Belief Revision for AgentSpeak Agents

Natasha Alechina
University of Nottingham
School of Computer Science
Nottingham, UK
nza@cs.nott.ac.uk

Rafael H. Bordini
University of Durham
Dept. of Computer Science
Durham, UK
r.bordini@durham.ac.uk

Jomi Fred Hübner
Univ. Regional de Blumenau
Dept. Sistemas e Computação
Blumenau, SC, Brazil
jomi@inf.furb.br

Mark Jago
University of Nottingham
School of Computer Science
Nottingham, UK
mtw@cs.nott.ac.uk

Brian Logan
University of Nottingham
School of Computer Science
Nottingham, UK
bsl@cs.nott.ac.uk

ABSTRACT

The AgentSpeak agent-oriented programming language has recently been extended with a number of new features, such as speech-act based communication, internal belief additions, and support for ontological reasoning, which imply a need for belief revision within an AgentSpeak agent. In this paper, we show how a polynomial-time belief-revision algorithm can be incorporated into the ***Jason*** AgentSpeak interpreter. To the best of our knowledge, this is the first attempt to include belief revision within an interpreter for a practical agent programming language.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Intelligent Agents, Multiagent Systems*; I.2.5 [**Artificial Intelligence**]: Programming Languages and Software; I.2.3 [**Artificial Intelligence**]: Deduction and Theorem Proving—*Nonmonotonic reasoning and belief revision*

General Terms

Languages

Keywords

Belief Revision, BDI, AgentSpeak, Jason

1. INTRODUCTION

Agents can be viewed as rational entities characterised in terms of their beliefs, desires and intentions and the relationships between them, and the *Belief*, *Desire* and *Intention* (BDI) approach has emerged as perhaps the dominant paradigm in agent research and development [9, 11, 6]. A key problem for BDI agents is the dynamics of beliefs (and consequently of desires and intentions). Agents are continuously presented with a stream of new information (e.g., from their sensors or other agents) requiring changes in their beliefs. In simple cases, the problem of how the agent's beliefs should be updated is straightforward: for example, if an agent's battery sensor reports that the agent's battery is 50% charged, then the agent's existing beliefs about its battery level (and any derived beliefs regarding how far it can travel before recharging) can be simply overwritten with the appropriate new belief(s). However for agents with multiple sources of information and/or more complex belief states a simple 'overwriting' strategy is inadequate. *Belief revision* is necessary when the agent receives conflicting information from different sources, or information which is contradictory in the context of the agent's own beliefs. For example, an agent may be told by Alice that the next group meeting will be held on Wednesday and by Bob that the meeting is on Thursday. If the agent's ontology contains a rule stating that each meeting has a unique date, then, from the agent's point of view, this information is contradictory. In such cases the problem of how the agent's beliefs should be revised to restore consistency is non-trivial.

Current agent-oriented programming languages typically leave belief update up to the programmer [3] and provide no support for belief revision. However as agent's belief states and the inferences performed on them become richer and more complex, such ad-hoc approaches push an ever-greater burden onto the agent programmer. In this paper, we show how to integrate correct and efficient (polynomial-time) belief revision into the architecture of a BDI agent. We focus on the well-known AgentSpeak agent-oriented programming language. However our approach and algorithms are equally applicable to other BDI agent implementations where agent's beliefs can be represented in the form of ground literals and Horn clauses (e.g., rules, or plans in case of AgentSpeak). While there has been some initial work on belief revision in abstract programming languages [10], to the best of our knowledge, the work presented here is the first attempt to include belief-revision within an interpreter for a *practical* agent-oriented programming language.

2. AgentSpeak AND *Jason*

The AgentSpeak(L) programming language was introduced in [8]. It is based on logic programming and provides an elegant abstract framework for programming BDI agents.

AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.

An AgentSpeak agent is defined by a set of *beliefs* (ground literals) which constitute the agent's *belief base*, and a set of plans which form its *plan library*. An AgentSpeak plan has a *head* which consists of a triggering event (the event(s) for which that plan is *relevant*), and a conjunction of belief literals representing a *context*. The conjunction of literals in the context must be a logical consequence of the agent's current beliefs for the plan to be considered *applicable* when triggered. A plan also has a *body*, which is a sequence of basic actions (i.e., atomic operations the agent can perform so as to change the environment), (sub)goals that the agent has to achieve (or test) when the plan is executed, and belief changes that can serve as "mental notes". Plans are triggered by the addition or deletion of beliefs due to communication or perception of the environment, or due to the addition or deletion of goals as a result the execution of plans triggered by previous events.

The ***Jason*** interpreter implements the operational semantics of AgentSpeak as given in, e.g., [5]. ***Jason*** in written in Java, and its IDE supports the development and execution of distributed multi-agent systems [4].Recent work has made important additions to AgentSpeak, which have also been (or are in the process of being) implemented in ***Jason***. Some of these new features, such as speech-act based communication, internal belief additions, and support for reasoning with ontological knowledge, presuppose a *belief revision* capability as part of agents' reasoning cycle. For example, in [7], an extension of AgentSpeak was proposed which incorporates ontological reasoning within the AgentSpeak interpreter. In the extended language, called AgentSpeak-DL, queries to the belief base are more expressive as their results do not depend only on the agent's explicit knowledge but also on what can be inferred from the ontology; the search for a relevant plan for a particular event is more flexible as this is not based solely on unification, but also on the subsumption relation between concepts; and agents may share knowledge by using web ontology languages such as OWL. These new capabilities can result in inconsistencies in an agent's belief base which cannot be eliminated by a simple 'overwriting' strategy.

The current implementation of ***Jason*** provides a basic belief update capability. This deletes any perceptual beliefs which are no longer perceived from the belief base and adds any new percepts as new perceptual beliefs. Belief additions executed within a plan as well as any information communicated by trusted sources is simply added to the belief base (without checking for consistency). While this approach ensures that the agent's perceptual beliefs remain consistent, it is incapable of handling derived inconsistencies, e.g., if a property is asserted of an individual which is inconsistent with the corresponding concept description. The elimination of such inconsistencies remains the responsibility of the agent developer and must be coded anew for each agent application. Our aim is to extend the ***Jason*** implementation with a belief revision capability which automatically restores belief base consistency in the face of derived inconsistencies.

3. THE BELIEF REVISION ALGORITHM

We have two main objectives when introducing belief revision in AgentSpeak. First the algorithm should be theoretically well motivated, in the sense of producing revisions which conform to a generally accepted set of postulates characterising rational belief revision. Second, we want the resulting language to be practical, which means that the belief revision algorithm must be efficient.

In the belief revision literature, these two goals have traditionally been seen as incompatible. Our approach draws on recent work [2] on efficient (polynomial-time) belief revision algorithms for rule-based agents which satisfy the well-known Alchourrón, Makinson and Gärdenfors (AGM) postulates [1] characterising rational belief revision and contraction. In this section we briefly describe the linear-time belief contraction algorithm introduced in [2]. We then explain how belief revision can be defined in terms of contraction.

We distinguish two kinds of contraction operation. AGM (or coherence) contraction of the agent's belief base K by a belief literal A is defined as the removal of A and sufficient literals from K such that A is no longer derivable using the set of plan instances executed to date. Reason-maintenance contraction additionally removes any beliefs for which A is (recursively) the sole support. To facilitate both AGM and reason-maintenance contraction, the inferential relationships between the beliefs comprising the agent's belief base are represented as a directed graph, where the nodes are beliefs and *justifications*. A justification consists of a belief and a *support list* containing the context (and possibly the triggering event) of the plan used to derive this belief: for example, the assertion of a belief A by a plan with context B triggered by a belief addition event C (or derived by an ontology rule $B, C \rightarrow A$) would have a justification $(A, [B, C])$. Foundational beliefs which were not derived, have a justification of the form $(D, [])$. In the graph, each justification has one outgoing edge to the belief it is a justification for, and an incoming edge from each belief in its support list. We assume that each support list s has a designated *least preferred* member $w(s)$. Intuitively, this is a belief which is not preferred to any other belief in the support list, and which we would be prepared to discard first, if we have to give up one of the beliefs in the list. We assume that we have constant time access to $w(s)$.

The algorithm for AGM contraction by A is then:

```
for each of A's outgoing edges
   to a justification (C, s),
   remove (C,s) from the graph
for each of A's incoming edges
   from a justification (A, s),
   if s is empty, remove (A, s);
   else contract by w(s);
remove A.
```

To implement reason-maintenance type contraction, we also remove beliefs which have no incoming edges.

In [2], it was shown that the contraction operator defined by the algorithm satisfies the AGM postulates for contraction (K-1)–(K-4) and (K-6) [1] (the recovery postulate, (K-5), is not satisfied). Closure under consequence in classical logic is replaced with closure in a weaker logic W which has a single inference rule of generalised modus ponens.

The algorithm runs in time $O(kr + n)$, where r is the number of plans, k the maximal number of beliefs in any support list, and n the number of literals in the belief base. Indeed, the upper bound on the number of steps required to remove justifications corresponding to plan instances is $r(k + 1)$ (one constant time operation for each belief in a context of the plan and one for the belief asserted by the plan). Removing all justifications corresponding to foundational beliefs costs n steps. The last step in the contraction algorithm (removing a belief) is executed at most n times.

The AGM (resp. reason-maintenance) revision of the set of literals in the agent's belief base K can then be defined as: add A, close under consequence and AGM (resp. reason-maintenance) contract by all contradictions. Note that for agents which reason using ground literals and Horn clauses, closure under consequence can be computed in polynomial time.

4. BELIEF REVISION IN *Jason*

Future releases of ***Jason*** will incorporate an implementation of the belief revision algorithm presented above as a user-selectable

option. The new implementation clearly distinguishes between *belief update*, i.e., the addition and deletion of perceptual beliefs following perception, and *belief revision* proper, i.e., the elimination of inconsistencies following belief additions by plans or ontological reasoning. A belief addition may be discarded (as at present) or may result in the deletion of some other belief(s) in order to allow the new belief to be consistently added to the belief base. Which beliefs are deleted is determined by a user-specified preference order (see below). Here, we assume that the agent's plans contain no explicit belief deletions, and that deletion of beliefs only occurs as a result of belief update and revision.

Each belief literal added to the belief base has an associated "dependencies list" (the literals that allowed the derivation of the literal in question), and a "justifies list" (the literals which the literal in question justifies, that is, it appears in their dependencies list). For example, if the plan that generated the belief change, say $+bl$, has the form "`@p` te: l_1 `&` ... `&` l_n `<-` bd", where te is a triggering event and bd a plan body, the support list of the corresponding justification is simply the ground literals from the plan context, "[l_1,...,l_n]". Note that if the triggering event, te, is itself a belief addition, the literal in te is included together with the context literals in the support list. Further, for each literal $l_1, \ldots, l_n$ we include the newly added belief in the literal's "justifies" list. We also record the time at which the belief was added as part of the justification. When deleting perceptual beliefs during belief update, we remove any justification which has the deleted perceptual belief in its support list, and mark the justified beliefs as self supporting (foundational) beliefs. This additional bookkeeping reflects the special status of perceptual beliefs: perceptual beliefs can trigger or justify the addition of other beliefs, but their deletion is not in itself a reason for removing a derived belief.

The belief revision algorithm also requires the definition of a partial order expressing the preference, or entrenchment, of each belief. To allow for user customisation, this is defined as a separate method that can also be overridden. The default method gives preference to perceived information over communicated information (as also happens in [10]), and in case of information from the same source, it gives preference to newer information over older information.

The implementation described above is conservative in revising only the agent's belief state. The agent's plans are considered part of the agent's program and are not revised (though revising, e.g., plans received from other agents would be an interesting extension). Similarly, when revising beliefs derived using ontological rules, we assume the ontology used by the agent to be immutable and consistent, and that it is consistent with every other ontology it references.

5. CONCLUSIONS AND FUTURE WORK

Experience has shown that correct and efficient implementations of agent-oriented programming languages are key in the widespread adoption of agent technology. As agent programming languages become richer, it becomes harder for the agent programmer to ensure that the belief states of agents developed using these languages are consistent. In this paper we briefly summarised the rationale for including automatic belief revision in a BDI agent programming language, and described the integration of an efficient (polynomial-time) belief-revision algorithm into the ***Jason*** AgentSpeak interpreter. Our approach is theoretically well motivated in the sense of producing revisions which conform to a generally accepted set of postulates characterising rational belief revision. We believe that other BDI agent-oriented programming languages and their platforms [3] which leave the problem of maintaining a consistent belief state entirely to agent programmer, can also benefit from our approach.

In future work, we plan to further investigate the implications of integrating belief revision into the AgentSpeak execution cycle (e.g., allowing belief deletions in plans, ontological reasoning). On the more practical side, we plan to develop large-scale agent applications to assess the performance of ***Jason*** with belief revision.

Acknowledgements

Rafael Bordini gratefully acknowledges the support of The Nuffield Foundation (grant number NAL/01065/G).

6. REFERENCES

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

[2] N. Alechina, M. Jago, and B. Logan. Resource-bounded belief revision and contraction. In *Proceedings of the 3rd International Workshop on Declarative Agent Languages and Technologies (DALT 2005)*, 2005.

[3] R. H. Bordini, M. Dastani, J. Dix, and A. El Fallah Seghrouchni, editors. *Multi-Agent Programming: Languages, Platforms and Applications*. Number 15 in Multiagent Systems, Artificial Societies, and Simulated Organizations. Springer, 2005.

[4] R. H. Bordini, J. F. Hübner, et al. ***Jason****: A Java-based agentSpeak interpreter used with saci for multi-agent distribution over the net*, manual, release version 0.7 edition, August 2005. `http://jason.sourceforge.net/`.

[5] R. H. Bordini and Á. F. Moreira. Proving BDI properties of agent-oriented programming languages: The asymmetry thesis principles in AgentSpeak(L). *Annals of Mathematics and Artificial Intelligence*, 42(1–3):197–226, 2004.

[6] D. Kinny. Agents — the challenge of relevance to the it mainstream. In *Programming Multi-Agent Systems, Second International Workshop ProMAS 2004*, LNCS Vol. 3346, pages 38–43. Springer-Verlag, 2005.

[7] A. F. Moreira, R. Vieira, R. H. Bordini, and J. Hübner. Agent-oriented programming with underlying ontological reasoning. In *Proceedings of the 3rd International Workshop on Declarative Agent Languages and Technologies (DALT 2005)*, 2005.

[8] A. S. Rao. AgentSpeak(L): BDI agents speak out in a logical computable language. In *Proceedings of the Seventh Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW'96)*, LNAI Vol. 1038, pages 42–55, London, 1996. Springer-Verlag.

[9] A. S. Rao and M. P. Georgeff. Modeling rational agents within a BDI-architecture. In *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning (KR'91)*, pages 473–484, 1991.

[10] R. M. van Eijk, F. S. de Boer, W. van der Hoek, and J.-J. C. Meyer. Information-passing and belief revision in multi-agent systems. In *Intelligent Agents V — Agent Theories, Architectures, and Languages*, LNCS Vol. 1555, pages 29–45, Berlin, 1999. Springer-Verlag.

[11] M. Wooldridge. *Reasoning About Rational Agents*. MIT Press, 2000.