline reasoning becomes too expensive to carry out. Second, the systems generated in this way are not capable of *learning* (i.e., they are not capable of adapting their "program" at run-time). Finally, as with direct execution approaches, agent specification frameworks tend to have no concrete computational interpretation, making such a synthesis impossible.

Verification

Once we have developed a concrete system, we need to show that this system is correct with respect to our original specification. This process is known as *verification*,

and it is particularly important if we have introduced any informality into the development process. We can divide approaches to the verification of systems into two broad classes: (1) *axiomatic*; and (2) *semantic* (model checking). In the subsections that follow, we shall look at the way in which these two approaches have evidenced themselves in agent-based systems.

Axiomatic approaches. Axiomatic approaches to program verification were the first to enter the mainstream of computer science, with the work of Hoare in the late 1960s [47]. Axiomatic verification requires that we can take our concrete program, and from this program systematically derive a logical theory that represents the behavior of the program. Call this the program theory. If the program theory is expressed in the same logical language as the original specification, then verification reduces to a proof problem: show that the specification is a theorem of (equivalently, is a logical consequence of) the program theory. The development of a program theory is made feasible by *axiomatizing* the programming language in which the system is implemented. For example, Hoare logic gives us more or less an axiom for every statement type in a simple PASCAL-like language. Once given the axiomatization, the program theory can be derived from the program text in a systematic way.

Perhaps the most relevant work from mainstream computer science is the specification and verification of reactive systems using temporal logic, in the way pioneered by Pnueli, Manna, and colleagues [72]. The idea is that the computations of reactive systems are infinite sequences, which correspond to models for linear temporal logic. Temporal logic can be used both to develop a system specification, and to axiomatize a programming language. This axiomatization can then be used to systematically derive the theory of a program from the program text. Both the specification and the program theory will then be encoded in temporal logic, and verification hence becomes a proof problem in temporal logic.

Comparatively little work has been carried out within the agent-based systems community on axiomatizing multi-agent environments. We shall review just one approach. In [111], an axiomatic approach to the verification of multi-agent systems was proposed. Essentially, the idea was to use a temporal belief logic to axiomatize the properties of two multi-agent programming languages. Given such an axiomatization, a program theory representing the properties of the system could be systematically derived in the way indicated above. A temporal belief logic was used for two reasons. First, a temporal component was required because, as we observed above, we need to capture the ongoing behavior of a multi-agent system. A belief component was used because the agents we wish to verify are each symbolic AI systems in their own right. That is, each agent is a symbolic reasoning system, which includes a representation of its environment and desired behavior. A belief component in the logic allows us to capture the symbolic representations present within each agent. The two multi-agent programming languages that were axiomatized in the temporal belief logic were Shoham's AGENT0 [101], and Fisher's Concurrent METATEM (see above). Note that this approach relies on the operation of agents being sufficiently simple that their properties can be axiomatized in the logic. It works for Shoham's AGENT0 and Fisher's Concurrent METATEM largely because these languages have a simple semantics, closely related to rule-based systems, which in turn have a simple logical semantics. For more complex agents, an axiomatization is not so straightforward.

Also, capturing the semantics of concurrent execution of agents is not easy (it is, of course, an area of ongoing research in computer science generally).

Semantic approaches: model checking. Ultimately, axiomatic verification reduces to a proof problem. Axiomatic approaches to verification are thus inherently limited by the difficulty of this proof problem. Proofs are hard enough, even in classical logic; the addition of temporal and modal connectives to a logic makes the problem considerably harder. For this reason, more efficient approaches to verification have been sought. One particularly successful approach is that of *model checking* [20]. As the name suggests, whereas axiomatic approaches generally rely on syntactic proof, model checking approaches are based on the semantics of the specification language.

The model checking problem, in abstract, is quite simple: given a formula φ of language L, and a model M for L, determine whether or not φ is valid in M, i.e., whether or not $M \models_L \varphi$. Model checking-based verification has been studied in connection with temporal logic. The technique once again relies upon the close relationship between models for temporal logic and finite-state machines. Suppose that φ is the specification for some system, and π is a program that claims to implement φ. Then, to determine whether or not π truly implements φ, we take π, and from it generate a model M_π that corresponds to π, in the sense that M_π encodes all the possible computations of π; determine whether or not $M_\pi \models \varphi$, i.e., whether the specification formula φ is valid in M_π; the program π satisfies the specification φ just in case the answer is 'yes'. The main advantage of model checking over axiomatic verification is in complexity: model checking using the branching time temporal logic CTL [20] can be done in polynomial time, whereas the proof problem for most modal logics is quite complex.

In [95], Rao and Georgeff present an algorithm for model checking BDI logic. More precisely, they give an algorithm for taking a logical model for their (propositional) BDI agent specification language, and a formula of the language, and determining whether the formula is valid in the model. The technique is closely based on model checking algorithms for normal modal logics [40]. They show that despite the inclusion of three extra modalities (for beliefs, desires, and intentions), into the CTL branching time framework, the algorithm is still quite efficient, running in polynomial time. So the second step of the two-stage model checking process described above can still be done efficiently. However, it is not clear how the first step might be realized for BDI logics. Where does the logical model characterizing an agent actually come from—can it be derived from an arbitrary program π, as in mainstream computer science? To do this, we would need to take a program implemented in, say, JAVA, and from it derive the belief, desire, and intention accessibility relations that are used to give a semantics to the BDI component of the logic. Because, as we noted earlier, there is no clear relationship between the BDI logic and the concrete computational models used to implement agents, it is not clear how such a model could be derived.

One approach to this problem was presented in [113], where an imperative programming language called MABLE was presented, with an explicit BDI semantics. Model checking for the language was implemented by mapping the language to the input language for the SPIN model checking system [56], and by reducing formulae in a restricted BDI language to the Linear Temporal Logic format required by SPIN. Here, for example, is a sample claim that may be made about a MABLE system, which may be automatically verified by model checking:

```
claim
[]
((believe agent2
            (intend agent1
                    (believe agent2 (a == 10))))
   ->
   <>(believe agent2 (a == 10))
);
```

This claim says that it is always ([]) the case that if agent 2 believes that agent 1 intends that agent 2 believes that variable *a* has the value 10, then subsequently (<>), agent 2 will itself believe that *a* has the value 10. MABLE was developed primarily as a testbed for exploring possible semantics for agent communication, and was not intended for large-scale system verification.

Several model checkers for logics combining knowledge, time, and other modalities have become developed in recent years. For example, using techniques similar to those used for CTL model checkers [20], Raimondi and Lomuscio implemented MCMAS, a model checker that supports a variety of epistemic, temporal, and deontic logics [87, 71]. Another recent approach to model checking multi-agent systems is [49], which involves model checking temporal epistemic logics by reducing the model checking problem to a conventional LTL model checking problem.

24.3 Representing the Strategic Structure of a System

The second main strand of research that we describe focuses not on the cognitive states of agents, but on the *strategic structure* of the environment: what agents can achieve, either individually or in groups. The starting point for such formalisms is a model of *strategic ability*.

Over the past three decades, researchers from many disciplines have attempted to develop a general purpose logic of strategic ability. Within the artificial intelligence (AI) community, it was understood that such a logic could be used in order to gain a better understanding of planning systems [31, 68, 5]. The most notable early effort in this direction was Moore's dynamic epistemic logic, described above [76, 77]. Moore's work was subsequently enhanced by many other researchers, perhaps most notably, Morgenstern [78, 79]. These distinctions also informed later attempts to integrate a logic of ability into more general logics of rational action in autonomous agents [115, 112] (see [114] for a survey of such logics).

In a somewhat parallel thread of research, researchers in the philosophy of action developed a range of logics underpinned by rather similar ideas and motivations. A typical example is that of Brown, who developed a logic of individual ability in the mid-1980s [18]. Brown's main claim was that modal logic was a useful tool for the analysis of ability, and that previous—unsuccessful—attempts to characterize ability in modal logic were based on an over-simple semantics. Brown's account of the semantics of ability was as follows [18, p. 5]:

> [An agent can achieve *A*] at a given world iff *there exists* a relevant cluster of worlds, at *every* world of which *A* is true.

Notice the $\exists\forall$ pattern of quantifiers in this account. Brown immediately noted that this gave the resulting logic a rather unusual flavor, neither properly existential nor properly universal [18, p. 5]:

> Cast in this form, the truth condition [for ability] involves *two* metalinguistic quantifiers (one existential and one universal). In fact, [the character of the ability operator] should be a little like each.

More recently, there has been a surge of interest in logics of strategic ability, which has been sparked by two largely independent developments: Pauly's development of Coalition Logic [83, 82, 81, 84], and the development of ATL by Alur, Henzinger, and Kupferman [8, 38, 27]. Although these logics are very closely related, the motivation and background to the two systems is strikingly different.

24.3.1 Coalition Logic

Pauly's Coalition Logic was developed in an attempt to shed some light on the links between logic—and in particular, modal logic—and the mathematical theory of games [80]. Pauly showed how the semantic structures underpinning a family of logics of cooperative ability could be formally understood as games of various types; he gave correspondence results between properties of the games and axioms of the logic, gave complete axiomatizations of the various resulting logics, determined the computational complexity of the satisfiability and model checking problems for his logics, and in addition, demonstrated how these logics could be applied to the formal specification and verification of social choice procedures. The basic modal operator in Pauly's logic is of the form $[C]\varphi$, where C is a set of agents (i.e., a subset of the grand coalition Σ), and φ is a sentence; the intended reading is that "C can cooperate to ensure that φ".

The semantics of cooperation modalities are given in terms of an *effectivity function*, which defines for every coalition C the states that C can cooperate to bring about; the effectivity function $\mathsf{E}\colon S \to (\mathcal{P}(\Sigma) \to \mathcal{P}(\mathcal{P}(S)))$, gives, for any state t and coalition C a set of sets of end-states $\mathsf{E}_C(t)$, with the intended meaning of $S \in \mathsf{E}_C(t)$ that C can enforce the outcome to be in S (although C may not be able to pinpoint the exact outcome that emerges with this choice; this generally depends on the choices of agents outside C, or 'choices' made by the environment). This effectivity function comes on a par with a modal operator $[C]$ with truth definition

$$t \models [C]\varphi \text{ iff for some } S \in \mathsf{E}_C(t)\colon \text{ for all } s (s \models \varphi \text{ iff } s \in S).$$

In words: coalition is effective for, or can enforce φ if there is a set of states S that it is effective for, i.e., which it can choose, which is exactly the denotation of φ: $S = [\![\varphi]\!]$. It seems reasonable to say that C is also effective for φ if it can choose a set of states S that 'just' guarantees φ, i.e., for which we have $S \subseteq [\![\varphi]\!]$. This will be taken care of by imposing monotonicity on effectivity functions: we will discuss constraints on effectivity in the end of this section.

In games and other structures for cooperative and competitive reasoning, effectivity functions are convenient when one is interested in the *outcomes* of the game or the encounter, and not so much about *intermediate states*, or *how* a certain state is reached. Effectivity is also a level in which on can decide whether two interaction scenarios are

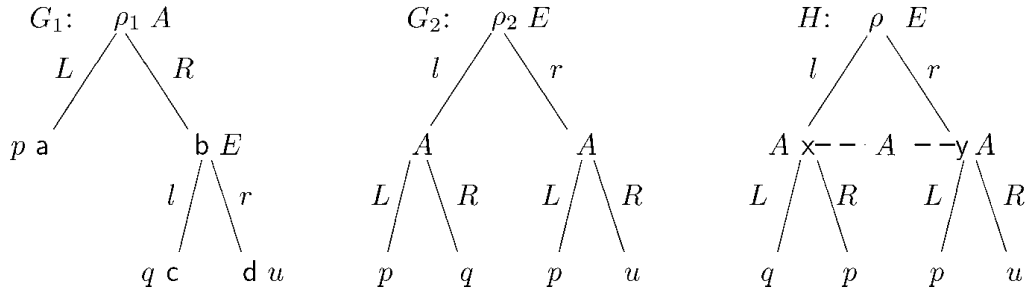

Figure 24.2: Two games G_1 and G_2 that are the same in terms of effectivity. H is an imperfect information game: see Section 24.3.3.

the same. The two games $G1$ and $G2$ from Fig. 24.2 are "abstract" in the sense that they do not lead to payoffs for the players but rather to states which satisfy certain properties, encoded with propositional atoms p, q and u. Such atoms could refer to which player is winning, but also denote other properties of an end-state, such as some distribution of resources, or "payments". Both games are two-player games: in $G1$, player A makes the first move, which he choses form L (Left) and R (Right). In that game, player E is allowed to chose between l and r, respectively, but only if A plays R: otherwise the game ends after one move in the state satisfying p. In game $G2$, both players have the same repertoire of choices, but the order in which the players choose is different. It looks like in $G1$ player A can hand over control to E, while the converse seems to be true for $G2$. Moreover, in $G2$, the player who is not the initiator (i.e., player A), will be allowed to make a choice, no matter the choice of his opponent.

Despite all these differences between the two games, when we evaluate them with respect to what each coalition can *achieve*, they are the same! To become a little more precise, let us define the powers of a coalition in terms of effectivity functions E. In game G_1, player A's effectivity gives $\mathsf{E}_A(\rho_1) = \{\{\mathsf{a}\}, \{\mathsf{c}, \mathsf{d}\}\}$. Similarly, player E's effectivity yields $\{\{\mathsf{a}, \mathsf{c}\}, \{\mathsf{a}, \mathsf{c}\}\}$: he can enforce the game to end in a or c (by playing l), but he can also force be the end-state among a and d (by playing r). Obviously, we also have $\mathsf{E}_{\{A,E\}}(\rho_1) = \{\{\mathsf{a}\}, \{\mathsf{c}\}, \{\mathsf{d}\}\}$: players A and E together can enforce the game to end in any end-state. When reasoning about this, we have to restrict ourselves to the properties that are true in those end states. In coalition logic, what we have just noted semantically would be described as:

$$G_1 \models [A]p \wedge [A](q \vee u) \wedge [E](p \vee q) \wedge [E](p \vee u) \wedge \\ [A, E]p \wedge [A, E]q \wedge [A, E]r.$$

Being equipped with the necessary machinery, it now is easy to see that the game G_2 verifies the same formula, indeed, in terms of what propositions can be achieved, we are in a similar situation as in the previous game: E is effective for $\{p, q\}$ (by playing l) and also for $\{p, u\}$ (play r). Likewise, A is effective for $\{p\}$ (play L) and for $\{q, u\}$ (play R). The alert reader will have recognized the logical law $(p \wedge (q \vee u)) \equiv ((p \wedge q) \vee (p \wedge u))$ resembling the 'equivalence' of the two games: $(p \wedge (q \vee u))$ corresponds to A's power in G_1, and $((p \wedge q) \vee (p \wedge u))$ to A's power in G_2. Similarly, the equivalence of E's powers is reflected by the logical equivalence $(p \vee (q \wedge u)) \equiv ((p \vee q) \wedge (p \vee u))$.

At the same time, the reader will have recognized the two metalinguistic quantifiers in the use of the effectivity function E, laid down in its truth-definition above. A set of outcomes S is in E_C iff *for some* choice of C, we will end up in S, under *all* choices of the complement of C (the other agents). This notion of so-called α-effectivity uses the $\exists\forall$-order of the quantifiers: what a coalition can establish through the truth-definition above, their α-ability, is sometimes also called $\exists\forall$-ability. Implicit within the notion of α-ability is the fact that C have *no knowledge* of the choice that the other agents make; they do not see the choice of $\overline{C}$ (i.e., the complement of C), and then decide what to do, but rather they must make their decision *first*. This motivates the notion of β-ability (i.e., "$\forall\exists$"-ability): coalition C is said to have the β-ability for φ if for every choice $\sigma_{\overline{C}}$ available to $\overline{C}$, there exists a choice σ_C for C such that if $\overline{C}$ choose $\sigma_{\overline{C}}$ and C choose σ_C, then φ will result. Thus C being β-able to φ means that no matter what the other agents do, C have a choice such that, if they make this choice, then φ will be true. Note the "$\forall\exists$" pattern of quantifiers: C are implicitly allowed to make their choice while being aware of the choice made by $\overline{C}$. We will come back to information of other player's moves in Section 24.3.3, and to the pairs of α and β ability in Section 24.3.4.

We end this section by mentioning some properties of α-abilities. The axioms for $[C]\varphi$ based on α-effectivity (or effectivity, for short) are summarized in Fig. 24.3; see also Pauly's [83]. The two extreme coalitions $\emptyset$ and the grand coalition Σ are of special interest. $[\Sigma]\varphi$ expresses that some end-state satisfies φ, whereas $[\emptyset]\varphi$ holds if no agent needs to do anything for φ to hold in the next state.

Some of the axioms of coalition logic correspond to restrictions on effectivity functions $\mathsf{E}\colon S \to (\mathcal{P}(\Sigma) \to \mathcal{P}(\mathcal{P}(S)))$. First of all, we demand that $\emptyset \notin \mathsf{E}_C$ (this guarantees axiom $\bot$). The function E is also assumed to be monotonic: For every coalition $C \subseteq \Sigma$, if $X \subseteq X' \subseteq S$, $X \in \mathsf{E}(C)$ implies $X' \in \mathsf{E}(C)$. This says that if a coalition can enforce an outcome in the set X, it also can guarantee the outcome to be in any superset X' of X (this corresponds to axiom (M)). An effectivity function E is *C-maximal* if for all X, if $\overline{X} \notin \mathsf{E}(\overline{C})$ then $X \in \mathsf{E}(C)$. In words: If the other agents $\overline{C}$ cannot guarantee an outcome outside X (i.e, in $\overline{X}$), then C is able to guarantee to be it in X. We require effectivity functions to be Σ-maximal. (This enforces axiom (N—Pauly's symbol for the grand coalition is N): if the empty coalition can not enforce an outcome satisfying φ, the grand coalition Σ can enforce φ. The final principle governs the formation of coalitions. It states that coalitions can combine their strategies to (possibly) achieve more: E is *superadditive* if for all X_1, X_2, C_1, C_2 such that $C_1 \cap C_2 = \emptyset$,

($\bot$)	$\neg[C]\bot$
(N)	$(\neg[\emptyset]\neg\varphi \to [\Sigma]\varphi)$
(M)	$[C](\varphi \wedge \psi) \to [C]\psi$
(S)	$([C_1]\varphi_1 \wedge [C_2]\varphi_2) \to [C_1 \cup C_2](\varphi_1 \wedge \varphi_2)$ where $C_1 \cap C_2 = \emptyset$
(MP)	from φ and $\varphi \to \psi$ infer ψ
(Nec)	from φ infer $[C]\varphi$

Figure 24.3: The axioms and inference rules of Coalition Logic.

$X_1 \in \mathsf{E}(C_1)$ and $X_2 \in \mathsf{E}(C_2)$ imply that $X_1 \cap X_2 \in \mathsf{E}(C_1 \cup C_2)$. This obviously corresponds to axiom (S).

24.3.2 Strategic Temporal Logic: ATL

In Coalition Logic one reasons about the powers of coalitions with respect to final outcomes. However, in many multi-agent scenarios, the strategic considerations continue during the process. It would be interesting to study a representation language for interaction that *is* able to express the temporal differences in the two games G_1 and G_2 of Fig. 24.2. *Alternating-time Temporal Logic* (ATL) is intended for this purpose.

Although it is similar to Coalition Logic, ATL emerged from a very different research community, and was developed with an entirely different set of motivations in mind. The development of ATL is closely linked with the development of branching-time temporal logics for the specification and verification of reactive systems [29, 28, 109]. Recall that CTL combines path quantifiers "A" and "E" for expressing that a certain series of events will happen on all paths and on some path respectively, and combines these with tense modalities for expressing that something will happen eventually on some path ($\Diamond$), always on some path ($\Box$) and so on. Thus, for example, using CTL logics, one may express properties such as "on all possible computations, the system never enters a fail state" ($\mathsf{A}\,\Box\,\neg \mathit{fail}$). CTL-like logics are of limited value for reasoning about *multi-agent* systems, in which system components (agents) cannot be assumed to be benevolent, but may have competing or conflicting goals. The kinds of properties we wish to express of such systems are the powers that the system components have. For example, we might wish to express the fact that "agents 1 and 2 can cooperate to ensure that the system never enters a fail state". It is not possible to capture such statements using CTL-like logics. The best one can do is either state that something will inevitably happen, or else that it may possibly happen: CTL-like logics have no notion of agency.

Alur, Henzinger, and Kupferman developed ATL in an attempt to remedy this deficiency. The key insight in ATL is that path quantifiers can be replaced by cooperation modalities: the ATL expression $\langle\!\langle C \rangle\!\rangle \varphi$, where C is a group of agents, expresses the fact that the group C can cooperate to ensure that φ. (Thus the ATL expression $\langle\!\langle C \rangle\!\rangle \varphi$ corresponds to Pauly's $[C]\varphi$.) So, for example, the fact that agents 1 and 2 can ensure that the system never enters a fail state may be captured in ATL by the following formula: $\langle\!\langle 1, 2 \rangle\!\rangle\, \Box\, \neg \mathit{fail}$. An ATL formula true in the root ρ_1 of game G_1 of Fig. 24.2 is $\langle\!\langle A \rangle\!\rangle \bigcirc \langle\!\langle E \rangle\!\rangle \bigcirc q$: A has a strategy (i.e., play R in ρ_1) such that in the next time, E has a strategy (play l) to enforce u.

Note that ATL generalizes CTL because the path quantifiers A ("on all paths...") and E ("on some paths...") can be simulated in ATL by the cooperation modalities $\langle\!\langle \emptyset \rangle\!\rangle$ ("the empty set of agents can cooperate to...") and $\langle\!\langle \Sigma \rangle\!\rangle$ ("the grand coalition of all agents can cooperate to...").

One reason for the interest in ATL is that it shares with its ancestor CTL the computational tractability of its model checking problem [20]. This led to the development of an ATL model checking system called MOCHA [9, 6]. With MOCHA, one specifies a model against which a formula is to be checked using a model definition language called REACTIVE MODULES [7]. REACTIVE MODULES is a guarded command language, which provides a number of mechanisms for the structured specification of

models, based upon the notion of a "module", which is basically the REACTIVE SYSTEMS terminology for an agent. Interestingly, however, it is ultimately necessary to define for every variable in a REACTIVE MODULES system which module (i.e., agent) controls it. The powers of agents and coalitions then derive from the ability to control these variables: and as we noted in the introduction, this observation was a trigger for [54] to develop a system for propositional control, CL-PC, as a system in its own right. We will come briefly back to this idea in Section 24.3.4.

ATL has begun to attract increasing attention as a formal system for the specification and verification of multi-agent systems. Examples of such work include formalizing the notion of role using ATL [100], the development of epistemic extensions to ATL [50, 52, 51], and the use of ATL for specifying and verifying cooperative mechanisms [84].

To give a precise definition of ATL, we must first introduce the semantic structures over which formulae of ATL are interpreted. An *alternating transition system* (ATS) is a 5-tuple

$$S = \langle \Pi, \Sigma, Q, \pi, \delta \rangle, \text{ where:}$$

- Π is a finite, non-empty set of *Boolean variables*;
- $\Sigma = \{a_1, \ldots, a_n\}$ is a finite, non-empty set of *agents*;
- Q is a finite, non-empty set of *states*;
- $\pi : Q \to 2^{\Pi}$ gives the set of Boolean variables satisfied in each state;
- $\delta : Q \times \Sigma \to 2^{2^Q}$ is the system transition function, which maps states and agents to the choices available to these agents. Thus $\delta(q, a)$ is the set of choices available to agent a when the system is in state q. We require that this function satisfy the requirement that for every state $q \in Q$ and every set $Q_1, \ldots, Q_n$ of choices $Q_i \in \delta(q, a_i)$, the intersection $Q_1 \cap \cdots \cap Q_n$ is a singleton.

One can think of $\delta(q, a)$ as the possible moves agent a can make in state q. Since in general he cannot determine the next state on his own, each specific choice that a makes at q yields a set of possible next states Q_a, which can be further constrained by the choices of the other agents. Indeed, the constraint that $Q_1 \cap \cdots \cap Q_n$ gives a singleton $\{q'\}$ resembles that the system as a whole is deterministic: once every agent a has made a decision Q_a at q, *the* next state q' of q is determined.

The games G_1 and G_2 of the previous section can be conceived of as special cases of alternating transition system: *turn based synchronous* systems, where, at every decision point (state) of the system, exactly one agent is responsible for the next state. For instance, we have, in G_1 that $\delta(\rho_1, A) = \{\{\mathsf{a}\}, \{\mathsf{b}\}\}$, and $\delta(\rho_1, E) = \{\{\mathsf{a}, \mathsf{b}\}\}$, denoting that E leaves the choice in ρ_1 to A. To make G_1 a real transition system, the transition function should specify choices for every state, also for the leaves a, c and d. One could do this for instance by looping those states to themselves: $\delta(\mathsf{a}, A) = \delta(\mathsf{a}, E) = \{\{\mathsf{a}\}\}$. In order to reason about them as leaves, one could add a proposition *end* that is true in exactly those states. Turn based systems satisfy the following property (cf. [52]),

which is not valid in ATL in general:

$$\langle\langle \Sigma \rangle\rangle \bigcirc \varphi \rightarrow \bigvee_{a \in \Sigma} \langle\langle a \rangle\rangle \bigcirc \varphi.$$

An ATL formula, formed with respect to an alternating transition system $S = \langle \Pi, \Sigma, Q, \pi, \delta \rangle$, is then defined by the following grammar:

$$\varphi ::= \top \mid p \mid \neg\varphi \mid \varphi \vee \varphi \mid \langle\langle C \rangle\rangle \bigcirc \varphi \mid \langle\langle C \rangle\rangle \Box \varphi \mid \langle\langle C \rangle\rangle \varphi \mathcal{U} \varphi$$

where $p \in \Pi$ is a Boolean variable, and $C \subseteq \Sigma$ is a set of agents. We assume the remaining connectives ("$\bot$", "$\rightarrow$", "$\leftarrow$", "$\leftrightarrow$", "$\wedge$") are defined as abbreviations in the usual way, and define $\langle\langle C \rangle\rangle \Diamond \varphi$ as $\langle\langle C \rangle\rangle \top \mathcal{U} \varphi$.

To give the semantics of ATL, we need some further definitions. For two states $q, q' \in Q$ and an agent $a \in \Sigma$, we say that state q' is an *a-successor* of q if there exists a set $Q' \in \delta(q, a)$ such that $q' \in Q'$. Intuitively, if q' is an a-successor of q, then q' is a possible outcome of one of the choices available to a when the system is in state q. We denote by $succ(q, a)$ the set of a successors to state q, and say that q' is simply a *successor* of q if for all agents $a \in \Sigma$, we have $q' \in succ(q, a)$; intuitively, if q' is a successor to q, then when the system is in state q, the agents Σ can cooperate to ensure that q' is the next state the system enters.

A *computation* of an ATS $\langle \Pi, \Sigma, Q, \pi, \delta \rangle$ is an infinite sequence of states $\lambda = q_0, q_1, \ldots$ such that for all $u > 0$, the state q_u is a successor of q_{u-1}. A computation $\lambda \in Q^\omega$ starting in state q is referred to as a *q-computation*; if $u \in \mathbb{N}$, then we denote by $\lambda[u]$ the uth state in λ; similarly, we denote by $\lambda[0, u]$ and $\lambda[u, \infty]$ the finite prefix $q_0, \ldots, q_u$ and the infinite suffix $q_u, q_{u+1}, \ldots$ of λ, respectively.

Intuitively, a *strategy* is an abstract model of an agent's decision-making process; a strategy may be thought of as a kind of plan for an agent. Formally, a strategy f_a for an agent $a \in \Sigma$ is a total function $f_a : Q^+ \rightarrow 2^Q$, which must satisfy the constraint that $f_a(\lambda \cdot q) \in \delta(q, a)$ for all $\lambda \in Q^*$ and $q \in Q$. Given a set $C \subseteq \Sigma$ of agents, and an indexed set of strategies $F_C = \{f_a \mid a \in C\}$, one for each agent $a \in C$, we define $out(q, F_C)$ to be the set of possible outcomes that may occur if every agent $a \in C$ follows the corresponding strategy f_a, starting when the system is in state $q \in Q$. That is, the set $out(q, F_C)$ will contain all possible q-computations that the agents C can "enforce" by cooperating and following the strategies in F_C. Note that the "grand coalition" of all agents in the system can cooperate to uniquely determine the future state of the system, and so $out(q, F_\Sigma)$ is a singleton. Similarly, the set $out(q, F_\emptyset)$ is the set of all possible q-computations of the system.

We can now give the rules defining the satisfaction relation "$\models$" for ATL, which holds between pairs of the form S, q (where S is an ATS and q is a state in S), and formulae of ATL:

$S, q \models \top$;

$S, q \models p$ iff $p \in \pi(q)$ (where $p \in \Pi$);

$S, q \models \neg\varphi$ iff $S, q \not\models \varphi$;

$S, q \models \varphi \vee \psi$ iff $S, q \models \varphi$ or $S, q \models \psi$;

$S, q \models \langle\langle C\rangle\rangle \bigcirc \varphi$ iff there exists a set of strategies F_C, such that for all $\lambda \in out(q, F_C)$, we have $S, \lambda[1] \models \varphi$;

$S, q \models \langle\langle C\rangle\rangle \Box \varphi$ iff there exists a set of strategies F_C, such that for all $\lambda \in out(q, F_C)$, we have $S, \lambda[u] \models \varphi$ for all $u \in \mathbb{N}$;

$S, q \models \langle\langle C\rangle\rangle \varphi \mathcal{U} \psi$ iff there exists a set of strategies F_C, such that for all $\lambda \in out(q, F_C)$, there exists some $u \in \mathbb{N}$ such that $S, \lambda[u] \models \psi$, and for all $0 \leqslant v < u$, we have $S, \lambda[v] \models \varphi$.

Pauly's Coalition Logic is then the fragment of ATL in which the only cooperation modalities allowed are of the form $\langle\langle C\rangle\rangle \bigcirc$ [81, 82, 38]. The truth of a Coalition Logic formula is determined on an ATS by using the first five items of the definition for satisfaction above. The satisfiability problem for ATL is EXPTIME-complete [27, 110], while for Coalition Logic it is PSPACE-complete in the general case [81, p. 63].

A number of variations of ATL have been proposed over the past few years, for example, to integrate reasoning about obligations into the basic framework of cooperative ability [116], to deal with quantification over coalitions [3], adding the ability to refer to strategies in the object language [107], and adding the ability to talk about preferences or goals of agents [2, 1]. In what follows, we will focus on one issue that has received considerable attention: the integration of *knowledge and ability*.

24.3.3 Knowledge in Strategic Temporal Logics: ATEL

The semantics of Coalition Logic and of ATL assume that agents have *perfect information* about the game. This is immediately apparent in the notion of strategy in ATL: by having an agent decide his next action given an element of Q^+, this makes two strong assumptions. First of all the agents have perfect information about the state they are in, which obviously is an idealized assumption: typically, agents do not know exactly what the state is. They may be unsure about certain facts in the state they are in, but also about the mental states of other agents, which is crucial in any strategic decision making. Secondly, the definition of a strategy assumes that the agents have *perfect recall*: they remember exactly what has happened in reaching the current state, so that they can make different decisions even in identical states.

We first address the issue of imperfect information. The paper [52] adds modalities for knowledge to ATL to obtain ATEL (Alternating-time Temporal Epistemic Logic). For every individual i, add an operator K_i to the language ($K_i\varphi$ is read as "i knows φ"), and for every coalition G, add operators E_G (everybody in G knows), D_G (it is distributed knowledge in G), and C_G (it is common knowledge in C).[2] The following examples of what can be expressed in ATEL are taken from [52].

Performing actions and knowledge interfere in at least two ways: for some actions, in order to be able to do them properly, some knowledge is required, and, on the other hand, actions may add to an agent's knowledge. We have already mentioned knowledge pre-conditions in Section 24.3. We can formulate knowledge pre-conditions quite naturally using ATEL and its variants, and the cooperation modality naturally and elegantly allows us to consider knowledge pre-conditions for *multi*-agent plans. The requirement that, in order for an agent a to be able to eventually bring about state

[2] A more detailed exposition on epistemic logic is given in Chapter 15 of this Handbook.

of affairs φ, it must know ψ, might, as a first attempt, be specified in ATEL as: $\langle\langle a\rangle\rangle \Diamond \varphi \rightarrow K_a\psi$. This intuitively says that knowing ψ is a *necessary* requirement for having the ability to bring about φ. However, this requirement is too strong. For instance, in order to be able to ever open the safe, I do not necessarily in general have to know the key right *now*. A slightly better formulation might therefore be $\langle\langle a\rangle\rangle \bigcirc \varphi \rightarrow K_a\psi$. As an overall constraint of the system, this property may help the agent to realize that he has to possess the right knowledge in order to achieve φ. But taken as a local formula, it does not tell us anything about what the agent should know if he wants to bring about φ the day after tomorrow, or "sometime" for that matter. Taken as a local constraint, a necessary knowledge condition to bring about φ might be $(\neg\langle\langle a\rangle\rangle \bigcirc \varphi)\mathcal{U}K_a\psi$. This expresses that our agent is not able to open the safe until he knows its key. The other way around, an example of an ability that is generated by possessing knowledge is the following, expressing that if Bob knows that the combination of the safe is s, then he is able to open it (o), as long as the combination remains unchanged.

$$K_b(c = s) \rightarrow \langle\langle b\rangle\rangle(\langle\langle b\rangle\rangle \bigcirc o)\mathcal{U}\neg(c = s). \tag{24.4}$$

One of the properties of the most widely embraced systems for knowledge is introspection, of which the positive variant says $K_a\varphi \rightarrow K_aK_a\varphi$. Another well-accepted principle of knowledged has it that from $K_a\varphi$ and $K_a(\varphi \rightarrow \psi)$ it follows that $K_a\psi$. Such idealized properties have been criticized since they assume agents to be perfect reasoners who know all consequences of their knowledge in a blow. One may also use ATEL-formulas to model limited reasoners, i.e., reasoners that do not make all inferences in one strike, but where this behavior can be approximated over time. Positive introspection might then look like

$$K_a\psi \rightarrow \langle\langle a\rangle\rangle \bigcirc K_aK_a\psi. \tag{24.5}$$

As a final example, in security protocols where agents a and b share some common secret (a key S_{ab}, for instance), what one typically wants is (24.6), expressing that a can send private information to b, without revealing the message to another agent c:

$$K_a\varphi \wedge \neg K_b\varphi \wedge \neg K_c\varphi \wedge \langle\langle a, b\rangle\rangle \bigcirc (K_a\varphi \wedge K_b\varphi \wedge \neg K_c\varphi). \tag{24.6}$$

Semantically, ignorance of the agents is usually modeled by specifying that each agent is unable to distinguish certain states: the more states he considers undistinguishable from a given state, the weaker his knowledge in that state. In game theory, such an indistinguishibility relation is often called a partition [10]. Take the game H in Fig. 24.2, for example. The dashed line labeled with agent A denotes that this agent does not know what E's move was: A cannot distinguish state x from y. It seems reasonable that do require strategies of agents to be *uniform*: if an agent does not know whether he is in state s or s', he should make the same decision in both. But there is more to adding knowledge to decision making. Let us assume that atom p in game H denotes a win for A. Then, in the root ρ we have that $[\![E]\!] \bigcirc \langle\langle A\rangle\rangle \bigcirc p$: saying that whichever strategy E plays in ρ, in the next state A will be able to reach a winning state in the next state. Note that this is even true if we restrict ourselves to uniform strategies! We even have $H, \mathsf{x} \models K_A\langle\langle A\rangle\rangle \bigcirc p$, saying that A knows that he has a winning strategy in x. This, of course, is only true in the *de dicto* reading of knowledge

of A: he knows in x that he has a uniform strategy to win, but he does not know which one it is! To obtain a *de re* type of reading of knowledge of strategies, work is still in progress, but we refer to [58] and the recent [105, 59].

Having discussed the issue of imperfect information in a state of the game, there is also the question how to represent what an agent knows about the past: in order to relate this to the example of DEL in Section 24.2.2, we present this for a system with operators for knowledge and change, not necessarily cooperation modalities. In a *synchronous system*, agents are aware of a clock and they know what time it is: in a game, this would mean that they know how many moves have played. In a game with *perfect recall*, agents recall what they have experienced: however, they need not be aware of the time, and also not aware of moves that does not have an impact on their information. In a logical language for knowledge and time one might expect that perfect recall corresponds to

$$K_i\varphi \to \Box K_i\varphi. \tag{24.7}$$

But this can in general not be. First of all, φ might refer to some specifics of the moment of evaluation. For instance, knowing that it is Wednesday should not imply that I always know that it is Wednesday. Moreover, φ might refer to i's ignorance, i.e., φ might be $\neg K_i\varphi$. Then, if (24.7) would hold, the agent would for ever know that he does not know φ. Since in most logics of knowledge, $\neg K_i\varphi$ is equivalent to $K_i\neg K_i\varphi$, scheme (24.7) would give $\neg K_i\psi \to \Box\neg K_i\psi$, a rather pessimistic principle! It appears that the proper characterization for perfect recall is

$$K_i \bigcirc \varphi \to \bigcirc K_i\varphi.$$

For a further discussion about this scheme and perfect recall, we refer to [30]. Let us finally mention that Bonanno [14] studies a property about memory in games that is weaker than perfect recall. He calls a game a *Von Neumann game* if for every two states that an agent cannot distinguish in a game, the number of predecessors in that game must be the same. This would mean that an agent knows how many moves have been played, but not necessarily which ones. Let P be a temporal operator denoting 'always in the past', and $\Box$ 'always in the future', then the epistemic temporal property characterizing Von Neumann games is

$$K_i\varphi \to \Box K_i P K_i\varphi.$$

Going back to DEL of Section 24.2.2, our example of Fig. 24.1 is rich enough to show that DEL in general does not satisfy perfect recall. To see this, let α be $L_{12}(L_1?p \cup L_1?\neg p \cup !\top)$. We then have $N, s \models K_2[\alpha]\neg(K_1 p \vee K_1\neg p)$ (2 knows that if nothing happens, 1 will not find out whether p), but not $N, s \models [\alpha]K_2\neg(K_1 p \vee K_1\neg p)$. We *do* have in general the following weaker form of perfect recall, however. Let M, w be a static epistemic state, and α an action, represented by some action state M, w. Let A be the set of actions that agent i cannot distinguish from M, w. Then we have

$$M, w \models \bigwedge_{\beta\in A} K_i[\beta]\varphi \to [\alpha]K_i\varphi. \tag{24.8}$$

In words, in order for agent i to 'remember' what holds after performance of an action α, he should already now in advance that it will hold after *every epistemically possible execution* of that action.

24.3.4 CL-PC

Both ATL and Coalition Logic are intended as *general purpose* logics of cooperative ability. In particular, neither has anything specific to say about the *origin* of the powers that are possessed by agents and the coalitions of which they are a member. These powers are just assumed to be implicitly defined within the effectivity structures used to give a semantics to the languages. Of course, if we give a specific *interpretation* to these effectivity structures, then we will end up with a logic with special properties. In [54], a variation of Coalition Logic was developed that was intended specifically to reason about *control* scenarios, as follows. The basic idea is that the overall state of a system is characterized by a finite set of variables, which for simplicity are assumed to take Boolean values. Each agent in the system is then assumed to control some (possibly empty) subset of the overall set of variables, with every variable being under the control of exactly one agent. Given this setting, in the Coalition Logic of Propositional Control (CL-PC), the operator $\Diamond_C \varphi$ means that there exists some assignment of values that the coalition C can give to the variables under its control such that, assuming everything else in the system remains unchanged, then if they make this assignment, then φ would be true. The box dual $\Box_C \varphi$ is defined in the usual way with respect to the diamond ability operator $\Diamond_C$. Here is a simple example:

> Suppose the current state of the system is that variables p and q are false, while variable r is true, and further suppose then agent 1 controls p and r, while agent 2 controls q. Then in this state, we have, for example: $\Diamond_1(p \wedge r)$, $\neg\Diamond_1 q$, and $\Diamond_2(q \wedge r)$. Moreover, for any satisfiable propositional logic formula ψ over the variables p, q, and r, we have $\Diamond_{1,2}\psi$.

The ability operator $\Diamond_C$ in CL-PC thus captures *contingent* ability, rather along the lines of "classical planning" ability [68]: ability under the assumption that the world only changes by the actions of the agents in the coalition operator $\Diamond_C$. Of course, this is not a terribly realistic type of ability, just as the assumptions of classical planning are not terribly realistic. However, in CL-PC, we can define α effectivity operators $\langle\!\langle C \rangle\!\rangle_\alpha \varphi$, intended to capture something along the lines of the ATL $\langle\!\langle C \rangle\!\rangle \bigcirc \varphi$, as follows:

$$\langle\!\langle C \rangle\!\rangle_\alpha \hat{=} \Diamond_C \Box_{\bar{C}} \varphi.$$

Notice the quantifier alternation pattern $\exists\forall$ in this definition.

One of the interesting aspects of CL-PC is that, by using this logic, it becomes possible to explicitly reason in the object language about who controls what. Let i be an agent, and let p be a system variable; let us define $ctrl(i, p)$ as follows:

$$ctrl(i, p) \hat{=} (\Diamond_i p) \wedge (\Diamond i \neg p).$$

Thus $ctrl(i, p)$ means that i can assign p the value true, and i can also assign p the value false. It is easy to see that if $ctrl(i, p)$ is true in a system, then this means that the variable p must be under the control of agent i. Starting from this observation, a more detailed analysis of characterizing control of arbitrary formulae was developed, in terms of the variables controlled by individual agents [54]. In addition, [54] gives a complete axiomatization of CL-PC, and shows that the model checking and satisfiability problems for the logic are both PSPACE-complete. Building on this basic formalism, [53] investigates extensions into the possibility of *dynamic* control, where variables can be "passed" from one agent to another.

24.3.5 Applications of Strategic Cooperation Logics

One of the fascinating aspects of coalition logic is its use in social choice theory, and in particular in the specification, development, and verification of *social choice procedures*. Consider the following scenario, adapted from [81].

> Two individuals, A and B, are to choose between two outcomes, p and q. We want a procedure that will allow them to choose that will satisfy the following requirements. First, we definitely want an outcome to be possible—that is, we want the two agents to bring about either p or q. We do not want them to be able to bring about both outcomes simultaneously. Similarly, we do not want either agent to dominate: we want them both to have equal power.

The first point to note is that we can naturally axiomatize these requirements using coalition logic:

$$\langle\langle A, B\rangle\rangle \bigcirc x, \quad x \in \{p, q\}$$
$$\neg\langle\langle A, B\rangle\rangle \bigcirc (p \wedge q)$$
$$\neg\langle\langle x\rangle\rangle \bigcirc p, \quad x \in \{A, B\}$$
$$\neg\langle\langle x\rangle\rangle \bigcirc q, \quad x \in \{A, B\}$$

It should be immediately obvious how these axioms capture the requirements as stated above. Now, given a particular voting procedure, a model checking algorithm can be used to check whether or not this procedure implements the specification correctly. Moreover, a constructive proof of satisfiability for these axioms might be used to *synthesize* a procedure; or else announce that no implementation exists.

24.4 Conclusions

In this paper, we have motivated and introduced a number of logics of rational agency; moreover, we have investigated the role(s) that such logics might play in the *development* of artificial agents. We hope to have demonstrated that logics for rational agents are a fascinating area of study, at the confluence of many different research areas, including logic, artificial intelligence, economics, game theory, and the philosophy of mind. We also hope to have illustrated some of the popular approaches to the theory of rational agency.

There are far too many research challenges open to identify in this article. Instead, we simply note that the search for a logic of rational agency poses a range of deep technical, philosophical, and computational research questions for the logic community. We believe that all the disparate research communities with an interest in rational agency can benefit from this search.

Bibliography

[1] T. Agotnes, W. van der Hoek, and M. Wooldridge. On the logic of coalitional games. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2006)*, Hakodate, Japan, 2006.

[2] T. Agotnes, W. van der Hoek, and M. Wooldridge. Temporal qualitative coalitional games. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2006)*, Hakodate, Japan, 2006.

[3] T. Agotnes, W. van der Hoek, and M. Wooldridge. Quantified coalition logic. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI-07)*, Hyderabad, India, 2007.

[4] C.E. Alchourrón, P. Gärdenfors, and D. Makinson. On the logic of theory change: partial meet contraction and revision functions. *Journal of Symbolic Logic*, 50:510–530, 1985.

[5] J.F. Allen, J. Hendler, and A. Tate, editors. *Readings in Planning*. Morgan Kaufmann Publishers, San Mateo, CA, 1990.

[6] R. Alur, L. de Alfaro, T.A. Henzinger, S.C. Krishnan, F.Y.C. Mang, S. Qadeer, S.K. Rajamani, and S. Taşiran. MOCHA user manual. University of Berkeley Report, 2000.

[7] R. Alur and T.A. Henzinger. Reactive modules. *Formal Methods in System Design*, 15(11):7–48, July 1999.

[8] R. Alur, T.A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49(5):672–713, September 2002.

[9] R. Alur, T.A. Henzinger, F.Y.C. Mang, S. Qadeer, S.K. Rajamani, and S. Taşiran. Mocha: Modularity in model checking. In *CAV 1998: Tenth International Conference on Computer-aided Verification*, *LNCS*, vol. 1427, pages 521–525. Springer-Verlag, Berlin, Germany, 1998.

[10] R.J. Aumann. Interactive epistemology I: Knowledge. *International Journal of Game Theory*, 28:263–300, 1999.

[11] A. Baltag and L. Moss. Logics for epistemic programs. *Synthese*, 139(2):165–224, 2004 (In the section 'Knowledge, Rationality and Action').

[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*, vol. 430, pages 94–129. Springer-Verlag, Berlin, Germany, June 1989.

[13] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Cambridge University Press, Cambridge, England, 2001.

[14] G. Bonanno. A characterisation of von Neumann games in terms of memory. *Synthese*, 139(2):281–295, 2004 (In the section 'Knowledge, Rationality and Action').

[15] M.E. Bratman. *Intention, Plans, and Practical Reason*. Harvard University Press, Cambridge, MA, 1987.

[16] 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.

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

[18] M.A. Brown. On the logic of ability. *Journal of Philosophical Logic*, 17:1–26, 1988.

[19] B. Chellas. *Modal Logic: An Introduction*. Cambridge University Press, Cambridge, England, 1980.

[20] E.M. Clarke, O. Grumberg, and D.A. Peled. *Model Checking*. The MIT Press, Cambridge, MA, 2000.

[21] W.F. Clocksin and C.S. Mellish. *Programming in Prolog*. Springer-Verlag, Berlin, Germany, 1981.

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

[23] 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.

[24] P.R. Cohen and H.J. Levesque. Teamwork. *Nous*, 25(4):487–512, 1991.

[25] D.C. Dennett. *The Intentional Stance*. The MIT Press, Cambridge, MA, 1987.

[26] M. d'Inverno, D. Kinny, M. Luck, and M. Wooldridge. A formal specification of dMARS. In M.P. Singh, A. Rao, and M.J. Wooldridge, editors. *Intelligent Agents IV*, *LNAI*, vol. 1365, pages 155–176. Springer-Verlag, Berlin, Germany, 1997.

[27] G. van Drimmelen. Satisfiability in alternating-time temporal logic. In *Eighteenth Annual IEEE Symposium on Logic in Computer Science (LICS 2003)*, pages 208–217, Ottawa, Canada, 2003.

[28] E.A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor. *Handbook of Theoretical Computer Science Volume B: Formal Models and Semantics*, pages 996–1072. Elsevier Science Publishers B.V., Amsterdam, The Netherlands, 1990.

[29] E.A. Emerson and J. Srinivasan. Branching time logic. In J.W. de Bakker, 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*, vol. 354, pages 123–172. Springer-Verlag, Berlin, Germany, 1988.

[30] R. Fagin, J.Y. Halpern, Y. Moses, and M.Y. Vardi. *Reasoning About Knowledge*. The MIT Press, Cambridge, MA, 1995.

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

[32] K. Fischer, J.P. Müller, and M. Pischel. A pragmatic BDI architecture. In M. Wooldridge, J.P. Müller, and M. Tambe, editors. *Intelligent Agents II*, *LNAI*, vol. 1037, pages 203–218. Springer-Verlag, Berlin, Germany, 1996.

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

[34] D.M. Gabbay, A. Kurucz, F. Wolter, and M. Zakharyaschev. *Many-Dimensional Modal Logics: Theory and Applications*. Elsevier Science Publishers B.V., Amsterdam, The Netherlands, 2003.

[35] 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.

[36] J.R. Galliers. A theoretical framework for computer models of cooperative dialogue, acknowledging multi-agent conflict. PhD thesis, Open University, UK, 1988.

[37] 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.

[38] V. Goranko. Coalition games and alternating temporal logics. In J. van Benthem, editor, *Proceeding of the Eighth Conference on Theoretical Aspects of Rationality and Knowledge (TARK VIII)*, pages 259–272, Siena, Italy, 2001.

[39] J.Y. Halpern and M.Y. Vardi. The complexity of reasoning about knowledge and time. I. Lower bounds. *Journal of Computer and System Sciences*, 38:195–237, 1989.

[40] J.Y. Halpern and M.Y. Vardi. Model checking versus theorem proving: A manifesto. In V. Lifschitz, editor. *AI and Mathematical Theory of Computation—Papers in Honor of John McCarthy*, pages 151–176. The Academic Press, London, England, 1991.

[41] D. Harel. *First-Order Dynamic Logic*. *LNCS*, vol. 68. Springer-Verlag, Berlin, Germany, 1979.

[42] D. Harel. Dynamic logic. In D. Gabbay and F. Guenther, editors. *Extensions of Classical Logic*, *Handbook of Philosophical Logic*, vol. II, pages 497–604. D. Reidel Publishing Company, Dordrecht, The Netherlands, 1984 (*Synthese Library*, vol. 164).

[43] D. Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic*. The MIT Press, Cambridge, MA, 2000.

[44] K. Hindriks, F.S. de Boer, W. van der Hoek, and J.-J.Ch. Meyer. A formal embedding of agentspeak(l) in 3APL. In G. Antoniou and J. Slaney, editors. *Advanced Topics in Artificial Intelligence*, *LNAI*, vol. 1502, pages 155–166. Springer, 1998.

[45] K.V. Hindriks, F.S. de Boer, W. van der Hoek, and J.-J.Ch. Meyer. Agent programming in 3APL. *Autonomous Agents and Multi-Agent Systems*, 2(4):357–402, 1999.

[46] K.V. Hindriks, F.S. de Boer, W. van der Hoek, and J.-J.Ch. Meyer. A formal semantics for the core of AGENT-0. In E. Postma and M. Gyssens, editors, *Proceedings of Eleventh Belgium–Netherlands Conference on Artificial Intelligence*, pages 27–34, 1999.

[47] C.A.R. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12(10):576–583, 1969.

[48] W. van der Hoek, J.-J.Ch. Meyer, and J.W. van Schagen. Formalizing potential of agents: The Karo framework revisited. In M. Falle, S. Kaufmann, and M. Pauly, editors. *Formalizing the Dynamics of Information*, *CSLI Lecture Notes*, vol. 91, pages 51–67. CSLI Publications, Stanford, 2000.

[49] W. van der Hoek and M. Wooldridge. Model checking knowledge and time. In D. Bošnački and S. Leue, editors. *Model Checking Software, Proceedings of SPIN 2002*, *LNCS*, vol. 2318, pages 95–111. Springer-Verlag, Berlin, Germany, 2002.

[50] W. van der Hoek and M. Wooldridge. Tractable multiagent planning for epistemic goals. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2002)*, pages 1167–1174, Bologna, Italy, 2002.

[51] W. van der Hoek and M. Wooldridge. Model checking cooperation, knowledge, and time—a case study. *Research in Economics*, 57(3):235–265, September 2003.

[52] W. van der Hoek and M. Wooldridge. Time, knowledge, and cooperation: Alternating-time temporal epistemic logic and its applications. *Studia Logica*, 75(1):125–157, 2003.

[53] W. van der Hoek and M. Wooldridge. On the dynamics of delegation and cooperation, and control: A logical account. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2005)*, pages 701–708, Utrecht, The Netherlands, 2005.

[54] W. van der Hoek and M. Wooldridge. On the logic of cooperation and propositional control. *Artificial Intelligence*, 164(1–2):81–119, May 2005.

[55] D.R. Hofstadter. Metamagical themas: A coffeehouse conversation on the Turing test to determine if a machine can think. *Scientific American*, pages 15–36, May 1981 issue.

[56] G. Holzmann. The Spin model checker. *IEEE Transactions on Software Engineering*, 23(5):279–295, May 1997.

[57] M. Huber. JAM: A BDI-theoretic mobile agent architecture. In *Proceedings of the Third International Conference on Autonomous Agents (Agents 99)*, pages 236–243, Seattle, WA, 1999.

[58] W. Jamroga and W. van der Hoek. Agents that know how to play. *Fundamenta Informaticae*, 63(2–3):185–219, 2004.

[59] W. Jamroga and T. Ågotnes. Constructive knowledge: What agents can achieve under incomplete information. Technical Report IfI-05-10, Clausthal University of Technology, 2005. *Journal of Applied Non-Classical Logics*, Submited for publication. Full paper downloadable at http://www.in.tu-clausthal.de/fileadmin/homes/techreports/ifi0510jamroga.pdf. Short version to appear in *Proc. AAMAS 2006*

[60] 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 in a Multi-Agent World (MAAMAW-91)*, pages 93–102. Elsevier Science Publishers B.V., Amsterdam, The Netherlands, 1992.

[61] 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.

[62] A. Kenny. *Will, Freedom and Power*. Basil Blackwell, Oxford, 1975.

[63] K. Konolige. *A Deduction Model of Belief*. Pitman Publishing/Morgan Kaufmann, London/San Mateo, CA, 1986.

[64] W. van der Hoek, K.V. Hindriks, F.S. de Boer, and J.-J.Ch. Meyer. Agent programming with declarative goals. In C. Castelfranchi and Y. Lespérance, editors. *Intelligent Agents VII, Proceedings of the 6th workshop on Agent Theories, Architectures, and Languages (ATAL)*, *LNAI*, vol. 1986, pages 228–243. Springer, 2001.

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

[66] H. Levesque, R. Reiter, Y. Lespérance, F. Lin, and R. Scherl. Golog: A logic programming language for dynamic domains. *Journal of Logic Programming*, 31:59–84, 1996.

[67] 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.

[68] 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.

[69] B. van Linder, W. van der Hoek, and J.-J.Ch. Meyer. Actions that make you change your mind. In A. Laux and H. Wansing, editors. *Knowledge and Belief in Philosophy and AI*, pages 103–146. Akademie-Verlag, 1995.

[70] B. van Linder, W. van der Hoek, and J.J.-Ch. Meyer. Formalizing abilities and opportunities of agents. *Fundamenta Informaticae*, 34(1–2):53–101, 1998.

[71] A. Lomuscio and F. Raimondi. MCMAS: a tool for verifying multi-agent systems. In *Proceedings of The Twelfth International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS-2006)*. Springer-Verlag, Berlin, Germany, 2006.

[72] Z. Manna and A. Pnueli. *Temporal Verification of Reactive Systems—Safety*. Springer-Verlag, Berlin, Germany, 1995.

[73] 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*, pages 463–502. Edinburgh University Press, 1969.

[74] J.-J.Ch. Meyer, W. van der Hoek, and B. van Linder. A logical approach to the dynamics of commitments. *Artificial Intelligence*, 113:1–40, 1999.

[75] J.-J.Ch. Meyer and R.J. Wieringa, editors. *Deontic Logic in Computer Science—Normative System Specification*. John Wiley & Sons, 1993.

[76] R.C. Moore. Reasoning about knowledge and action. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence (IJCAI-77)*, Cambridge, MA, 1977.

[77] 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.

[78] 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.

[79] 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.

[80] M.J. Osborne and A. Rubinstein. *A Course in Game Theory*. The MIT Press, Cambridge, MA, 1994.

[81] M. Pauly. Logic for social software. PhD thesis, University of Amsterdam, 2001. ILLC Dissertation Series 2001-10.

[82] M. Pauly. A logical framework for coalitional effectivity in dynamic procedures. *Bulletin of Economic Research*, 53(4):305–324, 2002.

[83] M. Pauly. A modal logic for coalitional power in games. *Journal of Logic and Computation*, 12(1):149–166, 2002.

[84] M. Pauly and M. Wooldridge. Logic for mechanism design—a manifesto. In *Proceedings of the 2003 Workshop on Game Theory and Decision Theory in Agent Systems (GTDT-2003)*, Melbourne, Australia, 2003.

[85] A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proceedings of the Sixteenth ACM Symposium on the Principles of Programming Languages (POPL)*, pages 179–190, January 1989.

[86] S. Popkorn. *First Steps in Modal Logic*. Cambridge University Press, Cambridge, England, 1994.

[87] F. Raimondi and A. Lomuscio. Symbolic model checking of multi-agent systems via OBDDs: an algorithm and its implementation. In *Proceedings of the Third International Conference on Autonomous Agents and Multiagent Systems (AAMAS-04)*, pages 630–637, New York, NY, 2004.

[88] A.S. Rao. AgentSpeak(L): BDI agents speak out in a logical computable language. In W. Van de Velde and J.W. Perram, editors. *Agents Breaking Away: Proceedings of the Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, *LNAI*, vol. 1038, pages 42–55. Springer-Verlag, Berlin, Germany, 1996.

[89] A.S. Rao. Decision procedures for propositional linear-time Belief-Desire-Intention logics. In M. Wooldridge, J.P. Müller, and M. Tambe, editors. *Intelligent Agents II*, *LNAI*, vol. 1037, pages 33–48. Springer-Verlag, Berlin, Germany, 1996.

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

[91] A.S. Rao and M. Georgeff. Decision procedures for BDI logics. *Journal of Logic and Computation*, 8(3):293–344, 1998.

[92] A.S. Rao and M.P. Georgeff. Asymmetry thesis and side-effect problems in linear time and branching time intention logics. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 498–504, Sydney, Australia, 1991.

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

[94] A.S. Rao and M.P. Georgeff. An abstract architecture for rational agents. In C. Rich, W. Swartout, and B. Nebel, editors, *Proceedings of Knowledge Representation and Reasoning (KR&R-92)*, pages 439–449, 1992.

[95] A.S. Rao and M.P. Georgeff. A model-theoretic approach to the verification of situated reasoning systems. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI-93)*, pages 318–324, Chambéry, France, 1993.

[96] A.S. Rao, M.P. Georgeff, and E.A. Sonenberg. Social plans: A preliminary report. In E. Werner and Y. Demazeau, editors. *Decentralized AI 3—Proceedings of the Third European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW-91)*, pages 57–76. Elsevier Science Publishers B.V., Amsterdam, The Netherlands, 1992.

[97] R. Reiter. *Knowledge in Action*. The MIT Press, Cambridge, MA, 2001.

[98] 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.

[99] S.J. Rosenschein and L.P. Kaelbling. A situated view of representation and control. In P.E. Agre and S.J. Rosenschein, editors. *Computational Theories of Interaction and Agency*, pages 515–540. The MIT Press, Cambridge, MA, 1996.

[100] M. Ryan and P.-Y. Schobbens. Agents and roles: Refinement in alternating-time temporal logic. In J.-J.Ch. Meyer and M. Tambe, editors. *Intelligent Agents VIII: Proceedings of the Eighth International Workshop on Agent Theories, Architectures, and Languages, ATAL-2001*, *LNAI*, vol. 2333, pages 100–114. Springer, 2002.

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

[102] 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.

[103] V.S. Subrahmanian, P. Bonatti, J. Dix, T. Eiter, S. Kraus, F. Ozcan, and R. Ross. *Heterogeneous Agent Systems*. The MIT Press, Cambridge, MA, 2000.

[104] M. Tambe. Towards flexible teamwork. *Journal of AI Research*, 7:83–124, 1997.

[105] N. Troquard and A. Herzig. Uniform choices in STIT. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AMAS)*, 2006.

[106] R. Turner. *Truth and Modality for Knowledge Representation*. Pitman Publishing, London, 1990.

[107] W. van der Hoek, W. Jamroga, and M. Wooldridge. A logic for strategic reasoning. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2005)*, pages 157–153, Utrecht, The Netherlands, 2005.

[108] H.P. van Ditmarsch, W. van der Hoek, and B.P. Kooi. *Dynamic Epistemic Logic*. *Synthese Library*, vol. 337. Springer, Berlin, 2007.

[109] M.Y. Vardi. Branching vs. linear time: Final showdown. In T. Margaria and W. Yi, editors. *Proceedings of the 2001 Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2001*, *LNCS*, vol. 2031, pages 1–22. Springer-Verlag, Berlin, Germany, April 2001.

[110] D. Walther, C. Lutz, F. Wolter, and M. Wooldridge. ATL satisfiability is indeed ExpTime-complete. *Journal of Logic and Computation*, 16:765–787, 2006.

[111] M. Wooldridge. The logical modelling of computational multi-agent systems. PhD thesis, Department of Computation, UMIST, Manchester, UK, October 1992.

[112] M. Wooldridge. *Reasoning about Rational Agents*. The MIT Press, Cambridge, MA, 2000.

[113] M. Wooldridge, M.-P. Huget, M. Fisher, and S. Parsons. Model checking multi-agent systems: The MABLE language and its applications. *International Journal on Artificial Intelligence Tools*, 15(2):195–225, April 2006.

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

[115] M. Wooldridge and N.R. Jennings. The cooperative problem solving process. *Journal of Logic and Computation*, 9(4):563–592, 1999.
[116] M. Wooldridge and W. van der Hoek. On obligations and normative ability. *Journal of Applied Logic*, 3:396–420, 2005.
[117] E.N. Zalta. Stanford encyclopedia of philosophy. See http://plato.stanford.edu/.