Belief Update in AgentSpeak-DL

Álvaro F. Moreira[1] and Renata Vieira[2]

[1] Universidade Federal do Rio Grande do Sul
afmoreira@inf.ufrgs.br

[2] Pontifcia Universidade Catlica do Rio Grande do Sul
renata@pucrs.br

Abstract. In previous work [8] we proposed an extension for the belief base of AgentSpeak agents based on Description Logic, aiming at enabling agent oriented programming to cope with recently proposed technologies for the Semantic Web. In such an extension an agent belief base contains the definition of complex concepts, besides specific factual knowledge. The foreseen advantages are: (i) more expressive queries to the belief base; (ii) a refined notion of belief update, which considers consistency of a belief addition; (iii) flexibility in plan searching allowed by subsumption relation between concepts; and (iv) knowledge sharing in a semantic web context (based on OWL). Following this proposal an extension of the well know Agent Speak interpreter, Jason, was presented in [13]. Among the interesting open issues is how to deal with the addition of beliefs which violates ontology consistency. In this work we discuss this problem related to ABox updating in the context of AgentSpeak-DL

1 Introduction

Developing applications that make full use of machine-readable knowledge sources as promised by the Semantic Web vision is attracting much of current research interest.

Among the key components of the Semantic Web are *domain ontologies* [4]. They are responsible for the specification of the domain knowledge, and as they can be expressed logically, they can be the basis for sound reasoning in the specified domain. Another key component of the Semantic Web technology is the work on intelligent agents which are responsible for making use of the available knowledge, autonomously interacting with other agents, so as to act on the user's best interest.

In [8] we bring these two key Semantic Web components together by proposing an extension to the BDI agent programming language AgentSpeak [3]; there has been much work on extending this language so that it becomes a fully-fledged programming language for multi-agent systems [10], [12]. The AgentSpeak extension proposed in [8] is based on Description Logic (DL) [1] rather than classical (predicate) logic. With DL, the belief base of an AgentSpeak agent consists of the definition of complex concepts and relationships among them, as well as specific factual knowledge (or beliefs, in this case) — in DL terminology, these are called TBox and ABox respectively.

Description logics are at the core of widely known ontology languages, such as the Ontology Web Language (OWL) [2]. An extension of AgentSpeak with underlying automatic reasoning over ontologies expressed in such languages can have a major impact

Dagstuhl Seminar Proceedings 08361
Programming Multi-Agent Systems
http://drops.dagstuhl.de/opus/volltexte/2008/1641

on the development of agents and multi-agent systems that can operate in a Semantic Web context.

Following the proposal in [8], JASDL, an extension of the well know AgentSpeak interpreter, Jason [10], was presented in [13]. Among the interesting open issues is how to deal with the addition of beliefs which violate ontology consistency. In this work we will discuss this problem related to ABox updating in the context of AgentSpeak-DL. The basic idea is to adapt the algorithms proposed in [6] and [7] for the updating of ontologies described in a description logic.

2 The AgentSpeak-DL Language

AgentSpeak-DL is essentially the same as predicate logic AgentSpeak, the only difference being that in predicate-logic AgentSpeak the belief base of an agent consists solely of ground atoms, whereas in AgentSpeak-DL the belief base contains the definition of complex concepts and relationships, besides factual knowledge. An AgentSpeak-DL agent specification *ag* is thus given by an ontology *Ont* and a set *ps* of plans, as defined by the grammar in Figure 1.

$$
\begin{array}{lll}
ag & ::= & Ont \quad ps \\
 & & \\
Ont & ::= & TBox \;\; ABox \\
at & ::= & C(t) \mid R(t_1, t_2) \\
 & & \\
ps & ::= & p_1 \ldots p_n \qquad (n \geq 1) \\
p & ::= & te : ct \leftarrow h \\
te & ::= & +at \mid -at \mid +g \mid -g \\
ct & ::= & at \mid \neg at \mid ct \;\wedge\; ct \mid \mathsf{T} \\
h & ::= & h_1 ; \mathsf{T} \mid \mathsf{T} \\
h_1 & ::= & a \mid g \mid u \mid h_1 ; h_1 \\
g & ::= & !at \mid ?at \\
u & ::= & +at \mid -at
\end{array}
$$

Fig. 1. AgentSpeak-DL Syntax.

A TBox is a set of class and property descriptions, and axioms establishing equivalence and subsumption relationships between classes (unary predicates) and properties (binary predicates). An ABox describes the state of an application domain by asserting that certain individuals are instances of certain classes and that certain individuals are related by a property.

In order to keep the formal treatment and examples simple, from now on we will consider TBoxes with classes only, and ABoxes with instances of those classes; that is, we assume a simplified language with no properties.

A plan is formed by a *triggering event* — denoting the events for which that plan should be considered *relevant* — followed by a conjunction of belief literals represent-

ing a *context*. The context must be a logical consequence of that agent's current beliefs for the plan to be *applicable*. The remainder of the plan is a sequence of basic actions or (sub)goals that the agent has to achieve (or test) when the plan, if applicable, is chosen for execution. In Figure 2, we give examples of AgentSpeak-DL plans that were written to deal with the event resulting from the arrival of a presenter/speaker in the room.

```
+paperPresenter(P)
   : late(P)
      ← !reschedule(P).

+invitedSpeaker(P)
   : late(P)
      ← !apologise;
         !announce(P).

+presenter(P)
   : ¬late(P)
      ← !announce(P).
```

Fig. 2. Examples of AgentSpeak plans.

The first plan in Figure 2 says that if a presenter of a paper is late he is rescheduled to the end of the session (and the session goes on). If an invited speaker is late, apologies are given to the audience and the speaker is announced. The third plan just announces any presenter (presenter being a concept that is the union of paperPresenter and invitedSpeaker) if he is not late.

An example of TBox components using the language above is as follows:

$$\mathit{invitedSpeaker} \sqsubseteq \mathit{presenter}$$
$$\mathit{paperPresenter} \sqsubseteq \mathit{presenter}$$

This TBox asserts that the concepts *invitedspeaker* and *paperpresenter* are subconcepts of *presenter*. Examples of elements of an ABox defined with respect to the TBox above are:

$$\mathit{invitedSpeaker}(\mathit{john})$$
$$\mathit{paperPresenter}(\mathit{mary})$$

3 Ontological reasoning in AgentSpeak-DL

The steps in the reasoning cycle are essentially the same in AgentSpeak-DL as for predicate-logic AgentSpeak, with the exception of the following aspects that are affected by the introduction of ontological reasoning:

– *plan search*

– *querying the belief base*
– *belief updating*

3.1 Plan Search in AgentSpeak-DL

The reasoning cycle of an agent can be better understood by assuming that it starts with the selection of an event from the set of events. The next step in the reasoning cycle is the search for relevant plans for dealing with the selected event.

As an example let us consider the case of checking for plans that are relevant for a particular event in the smart meeting-room scenario again. Suppose that a sensor in a smart meeting-room has somehow detected in the environment the arrival of the invited speaker *john*. This causes the addition of the external event $\langle +\texttt{invitedSpeaker}(\texttt{john}), \top \rangle$ to the set of events. Suppose also that *invitedSpeaker* $\sqsubseteq$ *presenter* can be inferred from the ontology. Under these circumstances, a plan with triggering event $+\texttt{presenter}(\texttt{X})$ is also considered relevant for dealing with the event. Observe that using subsumption instead of unification alone as the mechanism for selecting relevant plans potentially results in a larger set of plans than in predicate-logic AgentSpeak.

3.2 Querying the Belief Base

The evaluation of a test goal $?at$ in the body of a plan is more expressive in AgentSpeak-DL than in predicate-logic AgentSpeak. In predicate-logic AgentSpeak, the execution of a test goal consists in testing if *at* is a logical consequence of the agent's beliefs. The crucial difference is that now the reasoning capabilities of DL allows agents to infer knowledge that is implicit in the ontology. As an example, suppose that the agent belief base does not refer to instances of *attendee*, but has instead the facts *invitedSpeaker*(*john*) and *paperPresenter*(*mary*). If in the TBox we define that a paper presenter and an invited speaker are also attendees, a test goal such as $?attendee(A)$ succeeds producing substitutions that map *A* to *john* and *mary*.

3.3 Belief Updating

Both plan search and querying are well resolved both in the formal semantics and in the implementation of AgentSpeak-DL. Opens issues still remain in relation to belief updating. Suppose for instance that the TBox of an ontology is such that can be inferred from it that the concepts *chair* and *bestPaperWinner* are disjoint. Clearly, if the ABox asserts that *chair*(*mary*), the assertion *bestPaperWinner*(*mary*) is not to be added to it, otherwise the ontology would become inconsistent. In the process of developing an ontology a reasoner will inform the ontology developer of this inconsistency so that he can take an action by probably reviewing his definition, but in the context of multi-agent systems belief updating reveals to be more subtle.

In the first versions of predicate-logic AgentSpeak, the addition of a belief to the belief base has no further implications . If the addition of a belief would make the belief base inconsistent the belief would not be added. Later, [9] proposed the adoption of a principled notion of belief update following (most of) the rational postulates of [11].

The approach taken in JASDL, an implementation of AgentSpeak-DL in the Jason platform [13] is: if the update will make the ABox inconsistent it is not performed. So in this respect it is not different from the approach taken in the predicate logic based version of AgentSpeak. But that is not satisfactory since the updating has to be performed to reflect the new state of affairs.

An interesting approach for update of ontologies is that of [6]. In that approach the TBox is taken to be invariant and it is the ABox that has to be fixed in order to keep consistency. We believe that this is not a restrictive assumption since, in the real world, ABox changes are more frequent. In what follows we illustrate the main points of the algorithm proposed in [6] and we use their running example.

Suppose we have the following very imple DL-Lite ontology describing a basketball players' domain:

- TBox: $\{\exists.\mathit{WillPlay} \sqsubseteq \mathit{AvailablePlayer}, \mathit{AvailablePlayer} \sqsubseteq \mathit{Player}, \mathit{Injured} \sqsubseteq \neg\mathit{AvailablePlayer}\}$
- ABox: $\{\mathit{WillPlay}(\mathit{John}, \mathit{allstargame})\}$

Observe that we can infer from the ontology above that $\mathit{AvailablePlayer}(\mathit{John})$, $\mathit{Player}(\mathit{John})$, and that $\neg\mathit{Injuried}(\mathit{John})$. Suppose now that John gets injured so the ABox has to be updated with $\mathit{Injured}(\mathit{John})$ to reflect this new state of affairs. But simply adding $\mathit{Injured}(\mathit{John})$ to the ABox $\{\mathit{WillPlay}(\mathit{John}, \mathit{allstargame})\}$ will produce an inconsistent ABox.

Observe also that the fact $\mathit{Injured}(\mathit{John})$ means, according to the TBox, that he is **not an available player anymore**, and not being an available player also means that he **will not play a game**. However, he **remains a player** and if we simply remove $\mathit{WillPlay}(\mathit{John}, \mathit{allstargame})$ form the ABox and add $\mathit{Injured}(\mathit{John})$, the fact that John remains a player is not captured anymore. In order to fix this $\mathit{Player}(\mathit{John})$ has to be added to the ABox.

The update algorithm porposed by [6] can be briefly described as follows:

- Input:
  - a set $\mathcal{F}$ of facts
  - a DL-Lite ontology: TBox $\mathcal{T}$ and ABox $\mathcal{A}$
- Output:
  - ERROR if $< \mathcal{T}, \mathcal{F} >$ is inconsistent
  - a new ABox $\mathcal{A}'$ otherwise

For the example above the main steps of the algorithm are the following:

- $\mathcal{A}' = \mathcal{A} \cup \mathcal{F}$, i.e $\mathcal{A}' = \{\mathit{Willplay}(\mathit{John}, \mathit{allstargame})\} \cup \{\mathit{Injured}(\mathit{John})\}$
- $\mathcal{F}'$ = what is inferred from $< \mathcal{T}, \mathcal{A} >$ (original) and is contradictory with $\mathcal{F}$ i.e $\mathcal{F}'$ = $\{\mathit{AvailablePlayer}(\mathit{John}), \mathit{WillPlay}(\mathit{John}, \mathit{allstargame})\}$
- remove from $\mathcal{A}'$ what is in $\mathcal{F}'$. We then have $\mathcal{A}'$ = $\{\mathit{Injured}(\mathit{John})\}$
- add to $\mathcal{A}'$ what can be derived from what has been removed and do not contradict $\mathcal{F}$ i.e $\mathcal{A}'$= $\{\mathit{Injured}(\mathit{John})\} \cup \{\mathit{Player}(\mathit{John})\}$

4 Conclusions and Future Work

This is a draft paper exposing preliminary ideas for belief updating in AgentSpeak-DL based on an algorithm for ontology updating. Most of the work still has to be done and we foresee the following activities: we are now working on the formal semantics of AgentSpeak-DL with belief updating. This formal semantics will be the starting point both for implementing a version of JASDL with principled belief updating and also for investigating which belief revision postulates [11] are followed by agents developed with the AgentSpeak-DL programming language.

We also would like to extend the belief modification in AgentSpeak-DL in order to incorporate principled belief removal. As a starting point for this investigation we will adopt the algorithm for belief update and removal proposed in [7].

References

1. F. Baader, D. Calvanese, D. N. D. McGuinness, and P. Patel-Schneider, editors. *Handbook of Description Logics*. Cambridge University Press, Cambridge, 2003.
2. D. L. McGuinness and F. van Harmelen, editors. *OWL Web Ontology Language overview. W3C Recommendation*. Avalilable at http://www.w3.org/TR/owl-features/, February 2004.
3. A. S. Rao. AgentSpeak(L): BDI agents speak out in a logical computable language. In W. Van de Velde and J. Perram, editors, *Proceedings of the Seventh Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW'96), 22–25 January, Eindhoven, The Netherlands*, number 1038 in LNAI, pages 42–55, London, 1996. Springer-Verlag.
4. S. Staab and R. Studer, editors. *Handbook on Ontologies*. International Handbooks on Information Systems. Springer, 2004.
5. R. Stevens, C. Wroe, P. W. Lord, and C. A. Goble. Ontologies in bioinformatics. In Staab and Studer [4], pages 635–658.
6. G. De Giacomo, M. Lenzerini, A. Poggi, and R. Rosati. On the Update of Description Logic Ontologies at the Instance Level. In *AAAI*, 2006.
7. G. De Giacomo, M. Lenzerini, A. Poggi, and R. Rosati. On the Approximation of Instance Level Update and Erasure in Description Logics. In *AAAI*, pages 403–408, 2007.
8. Á. Moreira, R. Vieira, R. Bordini, and J. Hübner. Agent-Oriented Programming with Underlying Ontological Reasoning. In *DALT*, pages, 155–170, 2005.
9. N. Alechina, R. H. Bordini, J. F. Hübner, M. Jago, and B. Logan. Belief revision for agentspeak agents. In *AAMAS*, pages 1288–1290, 2006.
10. R. H. Bordini and J. F. Hübner. ***Jason****: a Java-based interpreter for an extended version of AgentSpeak*, 2007. `http://jason.sourceforge.net/`.
11. J. Doyle. Reason maintenance and belief revision – foundation vs. coherence theories. In P. Gärdenfors, editor, *Belief Revision*, Cambridge Tracts in Theoretical Computer Science. Cambridge, Cambridge, 2004.
12. R. Vieira, Á. F. Moreira, M. Wooldridge, and R. H. Bordini. On the formal semantics of speech-act based communication in an agent-oriented programming language. *J. Artif. Intell. Res. (JAIR)*, 29:221–267, 2007.
13. T. Klapiscak and R. Bordini JASDL: A Practical Programming Approach Combining Agent and Semantic Web Technologies In DALT, 2008