SAM
Semantic Agent Model for SWRL rule-based agents

Julien Subercaze
Université de Lyon, LIRIS UMR 5205, INSA de Lyon, Villeurbanne, France
julien.subercaze@liris.cnrs.fr

Pierre Maret
Université de Lyon, LaHC UMR 5516, Université de Saint-Etienne, France
pierre.maret@univ-st-etienne.fr

Keywords: Autonomous agent, agent architecture, rule-based agent, semantic web, behavior exchange, consistency checking, SWRL.

Abstract: Semantic Web technologies are part of multi-agent engineering, especially regarding knowledge base support. Recent advances in the field of logic for the semantic web enable a new range of applications. Among them, programming agents based on semantic rules is a promising field. In this paper we present a semantic agent model that allows SWRL programming of agents. Our approach, based on the extended finite state machine concept, results in a three layers architecture. We detail the architecture, the syntax of the rules , the agent interpreter cycle and present a prototype validating the concept. We present two distinguished features of our approach : behavior exchanges and consistency checking.

1 Introduction and motivation

Since the publishing of the agent roadmap in 2003 (Luck et al., 2003) that pointed out the lack of connection between Multi-Agent Systems and Semantic Web technologies, applications and frameworks have been developed to bridge this gap.

But agent behaviour programming has difficulties to take advantage of these technologies. The S-APL (Semantic agent programming language) was introduced by Katasonov (Katasonov and Terziyan, 2008). This language, which is the most advanced attempt of agent semantic programming is built on top of JADE and CWM (Closed World Machine), a rule based reasoning engine based on Horn Clauses and Closed World assumption. Closed world assumption implies that evertything that is not known to be true, is false. The opposite of closed world assumption is the open world assumption. Open world assumption states that everything that is not known is undefined. As stated in (Damásio et al., 2006), the incompleteness of knowledge owned by agents is the reason for using the open world assumption in MAS. Our motivation is to build an agent model that takes advantage of Description logics expressivity used in semantic web technologies OWL and SWRL (Semantic Web Rule Language). Description logics are more expressive than standard Horn clauses and are based upon open world assumption. Due to recent advances in implementation, it is now possible to develop agents based on SWRL rules.

In the next section we explain the construction of our agent's model. We first introduce the layered architecture, then detail the control structure and give a formal description of the SAM grammar. After the description of the agent architecture, we present in section 3 behaviour exchanges and consistency checking features. An implementation of the architecture is presented in section 4. Our conclusions are presented in section 5.

2 Building Agents with Semantic Rules

2.1 SAM agent Architecture

Programming agent behaviour using a rule language can be carried out in two ways. The first way consists in extending a logic programming language in order to support traditional agent features (i.e. message passing, threading, etc.). The second way consists

in building a layered architecture using the rule language at an upper layer. Agent features are delegated to a lower layer. Commonly, in this type of architecture, the lower level language (i.e. Java, C++,etc.) is used to handle communication, file access, thread management, etc. The main idea behind this approach is to reuse the required features for MAS that are already implemented in another language and to define an agent interpreter to support a particular architecture, such as BDI for instance. The literature shows examples of both approaches. Clark et al. (Clark et al., 2001) follows the first approach by extending Qu-Prolog with multi-threading support and inter-thread message communication. However, this approach is not scalable and does not comply with the Agent Communication Language (ACL) specified by the FIPA [1]. FIPA-ACL is currently recognized as the standard for agent communication and ensures interoperability between MAS frameworks. S-APL, that we discussed in the previous section, follows the same approach but some direct calls to JAVA functions are inserted into the rules.

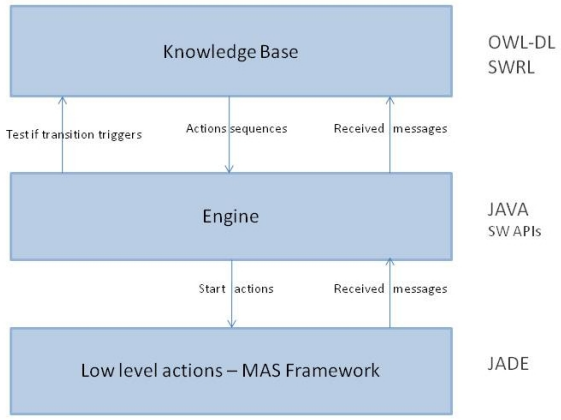

Figure 1: SAM Agent Architecture

Standard MAS languages rely on the second approach. Agent0, the first agent dedicated language, which is an implementation of Shoham's Agent Oriented Programming was developed on top of LISP. Equally, 3APL, 3APL-m, JASON and the BDI agent system Jadex are based on JAVA.

Our architecture follows the second approach and results in the following layered architecture (Fig. 1) :

Knowledge Base The knowledge base is the upper layer of the SAM architecture. The knowledge base contains the knowledge of the agent which, in our approach, is composed of static knowledge and behaviour. Behaviour of agent is expressed using SWRL rules. As SWRL is based upon OWL, terms of the knowledge base are directly manipulated in the rules. Terms of the knowledge base can appear in both antecedent and consequent of rules. A formal specification of the rule syntax is given in section 2.3.

Engine As SWRL buid-ins do not cover all the requirements for agent programming , we have introduced additional low level actions (3rd layer). This middle layer is the control structure that make the interface between the rules contained in the knowledge base and the low level actions. Rules from the knowledge base are fired by the engine, one at a time. If the rule implies to call low level actions, the engine layer carries out this call.

Low level actions and MAS Framework This layer contains the implementation of the low level actions that are complementary to SWRL built-ins. Notice that these actions are introduced as instances of OWL class Actions in the syntax of the rules (top layer). Communication between agents relies on an existing MAS framework. Messages are structured following the FIPA-ACL standard, consequently the MAS framework has to be FIPA compliant (our implementation is based upon JADE). Messages from other agents are received through the MAS framework, then converted into an OWL representation and finally added to the knowledge base.

2.2 Control structure

Rule-based agents constitutes an important part of the research on MAS. In (Hindriks et al., 1999b), Hindriks et al. define the requirement for a minimal agent programming language that includes rules and goals. They also defined formalization tools that were applied to three standard agent programming languages AGENT-0(Shoham, 1991), AgentSpeak(L)(Rao, 1996) (that was later implemented and extended in JASON(Bordini and Hubner, 2006)) and 3APL(Hindriks et al., 1999a). Their definition of an agent program for goal directed agents includes a set of rules Γ called the rule base of the agent. They identify rule ordering as a crucial issue in rule-based agents. However, this presents us with the following problem : when several rules from the ruleset can be fired, there must be an order to determine the sequence of execution of those rules. So the order in which the rules will be sorted must be defined. Hindriks et al. (Hindriks et al., 1999b) proposed that all rules fall into one of the following categories :

[1] http://www.fipa.org/repository/aclspecs.html

```
SAMrule       ::= 'Implies(' [ URIreference ]
                  { annotation }
                  SAMantecedent SAMconsequent ')'

SAMantecedent::= currentState('i-variable')'
                 hasStateValue'('i-variable')' atom*

SAMconsequent::= hasNextState'('i-variable')'
                 hasActionList'(a-list')' atom*

a-list        ::= hasValue(action) hasNext(a-list)
                | endlist

action        ::= URIreference hasParameterName(a-name)

a-name        ::= hasParameterValue(i-object)
```

Figure 2: Extension of SWRL EBNF

reactive(R), *means-end(M)*, *failure(F)* and *optimisation(O)* with an order based on intuition :

$$R > F > M > O$$

As SWRL doesn't support rule ordering, we are also confronted with the same issue. However, instead of deciding an arbitrary order, we have decided to use another model of behavior, a slightly modified version of the Extended Finite State Machine (EFSM) model (Cheng and Krishnakumar, 1993), that guarantees the execution of only one rule at a time.

2.3 Language Syntax

The syntax of the rule language that we designed (given in figure 2) is expressed in Extended Backus-Naur Form (EBNF). This syntax is based on the existing SWRL EBNF syntax as specified in (Horrocks et al., 2004). SAM grammar is a subset of the SWRL grammar. In the antecedent of a SAM rule (*SAMantecedent*) it is mandotary to specify to which state the rule applies. This is set up by the *hasStateValue* property. The previous property, *currentState*, ensures that the rule will be fired when the current state of the EFSM is the one to which the rule applies. The second part of the antecedent contains the triggering conditions. In this part, conditions under which the transition will be triggered are defined. The range of these conditions is the knowledge base of the agent. These conditions are represented by *atom** which is not modified from the original SWRL specification. Conditions can test the validity of class belonging, property between classes or between individuals, including received messages.

The rule consequent term (*SAMconsequent*) specifies the destination state of the transition and the sequence of atomic actions to be executed. *Each action has different parameters. Parameters are passed using two properties, *hasParameterName* and *hasParameterValue*. The first property applies to the action which is to be executed and specifies the name of the parameter. Then *hasParameterValue* is applied to the name of the parameter in order to specify its value.

Internal Actions :

Among internal actions, we made the distinction between SWRL built-ins that are executed by the rule engine and the other required actions that in our model, are the low level atomic actions. These latter are called by the agent interpreter.

External Actions refer to the agents' interactions with their environment. We restrict our scope to software agents that evolve in an electronic environment. Interactions are then limited to message exchanges between agents. We rely on the FIPA ACL specification for the message structures. Received messages are stored in the messagelist. From those simple actions, it is possible to build complex interactions between actions, for instance FIPA ACL specifies an extensive communicative act library including query-answer, contracting, proposal, subscribing. Different fields of the message are represented in the OWL knowledge Base using properties, i.e. *hasPerformative*, *hasContent*, *hasSender*.

3 Features and Benefits

This architecture, as described in previous sections presents two main advantages over existing semantic agent architectures. Firstly, in SAM, behaviors are represented using a rule language that follows the same syntax as the knowledge representation (SWRL and OWL). This way, behaviour and knowledge are represented and stored in the same layer of the architecture. Combining this feature with the fact that agents are aware of their own architecture, behaviour exchanges among agents are natively enabled in SAM.

Secondly, the logical foundation of SAM agents differs from existing agents framework. Description Logic tools provide reasoning mechanism that allows consistency checking. For a single agent, consistency checking ensures that the received knowledge from external sources is consistent with its internal knowledge.

We implemented examples for both behavior exchanges and consistency checking in the prototype presented in the next section.

4 Implementation

We have developed a JAVA interpreter that communicates with the knowledge Base using the Protege-OWL API [2]. Pellet [3] is used in combination with Jena [4] as a OWL and SWRL reasoner. The JADE framework is used for the low level external actions and to provide communication facilities between agents. The framework handles agent registration, service discovery and message passing. It also provides an environment that is FIPA-ACL compliant and thus ensures interoperability with FIPA-ACL compliant frameworks. Since OWL does not support RDF lists, we used OWLList to represent action sequences and for the queue of received messages. The open-source prototype is available online [5].

Along to the validation of the model, the implementation showed us some limitations. We have used Pellet as a SWRL reasoner, since it is currently the most advanced open-source implementation of SWRL. As developments stands at the moment, several important features are not supported by Pellet, for instance some SWRL built-ins are not yet available.

The implementation results show the feasability of the proposal and we intend to further develop the prototype to make it fully suitable for the development of applications.

5 Conclusion and perspectives

In this paper we showed how the next generation of Semantic Web technologies can be applied in MAS programming. We presented an agent model called SAM that enables agent development using the Semantic Web Rule Language (SWRL). We described the three layer architecture, the OWL agent model, its rule syntax and we validated our approach by the implementation of a prototype.

We presented two main features of SAM. We described how behaviours exchanges in SAM can be made regardless of the low level implementation language, making SAM agents "real" semantic agents. Description Logic that underpins and which is inherent in SWRL is a very powerful logic and it allows greater agent reasoning capabilities than standard Prolog. Description Logic tools enable dynamic consistency checking, that is used to maintain agent's knowledge base consistency and to provide explanation of inconsistencies. Further research will focus on the development of applications based on these features, especially in the field of multi-agent argumentation and negotiation.

[2] http://protege.stanford.edu/plugins/owl/

[3] http://clarkparsia.com/pellet/

[4] http://jena.sourceforge.net/

[5] http://code.google.com/p/semanticagent/

REFERENCES

Bordini, R. and Hubner, J. (2006). BDI agent programming in AgentSpeak using Jason. In *Proceedings of 6th International Workshop on Computational Logic in Multi-Agent Systems (CLIMA VI)*, volume 3900, pages 143–164. Springer.

Cheng, K. and Krishnakumar, A. (1993). Automatic functional test generation using the extended finite state machine model. In *Proceedings of the 30th international conference on Design automation*, pages 86–91. ACM New York, NY, USA.

Clark, K., Robinson, P., and Hagen, R. (2001). Multi-threading and message communication in Qu-Prolog. *Theory and Practice of Logic Programming*, 1(03):283–301.

Damásio, C. V., Analyti, A., Antoniou, G., and Wagner, G. (2006). Open and closed world reasoning in the semantic web. In *Proceedings of IPMU 2006), special session Works on the Semantic Web*, pages 1850–1857, Paris, France. Editions E.D.K. Participaçãoo por convite e sujeita a avaliação.

Hindriks, K., De Boer, F., Van der Hoek, W., and Meyer, J. (1999a). Agent programming in 3APL. *Autonomous Agents and Multi-Agent Systems*, 2(4):357–401.

Hindriks, K., De Boer, F., Van Der Hoek, W., and Meyer, J. (1999b). Control structures of rule-based agent languages. In *Atal'98: Paris, France*, page 384. Springer.

Horrocks, I., Patel-Schneider, P., Boley, H., Tabet, S., Grosof, B., and Dean, M. (2004). SWRL: A semantic web rule language combining OWL and RuleML. *W3C Member Submission*, 21.

Katasonov, A. and Terziyan, V. (2008). Semantic agent programming language (S-APL): A middleware platform for the Semantic web. In *Proc. 2nd IEEE International Conference on Semantic Computing*, pages 504–511.

Luck, M., McBurney, P., and Preist, C. (2003). *Agent technology: Enabling next generation computing*. AgentLink II.

Rao, A. (1996). AgentSpeak (L): BDI agents speak out in a logical computable language. *Lecture Notes in Computer Science*, 1038:42–55.

Shoham, Y. (1991). AGENT0: A simple agent language and its interpreter. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, volume 2, pages 704–709.