Computational Intelligence, Volume 2, Number 5, 1993

COMPUTATIONAL LOGICS AND AGENTS: A ROADMAP OF CURRENT TECHNOLOGIES AND FUTURE TRENDS

Michael Fisher
Department of Computer Science, University of Liverpool, United Kingdom

Rafael H. Bordini
Department of Computer Science, University of Durham, United Kingdom

Benjamin Hirsch
DAI-Labor, Technische Universität Berlin, Germany

Paolo Torroni
Dipartimento di Elettronica, Informatica e Sistemistica, Università di Bologna, Italy

The concept of an *agent* is increasingly used in contemporary software applications, particularly those involving the Internet, autonomous systems, or cooperation. However, with dependability and safety in mind, it is vital that the mechanisms for representing and implementing agents are clear and consistent. Hence there has been a strong research effort directed at using formal logic as the basis for agent descriptions and agent implementation. Such a logical basis not only presents the clarity and consistency required but also allows for important techniques such as logical verification to be applied.

We present a roadmap of research into the use of computational logic in agent-based systems and survey much of the recent work in these areas. Even though, with such a rapidly changing field, it is impossible to cover every development, we aim to give the reader sufficient background to understand the current research problems and potential future developments in this maturing area.

Keywords: logic; autonomy; rational agents; multi-agent systems; agent specification; agent verification; logical implementation

1. INTRODUCTION

Representing and manipulating systems comprising large numbers of evolving and interacting computational entities is difficult. The concept of an *agent* was introduced as a useful abstraction for describing such computational entities, with the larger structures forming *multi-agent systems*. Representing these computational elements as agents and agent *organisations* provides a simple and intuitive metaphor for both individual and collective computation. However, it is important that software designers have appropriate and semantically clear mechanisms available for representing not only how individual agents adapt and evolve, but also how agents interact and combine to form new systems. Without the ability to represent and control agents, the practical development of complex multi-agent systems remain difficult, and agents will not be trusted for use in critical applications.

In this document, we describe the use of *formal logic* as the basis for both determining and representing the behaviour of individual agents as well as agent organisations. The fact that formal logics have been used for this purpose is not surprising since they provide a framework that is both unambiguous and well understood, and in which specific logics can be designed to suit different scenarios. In particular, when an appropriate logic is chosen, it can provide a level of abstraction close to the key concepts of the software to be developed, as is the case with logics for multi-agent systems. We provide a survey of how computational logic (i.e., formal logic applicable to computing, for decision procedures, operational programming languages, etc.) can be used for agent-based systems. In particular, we will address:

- *agent description*, through logical specification and semantic techniques;
- *agent implementation*, through the execution of logical statements; and
- *agent analysis*, through logic-based verification techniques.

The link between agents and logic is important to both sides: logic-based approaches to agents are likely to grow in importance especially as verification becomes more necessary, while agents (particularly those that are required to *reason*) represent a key challenge for computational logic developments and application. Thus, we aim to describe the state of the art in key areas concerning logic and agents. In addition, we will also consider future trends in computing where computational logic approaches to agents ought to be relevant — this will serve to identify key future research issues and key requirements for wider use of these techniques.

2. LOGIC-BASED AGENT DESCRIPTION [FORMAL SPECIFICATION]

Although we will not delve in detail into the "what is an agent" debate, we will briefly outline the variety of agents we consider. While an object can be seen essentially as a structure *encapsulating* behaviour (e.g., state, methods, ...), an agent is an *autonomous* object. Thus, the agent makes its own decisions about what to pursue. We are particularly concerned with *rational agents*, which can be seen as agents that make their decisions in a rational and explainable way. Thus, one of the key new aspects that the agent abstraction brings is that, for both design and verification, we need to consider not just what they do but *why* they do it. Since agents are autonomous, understanding why an agent chooses a course of action is vital. Wooldridge and Jennings (1995) describe a number of additional aspects of agents, in particular their *pro-active*, *reactive* and *social* attributes. Although we do not see these as essential, most of the agents considered here do, indeed, incorporate such notions.

A number of logical theories of (rational) agency have been developed, such as the BDI (Rao and Georgeff, 1991, 1995) and KARO (van der Hoek *et al.*, 1993b; van Linder, 1996) frameworks. One reason for using logic in agent-based systems is that a formal semantics comes (more or less) for free. From this it follows that the semantics of logic-based agents is strongly dependent on their underlying logic. The foundations of such agent description frameworks are usually represented as (often complex) non-classical logics. In addition to their use in agent theories, where the basic representation of agency and rationality is explored, these logics often form the basis of formal methods for agent systems.

The leading agent theories and formal methods in this area all share similar logical properties. In particular, the logics used bring together:

- an *informational* component, so as to be able to represent the agent's beliefs or knowledge;
- a *motivational* component, in order to representing the agent's desires or goals; and
- a *dynamic* component, allowing the representation of the agent's activity.

Thus, the predominant approaches use relevant logical combinations: the BDI approach combines dynamic aspects via propositional temporal logic, informational aspects via modal logic $\mathsf{KD45}$, and motivational aspects via modal logic KD; the KARO approach combines dynamic aspects via propositional dynamic logic, informational aspects via modal logics $\mathsf{S5}/\mathsf{KD45}$, and motivational aspects via modal logic KD. Unfortunately, these combinations can often be complex, both in terms of understandability and reasoning (Fagin *et al.*, 1995; Bennett *et al.*, 2002).

To give a flavour of the different approaches, we provide below brief overviews of some approaches to logic-based agent description.

- **BDI**
  The BDI framework was developed by Rao and Georgeff over a number of years (Rao and Georgeff, 1991, 1995) and actually comprises two aspects: an agent theory and a practical agent architecture. The former is a well-known agent logic, or rather a framework for agent logics, based on Bratman's theory of practical reasoning (Bratman, 1999). The latter is a much-copied architecture for building agents that are able to *reason* about their environment and *deliberate* about their actions.
  The BDI theory describes agent behaviour using 'mentalistic' attributes, in particular **B**eliefs, **D**esires, and **I**ntentions, and uses these three components to form an agent's *mental state*. Belief corresponds to the information the agent has about its environment; desires (or goals) represent *options* available to the agent; and Intentions are the goals the agent has chosen to tackle immediately. Rao and Georgeff use branching time temporal logic (Emerson and Srinivasan, 2001) to incorporate the dynamic aspect (of action within an environment), and a Kripke structure (Kripke, 1963) to model the mentalistic notions. Thus, each Kripke world consists of a time tree in which the result of executing (or failing to execute) each action is represented.
  The basic BDI architectural model, for example exemplified within both the Distributed Multi-Agent Reasoning System (dMARS) and the Procedural Reasoning System (PRS) (Georgeff and Lansky, 1987; Myers, 1993; Rao and Georgeff, 1995) can be summarized as follows. Essentially, such agents:
  - observe the world and the agent's internal state, and update the event queue to reflect the events that have been observed;
  - generate new possible desires (tasks), by finding plans whose trigger event matches an event in the event queue;
  - from this set of matching plans, select one for execution (an intended means);
  - push the intended means onto an existing or new intention stack, according to whether or not the event is a subgoal; and
  - select an intention stack, take the topmost plan (the intended means), and execute the next step of this current plan — if the step is an action, perform it, otherwise, if it is a subgoal, post this subgoal on the event queue.

  This approach has been extremely influential and is the basis for several agent systems and programming languages such as the PRS (Georgeff and Lansky, 1987; Myers, 1993), dMARS (Rao and Georgeff, 1995; d'Inverno *et al.*, 1998), and AgentSpeak(L) (Rao, 1996).
- **KARO**
  The foundations of KARO (van der Hoek *et al.*, 1993b; van Linder, 1996) originate in the work of Moore on knowledge and action (Moore, 1984). Its name represents the main ingredients of the theory: **K**nowledge, **A**bilities, **R**esults, and **O**pportunity. The core logic consists of dynamic logic (Harel, 1984), used to model abilities and opportunities, and the well-known S5 logic (Fagin *et al.*, 1995), used to model knowledge. Abilities and opportunities are formalised as *potentials* using propositional dynamic logic and logics of knowledge. Numerous extensions exist, augmenting the core language with formalisations for commitments, wishes, and goals (van Linder *et al.*, 1997; van der Hoek *et al.*, 1993a; van Linder *et al.*, 1998; Meyer *et al.*, 1999; van der Hoek *et al.*, 2000). Proof methods have been investigated, as has the influence of (non-)determinism (van der Hoek *et al.*, 1993c; Meyer *et al.*, 2001; Hustadt *et al.*, 2001).

- **Temporal Logics**
  While some approaches use *dynamic logics* to represent the underlying dynamic activity of an agent, many also use *temporal logics* (Emerson, 1990; Goldblatt, 1992) for this purpose. A straightforward use of discrete, linear temporal logics in specifying (and, as we will see later, verifying and executing) agents is given by Wooldridge (1992); Fisher (1995); Fisher and Wooldridge (1997); Fisher (2005b); this approach is in turn based upon work on temporal specifications of distributed and concurrent processes (Barringer *et al.*, 1996). Much of this work uses a separated normal form (SNF) (Fisher, 1997b), in which temporal formulæ describe what the agent must do *now*, what it must do *next*, and what it guarantees to do at *some* point in the future. Temporal formulæ in SNF form can be *directly executed*.
  However, just as there is a *wide* variety of temporal logics, so there are numerous different uses of temporal logics in agent-based systems. For example, approaches to temporal semantics of agent computation have been developed using techniques as disparate as Temporal Logic of Actions (Merz *et al.*, 2003), *dense* temporal logics (Fisher, 1996), and Temporal Z (Regayeg *et al.*, 2004). In addition, temporal techniques have been developed for multi-agent systemas as well as autonomous agents; see, for example, Singh (1996); Jonker *et al.* (2001). In many cases the basic underlying temporal formalism has been extended with other logical dimensions such as those for knowledge, belief, goals, and motivations (for example Fisher (2005b)). Thus, this seemingly simple language is flexible enough to describe individual agents in terms of their dynamic behaviour, how their knowledge/information evolves, and how their goals/motivations evolve.
- **Deontic Logic**
  Deontic logic is a form of modal logic where operators concern aspects such as *permission* and *obligation*. Deontic notions have been the basis of a number of agent descriptions, two of which are the IMPACT and BOID systems. IMPACT is a framework for coordinating (using hierarchical planning) legacy systems which is based on a logical foundation combining time, uncertainty, and deontic (Eiter and Subrahmanian, 1998) operators (we will discuss IMPACT further later).
  An approach to agent programming that extends the BDI concepts with deontic notions, particularly obligations, is the BOID architecture (**B**elief, **O**bligations, **I**ntentions, **D**esires) (Broersen *et al.*, 2001, 2002). The argument here is that rational agents need to take into consideration norms and obligations that are set by the (agent) society (Dignum, 1999), and so must reason about them. The BOID architecture is an extension of the BDP logic (Thomason, 2000), which in turn is rooted in Reiter's default logic (Reiter, 1980). It is based on conflict resolution for conditional beliefs, desires, obligations, and intentions. Instead of using modalities to model each attitude, the different notions are defined by sets of rules described by propositional logic. Default rules for each set are iteratively applied to the current theory. Whenever a conflict arises, a preference relation over the default rules is used to solve it.
- **Alternating-Time Temporal Logic**
  Temporal logics (Emerson, 1990) have been studied for many years and form the basis of much of the work on verification of computational systems, as we will see later. Such logics come, traditionally, in two flavours: linear-time or branching-time. A third, more general, alternative is Alternating-Time Temporal Logics (ATL), introduced by Alur *et al.* (2004) and extended by Alur *et al.* (2002). An interesting characteristic of this alternative to standard temporal logics is that one can quantify, selectively, over paths that are seen as possible outcomes of game-theoretic interactions (to represent, for example, the possible interactions of an agent and its environment). In ATL, the individual sources of non-determinism (i.e., choices on "actions" that effect changes in

the system or environment) can be referred to explicitly, hence the interest in the use of such logics for describing and reasoning about multi-agent systems. For example, Ryan and Schobbens (2002) use ATL to define a refinement relation between agents, which is useful for agent verification.

A combined logic of potential interest for multi-agent systems, introduced by van der Hoek and Wooldridge, is ATEL (van der Hoek and Wooldridge, 2003): an epistemic version of ATL which allows one to reason about what knowledge (or common knowledge, etc.) a coalition can effectively bring about or, indeed, what knowledge is required from agents so that they can effectively cooperate to bring about certain states of affairs. In combined logics, however, the semantics of interactions among the various modalities can often lead to various difficulties and complexities.

- **Situation Calculus**
  The Situation Calculus (McCarthy and Hayes, 1969; Levesque *et al.*, 1998a) is a first order language with some second order features that was originally developed to formalise dynamic aspects of the world, including concepts of ability and belief (Levesque *et al.*, 1998b; Pirri and Reiter, 1999). The logic itself is rather versatile, and has been expanded to include, among other things, concurrency (De Giacomo *et al.*, 2000), time (Pinto and Reiter, 1993), non-monotonic reasoning (Baker, 1991), and events (Lakemeyer, 1999; Lin, 1995).
  Formulæ in the situation calculus are usually interpreted over a set of axioms. A situation can be seen as a possible world history, i.e., a sequence of actions that led to the current state. Situations change by means of *fluents* (functions that are applied to situations). Fluents have pre- and post-conditions in order to minimise the amount of possible successor states. The situation calculus has been used to specify and implement agents, as we will see when the Golog programming language is mentioned later in this paper.
- **Z**
  The formal specification language Z (Spivey, 1992) is based on set theory and first order predicate calculus. It extends the use of these languages by allowing an additional mathematical type known as *schema type*. Z schemas consist of a declarative part that declares variables and their types, and a predicate part that relates and constrains those variables.
  A number of attempts have been made to create a theory in Z upon which agents can be based, with probably the most successful being those languages that provide Z specifications for objects (Mahony and Dong, 2000; Smith, 2000). However, capturing an agent's *motivations* as part of the core of Z appears problematic. Rather than creating an original agent theory based on Z, several existing agent theories and languages have been modelled using the Z specification language, for example AgentSpeak(L) (d'Inverno and Luck, 1998) and dMARS (d'Inverno *et al.*, 1998).
- **Nonmonotonic Logics**
  Nonmonotonic logics are used to formalise plausible reasoning (Morgenstern, 1999). Reiter's *default logic* (Reiter, 1980), which we have mentioned in relation with the BOID architecture, is a milestone in this area. Its key feature is the notion of *default rules*, used to model what typically happens given some assumptions and under certain conditions. Beside Reiter's work, among nonmonotonic logics we count: Moore's *autoepistemic logic* (Moore, 1985), relating nonmonotonicity with rational agent beliefs; McCarthy's logic of *circumscription* (McCarthy, 1986), formalizing nonmonotonicity by means of second-order axioms that limit the extension of certain "abnormal" predicates; and Alchourron et al.'s work on Belief Revision (Alchourron *et al.*, 1985). Here, the authors propose a number of postulates, known in the literature as AGM postulates, capturing the proper-

ties that a belief revision operator should satisfy in order to be considered rational. It is essentially a study of nonmonotonic reasoning from the dynamic point of view, and as such it has been considered very suitable to model rational agents. The general principle that should govern belief revision is indeed that of *minimal change* of information, while adapting to new input from the outside. Among the nonmonotonic forms of reasoning that are very relevant in agent research, we particularly identify *hypothetical reasoning* and *abduction*. Originally intended to model the reasoning process that leads from observations to plausible explanations (i.e. *hypotheses* (Hartshorne and Weiss, 1965)), abduction has been very much adopted as a general modelling and reasoning framework to extend incomplete belief or knowledge bases. This is very appealing in the agent domain, where knowledge incompleteness typically arises either because agents have to interact with other agents based on limited information, or where agents are situated in dynamic environments, subject to continuous change and evolution.
Because of the intrinsic complexity of such theories, the application of nonmonotonic reasoning in the agent context is as yet limited, with a few exceptions. For this reason, *belief revision for resource-bounded agents* has become an increasingly popular research area, leading to implementable logics for agent (Alechina *et al.*, 2006b,a). On the other hand, logic programming and extensions have provided efficient techniques to handle many types of nonmonotonic and hypothetical reasoning (we elaborate on this later).

3. LOGIC-BASED AGENT IMPLEMENTATION [PROGRAMMING LANGUAGES]

As we have seen in the previous section, there are many ways in which logic can be used to specify (and so provide semantics for) agent-based systems. However, we are also interested in *building* agent-based systems using techniques based upon logic. In other words, we are interested in the development and application of *computational logic* in this area. Computational logic has been very successful in the past, particularly through languages such as Prolog, in providing high-level abstractions for programming complex systems, while retaining a close link to formal semantics. The transparent formal semantics of logic-based approaches allows the possibility of applying formal verification techniques, thus enhancing the trustworthiness of the software.

In this section, we will survey languages that have the notions of agency and computational logic at their core. There is a wide variety of such languages (Bordini *et al.*, 2005a); while it is impossible to cover all of them, we aim to highlight a range of languages that use logical machinery to implement rational agents. It should be noted that this covers much more than logic programming, which is but one approach to implementation using logic. For example, languages such as AGENT-0 and Go! contain some elements of functional programming, while MetateM is based on forward chaining model construction. In addition, while there is a wide range of logic-based languages that implement very basic forms of agency, these are often little more than extensions of logic programming that capture communication and encapsulation. Thus, we will concentrate on a number of different mechanisms for using computational logic to implement specifically *rational* agents.

- **AGENT-0**
  One of the seminal works in this area was produced by Shoham (1993) who proposed *Agent-Oriented Programming* (AOP) as a new paradigm for implementing (rational) agents. This framework consists of a logical system, called AGENT-0, for defining the mental state of agents, together with an interpreted language for programming those

agents (Thomas, 1995). The logical component of AGENT-0 is a quantified multi-modal logic, with three modalities: belief, commitment, and ability. An example of an acceptable formula of AGENT-0 is as follows:

$$CAN^{4}_{pele} pass(ball)^{9} \;\Rightarrow\; B^{4}_{maradona} CAN^{10}_{maradona} score(goal)^{11}$$

The intended meaning of this statements is

> *if at time 4 pele can ensure that he will pass the ball at time 9, then at time 4 maradona believes that at time 10 he (maradona) can ensure a goal is scored at time 11.*

As can be seen from the example above, the logical basis of AOP contains notions of time, as well as other modalities, and the programming language provides syntactic support for such concepts (though it is not clear what the formal link between the programming language and the logic is (Wooldridge and Jennings, 1995)). Rather than being widely used itself, it is the idea of a language for developing agents based upon multi-modal logics that has been taken up by other researchers. AGENT-0 has been the inspiration for many subsequent agent programming languages.

- **MetateM** and **Concurrent MetateM**
  While most agent specifications are based on a combination of temporal or dynamic and modal logics, MetateM (Barringer *et al.*, 1995) was initially based on *temporal logic* alone. Temporal formulæ in a particular normal form (called SNF (Fisher, 1997b)) describe what the agent must do *now*, what it must do *next*, and what it guarantees to do at *some* point in the future. Most importantly, temporal formulæ in SNF form can be *directly executed* in order to implement the behaviour of an agent. This basis was extended for increasingly sophisticated forms of agency (Fisher, 1995), involving belief, deliberation (Fisher, 1997a), resource-bounded reasoning (Fisher and Ghidini, 1999) and confidence (Fisher and Ghidini, 2002). In parallel, the language was developed into a concurrent version, for multi-agent systems (Fisher, 1994, 2005a) and, more recently, work on this has moved towards flexible group organisation (Fisher *et al.*, 2003), multi-agent collaboration (Fisher *et al.*, 2004), and applications in pervasive/ubiquitous computing (Hirsch *et al.*, 2005). For an overview of these developments, see Fisher (2005a).
- **Golog** and **ConGolog**
  Golog (Levesque *et al.*, 1997) is a high-level specification language, based on the situation calculus, extending it with control structures such as sequencing, test actions, non-deterministic choice (of both actions and programs), and procedures (essentially macros). The execution of a Golog program involves proving the theorem expressed by the program with respect to some background theory (expressed by its axioms). As a byproduct, variables are instantiated and so provide an execution trace. In this, it is reminiscent of Prolog and, not surprisingly, a Golog interpreter can easily be described in Prolog.
  As with the situation calculus, the basic Golog programming language can be extended considerably, for example with sensing actions relevant to robotics (Lakemeyer, 1999). One of the most prominent extensions is ConGolog, a concurrent variety of Golog (De Giacomo *et al.*, 2000). The concurrency described is not true concurrency but rather an interleaving of processes. (Of course, on a one-processor machine this is always the case, but "truly" concurrent programs let the operating system take care of the actual interleaving, setting only their priority.) However, since ConGolog incorporates thread priorities and interrupts, either coarse-grained or fine-grained interleaving can be described. In the extreme case, it is possible to interleave whole programs rather than program steps.

Golog has been particularly applied in the (cognitive) robotics area, but is also being developed for use in the semantic web (McIlraith and Son, 2001, 2002).

- **AgentSpeak**
  While BDI architectures have been used in many practical agent systems, there are few high-level programming languages based systematically on the principles of BDI theory. AgentSpeak(L) is such a language, defined to be an extension of logic programming incorporating beliefs, events, goals, actions, plans, intentions (Rao, 1996); its reasoning cycle is very similar to the dMARS cycle described above. The behaviour of the agent (i.e., its interaction with the environment) is dictated by the plans written in AgentSpeak. Although the beliefs, desires, and intentions of the agent are not explicitly represented as modal formulæ, such mental attitudes can be formally ascribed or checked given a representation of an agent's current state.
  The practial use of AgentSpeak is being facilitated by a new interpreter and multi-agent platform called *Jason* (Bordini and Hübner, 2006). The interpreter implements, in Java, the operational semantics of an extended version of AgentSpeak (Bordini *et al.*, 2005b). *Jason* is available *open source* (see `http://jason.sourceforge.net`).
- **Logic Programming and Non-Monotonic Reasoning**
  Following Kowalski and Sadri's work on the *observe-think-act* agent cycle and architecture to reconcile rationality with reactivity (Kowalski and Sadri, 1999), a conspicuous body of literature introduced agent theories based on Logic Programming (Lloyd, 1987). This choice is often justified by the aim to link declarative semantic agent descriptions to execution models. Therefore, even when not as expressive as other formalisms, logic programming languages and variations become appealing when they come together with formal results that link declarative and operation semantics.
  Among the various logic programming frameworks and languages, the most commonly used for agent theories are: Answer Set Programming (Baral, 2003), Abductive Logic Programming (Kakas *et al.*, 1993) for hypothetical and temporal reasoning, Dynamic Logic Programming (Alferes *et al.*, 1998) for knowledge and program updates, and Logic Programming without Negation as Failure (Dimopoulos and Kakas, 1995) for agent deliberation.
- **3APL** [`http://www.cs.uu.nl/3apl`]
  3APL is a programming language for implementing cognitive agents (Hindriks *et al.*, 1998). It provides programming constructs for implementing an agent's beliefs, goals, basic capabilities such as belief updates or motor actions, and a set of practical reasoning rules through which the agent's goals can be updated or revised (Dastani *et al.*, 2003). 3APL programs are executed by an interpreter that implements an deliberation cycle using such constructs.
  The 3APL language is a combination of imperative and logic programming. From the imperative programming viewpoint, 3APL inherits the full range of regular programming constructs, including recursive procedures and state-based computation. States of agents in 3APL, however, are belief (or knowledge) bases, which are different from the usual variable assignments of imperative programming. From the computational logic perspective, answers to belief-base queries of a 3APL agent are proofs in the logic programming sense.
  The authors of 3APL have also examined a number of related agent languages, including **Dribble** (van Riemsdijk *et al.*, 2003) and **GOAL** (Hindriks *et al.*, 2001).
- **IMPACT** [`http://www.cs.umd.edu/projects/impact`]
  IMPACT (Arisha *et al.*, 1999) evolved through an international research project led by the University of Maryland. The main aim of the IMPACT project was to "develop both a theory as well as a software implementation facilitating the creation, deployment,

interaction, and collaborative aspects of software agents in a heterogeneous, distributed environment" (Eiter *et al.*, 1999; Eiter and Subrahmanian, 1999; Eiter *et al.*, 2000, 2002). Although IMPACT is only partially based on logic, providing a "thin" logic component on top of traditional data structures, the logic component involves time, uncertainty and deontic logics (Eiter and Subrahmanian, 1998). It also makes use of stable models (Gelfond and Lifschitz, 1988) for implementing agent rationality.
On the practical side, IMPACT provides a set of servers (yellow pages, thesaurus, registration, type and interface) that facilitate agent interoperability in an application independent manner. It also provides an Agent Development Environment for creating, testing, and deploying agents. Thus, IMPACT provides techniques and tools to build agents on top of existing legacy code, and algorithms supporting a variety of applications (command and control, logistics, e-commerce, data mining, etc.).

- **JIAC IV** [`http://www.jiac.de`]
JIAC is an agent based service-ware framework geared towards telematics and telecommunication services. It has three major elements: it employs a flexible component structure which allows agents to quickly create and dynamically adapt their capabilities; it is knowledge based, as the agent architecture follows a BDI approach, based on three-valued predicate logic; and its communication model is structured around the service concept. Agents are programmed using JADL (JIAC Agent Description Language), which comprises ontology definitions, declarative plan elements, and procedural script-like elements. Java objects can be invoked from JADL using AgentBeans. Due to its strong connection with industry the framework features advanced security as well as management facilities.
- **SOCS** [`http://lia.deis.unibo.it/research/socs`]
SOCS (standing for Societies Of ComputeeS) was an European project aimed at defining a computational logic model for the description, analysis and verification of global and open societies of heterogeneous *computees*, i.e., agents in computational logic (Toni, 2006). The main outcomes of the project are definitions of the agent and society models and the proof-procedures implementing the operational models. The computee model is based on the $\mathcal{KGP}$ ($\mathcal{K}$nowledge, $\mathcal{G}$oals, $\mathcal{P}$lans) agent architecture (Kakas *et al.*, 2004b; Sadri and Toni, 2006; Sadri, 2006), whereas the $SOCS$ social model is based on Social Integrity Constraints ($\mathcal{IC}_S$) (Alberti *et al.*, 2005b; Chesani *et al.*, 2006). The proof-procedures to implement the operational models are the $\mathcal{S}CIFF$ (Alberti *et al.*, 2005a) for interaction verification and $CIFF$ (Endriss *et al.*, 2004) for temporal reasoning and planning, both of them extensions of Fung and Kowalski's *IFF* abductive proof-procedure (Fung and Kowalski, 1997), and Gorgias (Kakas and Moraitis, 2003) for preference reasoning and deliberation, based on LPwNF (Dimopoulos and Kakas, 1995). A prototype of *societies of computees* is implemented in the PROSOCS architecture (Bracciali *et al.*, 2006; Alberti *et al.*, 2006).
The distinguishing features of the $\mathcal{KGP}$ model are: a modular architecture, where reasoning capabilities can be implemented independently of each other, and a flexible agent cycle that can be programmed using preference rules. The $SOCS$ social model is mainly oriented towards specification and verification of agent interaction, and $\mathcal{IC}_S$, its main component, can be used to express commitment-based semantics for agent communication languages and to define agent protocols, as well as for run-time verification of whether agents comply such specifications.
- **Agent reasoning and coordination based on Abductive Logic Programming**
Kowalski *et al.* were the first authors to propose an agent architecture based on Abductive Logic Programming (Kakas *et al.*, 1993), to reconcile reactivity with rationality. KS agents adopt an *observe-think-act* cycle, where *think* is a (fixed) number of computational steps that the IFF proof-procedure (Fung and Kowalski, 1997) uses to compute actions

to perform, based on the agent's current beliefs, goals, and inputs. The agent goals are a conjunction of Prolog terms and integrity constraints, i.e., forward rules that must be true at all times during the operation of the agent.
The KS agent architecture has been extended by Dell'Acqua and colleagues (Dell'Acqua *et al.*, 2002) to accommodate *updates*, i.e., dynamic changes to the agent programs. KS agents have also been used by Sadri *et al.* to implement agents that engage in negotiation dialogues in the context of resource allocation (Sadri *et al.*, 2001, 2002, 2003). A more elaborate agent model which can be considered as a descendant of KS agents is the $\mathcal{KGP}$ model mentioned above.
Agent deliberation is the main focus of work done by Kakas *et al.* based on Logic Programming without Negation as Failure (LPwNF) (Kakas and Moraïtis, 2002; Kakas and Moraitis, 2003; Kakas *et al.*, 2004a). The agent decision making process is specified by a number of preference rules structured in a multi-layered program. LPwNF is used to implement an argumentation framework where policies are combined with context rules. Argumentation is used to solve conflicts arising, for instance, from an agent policy giving directives that are incompatible with those given by the context rules.
Orthogonal interest has been devoted by several authors to the issue of coordination of multi-agent reasoning activity. Two noteworthy bodies of work are the research conducted by Ciampolini *et al.* on ALIAS (Ciampolini *et al.*, 2003, 2002) and that by Satoh *et al.* on speculative computation (Satoh *et al.*, 2000; Satoh, 2002; Satoh and Yamamoto, 2002). ALIAS is a multi-agent architecture where agents can reason hypothetically, making assumptions about missing pieces of information (as for KS agents), and where they can form groups ("bunches") to coordinate their reasoning activity so as to ensure that their beliefs about unknown information are consistent with one another's integrity constraints. This kind of interaction is kept at a separate level from that of reasoning, and is described by a language called LAILA (Ciampolini *et al.*, 2002). Speculative computation is another form of collective and coordinated hypothetical reasoning, based on abductive logic programming. Its main purpose is to enable agents to continue their execution while waiting for pieces of information from the outside, based on default assumptions, and to switch to a different operation in case some incoming information contradicts their defaults.

- **Dynamic Logic Programming** and **Updates**
  An increasingly popular logic programming based formalism for modelling multi-agent reasoning is Answer Set Programming, and extensions of Dynamic Logic Programming to accommodate adaptation and changes. Leite *et al.* developed an agent architecture, called $\mathcal{MINERVA}$, based on Multi-Dimensional Dynamic Logic Programming ($\mathcal{MDLP}$) (Leite *et al.*, 2001) and on the update command language $LUPS$ (Alferes *et al.*, 2002), where several specialised, possibly concurrent, sub-agents perform various tasks while reading and manipulating a common knowledge base. Related to $\mathcal{MINERVA}$ is the work on DALI (Costantini and Tocchio, 2006), an agent programming language encompassing basic patterns for reactivity, pro-activity, internal thinking, and memory. The semantics of a DALI program is defined in terms of another program, where reactive and proactive rules are reinterpreted in terms of standard Horn Clause rules.
- **April** and **Go!**
  April (McCabe and Clark, 1995) combines logic and functional programming. Although a high-level language, it offers a simple interface to native code in languages such as C. As the authors put it: "April is more an object based concurrent language with objects as processes". Go! (McCabe and Clark, 2004) is a descendant of April that is multi-threaded, strongly typed, and higher order. Agents in Go! contain both reactive and deliberative aspects, and coordinate using BDI structures.

- **Qu-Prolog**
  Finally, Qu-Prolog (Clark *et al.*, 2001) is a multi-threaded and distributed version of Prolog with support for high-level communication between the threads. Agents can communicate using thread-to-thread messages, a publish-and-subscribe event notification system, or via the shared Prolog dynamic database. The first two alternatives allow threads in different Qu-Prolog processes anywhere on the internet to communicate; the last communication alternative is restricted to threads in the same process (Clark *et al.*, 2006).

Agent Communication Languages and Ontologies

An important aspect of multi-agent systems development is agent interaction. For this, Agent Communication Languages (ACL) with appropriate formal semantics are required, as is the use of ontologies so that agents agree on the semantics of the vocabulary used in communication. In this section, we briefly mention the most prominent work on ACL and ontologies, discussing also their logical basis.

- **KQML** [`http://www.cs.umbc.edu/kqml`]
  The Knowledge Query and Manipulation Language (KQML) is a language and protocol for knowledge exchange (Labrou and Finin, 1994). It was developed as part of a larger effort, the ARPA Knowledge Sharing Effort which was aimed at developing techniques and methodologies for building large-scale, sharable, reusable knowledge bases. KQML is both a message format and a message-handling protocol to support run-time knowledge sharing among agents. It can be used as a language for an application program to interact with an intelligent system or for two or more intelligent systems to share knowledge in support of cooperative problem solving.
  KQML focuses on an extensible set of performatives, which define the permissible operations that agents may attempt on each other's knowledge and goal stores. The performatives comprise a substrate on which to develop higher-level models of inter-agent interaction such as the contract net protocol and negotiation techniques. In addition, KQML provides a basic architecture for knowledge sharing through a special class of agents called "communication facilitators", which coordinate the interactions of other agents. The ideas underlying the design of KQML have been explored through experimental prototype systems which were used to support several testbeds in areas such as concurrent engineering, intelligent design, and intelligent planning and scheduling. Even though FIPA (see below) is becoming the standard ACL, it took most of its ideas from KQML, which also has a historical importance in being the first ACL to achieve widespread use.
- **FIPA** [`http://www.fipa.org/activities/semantics.html`]
  The Foundation for Intelligent Physical Agents (FIPA) was formed in 1996 to produce software standards for heterogeneous and interacting agents and agent-based systems. In the development of these standards, FIPA obtained input and collaboration from its membership and from the agent's field in general to build specifications that can be used to achieve interoperability between agent-based systems developed by different companies and organisations.
  FIPA is currently working on a new semantic framework to reflect the needs of verifiability and conformance. In particular, the objective is to adopt or define a semantic framework that can give an account of FIPA's existing aspects as well as a number of additional constructs such as contracts, agreements, policies, and trust.
- **OWL** [`http://www.w3.org/2004/OWL/`]

It is increasingly clear and widely recognised that in order to provide true interoperability, it is not enough to standardise on agent communication languages. The actual content of the messages also has to be interpreted in a semantically compatible way. Interpretation of information by computing systems is at the centre of the semantic web vision. In that area, the aim is to augment the syntactic description of web pages with semantic information.
While HTML allows us to display the information available on the web, it does not provide much capability for describing the information in ways that facilitate the use of software programs to find or interpret it. As an example, the word 'agent' on a web site might represent an aspect of Computer Science, an aspect of Chemistry, or a participant of an espionage plot.
OWL (McGuinness and van Harmelen, 2004), the successor of the DAML+OIL (van Harmelen *et al.*, 2001; Horrocks *et al.*, 2002) language, provides a rich set of constructs with which to create ontologies and to markup information so that it is machine readable and understandable. A key approach here is the use of Description Logics (Baader *et al.*, 2003), which allows reasoning about semantic information. There are three variants of the OWL language: OWL-light, OWL-DL (description logic), and OWL-Full. While OWL-Light allows for easy and efficient implementations, OWL-DL makes use of the expressive power of description logic while still being computable. OWL-Full violates some constraints of description logic (Bechhofer *et al.*, 2004). OWL is a W3C Recommendation since February 2004.

4. LOGIC-BASED AGENT VERIFICATION [AGENT ANALYSIS]

Agents appear to be an appropriate abstraction for many systems that operate in complex, dynamic, and unpredictable environments. These include application areas such as air-traffic control, autonomous spacecraft control, health care, and industrial systems control, to name just a few. These are areas of application that clearly require *dependable* computational systems and, in particular, that such systems should be *verifiable*. In this section we address this issue, providing an overview of several approaches that aim to develop techniques for the (automatic or semi-automatic) verification of agent-based systems.

When using logical techniques, there are two common approaches to the verification of computational systems: model checking (Clarke Jr. *et al.*, 1999) and theorem proving. Model checking essentially requires a finite-state description of the possible behaviours of the system, together with a property to be checked. A model checker then carries out an exhaustive search through the finite-state description attempting to find a way where the property can be falsified. If the model checker is unable to find such a counter-example, then the property is satisfied on *all* possible executions of the system. Model checking has become very popular over recent years, with theoretical advances in the use of model checking combined with significant efficiency improvements in model checking tools leading to a marked increase in the practical use of model checkers. Alternatively, theorem proving requires a logical formula that captures *all* the behaviours of the system (whether finite or not) and a property, expressed as a logical formula. Then a proof must be carried out, showing that the formula describing the property is a *logical consequence* of the formula describing the system. While model checking is usually automatic, theorem-proving usually involves human intervention. Between these two extremes are proof problems that can be solved automatically, yet involve infinite-state behaviours.

Typically, autonomous agents in a multi-agent system need to share and/or compete for resources. The usual approach is that resources are owned by specific agents and the

allocation mechanism is based on some form of negotiation technique. Usually, the communication and allocation protocols used need themselves be verified as they could incur in well known faults (e.g., starvation). Indeed, any such protocols can be independently checked for properties that guarantee no such faults can occur. Ideally, we will have, in the near future, libraries of such high-level agent coordination mechanisms associated with properties that have already been verified. Developers of multi-agent systems could then use these "off the shelf", while being aware of their properties (as well as the requirements with which agents should comply in order to ensure appropriate interaction).

Although still at an early stage, this points to the use of compositional reasoning for the formal verification of multi-agent systems. In addition, verification must take into account the use of *legacy* code, something which is often used in multi-agent systems. Further, it is important to note that verification of multi-agent systems can work on several levels of abstraction: the level of social interactions and social organisations; the individual level where the agent's internal reasoning is targeted; and also an even lower level concerning specific algorithms or legacy code used internally to an agent.

While most of this research aims to reason on agent specifications or models with an a-priori approach, a different aspect of verification in multi-agent systems is run-time verification. This accounts to monitoring and checking whether a particular implemented, running agent does indeed operate according to its specification. There exist logic-based tools that reason upon the externally observable behaviour of interacting agents and verify whether it complies to predefined norms or protocols. In particular, the SOCS-SI tool (Alberti *et al.*, 2006) (already mentioned in the previous section in relation with SOCS), uses the abductive logic programming SCIFF proof-procedure to consider messages exchanged by agents plus other events and carry out such a run-time interaction verification task. Though comparatively less studied, this is also an important aspect of developing dependable agent systems.

However, while model checking techniques can be seen as appropriate solutions to many of the above problems, they are less satisfactory when we consider issues such as openness (i.e., an arbitrary number of autonomous agents may be involved) and emergence (i.e., coping with evolving agent behaviours). Here, theorem-proving appears to have advantages over model-checking. Although it is early days, our view is that combining deductive (Fisher, 2005b) and algorithmic techniques (i.e., the various approaches based on model-checking discussed below) will have an important role in the verification of multi-agent systems in the future.

Regardless of how long it takes to achieve these more ambitious objective, in the short term we expect to see an increase in the number of researchers interested in formal verification of multi-agent systems, mainly because of the adequacy of such systems for many areas of application that demand large-scale dependable systems. In particular, we expect to see both theorem-proving and model-checking techniques, and hopefully combinations of both, being developed for the many logics of interest for multi-agent systems. Various examples are given below in this section, showing which are the logics of interest, and also the kinds of combined logics that are required, as most of those logics that appear in the literature so far cover only a part of the features that multi-agent systems normally have (such as epistemic properties, temporal properties, motivational properties, coalitional power, etc.).

4.1. Model Checking

Model checking essentially takes a finite state model of a program and a temporal property and then checks that the temporal property is satisfied on *all* possible executions of the program (Clarke Jr. *et al.*, 1999). In the case of agent verification, we again need a finite state agent description (often a program) and a property to be checked on all possible executions of the agent. In using model checking to verify properties of agent-based systems,

we are often using properties based on some combination of modal and temporal logics, such as BDI logics, ATL, or temporal logics of knowledge. As in the standard case, problems of space requirements and of handling essentially infinite-state programs, are the key issues in agent verification. A further complication is that, once multi-agent systems are tackled, the complex interactions between such agents often mean the potential state space of the whole system is very large.

- **Model Checking Algorithms for BDI-like Logics**
  Benerecetti *et al.* (1998) presented a model-checking algorithm for a BDI logic (using CTL as underlying temporal logic) (see Rao and Georgeff (1998); Wooldridge (2000); Singh *et al.* (1999) for details on BDI-like logics). The approach formalises the BDI modalities using a variation of multi-language hierarchical logics (Giunchiglia and Serafini, 1994; Ghidini and Giunchiglia, 2001); the idea is to have a tree hierarchy of related theories, representing the nesting of modalities (that is, the mental model an agent has, possibly about another agent). In Benerecetti and Cimatti (2002), the approach was extended to consider symbolic model-checking (Clarke Jr. *et al.*, 1999) in particular (previous work had given algorithms for explicit-state model checking). One drawback of the approach is that it is not clear how the types of models they require are to be generated from actual systems, therefore their techniques might not be easily applicable to practical verification of multi-agent systems.
- **Model Checking AgentSpeak Programs**
  In a series of papers, Bordini *et al.* have developed model-checking techniques that apply directly to multi-agent programs written in AgentSpeak. The approach is to translate AgentSpeak multi-agent systems into either Promela (Bordini *et al.*, 2003a) or Java (Bordini *et al.*, 2004c) models, then using, respectively, SPIN (Holzmann, 2003) or JPF (Visser *et al.*, 2000) as model checkers. The property specification language is a much simplified BDI logic which is suitable for AgentSpeak programs. The tools that have been developed to automate the translation process were briefly described in Bordini *et al.* (2003b). Bordini *et al.* (2004b) introduced a slicing algorithm for AgentSpeak, and initial experimental results shown in that paper indicate that the technique can have a significant impact in reducing the state-space that needs to be considered in practical model-checking for AgentSpeak programs. The whole approach was summarised by Bordini *et al.* (2004a, 2006).
- **Bounded Model Checking for Epistemic Temporal Logics**
  Penczek and Lomuscio have defined bounded semantics of CTLK (Penczek and Lomuscio, 2003), a combined logic of knowledge and time. Based on that semantics, they define a bounded model-checking algorithm for that logic. The approach is to translate the (bounded) model of the system (given as an interpreted system) and a formula ϕ — the (negation of the) property to be verified — to sets of propositional formulæ in such as way that they can only be satisfied if, and only if, the system satisfies ϕ. This means that, in their approach, a model checking exercise can be carried out by existing SAT-solvers (which can at times, they argue, turn out to be rather efficient). More recently (van Riemsdijk *et al.*, 2004), they have defined similar translations to OBDDs, the approach used for symbolic model checking (Clarke Jr. *et al.*, 1999).
  Experimental results on the "attacking generals" problem, which has applications for agent coordination, were reported in (Lomuscio *et al.*, 2003). It remains to be seen if the performance of both SAT- and OBDD-based model-checking (and the associated translations) for practical verification of multi-agent systems will be satisfactory.
- **Model Checking for (variations of) ATL**
  The emphasis in ATL verification is usually on model checking techniques even though,

with complete ATL axiomatizations (Goranko and van Drimmelen, 2006; Walther *et al.*, 2006), there is potential for deductive approaches. Another logic with a game-theoretic basis that has been used for reasoning about teams, coalitions, etc., is Coalition Logic (Pauly, 2002), which can be seen as a generalisation of the modal base of Game Logic (Parikh, 1983; Pauly and Parikh, 2003) (itself an extension of Propositional Dynamic Logic (Harel *et al.*, 2000)). Practical model checking techniques for ATEL (van der Hoek and Wooldridge, 2003), a version of ATL with added epistemic modalities, were presented by van Otterloo *et al.* (2004) (by reduction to ATL); the paper includes an exercise in model checking the Russian cards scenario using Mocha (Alur *et al.*, 2000), an ATL model checker.

Further work based on ATL for agent verification has been carried out by Ryan and Schobbens (2002). They define a refinement relation between agents in a multi-agent system based on ATL. This allows the assume-guarantee paradigm (a well-known form of compositional reasoning (Clarke Jr. *et al.*, 1999)) to be used in combination with model checking, thus extending (in principle) the use of formal verification methods to much larger systems, despite the state-space explosion problem.

- **Deontic Interpreted Systems**
  Interpreted systems have arguably a more clear correspondence with states of distributed computational systems than Kripke structures, the preferred semantic formalism in philosophical logic. Lomuscio and Sergot (2002) have extended interpreted systems with deontic notions (Aqvist, 1984); in particular, this allows one to reason about correct and incorrect behaviour of individual agents and of the multi-agent system. Lomuscio and Sergot (2001) extended the framework so as to combine it with standard epistemic notions (Fagin *et al.*, 1995), thus allowing the expression of properties such as the knowledge that agents have, or require, when they function properly (with regards to a certain protocol). See also Lomuscio and Sergot (2003) for an extended paper on the subject. Regarding actual verification, Penczek and Lomuscio have applied bounded model-checking techniques to deontic interpreted systems (Wozna *et al.*, 2005).
- **Checking Communication Protocols within Electronic Institutions**
  Cliffe and Padget (2002) introduced a language that is suitable for modelling communication protocols in the context of Electronic Institutions (Esteva *et al.*, 2001). The language used to write the models of communication protocols provides a Prolog-like notation for defining a transition system, which is then translated into the input language of NuSMV (Cimatti *et al.*, 2002), a well-known symbolic model checker (model checking is then carried out by NuSMV). It is not yet clear how applicable the approach will be in general.
- **Model Checking Knowledge-Based Programs**
  van der Meyden (1998) examined the model checking problem for epistemic logic (with common knowledge formulæ) and systems of multiple agents (with perfect recall) situated in finite state environments. Model checking techniques for a logic combining knowledge and temporal modalities are described in van der Meyden and Wong (2003). This is the underlying theory for the MCK model checker (Gammie and van der Meyden, 2004), available at `http://www.cse.unsw.edu.au/~mck/`. MCK is a symbolic model checker implemented in Haskell, but relying on David Long's BDD package which is implemented in C (`http://www.cs.cmu.edu/~modelcheck/bdd.html`).
- **Model Checking Agent Dialogues**
  A language for describing agent dialogues, called MAP and based on CCS, was defined by Walton (2005). The language is particularly targeted at modelling dialogues in the context of Electronic Institutions (Esteva *et al.*, 2001) (the same application area as work by Cliffe & Padget described earlier). As in the work on model checking for AgentSpeak

described above, the approach here is to translate MAP protocols into Promela and use SPIN as model checker. Another use of the MAP language, for verifying agent communication in interaction with web services, is mentioned by Walton (2004).

- **Verifying Commitments**
  Both Singh and Wooldridge have argued about the difficulty of checking compliance of agents to communication protocols because of the mentalistic approach used in giving semantics to agent communication languages (Singh, 1998; Wooldridge, 1998). An alternative to the mentalistic approach is to use social commitments as the basis for speech-act semantics (Singh, 1998). Yolum and Singh (2002, 2004) present an approach to representing and reasoning about commitment protocols, based on the Event Calculus (Kowalski and Sergot, 1986). They use an event calculus planner to allow agents to choose flexibly a course of action that is compliant with given protocols. Another approach to commitment protocols is presented by Wan and Singh (2004). The idea there is to map Dooley graphs (often used for designing multi-agent interaction protocols) and commitment causal relations to the π-calculus (Milner *et al.*, 1992), which can then be used to check consistency of multi-agent systems.

4.2. Theorem Proving

In contrast to model checking, the deductive approach to agent verification uses a logical formula to describe all possible executions of the agent system, and then attempts to *prove* the required property from this logical formula. Again, in considering rational agents, the properties required are often captured using combinations of modal and temporal logics, therefore the deduction problem can be quite difficult. As usual with theorem proving, there are two main varieties: those that involve decidable problems, where the deduction is guaranteed to terminate, and much more complex problems, where human interaction is required.

- **Temporal Logics**
  In situations where agent descriptions are solely given using temporal logics (e.g., MetateM above), then temporal proof may be carried out. In general, first order temporal problems are very complex, but there are automatic methods for both propositional (Fisher *et al.*, 2001; Hustadt and Konev, 2003) and significant fragments of first-order (Degtyarev *et al.*, 2006; Konev *et al.*, 2005; Hustadt *et al.*, 2004) temporal logics. While individual modal, temporal, or description logics have been studied extensively in the past, much less is known about combinations of modal logics and, in particular, about practical proof methods for such combinations (Bennett *et al.*, 2002). However, both resolution- and tableau-based proof methods for combinations of propositional (linear or branching time) temporal logics with modal logics (Dixon *et al.*, 1998, 2002; Wooldridge *et al.*, 1998; Hustadt *et al.*, 2000) have been studied. For example, the combination of modal and temporal logics and translations into fragments of classical first-order logics have been used to reason about the KARO agent framework (Hustadt *et al.*, 2006), and about extensions of the basic MetateM framework (Fisher, 2005b).
- **CASL**
  Shapiro *et al.* (2002) introduced the Cognitive Agent Specification Language (CASL), which is based on the situation calculus, knowledge and goal operators (for representing an agent's mental states), and uses ConGolog (see above) for describing an agent's behaviour. They also defined CASLve, a verification environment for CASL that consists of representing a CASL specification within the framework of the PVS verification system (Owre *et al.*, 1992).

- **BDI and Logics of Concurrency** Schild (2000) provides a mapping from Rao and Georgeff's BDI logic (Rao and Georgeff, 1998) to μ-calculus (Kozen, 1983). The BDI model of Rao and Georgeff is based on a combination of the branching time logic CTL* (Emerson and Srinivasan, 2001) and modal operators for belief, desire, and intention. While it had been shown that CTL* is subsumed by the μ-calculus (Emerson, 1990), it was much less clear how to deal with the orthogonality that Rao and Georgeff's logic exhibits. Schild tackles this by collapsing the two dimensions of time and modalities onto a one dimensional structure. The translation into μ-calculus provides a full axiomatisation of Rao and Georgeff's BDI logic.
  Schild proves that, for CTL^n (containing formulae with at most n nested branching operators), the translation is decidable and succinct. Some axiom schemata (such as weak realism) can also be decided, while others (such as strong realism) might yield non-succinct translations, and are therefore less usable for automatic verification.

5. CONCLUDING REMARKS

5.1. Related Work

While this paper focusses on the use of computational logic within research on agents, there are many non-logical approaches to programming agents. Without claiming to cover all (important) frameworks, we briefly review some of the most prominent ones.

- **FIPA-OS** [`http://fipa-os.sourceforge.net`]
  FIPA-OS is a component-based toolkit enabling rapid development of FIPA compliant agents. FIPA-OS supports the majority of the FIPA experimental specifications and is being continuously improved as a managed *Open Source* community project, making it a suitable choice on which to base FIPA-compliant agent development activity. FIPA-OS was first released in August 1999 as the first publicly available implementation of FIPA technology.
- **PlanGent** [`http://www.toshiba.co.jp/plangent`]
  PlanGent is an intelligent agent system that performs tasks for human users. An agent in the PlanGent system has movement and planning capabilities; it can move around the network, determine its best course of action in various situations, and act autonomously.
- **JACK** [`http://www.agent-software.com`]
  JACK Intelligent Agents is an environment for building, running, and integrating commercial-grade multi-agent systems using a component-based approach. This approach is based heavily on the BDI architecture and appropriate Java classes for agent programming are provided. The JACK Agent Language is the programming language that extends Java with agent-oriented concepts, such as agents, capabilities, events, plans, agent knowledge bases (databases), and resource and concurrency management.
- **Jade** [`http://jade.tilab.com/`]
  JADE (Java Agent DEvelopment Framework) is a software framework implemented in Java. It supports FIPA-based communication and provides a set of graphical tools that aid the programmer during multi-agent systems development. In recent years, JADE has become one of the most used agent platforms within the research community, and its middleware is quickly becoming a *de facto* standard. An extension to JADE called Jadex extends the basic framework with a BDI-like architecture (`http://vsis-www.informatik.uni-hamburg.de/projects/jadex/`).
- **Aglets** [`http://aglets.sourceforge.net/`]

Aglets have been developed by the IBM Tokio Research Laboratory, and provide an applet-like approach to building agents using Java. It consists of a daemon application which serves as a basic framework, and mobile agents which can communicate using the framework. Through the use of Java security mechanisms, aglets support advanced security features.

- **Cougaar** [http://www.cougaar.org/]
  Cougaar is a Java-based framework for the creation of large-scale distributed applications. Originally conceived as part of a DARPA project, it has since become an open-source project. Special concern has been "survivability" in case of failures in parts of the system, as well as security. Based on a flexible component model, it implements various communication approaches, such as blackboard, and various protocols, such as CORBA, RMI, HTTP, and UDP.

5.2. Frequently Asked Questions

Q: . Why should we use logical approaches to agents?

A: Two reasons. The first is clarity – logical descriptions are unambiguous and often concise, thus providing a basis for detailed analysis and verification. The second is that, when using logical approaches, there is a vast number of different logics (or combinations) to choose from. This gives the system designer a very wide range of possible approaches. Choosing the best logic can give very intuitive and useful abstractions for programming complex systems.

Q: . Have the "right" logics been used?

A: Unfortunately, there is no "right" logic! In most cases, particular logical mechanisms have been developed for the application area being considered. Even the BDI approach, which is probably the most widespread in this area, has been modified, or even avoided, for certain applications.

Q: . Logics are is complex, so isn't using them likely to be inefficient?

A: Yes and no! Reasoning with expressive logics requires, in general, computationally expensive manipulation and so, typically, one has to trade efficiency for generality and expressiveness. However, in many cases specific logical fragments have been devised to shift the balance toward efficiency, while (possibly) reducing generality. Some examples: simplified BDI logics are adopted to translate AgentSpeak specifications into models that can be efficiently processed by off-the-shelf model checkers like SPIN or JPF; Constraint Logic Programming techniques have been used to implement forms of hypothetical reasoning in agents in an efficient way; and implementable agent logics have been proposed under the assumption of resource-boundedness.

Q: . What about "the market?"

A: It is true that logical approaches to multi-agent systems are not widely used in "the market" (yet). However, we witness a growing interest of the stakeholders in technologies such as autonomic computing, service oriented architectures, mobile robotics and e-trade. These are all domains very much related to the research being carried out in the MAS community. As concerns increase over the reliability and security of such systems and over the public's *trust* in these systems, so the use of logical approaches is likely to increase. Moreover, domains

such as Semantic Web services have a huge market potential and enjoy extensive input from (specifically) computational logic-based agent systems research (see below).

Q: . Is there a killer application?

A: Agent research is a broad domain! Many, equally good, understandings of agency have co-existed in the literature since its early days. It is hard to imagine, at this stage, that one killer application will suit to *all* (computational logic-based) agent research. However, There are areas where computational approaches to agency may become dominant (see below).

5.3. Future Directions

Below is a subjective and, to some extent, speculative view about the future of computational logics and agents.

Agent Programming. We have yet to see a truly successful agent programming language. Often agents are still programmed just using Java. Thus, there is an opportunity for a computational logic approach to programming agents to take a lead here.

Agent Analysis. Agent verification will surely increase. When unknown agents are sent to you (e.g., in the form of Java bytecodes) and your agents are expected to negotiate (sometimes sensitive) information with unknown agents, then verification will be increasingly required for questions of *security* and *trust*.

Agent Applications. We conclude the paper by suggesting four areas where computational logic approaches to agency may be particularly influential.

1. Ubiquitous Computing
   With the mobility, openness, and organisational aspects, both programming and analysing ubiquitous computing applications can be problematic. By providing a logical basis for programming both individual agents (through key aspects such as deliberation) and multi-agent systems (through groups and teams), computational logic can help the principled development of reliable ubiquitous computing applications. In addition, logical verification techniques can be used to analyse the behaviour of such potentially complex applications.
2. Autonomous Control
   Control systems are highly developed. However, the requirement for systems to act autonomously and *rationally* does not necessarily sit well with traditional control systems. As we move from systems where we need to know *what* the system does, to those where we need to know *why* it did something, we believe that logical approaches to the control and analysis of complex systems will come into their own.
3. Semantic Web
   While we have said very little about the Semantic Web in this report, web services are complimentary to agent technology. Consequently, many techniques for agent-based systems can be transferred to the Semantic Web. In terms of computational logic, this is even more evident when we see that description logics are at the core of Web ontologies. Thus, computational logic has a key role to play both in programming web-service description and discovery, and in analysing web-service collections.
4. Distribution
   This covers a wide variety of aspects, from negotiation, through resource allocation/balancing and stock markets, to e-commerce and financial applications. In many of these cases,

varieties of economic games are used. However, there is growing indication that computational logic might take a more central role in this area, possibly through combined logics/economics approaches.

References

Alberti, M., Gavanelli, M., Lamma, E., Mello, P., and Torroni, P. (2005a). The $\mathcal{S}$CIFF abductive proof-procedure. In *Advances in Artificial Intelligence (AI*IA)*, volume 3673 of *Lecture Notes in Artificial Intelligence*, pages 135–147, Heidelberg, Germany. Springer-Verlag.

Alberti, M., Chesani, F., Gavanelli, M., Lamma, E., Mello, P., and Torroni, P. (2005b). The SOCS computational logic approach for the specification and verification of agent societies. In *Global Computing: IST/FET International Workshop*, volume 3267 of *Lecture Notes in Artificial Intelligence*, pages 324–339. Springer-Verlag, Heidelberg, Germany.

Alberti, M., Chesani, F., Gavanelli, M., Lamma, E., Mello, P., and Torroni, P. (2006). Compliance verification of agent interaction: a logic-based tool. *Applied Artificial Intelligence*, **20**(2-4), 133–157.

Alchourron, C. E., Gardenfors, P., and Makinson, D. (1985). On the logic of theory change: Partial meet functions for contraction and revision. *Journal of Symbolic Logic*, **50**, 510–530.

Alechina, N., Bordini, R. H., Hübner, J. F., Jago, M., and Logan, B. (2006a). Automating belief revision for AgentSpeak. In *Declarative Agent Languages and Technologies IV (DALT)*, volume 4327 of *Lecture Notes in Artificial Intelligence*, pages 61–77, Heidelberg, Germany. Springer-Verlag.

Alechina, N., Jago, M., and Logan, B. (2006b). Resource-bounded belief revision and contraction. In *Declarative Agent Languages and Technologies III (DALT)*, volume 3904 of *Lecture Notes in Artificial Intelligence*, pages 141–154, Heidelberg, Germany. Springer-Verlag.

Alferes, J. J., Leite, J. A., Pereira, L. M., Przymusinska, H., and Przymusinski, T. C. (1998). Dynamic logic programming. In *Proceedings of the 6th International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 98–111, San Fransisco, CA, USA. Morgan Kaufmann.

Alferes, J. J., Pereira, L. M., Przymusinska, H., and Przymusinski, T. C. (2002). Lups—a language for updating logic programs. *Artificial Intelligence*, **138**(1-2), 87–116.

Alur, R., de Alfaro, L., Henzinger, T. A., Krishnan, S. C., Mang, F. Y. C., Qadeer, S., Rajamani, S. K., and Taşiran, S. (2000). MOCHA user manual. Report, University of Berkeley, California, CA, USA.

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

Alur, R., Henzinger, T. A., and Kupferman, O. (2004). Alternating-time temporal logic. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 100–109, Washington, DC, USA. IEEE Computer Society.

Aqvist, L. (1984). Deontic logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic: Volume II: Extensions of Classical Logic*, pages 605–714. D. Reidel, Dordrecht, The Netherlands.

Arisha, K. A., Ozcan, F., Ross, R., Subrahmanian, V. S., Eiter, T., and Kraus, S. (1999). IMPACT: a Platform for Collaborating Agents. *IEEE Intelligent Systems*, **14**(2), 64–72.

Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., and Patel-Schneider, P. F., editors (2003). *The Description Logic Handbook: Theory, Implementation, and Applications*, New York, NY, USA. Cambridge University Press.

Baker, A. B. (1991). Nonmonotonic reasoning in the framework of situation calculus. *Artificial Intelligence*, **49**(1-3), 5–23.

Baral, C. (2003). *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press, New York, NY, USA.

Barringer, H., Fisher, M., Gabbay, D. M., Gough, G., and Owens, R. (1995). METATEM: An introduction. *Formal Aspects of Computing*, **7**(5), 533–549.

Barringer, H., Fisher, M., Gabbay, D., Owens, R., and Reynolds, M., editors (1996). *The Imperative Future: Principles of Executable Temporal Logics*. John Wiley & Sons, Inc., New York, NY, USA.

Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D. L., Patel-Schneider, P. F., and Stein, L. A., editors (2004). *OWL Web Ontology Language Reference*, W3C Recommendation. World Wide Web Consortium. Latest version available at `http://www.w3.org/TR/owl-ref/`.

Benerecetti, M. and Cimatti, A. (2002). Symbolic model checking for multi-agent systems. In *Proceedings of the Workshop on Model Checking and Artificial Intelligence (MoChArt)*, Lyon, France.

Benerecetti, M., Giunchiglia, F., and Serafini, L. (1998). Model checking multiagent systems. *Journal of Logic and Computation*, **8**(3), 401–423.

Bennett, B., Dixon, C., Fisher, M., Franconi, E., Horrocks, I., and de Rijke, M. (2002). Combinations of modal logics. *Artificial Intelligence Review*, **17**(1), 1–20.

Bordini, R., Dastani, M., Dix, J., and El Fallah Seghrouchni, A., editors (2005a). *Multi-Agent Programming: Languages, Platforms and Applications*, volume 15 of *Multiagent Systems, Artificial Societies, and Simulated Organizations*, Heidelberg, Germany. Springer-Verlag.

Bordini, R. H. and Hübner, J. F. (2006). BDI agent programming in AgentSpeak using **jason** (tutorial paper). In *Computational Logic in Multi-Agent Systems VI (CLIMA)*, volume 3900 of *Lecture Notes in Artificial Intelligence*, pages 143–164, Heidelberg, Germany. Springer-Verlag.

Bordini, R. H., Fisher, M., Pardavila, C., and Wooldridge, M. (2003a). Model checking AgentSpeak. In *Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 409–416, New York, NY, USA. ACM Press.

Bordini, R. H., Visser, W., Fisher, M., Pardavila, C., and Wooldridge, M. (2003b). Model checking multi-agent programs with CASP. In *Proceedings of the 15th International Conference on Computer-Aided Verification (CAV)*, volume 2725 of *Lecture Notes in Computer Science*, pages 110–113, Heidelberg, Germany. Springer-Verlag. Tool description.

Bordini, R. H., Fisher, M., Visser, W., and Wooldridge, M. (2004a). Model checking rational agents. *IEEE Intelligent Systems*, **19**(5), 46–52.

Bordini, R. H., Fisher, M., Visser, W., and Wooldridge, M. (2004b). State-space reduction techniques in agent verification. In *Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 896–903, New York, NY, USA. ACM Press.

Bordini, R. H., Fisher, M., Visser, W., and Wooldridge, M. (2004c). Verifiable multi-agent programs. In *Programming Multi-Agent Systems (PROMAS)*, volume 3067 of *Lecture Notes in Artificial Intelligence*, pages 72–89, Heidelberg, Germany. Springer-Verlag.

Bordini, R. H., Hübner, J. F., and Vieira, R. (2005b). ***Jason*** and the golden fleece of agent-oriented programming. In *Multi-Agent Programming: Languages, Platforms and Applications*, number 15 in Multiagent Systems, Artificial Societies, and Simulated Organizations, pages 3–37, Heidelberg, Germany. Springer-Verlag.

Bordini, R. H., Fisher, M., Visser, W., and Wooldridge, M. (2006). Verifying multi-agent programs by model checking. *Journal of Autonomous Agents and Multi-Agent Systems*, **12**(2), 239–256.

Bracciali, A., Endriss, U., Demetriou, N., Kakas, A. C., Lu, W., and Stathis, K. (2006). Crafting the mind of PROSOCS agents. *Applied Artificial Intelligence*, **20**(2-4), 105–131.

Bratman, M. E. (1999). *Intentions, Plans, and Practical Reason*. The David Hume Series: Philosophy and Cognitive Science Reissues. CSLI Publications, Stanford, CA, USA.

Broersen, J., Dastani, M., Huang, Z., Hulstijn, J., and van der Torre, L. (2001). The BOID architecture: conficts between beliefs, obligations, intentions and desires. In *Proceedings of 5th International Conference on Autonomous Agents*, pages 9–16. ACM Press, New York, NY, USA.

Broersen, J., Dastani, M., Huang, Z., Hulstijn, J., and van der Torre, L. (2002). Goal generation in the BOID architecture. *Cognitive Science Quarterly*, **2**, 428–447.

Chesani, F., Gavanelli, M., Alberti, M., Lamma, E., Mello, P., and Torroni, P. (2006). Specification and verification of agent interaction using abductive reasoning (tutorial paper). In *Computational Logic in Multi-Agent Systems VI (CLIMA)*, volume 3900 of *Lecture Notes in Artificial Intelligence*, pages 243–264, Heidelberg, Germany. Springer-Verlag.

Ciampolini, A., Lamma, E., Mello, P., and Torroni, P. (2002). LAILA: A language for coordinating abductive reasoning among logic agents. *Computer Languages*, **27**(4), 137–161.

Ciampolini, A., Lamma, E., Mello, P., Toni, F., and Torroni, P. (2003). Co-operation and competition in ALIAS: a logic framework for agents that negotiate. *Annals of Mathematics and Artificial Intelligence*, **37**(1-2), 65–91.

Cimatti, A., Clarke, E. M., Giunchiglia, E., Giunchiglia, F., Pistore, M., Roveri, M., Sebastiani, R., and Tacchella, A. (2002). NuSMV2: An opensource tool for symbolic model checking. In *Proceedings of the 14th International Conference on Computer Aided Verification (CAV)*, volume 2404 of *Lecture Notes in Computer Science*, pages 359–364, Heidelberg, Germany. Springer-Verlag.

Clark, K. L., Robinson, P., and Hagen, R. (2001). Multi-threading and message communication in Qu-Prolog. *Theory and Practice of Logic Programming*, **1**(3), 283–301.

Clark, K. L., Robinson, P. J., and Zappacosta-Amboldi, S. (2006). Multi-threaded communicating agents in qu-prolog (tutorial paper). In *Computational Logic in Multi-Agent Systems VI (CLIMA)*, volume 3900 of *Lecture Notes in Artificial Intelligence*, pages 186–205, Heidelberg, Germany. Springer-Verlag.

Clarke Jr., E. M., Grumberg, O., and Peled, D. A. (1999). *Model Checking*. The MIT Press, Cambridge, MA, USA.

Cliffe, O. and Padget, J. (2002). A framework for checking agent transactions within institutions. In *Proceedings of the Workshop on Model Checking and Artificial Intelligence (MoChArt)*, Lyon, France.

Costantini, S. and Tocchio, A. (2006). A logic programming language for multi-agent systems. In *Logics in Artificial Intelligence, European Conference (JELIA)*, volume 2424 of *Lecture Notes in Artificial Intelligence*, pages 1–13, Heidelberg, Germany. Springer-Verlag.

Dastani, M., de Boer, F., Dignum, F., and Meyer, J.-J. C. (2003). Programming agent deliberation: An approach illustrated using the 3APL language. In *Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 97–104, New York, NY, USA. ACM Press.

De Giacomo, G., Lespérance, Y., and Levesque, H. J. (2000). Congolog, a concurrent programming language based on the situation calculus. *Artificial Intelligence*, **121**(1-2), 109–169.

Degtyarev, A., Fisher, M., and Konev, B. (2006). Monodic temporal resolution. *ACM Transactions on Computational Logic*, **7**(1), 108–150.

Dell'Acqua, P., Nilsson, U., , and Pereira, L. M. (2002). A logic based asynchronous multi-agent system. *Electronic Notes in Theoretical Computer Science*, **70**.

Dignum, F. (1999). Autonomous agents and norms. *Artificial Intelligence and Law*, **7**, 69–79.

Dimopoulos, Y. and Kakas, A. C. (1995). Logic programming without negation as failure. In *Proceedings of the International Symposium on Logic Programming*, pages 369–384, Cambridge, MA, USA. The MIT Press.

d'Inverno, M. and Luck, M. (1998). Engineering AgentSpeak(L): A formal computational model. *Journal of Logic and Computation*, **8**(3), 233–260.

d'Inverno, M., Kinny, D., Luck, M., and Wooldridge, M. (1998). A formal specification of dMARS. In *Intelligent Agents IV*, volume 1365 of *Lecture Notes in Artificial Intelligence*, pages 155–176, Heidelberg, Germany. Springer-Verlag.

Dixon, C., Fisher, M., and Wooldridge, M. (1998). Resolution for Temporal Logics of Knowledge. *Journal of Logic and Computation*, **8**(3), 345–372.

Dixon, C., Fisher, M., and Bolotov, A. (2002). Clausal resolution in a logic of rational agency. *Artificial Intelligence*, **139**(1), 47–89.

Eiter, T. and Subrahmanian, V. S. (1998). Deontic action programs. In *Workshop on Foundations of Models and Languages for Data and Objects (FMLDO)*, pages 37–54, Ostfriesland, Germany.

Eiter, T. and Subrahmanian, V. S. (1999). Heterogeneous active agents, II: Algorithms and complexity. *Artificial Intelligence*, **108**(1-2), 257–307.

Eiter, T., Subrahmanian, V. S., and Pick, G. (1999). Heterogeneous active agents, I: Semantics. *Artificial Intelligence*, **108**(1-2), 179–255.

Eiter, T., Subrahmanian, V. S., and Rodgers, T. J. (2000). Heterogeneous active agents, III: Polynomially implementable agents. *Artificial Intelligence*, **117**(1), 107–167.

Eiter, T., Mascardi, V., and Subrahmanian, V. S. (2002). Error-tolerant agents. In *Computational Logic. Logic Programming and Beyond*, volume 2407 of *Lecture Notes in Artificial Intelligence*, pages 586–625, Heidelberg, Germany. Springer-Verlag.

Emerson, E. A. (1990). *Handbook of Theoretical Computer Science*, volume B, chapter Temporal and Modal Logic, pages 997–1072. Elsevier Science, Amsterdam, The Netherlands.

Emerson, E. A. and Srinivasan, J. (2001). Branching time temporal logic. In *Linear Time, Branching Time, and Partial Order in Logics and Models for Concurrency*, volume 1989 of *Lecture Notes in Computer Science*, pages 123–172, Heidelberg, Germany. Springer-Verlag.

Endriss, U., Mancarella, P., Sadri, F., Terreni, G., and Toni, F. (2004). The CIFF proof procedure for abductive logic programming with constraints. In *Logics in Artificial Intelligence, European Conference (JELIA)*, volume 3229 of *Lecture Notes in Artificial Intelligence*, pages 31–43, Heidelberg, Germany. Springer-Verlag.

Esteva, M., Rodríguez, J. A., Sierra, C., Garcia, P., and Arcos, J. L. (2001). On the formal specification of electronic institutions. In *Agent-mediated Electronic Commerce (The European AgentLink Perspective)*, volume 1991 of *Lecture Notes in Artificial Intelligence*, pages 126–147, Heidelberg, Germany. Springer-Verlag.

Fagin, R., Halpern, J. Y., Moses, Y., and Vardi, M. (1995). *Reasoning about Knowledge*, volume Cambridge, MA, USA. The MIT Press.

Fisher, M. (1994). A survey of concurrent MetateM — the language and its applications. In *Proceedings of the 1st International Conference on Temporal Logic (ICTL)*, volume 827 of *Lecture Notes in Artificial Intelligence*, pages 480–505, Heidelberg, Germany. Springer-Verlag.

Fisher, M. (1995). Representing and executing agent-based systems. In *Intelligent Agents*, volume 890 of *Lecture Notes in Artificial Intelligence*, pages 307–323, Heidelberg, Germany. Springer-Verlag.

Fisher, M. (1996). A temporal semantics for concurrent MetateM. *Journal of Symbolic Computation*, **22**(5/6), 627–648.

Fisher, M. (1997a). Implementing BDI-like systems by direct execution. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI)*, volume 1, pages 316–321, San Fransisco, CA, USA. Morgan Kaufmann.

Fisher, M. (1997b). A normal form for temporal logic and its application in theorem-proving and execution. *Journal of Logic and Computation*, **7**(4), 429–456.

Fisher, M. (2005a). MetateM: The story so far. In *Programming Multi-Agent Systems II (PROMAS)*, volume 3862 of *Lecture Notes in Artificial Intelligence*, pages 3–22, Heidelberg, Germany. Springer-Verlag.

Fisher, M. (2005b). Temporal development methods for agent-based systems. *Journal of Autonomous Agents and Multi-Agent Systems*, **10**(1), 41–66.

Fisher, M. and Ghidini, C. (1999). Programming resource-bounded deliberative agents. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 200–205, San Fransisco, CA, USA. Morgan Kaufmann.

Fisher, M. and Ghidini, C. (2002). The ABC of rational agent modelling. In *Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 849–856, New York, NY, USA. ACM Press.

Fisher, M. and Wooldridge, M. (1997). On the formal specification and verification of multi-agent systems. *International Journal of Cooperative Information Systems*, **6**(1), 37–65.

Fisher, M., Dixon, C., and Peim, M. (2001). Clausal temporal resolution. *ACM Transactions on Computational Logic*, **2**(1), 12–56.

Fisher, M., Ghidini, C., and Hirsch, B. (2003). Organising logic-based agents. In *Proceedings of the 2nd NASA/IEEE Goddard Workshop on Formal Approaches to Agent Based Systems (FAABS)*, volume 2699 of *Lecture Notes in Artificial Intelligence*, pages 15–27, Heidelberg, Germany. Springer-Verlag.

Fisher, M., Ghidini, C., and Hirsch, B. (2004). Programming groups of rational agents. In *Computational Logic in Multi-Agent Systems IV (CLIMA)*, volume 3259 of *Lecture Notes in Artificial Intelligence*, pages 16–33, Heidelberg, Germany. Springer-Verlag.

Fung, T. H. and Kowalski, R. A. (1997). The IFF proof procedure for abductive logic programming. *Journal of Logic Programming*, **33**(2), 151–165.

Gammie, P. and van der Meyden, R. (2004). MCK: Model checking the logic of knowledge. In *Proceedings of the 16th International Conference on Computer-Aided Verification (CAV)*, volume 3114 of *Lecture Notes in Computer Science*, pages 479–483, Heidelberg, Germany. Springer-Verlag.

Gelfond, M. and Lifschitz, V. (1988). The stable model semantics for logic programming. In *Proceedings of the Joint International Conference and Symposium on Logic Programming*, pages 1070–1080, Cambridge, MA, USA. The MIT Press.

Georgeff, M. P. and Lansky, A. L. (1987). Reactive reasoning and planning. In A. Press, editor, *Proceedings of the 6th National Conference on Artificial Intelligence (AAAI)*, pages 677–682, Menlo Park, CA, USA.

Ghidini, C. and Giunchiglia, F. (2001). Local models semantics, or contextual reasoning=locality+compatibility. *Artificial Intelligence*, **127**(2), 221–259.

Giunchiglia, F. and Serafini, L. (1994). Multilanguage hierarchical logics or: How we can do without modal logics. *Artificial Intelligence*, **65**(1), 29–70.

Goldblatt, R. (1992). *Logics of Time and Computation*, volume 7 of *CSLI Lecture Notes*. CSLI Publications, Stanford, CA, USA, 2nd edition.

Goranko, V. and van Drimmelen, G. (2006). Complete axiomatisation and decidability of alternating-time temporal logic. *Theoretical Computer Science*, **253**(1-3), 93–117.

Harel, D. (1984). Deontic logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic: Volume II: Extensions of Classical Logic*, pages 497–604. D. Reidel, Dordrecht, The Netherlands.

Harel, D., Kozen, D., and Tiuryn, J. (2000). *Dynamic Logic*. The MIT Press, Cambridge, MA, USA.

Hartshorne, C. and Weiss, P., editors (1965). *Collected Papers of Charles Sanders Peirce, 1931–1958*, volume 2, Cambridge, MA, USA. Harvard University Press.

Hindriks, K., de Boer, F., van der Hoek, W., and Meyer, J.-J. (1998). Formal semantics for an abstract agent programming language. In *Intelligent Agents IV*, volume 1365 of *Lecture Notes in Artificial Intelligence*, pages 215–229, Heidelberg, Germany. Springer-Verlag.

Hindriks, K., de Boer, F., van der Hoek, W., and Meyer, J.-J. (2001). Agent programming with declarative goals. In *Intelligent Agents VII*, volume 1986 of *Lecture Notes in Artificial Intelligence*, pages 228–243, Heidelberg, Germany. Springer-Verlag.

Hirsch, B., Fisher, M., Ghidini, C., and Busetta, P. (2005). Organising software in active environments. In *Computational Logic in Multi-Agent Systems V (CLIMA)*, volume 3487 of *Lecture Notes in Artificial Intelligence*, pages 265–280, Heidelberg, Germany. Springer-Verlag.

Holzmann, G. J. (2003). *The Spin Model Checker: Primer and Reference Manual*. Addison-Wesley Professional, Boston, MA, USA.

Horrocks, I., Patel-Schneider, P. F., and van Harmelen, F. (2002). Reviewing the design of DAML+OIL: An ontology language for the semantic web. In A. Press, editor, *Proceedings of the 18th National Conference on Artificial Intelligence (AAAI)*, pages 792–797, Menlo Park, CA, USA.

Hustadt, U. and Konev, B. (2003). TRP++ 2.0: A temporal resolution prover. In *Proceedings of the 19th International Conference on Automated Deduction (CADE)*, volume 2741 of *Lecture Notes in Artificial Intelligence*, pages 274–278, Heidelberg, Germany. Springer-Verlag.

Hustadt, U., Dixon, C., Schmidt, R., and Fisher, M. (2000). Normal forms and proofs in combined modal and temporal logics. In *Proceedings of the 3rd International Workshop on Frontiers of Combining Systems (FroCoS)*, volume 1794 of *Lecture Notes in Artificial Intelligence*, pages 73–87, Heidelberg, Germany. Springer-Verlag.

Hustadt, U., Dixon, C., Schmidt, R. A., Fisher, M., Meyer, J.-J., and van der Hoek, W. (2001). Reasoning about agents in the KARO framework. In *Proceedings of the 8th International Symposium on Temporal Representation and Reasoning (TIME)*, pages 206–213, Washington, DC, USA. IEEE Press.

Hustadt, U., Konev, B., Riazanov, A., and Voronkov, A. (2004). TeMP: A temporal monodic prover. In *Proceedings of the 2nd International Joint Conference on Automated Reasoning (IJCAR)*, volume 3097 of *Lecture Notes in Artificial Intelligence*, pages 326–330, Heidelberg, Germany. Springer-Verlag.

Hustadt, U., Dixon, C., Schmidt, R. A., Fisher, M., Meyer, J.-J., and van der Hoek, W. (2006). Verification within the KARO agent theory. In C. Rouff, M. Hinchey, J. Rash,

W. Truszkowski, and D. Gordon-Spears, editors, *Agent Technology from a Formal Perspective*, NASA Monographs in Systems and Software Engineering, pages 193–226. Springer-Verlag, Heidelberg, Germany.

Jonker, C. M., Treur, J., and Wijngaards, W. C. A. (2001). Dynamics within an organisation: Temporal specification, simulation and evaluation. In U. Press, editor, *Proceedings of the 3rd International Conference on Cognitive Science (ICCS)*, pages 546–550, Beijing, China.

Kakas, A. C. and Moraïtis, P. (2002). Argumentative agent deliberation, roles and context. In *Computational Logic in Multi-Agent Systems III (CLIMA)*, volume 70 of *Electronic Notes on Theoretical Computer Science*, Amsterdam, The Netherlands. Elsevier Science Publishers.

Kakas, A. C. and Moraitis, P. (2003). Argumentation based decision making for autonomous agents. In *Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 883–890, New York, NY, USA. ACM Press.

Kakas, A. C., Kowalski, R. A., and Toni, F. (1993). Abductive Logic Programming. *Journal of Logic and Computation*, **2**(6), 719–770.

Kakas, A. C., Torroni, P., and Demetriou, N. (2004a). Agent planning, negotiation, and control of operation. In I. Press, editor, *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI)*, pages 28–32, Amsterdam, The Netherlands.

Kakas, A. C., Mancarella, P., Sadri, F., Stathis, K., and Toni, F. (2004b). Reactive reasoning and planning. In I. Press, editor, *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI)*, pages 33–37, Amsterdam, The Netherlands.

Konev, B., Degtyarev, A., Dixon, C., Fisher, M., and Hustadt, U. (2005). Mechanizing first-order temporal resolution. *Information and Computation*, **199**(1-2), 55–86.

Kowalski, R. A. and Sadri, F. (1999). From logic programming towards multi-agent systems. *Annals of Mathematics and Artificial Intelligence*, **25**(3-4), 391–419.

Kowalski, R. A. and Sergot, M. J. (1986). A logic-based calculus of events. *New Generation Computing*, **4**(1), 67–95.

Kozen, D. (1983). Results on the propositional μ-calculus. *Theoretical Computer Science*, **27**, 333—354.

Kripke, S. (1963). Semantical considerations on modal logic. *Acta Philosophica Fennica*, **16**, 83–94.

Labrou, Y. and Finin, T. (1994). A semantics approach for KQML—a general purpose communication language for software agents. In *Proceedings of the 3rd International Conference on Information and Knowledge Management (CIKM)*, pages 447–455, New York, NY, USA. ACM Press.

Lakemeyer, G. (1999). On sensing and off-line interpreting in GOLOG. In *Logical Foundations for Cognitive Agents*, pages 173–189, Heidelberg, Germany. Springer-Verlag.

Leite, J. A., Alferes, J. J., and Pereira, L. M. (2001). Multi-dimensional dynamic knowledge representation. In *Proceedings of the 6th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR)*, volume 2173 of *Lecture Notes in Artificial Intelligence*, pages 365–378, Heidelberg, Germany. Springer-Verlag.

Levesque, H., Reiter, R., Lespérance, Y., Lin, F., and Scherl, R. (1997). GOLOG: A logic programming language for dynamic domains. *Journal of Logic Programming*, **31**(1-3), 59–84.

Levesque, H., Pirri, F., and Reiter, R. (1998a). Foundations for the situation calculus. *Electronic Transactions on Artificial Intelligence*, **2**, 159–178.

Levesque, H., Pirri, F., and Reiter, R. (1998b). Foundations for the situation calculus. *Linköping Electronic Articles in Computer and Information Science*, **3**(18).

Lin, F. (1995). Embracing causality in specifying the indirect effects of actions. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1985–1991, San Fransisco, CA, USA. Morgan Kaufmann.

Lloyd, J. W. (1987). *Foundations of Logic Programming*. Springer-Verlag, Heidelberg, Germany, 2nd edition.

Lomuscio, A. and Sergot, M. (2001). Extending interpreted systems with some deontic concepts. In *Proceedings of the 8th Conference on Theoretical Aspects of Rationality and Knowledge (TARK)*, volume 2699, pages 207–218, San Fransisco, CA, USA. Morgan Kaufmann.

Lomuscio, A. and Sergot, M. (2002). On multi-agent systems specification via deontic logic. In *Intelligent Agents VIII*, volume 2333 of *Lecture Notes in Artificial Intelligence*, pages 86–99, Heidelberg, Germany. Springer-Verlag.

Lomuscio, A. and Sergot, M. (2003). Deontic interpreted systems. *Studia Logica*, **75**(1), 63–92.

Lomuscio, A., Lasica, T., and Penczek, W. (2003). Bounded model checking for interpreted systems: Preliminary experimental results. In *Proceedings of the 2nd NASA/IEEE Goddard Workshop on Formal Approaches to Agent Based Systems (FAABS)*, volume 2699 of *Lecture Notes in Artificial Intelligence*, pages 115–125, Heidelberg, Germany. Springer-Verlag.

Mahony, B. and Dong, J. (2000). Timed communicating Object Z. *IEEE Transactions on Software Engineering*, **26**(2), 150–177.

McCabe, F. G. and Clark, K. L. (1995). `April` — agent process interaction language. In *Intelligent Agents*, volume 890 of *Lecture Notes in Artificial Intelligence*, pages 324–340, Heidelberg, Germany. Springer-Verlag.

McCabe, F. G. and Clark, K. L. (2004). Go! - a multi-paradigm programming language for implementing multi-threaded agents. *Annals of Mathematics and Artificial Intelligence*, **41**(2-4), 171–206.

McCarthy, J. (1986). Applications of circumscription to formalizing common-sense knowledge. *Artificial Intelligence*, **28**(1), 86–116.

McCarthy, J. and Hayes, P. J. (1969). Some philosophical problems from the standpoint of artificial intelligence. *Machine Intelligence*, **4**, 463–502.

McGuinness, D. L. and van Harmelen, K., editors (2004). *OWL Web Ontology Language — Overview*, W3C Recommendation. World Wide Web Consortium. Latest version available at `http://www.w3.org/TR/owl-features/`.

McIlraith, S. A. and Son, T. C. (2001). Adapting golog for programming in the semantic web. In *Proceedings of the 5th International Symposium on Logical Formalizations of Commonsense Reasoning*, pages 195–202.

McIlraith, S. A. and Son, T. C. (2002). Adapting golog for composition of semantic web services. In *Proceedings of the 8th International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 482–496, San Fransisco, CA, USA. Morgan Kaufmann.

Merz, S., Zappe, J., and Wirsing, M. (2003). A spatio-temporal logic for the specification and refinement of mobile systems. In *Fundamental Approaches to Software Engineering (FASE)*, volume 2621 of *Lecture Notes in Computer Science*, pages 87–101, Heidelberg, Germany. Springer-Verlag.

Meyer, J.-J., van der Hoek, W., and van Linder, B. (1999). A logical approach to the dynamics of commitments. *Artificial Intelligence*, **113**(1-2), 1–40.

Meyer, J.-J. C., de Boer, F., van Eijk, R., Hindriks, K., and van der Hoek, W. (2001). On programming KARO agents. *Logic Journal of the IGPL*, **9**(22), 261–272.

Milner, R., Parrow, J., and Walker, D. (1992). A calculus for mobile processes (parts I and II). *Information and Computation*, **100**(1), 1–40 and 41–77.

Moore, R. C. (1984). A formal theory of knowledge and action. Technical Note 320, SRI International Artificial Intelligence Center, Menlo Park, CA, USA.

Moore, R. C. (1985). Semantical considerations on nonmonotonic logic. *Artificial Intelligence*, **25**(1), 75–94.

Morgenstern, L. (1999). Nonmonotonic logics. In *The MIT Encyclopedia of the Cognitive Sciences*, Cambridge, MA, USA. The MIT Press.

Myers, K. L. (1993). *User's Guide for the Procedural Reasoning System*. SRI International Artificial Intelligence Center, Menlo Park, CA, USA.

Owre, S., Rushby, J. M., and Shankar, N. (1992). PVS: A prototype verification system. In *Proceedings of the 11th International Conference on Automated Deduction (CADE)*, volume 607 of *Lecture Notes in Artificial Intelligence*, pages 748–752, Heidelberg, Germany. Springer-Verlag.

Parikh, R. (1983). Propositional game logic. In *Proceedings of the 24th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 195–200, Washington, DC, USA. IEEE Computer Society.

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

Pauly, M. and Parikh, R. (2003). Game logic — an overview. *Studia Logica*, **75**(2), 165–182.

Penczek, W. and Lomuscio, A. (2003). Verifying epistemic properties of multi-agent systems via model checking. *Fundamenta Informaticae*, **55**(2), 167–185.

Pinto, J. and Reiter, R. (1993). Temporal reasoning in logic programming: A case for the situation calculus. In *Proceedings of the International Conference on Logic Programming (ICLP)*, pages 203–221, Cambridge, MA, USA. The MIT Press.

Pirri, F. and Reiter, R. (1999). Some contributions to the metatheory of the situation calculus. *Journal of the ACM*, **46**(3), 325–361.

Rao, A. S. (1996). AgentSpeak(L): BDI agents speak out in a logical computable language. In *Agents Breaking Away. Proceedings of the 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, (MAAMAW)*, volume 1038 of *Lecture Notes in Artificial Intelligence*, pages 748–752, Heidelberg, Germany. Springer-Verlag.

Rao, A. S. and Georgeff, M. P. (1991). Modeling rational agents within a BDI-architecture. In *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 473–484, San Fransisco, CA, USA. Morgan Kaufmann.

Rao, A. S. and Georgeff, M. P. (1995). BDI agents: From theory to practice. In *Proceedings of the 1st International Conference on Multi-Agent Systems (ICMAS)*, pages 312–319, Washington, DC, USA. IEEE Press.

Rao, A. S. and Georgeff, M. P. (1998). Decision procedures for BDI logics. *Journal of Logic and Computation*, **8**(3), 293–343.

Regayeg, A., Kacem, A. H., and Jmaiel, M. (2004). Specification and verification of multi-agent applications using Temporal Z. In *Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT)*, pages 260–266, Washington, DC, USA. IEEE Press.

Reiter, R. (1980). A logic for default reasoning. *Artificial Intelligence*, **13**(1-2), 81–132.

Ryan, M. and Schobbens, P.-Y. (2002). Agents and roles: Refinement in alternating-time temporal logic. In *Intelligent Agents VIII*, volume 2333 of *Lecture Notes in Artificial Intelligence*, pages 100–114, Heidelberg, Germany. Springer-Verlag.

Sadri, F. (2006). Using the KGP model of agency to design applications (tutorial paper). In *Computational Logic in Multi-Agent Systems VI (CLIMA)*, volume 3900 of *Lecture Notes in Artificial Intelligence*, pages 165–185, Heidelberg, Germany. Springer-Verlag.

Sadri, F. and Toni, F. (2006). A formal analysis of KGP agents. In *Logics in Artificial Intelligence, European Conference (JELIA)*, volume 4160 of *Lecture Notes in Artificial Intelligence*, pages 413–425, Heidelberg, Germany. Springer-Verlag.

Sadri, F., Toni, F., and Torroni, P. (2001). Logic agents, dialogues and negotiation: An abductive approach. In *Proceedings of the 3rd Annual AISB Convention*, York, UK. The Society for the Study of Artificial Intelligence and Simulation of Behaviour.

Sadri, F., Toni, F., and Torroni, P. (2002). Dialogues for negotiation: Agent varieties and dialogue sequences. In *Intelligent Agents VIII*, volume 2333 of *Lecture Notes in Artificial Intelligence*, pages 405–421, Heidelberg, Germany. Springer-Verlag.

Sadri, F., Toni, F., and Torroni, P. (2003). Minimally intrusive negotiating agents for resource sharing. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 796–804, San Fransisco, CA, USA. Morgan Kaufmann.

Satoh, K. (2002). Speculative computation and abduction for an autonomous agent. In *Proceedings of the 9th International Workshop on Non-Monotonic Reasoning (NMR)*, pages 191–199.

Satoh, K. and Yamamoto, K. (2002). Speculative computation with multi-agent belief revision. In *Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 897–904, New York, NY, USA. ACM Press.

Satoh, K., Inoue, K., Iwanuma, K., and Sakama, C. (2000). Speculative computation by abduction under incomplete communication environments. In *Proceedings of the 4th International Conference on Multi-Agent Systems (ICMAS)*, pages 263–270, Washington, DC, USA. IEEE Press.

Schild, K. (2000). On the relationship between BDI logics and standard logics of concurrency. *Journal of Autonomous Agents and Multi-Agent Systems*, **3**(3), 259–283.

Shapiro, S., Lespérance, Y., and Levesque, H. (2002). The cognitive agents specification language and verification environment for multiagent systems. In *Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 19–26, New York, NY, USA. ACM Press.

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

Singh, M. P. (1996). Semantical considerations on some primitives for agent specification. In *Intelligent Agents II*, volume 1037 of *Lecture Notes in Artificial Intelligence*, pages 49–64, Heidelberg, Germany. Springer-Verlag.

Singh, M. P. (1998). Agent communication languages: Rethinking the principles. *IEEE Computer*, **31**(12), 40–47.

Singh, M. P., Rao, A. S., and Georgeff, M. P. (1999). *Multiagent Systems—A Modern Approach to Distributed Artificial Intelligence*, chapter Formal Methods in DAI: Logic-Based Representation and Reasoning, pages 331–376. The MIT Press, Cambridge, MA, USA.

Smith, G. (2000). *The Object-Z Specification Language*, volume Dodrecht, The Netherlands of *Advances in Formal Methods*. Kluwer Academic Publishers.

Spivey, J. M. (1992). *The Z Notation*. Prentice Hall, Hemel Hempstead, UK, 2nd edition.

Thomas, S. R. (1995). The PLACA agent programming language. In *Intelligent Agents*, volume 890 of *Lecture Notes in Artificial Intelligence*, pages 307–319, Heidelberg, Germany. Springer-Verlag.

Thomason, R. (2000). Desires and defaults: A framework for planning with inferred goals. In *Proceedings of the 7th International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 702–713, San Fransisco, CA, USA. Morgan Kaufmann.

Toni, F. (2006). Multi-agent systems in computational logic: Challenges and outcomes of the SOCS project. In *Computational Logic in Multi-Agent Systems VI (CLIMA)*, volume 3900 of *Lecture Notes in Artificial Intelligence*, pages 420–426, Heidelberg, Germany. Springer-Verlag.

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

van der Hoek, W., van Linder, B., and Meyer, J.-J. (1993a). An integrated modal approach to rational agents. Technical Report UU-CS-1997-06, Department of Computer Science, Utrecht University, Utrecht, The Netherlands.

van der Hoek, W., van Linder, B., and Meyer, J.-J. (1993b). A logic of capabilities. Rapport IR-330, Vrije Universiteit, Amsterdam, The Netherlands.

van der Hoek, W., van Linder, B., and Meyer, J.-J. (1993c). Unravelling nondeterminism: On having the ability to choose. Technical Report RUU-CS-1993-03, Department of Computer Science, Utrecht University, Utrecht, The Netherlands.

van der Hoek, W., Meyer, J.-J., and van Schagen, J. (2000). Formalizing potential of agents: The KARO framework revisited. In *Proceedings of the 7th CSLI Workshop on Logic, Language, and Computation (LLC)*, number 91 in CSLI Lecture, pages 51–67, Stanford, CA, USA. CSLI Publications.

van der Meyden, R. (1998). Common knowledge and update in finite environments. *Information and Computation*, **140**(2), 115–157.

van der Meyden, R. and Wong, K. (2003). Complete axiomatizations for reasoning about knowledge and branching time. *Studia Logica*, **75**(1), 93–123.

van Harmelen, F., Horrocks, I., and Patel-Schneider, P. F., editors (2001). *A Model-Theoretic Semantics for DAML+OIL*, W3C Note. World Wide Web Consortium. Latest version available at `http://www.w3.org/TR/daml+oil-model`.

van Linder, B. (1996). *Modal Logics for Rational Agents*. Phd thesis, Universiteit Utrecht, Utrecht, The Netherlands.

van Linder, B., Meyer, J.-J., and van der Hoek, W. (1997). Formalizing motivaltional attitudes of agents using the KARO framework. Technical Report RUU-CS-1997-03, Department of Computer Science, Utrecht University, Utrecht, The Netherlands.

van Linder, B., van der Hoek, W., and Meyer, J.-J. C. (1998). Formalising abilities and opportunities of agents. *Fundamentae Informaticae*, **34**(1-2), 53–101.

van Otterloo, S., van der Hoek, W., and Wooldridge, M. (2004). Knowledge as strategic ability. In *Logic and Communication in Multi-Agent Systems (LCMAS)*, volume 85(2) of *Electronic Notes on Theoretical Computer Science*, Amsterdam, The Netherlands. Elsevier Science Publishers.

van Riemsdijk, B., van der Hoek, W., and Meyer., J.-J. (2003). Agent programming in dribble: from beliefs to goals with plans. In *Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 393–400, New York, NY, USA. ACM Press.

van Riemsdijk, B., van der Hoek, W., and Meyer., J.-J. (2004). Verification of multiagent systems via ordered binary decision diagrams: An algorithm and its implementation. In *Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 630–637, New York, NY, USA. ACM Press.

Visser, W., Havelund, K., Brat, G., and Park, S. (2000). Model checking programs. In *Proceedings of the 15th International Conference on Automated Software Engineering (ASE)*, pages 3–12, Washington, DC, USA. IEEE Computer Society.

Walther, D., Lutz, C., Wolter, F., and Wooldridge, M. (2006). Quantitative temporal logics: Pspace and below. *Information and Computation*, **205**(1), 99–123.

Walton, C. (2004). Model checking multi-agent web services. In *Proceedings of the AAAI Spring Symposium on Semantic Web Services*, pages 68–77, Menlo Park, CA, USA. AAAI Press.

Walton, C. (2005). Model checking agent dialogues. In *Declarative Agent Languages and Technologies II (DALT)*, volume 3476 of *Lecture Notes in Artificial Intelligence*, pages 132–147, Heidelberg, Germany. Springer-Verlag.

Wan, F. and Singh, M. P. (2004). Mapping dooley graphs and commitment causality to the π-calculus. In *Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 412–419, New York, NY, USA. ACM Press.

Wooldridge, M. (1992). *The Logical Modelling of Computational Multi-Agent Systems*. Phd thesis, Department of Computation, University of Manchester Institute of Science and Technology (UMIST), Manchester, UK.

Wooldridge, M. (1998). Verifiable semantics for agent communication languages. In *Proceedings of the 3rd International Conference on Multi-Agent Systems (ICMAS)*, pages 349–365, Washington, DC, USA. IEEE Press.

Wooldridge, M. (2000). *Reasoning about Rational Agents*. The MIT Press, Cambridge, MA, USA.

Wooldridge, M. and Jennings, N. R. (1995). Agent theories, architectures, and languages: A survey. In *Intelligent Agents*, volume 890 of *Lecture Notes in Artificial Intelligence*, pages 1–39, Heidelberg, Germany. Springer-Verlag.

Wooldridge, M., Dixon, C., and Fisher, M. (1998). A tableau-based proof method for temporal logics of knowledge and belief. *Journal of Applied Non-Classical Logics*, **8**(3), 225–258.

Wozna, B., Penczek, W., and Lomuscio, A. (2005). Bounded model checking for deontic interpreted systems. *Electronic Notes in Theoretical Computer Science*, **126**, 93–1143.

Yolum, P. and Singh, M. (2004). Reasoning about commitments in the event calculus: An approach for specifying and executing protocols. *Annals of Mathematics and Artificial Intelligence*, **42**(1-3), 227–253.

Yolum, P. and Singh, M. P. (2002). Flexible protocol specification and execution: Applying event calculus planning using commitments. In *Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 527–534, New York, NY, USA. ACM Press.