Where Logic and Agents Meet

Jürgen Dix and Michael Fisher
Where Logic and Agents Meet

Jürgen Dix and Michael Fisher

Clausthal University of Technology, Germany and University of Liverpool, United Kingdom

Abstract

Multi agent systems have been invented 20 years ago and the theory has matured in the last two decades. Logic has and still is playing a prominent part in the basic foundations of agency and also in the development of agent programming languages, the specification and verification of agent systems. This paper is a (subjective) overview of the points of contact of logic and agents as the authors perceived it over the years.

1 Introduction

The notion of an ‘agent’ came about through the confluence of several important ideas: distributed computation; object-based systems; Artificial Intelligence techniques, such as planning and learning; and ideas from Philosophy about rational action, choice and commitment. The basic idea is of a truly autonomous computational entity, able to make its own choices about targets, tasks, and plans. Yet, as the area developed, arguments about what exactly should constitute an agent [139][72] led on to more sophisticated definitions of a rational agent [140], essentially excluding autonomous computational systems based on neural networks or other, primarily stochastic, foundations. Our view of a ‘rational agent’ is as a distributed, autonomous, computational entity with its own goals, information and ways of making choices. When such an agent executes it will do so with some goal in mind but can, at any moment in time, decide to change between goals or between ways of achieving that goal [137], such changes come about as the result of deliberation [37][47].

*This work was partly funded by the NTH School for IT Ecosystems. NTH (Niedersächsische Technische Hochschule) is a joint university consisting of Technische Universität Braunschweig, Technische Universität Clausthal, and Leibniz Universität Hannover.

†This work was partly funded by the NTH School for IT Ecosystems. NTH (Niedersächsische Technische Hochschule) is a joint university consisting of Technische Universität Braunschweig, Technische Universität Clausthal, and Leibniz Universität Hannover.
Describing Agents

The use of one agent on its own can be interesting, but it is only when we consider multiple agents, or multi-agent systems that the full power of the agent metaphor becomes clear. As we move to a multi-agent scenario, not only must the agents themselves now be able to communicate, but it makes sense for each agent to have views on the other agents. Thus, while an agent typically has goals and information about itself, in a multi-agent context it can have goals about the activities of the multi-agent systems and knowledge about other agents in the system. This leads on to very sophisticated interaction, coordination, and cooperation within a multi-agent system. If we are to describe or analyze such multi-agent systems then we must surely take account of the social and well as the individual behaviours of the agents.

The multi-agent paradigm has become increasingly popular and is now the metaphor of choice within complex systems. Its popularity is to do both with the simplicity of the ideas, and with its ability to capture a very wide range of different aspects — not just functional, but societal, emotional, and economic. This, together with the vast array of languages, frameworks and tools now available means that practical construction of multi-agent systems can go together with theoretical analysis.

In this article we provide a short, and very subjective, description of how mathematical techniques, primarily those from logical foundations, have been used in multi-agent systems. This is certainly not a comprehensive survey, but gives some indication of where the fields of “Logic” and “Agents” have met in recent years. The Annals of Math and AI contain a surprising number of articles in this area, among them several special issues focussing on computational logic and its use in agent systems.

2 Describing Agents

How shall we describe an ‘agent’? As we commented above, to some extent we can see an agent as a distributed component, an object, an AI subsystem, or as a philosophical artifact. Indeed, some people have argued that there is nothing special about the agent view. By contrast, we contend that, by considering all these aspects together, the agent metaphor provides a high-level and holistic view of quite complex processes that cannot easily be seen just as distributed computation, object execution, etc.

A key development in describing and understanding agents, particularly rational agents, was the introduction of the BDI (Belief, Desires, Intentions) view [114]. This was based on both Bratman’s philosophical work on intentions [27] and on experiences with practical AI planning [73]. The BDI view contains two aspects: a theoretical framework, comprising distinguished beliefs, desires, and intentions together with the interactions between them [114], [113]; and a practical architecture providing a basis for agent-based implementations [115] [116]. Here we will mainly address the first of these, but
note that most agent systems, including the languages and frameworks mentioned later, are based on the ideas from the BDI architecture.

There have been, of course, logical techniques used for agents that are different to the BDI approach, with some of the most influential being \[91, 43, 44, 100\]. However, we will particularly focus on the BDI approach.

Recall that, in the introduction, we said that the core elements of a rational agent were some goals the agent has, some information it has, and some deliberation techniques for switching between goals based on this information. The BDI approach refines this a little further to incorporate:

- **beliefs** — describing the agent’s view of itself, other agents and its environment (note that the term ‘belief’ is used rather than ‘knowledge’ since the agent can not be certain of its information and it can only believe this information about the world and other agents);
- **desires** — describing the agent’s long term goals; and
- **intentions** — describing those long term goals that have been selected and are actively being pursued.

Thus,

*I believe I have one hour of uninterrupted time available, my desire is that eventually this article will be finished and published, and my immediate intention is to write this particular section.*

In the BDI architecture the above sets are usually supplemented by a set of **actions** describing the primitive ways the agent can affect the world, and a set of **plans** describing how the agent might achieve certain goals. We will see later that these play a significant part in practical agent programming but, for the moment, we will concentrate on beliefs, desires, and intentions.

In the logical formalisation of BDI agents, a key development was the use of various modal and temporal logics for this purpose \[113, 121\]. Thus, in describing agents we (at least) need some logical formalism for capturing the underlying dynamic nature of an agent, and then some (possibly different) formalism for capturing the BDI aspects. Typically, dynamic \[80, 130\] or temporal \[65\] logics have been used for the first of these. For the BDI aspects, the following are used.

1. A logic of belief, typically KD45 modal logic \[16\], but alternatively fuzzy logics, logics of probabilistic belief, or multi-context logics \[76, 63\], or even (if we are certain) logics of knowledge \[66\].

2. A logic of desires, typically KD modal logic, and

3. A logic of intentions, again a KD modal logic.
Agent Interactions and Group Dynamics

Importantly there are many interactions between these dimensions, for example

\[ \neg(I \varphi \land \neg D \varphi) \]

meaning that the agent should never intend to do something that it does not explicitly desire. These logical dimensions, together with the underlying temporal or dynamic basis, allow us to describe quite complex agent activities. For example

\[ I \text{write} \land \Box \text{can write} \Rightarrow B \Diamond \text{finished section} \]

might be interpreted as

“if I intend to write the section and can always write, then I believe that eventually the section will be finished”.

We can see from this that agent descriptions can become quite complex and indeed, since the logical basis is a combination \[94\] of temporal and (several) modal logics, the formal analysis of such descriptions (see later) can be quite expensive.

Before we conclude this section, we just note that, while the BDI approach is widely used, there are other options. One interesting one is the KARO \[131\] framework (where KARO stands for “Knowledge, Abilities, Results, and Opportunities”). Again, this is based on the idea of having an underlying dynamic component together with both informational and motivational components, but with a different approach to that of BDI. Thus, the KARO approach combines dynamic aspects via propositional dynamic logic, informational aspects via S5 modal logic, and motivational aspects via KD modal logic.

There are numerous other approaches, including those involving linear logic \[81\], dynamic logic \[122\], the classical situation calculus \[78\], and the event calculus \[142\].

A final aspect is that, when we move towards describing an agent’s environment we might well have to involve real-time \[4\], spatial \[45\] or probabilistic \[79\] logics.

3 Agent Interactions and Group Dynamics

As opposed to a single agent system, we face new and difficult challenges in a multi-agent system. In this section we briefly consider two of the most important challenges and list some significant developments in logical approaches to tackling them. The challenges are:

\[1\]From now on: I means “I intend”; D means “I desire”; B means “I believe”; \( \Box \) means “always”; and \( \Diamond \) means “sometimes”.

DEPARTMENT OF INFORMATICS
WHERE LOGIC AND AGENTS MEET

- How do agents interact with each other?
- What goals can groups of agents achieve?

Agents have to talk to each other and therefore they need a (standardized) language to do so. Some popular approaches for agent communication are KQML, KIF, and Fipa-ACL, which have their roots in the *speech act* theory of Searle from linguistics [124]. These approaches come from knowledge representation and the part of Philosophy that is closely related to formal logic [127, 134, 101, 77].

Agents developed within one multi-agent system should be able to deal with agents from other systems as well. As one cannot assume that they all have the same “background” (knowledge about the world), they need to have access to a knowledge base where the fundamental concepts they talk about are defined. This is where the idea of the semantic web and *ontologies* comes in. Again, this approach is perfectly suited for logic-based methodologies and many papers have been devoted to deal with it [110, 84, 8, 11] in addition to the vast amount of publications on ontologies and description logics in general [9].

We mention in passing another important part of interaction, namely *negotiation*: how to reach a satisfactory conclusion among self-interested agents? It turns out that many techniques from logic, *argumentation* and *game theory* can be successfully applied here: [60, 6, 93, 35, 29, 28]

While the notion of an agent as an autonomous entity is a very important one, the full power of the agent paradigm is only obtained by the interplay of multiple agents. Often, a single agent is not able to bring about a more complicated goal. Several agents may be able to reach this goal, but their success often depends on how the remaining agents behave. This leads to various branches in multi-agent systems, e.g. *coalition formation* (which coalitions emerge and how do they change over time?) and *what can coalitions bring about* (how can one describe and determine the behaviour of teams of agents?). Coalition formation typically depends on notions of common goals, shared intentions, and the agents’ level of knowledge about their environment. While the development of an agent system over time can be described by temporal logics, such as LTL [108], and CTL [64, 65, 41, 39], these logics are not appropriate to succinctly express the strategic behaviour of agents. For example:

*is there a strategy for some team of agents to make sure that a certain property holds?*

To encompass such requirements, standard temporal logics have been extended with game-theoretic concepts [123]. Thus, the logics ATL* and ATL $^5$ (*Alternating-time Temporal Logic*) and various extensions of them are logics of this kind. They contain *cooperation modalities* of the form $\langle A \rangle$ where ‘$A$’ is a
team of agents. The formula $\langle \langle A \rangle \rangle \gamma$ expresses the statement that team $A$ has a collective strategy that can enforce $\gamma$. The complexity of these logics, in particular the model checking problem, has been studied in detail \cite{30,33,57,129,95}. Various extensions of ATL which allow us to express well-known solution concepts from game theory, such as Nash equilibria, sub-game perfect Nash equilibria, etc., have also been developed \cite{34} and, recently, versions of ATL describing resource-bounded agents have also been defined \cite{32,31}.

Finally, there is the vast area of work covering norms, institutions and organisations in agent systems. The idea here is to look at how human societies have coped with decentralized control, coordination and cooperation, and to re-use such structures in multi-agent systems. Norms and conventions within human societies have evolved over time and research in this area tries to model the appropriate abstractions for multi-agent systems. In this area, we mainly highlight the workshop series devoted to this branch of agent systems \cite{106,107,126}.

4 Agent Programming

While the modelling and analysis of systems in terms of agents has many benefits, particularly for clarity and conciseness, it is important to have appropriate programming abstractions and robust languages in which agent-based systems can be implemented. While in some cases agents are implemented via the use of traditional programming languages it is increasingly clear that the novel aspects of agents require suitably novel programming languages.

Just as the concept of an ‘agent’ evolved from several different areas, so agent programming frameworks show a similarly diverse background \cite{18,55,19}. However, the main development began with AI languages such as PROLOG. Recall that, in PROLOG, we have an ordered list of ‘goals’, together with an ordered list of rules and facts. These rules/facts are searched in a sequential manner in order to try to reduce the first goal on the list to nothing (via a ‘fact’) or to further sub-goals (via a ‘rule’). If goal reduction fails then the execution backtracks to a previous choice point and continues. Many rational agent languages work in a similar way. However, a key aspect of an agent is that it can react to new information (e.g. sensor data) and modify its current goals. So, the PROLOG execution mechanism is essentially modified in two ways:

1. the current list of ‘goals’ can be re-ordered or modified at any moment — since computation concerns reducing the first goal on the list, this change of emphasis can be viewed as the deliberation the agent undertakes about its current goal \cite{67}; and
2. the list of facts and rules can be re-ordered — since this ordering describes
the next rule/fact to be attempted, then re-ordering corresponds
to the agent deliberating about, and changing its view of, how to reduce
the current goal.

There is, of course, much more to these languages but the above provides
a general framework. In BDI based languages the ‘facts’ the agent has are
expanded to include quite complex beliefs and the goals the agent works to-
wards are refined to be ‘intentions’. The key language in this areas is AGENTS-
PEAK [112] developed to be a version of standard declarative programming
adapted to the BDI framework. This has led on to very many interesting and
useful languages of a similar form, such as JASON [24, 25], 3APL, [52], and
GOAL [83]. Particularly the first two of these have been used in many practi-
cal applications.

As mentioned above, agent languages also evolved from many areas. So there are:

- extensions of Java with BDI architectures [136, 109, 102];
- agent extensions of logic programming [97, 92, 83, 119, 133, 117] and
  answer set programming [86, 42, 132, 46];
- languages providing an interaction layer for legacy systems [128, 62, 54]; and
- languages based on executable specification of concurrent agents such as
  METATEM [69, 68] and [10].

There are increasingly many such languages though important work is now
taking place assessing and evaluation the different approaches from both

5 Agent Verification

As the agent, and particularly multi-agent, metaphor is used in more sophisti-
cated and critical scenarios, so attention has turned to questions concerning
the reliability and predictability of such systems. Multi-agent systems are used
in many such scenarios, for example automated stock trading systems [82,
125], process control [120], space exploration [103, 36], sensor networks [96,
118], health-care systems [141], air traffic systems [105, 104], etc [88]. While
multi-agent systems are widely deployed there is often a lack of trust among
users that the agents will always work as expected in critical scenarios.

Unsurprisingly, therefore, there has been a move towards fully formal justi-
fication of the reliability of these agents [70]. As within standard Computer
Science, there is a wide array of techniques that can potentially be adapted
for agent verification [17]. Here, some formal (usually logical) description of the required behaviour of a system, say $Req$, is checked against some representation of the system. In deductive verification, a logical description of the system is given, say $Sys$, and then proof methods are used to establish that $Sys \Rightarrow Req$. An alternative approach is to use algorithmic verification whereby $Req$ is checked on all possible routes through some structure describing the behaviour of the system. In model checking [40], the structure explored is usually a finite-state automaton representing all the runs/executions of the system. This approach is certainly the most popular and successful variety of formal verification.

Mirroring the work of formal verification in other areas, the verification of multi-agent systems is indeed a very active research area [21, 70, 51]. Again, although deductive and rewriting approaches have been developed [71, 85, 2, 53] it is the techniques based on model-checking that have been particularly popular [14, 21, 138, 75, 13, 74, 90, 87, 89]. Work in this area has led to a variety of different practical agent verification systems. One class verifies the required property against a model (structure, as described above) representing possible agent behaviours; a leading example of this is MCMAS [98, 111, 99]. An alternative approach, based on the idea of program model checking [135], involves directly verifying the agent program code rather than a model of its executions. A leading example of this style of agent verification is MCAPL/AJPF [22, 20, 23].

6 Where to now?

Logic (particularly computational logic) continues to be central to the area of multi-agent systems: logical approaches are vital for the specification and semantics of such systems; programming metaphors based on computational logic are predominant amongst agent languages; and formal verification techniques are increasingly used to assess the safety and reliability of multi-agent systems. All of these areas look set to increase in activity and popularity.

The research community associated with these areas is particularly active. For example, the 11th international workshop on Computational Logic in Multi-Agent Systems (CLIMA) was held in 2010; the 12th edition will happen in 2011. CLIMA (http://centria.di.fc.ul.pt/~clima) brings together researchers tackling all the areas covered within this article, together with many more [56, 57]. The Annals of Mathematics in Artificial Intelligence has published special issues related to some of these CLIMA workshops, for example [61, 58], as well as special issues related to other activities linking logic and agents, for example [55, 70].

On the practical side, many tools, techniques and languages have been developed and an interesting and useful initiative involves the multi-agent programming contest. This started in [48], and continues today [49, 50, 12]. The
WHERE LOGIC AND AGENTS MEET

contest aims to provide an objective, and non-trivial, problem to which various different agent programming languages can be applied. This has led to a deeper understanding of the most appropriate ways to program multi-agent systems as well as, of course, improvements in the languages themselves.

As we have seen, logic and agents have a natural and productive interaction. Activity in this area is not only set to continue, but increase. Formal specification, automated synthesis, and deductive verification, via logics of agency are sure to come into their own. Agent programming languages are about to become mainstream. Formal agent verification techniques are essential for trusted, mobile agents. These involve not only the model-checking techniques mentioned above, but other lightweight techniques such as runtime verification. Negotiation, cooperation and planning [59] are all required within the increasingly sophisticated agent applications. And so on.

In summary, although we can only provide a brief (subjective) view of the field in this article, we hope the reader has finished it with a strong and positive feel for the area “where logic and agents meet”.

References


References


References


References


References


References


Technical Report IfI-10-08
References


