Integrating Semantic Web and Software Agents: Exchanging RIF and BDI Rules

Yiwei Gong, Delft University of Technology, the Netherlands
Sietse Overbeek, Delft University of Technology, the Netherlands
Marijn Janssen, Delft University of Technology, the Netherlands

ABSTRACT

Software agents and rules are both used for creating flexibility. Exchanging rules between Semantic Web and agents can ensure consistency in rules and support easy updating and changing of rules. The Rule Interchange Format (RIF) is a new W3C recommendation Semantic Web standard for exchanging rules among disparate systems. Yet, the contribution of RIF in rules exchange between Semantic Web and software agents is unclear. The BDI architectural style is regarded as the predominant approach for the implementation of intelligent agents. This paper proposes a development for integrating RIF and BDI agents to enhance agent reasoning capabilities. This approach consists of an integration architecture and equivalence principles for rule translation. The equivalence principles are demonstrated using examples. The results show that the approach allows the integration of RIF with BDI agent programming and realize the translation between the two systems.

Keywords: AgentSpeak, BDI, Business Process Management, Business Rule, Flexibility, Information Systems, RIF

INTRODUCTION

Ten years ago, Berners-Lee et al. (2001) unveiled a nascent vision of the Semantic Web: a highly interconnected network of data that could be easily accessed and understood by any software agent. Those intelligent software agents would head out on the Internet and automatically perform tasks like booking flights and hotels, updating medical records and giving a single, customized answer to a particular question without the human user having to search for information and pore through results. Software agents have the promise to help users in dealing with information overload and perform tasks automatically in a flexible way. After a decade of developments in both Semantic Web and Software Agent technology, this vision has not yet come true. The challenge for researchers is the need to develop mechanisms to enable connections, which are between the web content and its end-users (Hendler & Berners-Lee, 2010). This implies the need to connect the Semantic Web providing the content and agents who are acting on behalf of the end-users.

DOI: 10.4018/jssoe.2011010104
From a technology perspective, the Semantic Web can be regarded as a group of technologies to allow machines to understand the meaning of information on the Web. Among those technologies the key ones related to knowledge representation and exchanges are the Resource Description Framework (RDF), the Web Ontology Language (OWL) and the Rule Interchange Format (RIF). The term agent, or software agent, has been widely used and has found its way into a number of technologies.

In artificial intelligence, an agent is an entity which can observe and act upon an environment and directs its activity towards achieving goals (Russell & Norvig, 2003). In agent research, the belief-desire-intention (BDI) architecture style is regarded as the predominant approach to the implementation of intelligent or rational agents (Wooldridge, 2000). There are many agent languages and tools supporting this architecture, e.g., PRS, dMARS, and AgentSpeak(L) (Mascardi, Demergasso, & Ancona, 2005).

Nowadays the ideas for integrating Semantic Web and Agent Programming techniques are blooming (Klapiscak & Bordini, 2009). Both domains can benefit from each other by connecting with the other. One of the advantages is that groups of intercommunicating agents are available to simulate complex actions on behalf of the end-users of the Web in real-life applications. (Kravari, Kastori, Bassiliades, & Governatori, 2010). In this way, the development on the Web and its performance can be tested by the agents. Another advantage is to allow the agent system to use existing resources (e.g., an ontology) in the Semantic Web. This integration refers to the interchange and use of rules to regulate the behaviors of agents or share knowledge among collaborating agents. The key to achieve this was indicated as “agents can only flourish when standards are well established” (Shadbolt, Hall, & Berners-Lee, 2006, p. 96). The Web standards for expressing a shared meaning have progressed steadily over the past years. The prospering of OWL in the academic community establishes the foundation of knowledge sharing by giving an open representation of the knowledge. This lets the Semantic Web technologies open up new applications to rule-based technology including agents. To enable the interoperation among rule-based applications on the Web, standards are needed to represent rules. Responding to these circumstances RIF has recently been promoted as a W3C standard for exchanging rules among different systems and developing intelligent rule-based applications for the Semantic Web.

The sharing of business rules will enable the reuse of business rules in different systems, in this way ensuring consistent use among diverse systems. Furthermore, by retrieving business rules from a repository the actuality of the business rules can be ensured, as new versions and updates are automatically taken. This makes the system more flexible in dealing with changes. Generally speaking, different rule systems use their own way to represent rules, and a translation component is required to share rule knowledge between two systems. For example, JASDL (Jason AgentSpeak–DescriptionLogic) is developed to integrate OWL and AgentSpeak (Klapiscak & Bordini, 2009). Without a common rule interchange format, the rule interchanging between $n$ systems in which each system uses different rule format, might require $n(n-1)$ translation components for bilateral translation. In this case, the most obvious advantage of using RIF is that it only requires $2n$ translation components in total. Since RIF is a W3C recommended standard, it is widely accepted and compatible with other Semantic Web technologies.

If rules in RIF format can be translated into executable rules for BDI agents, and vice versa, the facility of other Semantic Web technologies which are compatible with RIF, such as OWL and RDF, can be used in agent systems. In this way agents can make use of the rules described in OWL and RDF. This can result in an enhancement of the agents’ intelligence, like their reasoning capabilities, the versioning of rules and other advanced functions that are difficult to implement in software agents. In this way RIF can be used as the intermediating language between other Semantic Web languages and the BDI language. This idea is conceptualized in
In this paper we will explore an approach for the bilateral translation between RIF rules and executable BDI rules to enhance agent reasoning capabilities. In the next section we will present the benefits of using Semantic Web and Agent technology in the research of Business Rules. Thereafter, the literature background of RIF and BDI will be introduced. Then we will explain an integration architecture and the equivalence principles of translating RIF rules into BDI rules. Thereafter, examples are provided to demonstrate the translation and the feasibility of the approach. Finally, conclusions are drawn and future research plans are discussed.

BUSINESS RULES AMONG SEMANTIC WEB AND AGENTS

In the Business Process Management (BPM) domain, two dominant approaches can be observed: one approach based on graphical models, and the other based on rule specifications. Generally, the rule-based approach provides more flexibility to deal with changes in the environment (Lu & Sadiq, 2007). In a classical rule-based approach, the logic of a process is represented by a set of rules, which are associated with business activities and specify the properties of the process, e.g., the preconditions of its execution.

The creation of flexible business processes has increasingly received attention by organizations to satisfy customer demands and to be able to react to the competitive environment (Gong & Janssen, 2010). A business process can be viewed as the time-dependent sequence of activities. Business process flexibility is the ability to adapt to changes (Li & Zhao, 2006). For example, organizations reacting on changing customer needs or implementing legal requirements need to adapt their processes accordingly. As this might occur frequently, organizations need to make sure that their business processes and supporting applications are flexible enough to adapt to changing situations. Yet, current BPM applications often include business process specifications in the context of web services composition in a Service-Oriented Architecture (SOA) environment, e.g., the Business Process Execution Language (BPEL), and are not easy to change. Traditional BPM approaches use business process models as a kind of scheme to define the ‘recipe’ of a process execution (Pesic & van der Aalst, 2006). In contrast, flexible business processes imply the real-time orchestration of web services in order to construct dynamic business processes. In this way, customized services can be delivered dynamically in an integrated and personalized manner (Overbeek, Klievink, & Janssen, 2009). Business rules can be more easily changed than hard-coded processes and in this way create flexibility (Orriëns, Yang, & Papazoglou, 2003).

From an information system’s perspective, a business rule is a statement that defines or constrains some aspects of the business, and it is a directive intended to influence or guide business behavior (Ross, 2003). The current development of business rules standards enables to describe them using Semantic technology. Examples of those standards are the Semantics of Business Vocabulary and Business Rules (SBVR) and the Rule Interchange Format (RIF). The SBVR specification is published by the Object Manage-
ment Group (OMG) for the documentation and interchange of business vocabularies and business rules among organizations and software tools (OMG, 2008). RIF has become a World Wide Web Consortium (W3C) recommendation recently. It is a standard for exchanging rules among rule systems. As a new Semantic Web recommendation, a few implementations of RIF are still under development (W3C, 2010a). It is still unclear how RIF impacts applications at the operation level. Further research on applying RIF in practice, such as BPM, should provide insight in its advantages and usage.

Agent-based approaches have been used for a long time to improve business processes. Agents can take over tasks and interact with each other to create and perform a business process. Agent-based simulation is currently in widespread use to catch the behaviors of complex systems in which multiple entities are involved (Janssen & Verbraeck, 2005). Agents provide a way of structuring a complex system around autonomous and communicative components, and lead to the construction of software tools and infrastructures to support the design metaphor (Luck, McBurney, Shehory, & Willmott, 2005). In this way, Business Process Management design concepts can be tested. In current BPM research, agent-based simulation and BDI agents are employed for testing the design solution for the creation of flexible business processes (Gong & Janssen, 2010; Lu & Sadiq, 2007). Comparing with other agent systems like JADE (Java Agent DEvelopment framework), which has a strong capability in terms of inter-agent communication, BDI is a sound foundation for modeling intra-agent behaviors. Therefore, it is more suitable for observing the impact of rule implementation on single agent behaviors and how this contributes to creating business processes.

Connecting Semantic Web and agents by allowing rule exchange can benefit rule-based applications, like the implementation of business rule systems. For example, using OWL can present the ontology of a domain which provides a vocabulary of terms in the domain and allows agents to perform inference among the terms. In this way, different agents are able to share the same ontology models and hold consistent understanding of the terms. When business rules are presented using the given terms in ontology models, agents can infer according to the models and to execute the business logic of the rules.

BACKGROUND

To translate rules between RIF and BDI agents, the syntaxes in both systems are important. In this section the background and syntax of the AgentSpeak language and RIF is introduced.

BDI Architecture and AgentSpeak Language

The BDI architecture is regarded as the predominant approach to the implementation of intelligent or rational agents (Wooldridge, 2000). As its name suggests, it contains the following main components: Beliefs - the agent’s knowledge about the world; Desires - the objectives to be accomplished; and Intentions - the courses of actions currently under execution to achieve the agent’s desires. AgentSpeak is an agent-oriented programming language based on logic programming, and inspired by the work on the BDI architecture (Bordini, & Hübner, 2006). Although a few other agent languages are also based on BDI, we choose to use AgentSpeak as it has very neat and elegant notations, and it complies better to the BDI architecture than other BDI-inspired programming languages (Bordini & Moreira, 2002). Jayson is an interpreter for an extended version of AgentSpeak. Jason implements the operational semantics of the AgentSpeak language, and provides a platform for the development of multi-agent systems with many user-customizable features (Hübner & Bordini, 2009). We employed the AgentSpeak language to present BDI. Due to space limitations, we only briefly introduce basic terms, formulas and necessary operators of AgentSpeak. For more details concerning
the syntax and semantics of AgentSpeak, we refer to Moreira and Bordini’s work (Moreira & Bordini, 2002).

Terms in AgentSpeak (Jason implementation) include: constant, variable and structure. Under the type of structure, the list is a special form. Those concepts are very common in logic programming. The terms are given in Figure 2.

An AgentSpeak agent is simply a specification containing a set of beliefs and a set of plans. A belief is an atomic (at) formula without variables (ground formula). The atomic formulas of the language are predicates given by the grammar in (2), where P is a predicate symbol and \( t_1, ..., t_n \) are standard terms of first-order logic. The agent has an initial belief about the environment where it is situated. When the environment changes, the set of beliefs changes accordingly. This involves events which can trigger the execution of plans. A plan (p) contains a triggering event (te), a context (ct) which is about the conditions that have to be hold for the plan to be applicable, and a sequence h of actions (a), goals (g) and updating of beliefs (u) to be executed. As context is the reasoning part of a plan, which determines whether the plan will be selected to execute, rules with the form of “Left: Right” can be used. In a rule, the left of the “::=” operator is true, if the right is satisfied. The syntax of AgentSpeak mentioned above is formally presented in the following list.

\[
\begin{align*}
\text{agent} &::= \text{beliefs plans} \\
\text{beliefs} &::= \text{at}_1, \ldots, \text{at}_n (n \geq 0) \\
\text{plans} &::= p_1 \ldots p_n (n \geq 1) \\
p &::= \text{te}: \text{ct} \leftarrow h \\
h &::= a | g | u | h \; ; \; h
\end{align*}
\]

A triggering event can be the addition (+) or deletion (-) of a belief, otherwise referred to as the addition or deletion of a goal. A goal can be an achievement goal (!at), which has to do with the execution of sub-plans, or a test goal (?ct) which is used to instantiate variables from unification with current beliefs of the agent when the plan is being executed.

\[
\begin{align*}
\text{te} &::= +\text{at} | -\text{at} | +\text{g} | -\text{g} \\
\text{g} &::= !\text{at} | ?\text{ct} \\
\text{u} &::= +\text{at} | -\text{at}
\end{align*}
\]

An AgentSpeak behaves in a reactive planning paradigm. Plan execution is triggered by an event due to perception of the environment, or due to the addition or deletion of goals as

---

Figure 2. Terms in AgentSpeak (adapted from Bordini, Hübner, & Wooldridge, 2007, p. 33)
a result of the execution of plans triggered by previous events. Agents are not just perceiving the environment, but also able to change the environment by perform actual behaviors besides logical reasoning. In multi-agent systems, environment is shared by different agents, and so that the action of an agent can interfere other agents by changing the environment. In the Jason implementation, a simulated environment can be programmed in Java and work with an AgentSpeak agent. An environment class provided by Jason can be extended by programmers to define a customized simulated environment. A user-defined class implementing an environment typically looks as seen in Box 1 (Bordini et al., 2007).

In the above code, `<EnvironmentName>` is the name of the environment class to be specified by the programmer. The `init` method is used to initialize the lists of percepts with the properties of the environment. In this example, those percepts are created using the `parseLiteral` method of the `Literal` class. This method adds literal `P(a)` to the global list of percepts. In this way, all agents will perceive `P(a)` if they sense the environment. After this, `P(a)` can be used in the agents’ code as a belief. Once `P(a)` is invoked, the action defined in the `executeAction` method and referenced at `P(a)` will be attempted to execute.

### Rules and RIF

Generally, rules can be classified as three types: deductive rules (or derivation rules), normative rules (or integrity rules), and reactive rules (or active rules) (Hu, Yeh, & Laun, 2009). Deductive rules are used to derive implicit facts. Normative rules pose constraints on the data or on the business logic and are rules of the form “it must be true that ...”. Reactive rules are further subdivided into event-condition-action (ECA) rules and production rules. ECA rules are rules of the form “ON Event IF Condition DO Action”. Production rules are rules of the form “IF Condition DO Action”.

RIF aims to become a standard for exchanging rules among disparate systems, especially on the Semantic Web. Due to the great diversity of rule languages, there is not an all-in-one language to bridge them all. To make it feasible, the RIF Working Group designed a family of languages, called dialects, with rigorously specified syntax and semantics. Central to the idea behind rule exchange through RIF is that “different systems will provide syntactic mappings from their native languages to RIF dialects and back” (W3C, 2010f). Consequently, these mappings are required to be “semantics-preserving”, and rule sets can be communicated between systems with a precondition that the
systems can talk through a suitable dialect, which they both support. The current RIF standard is focused on two kinds of dialects: logic-based dialects and dialects for rules with actions. Generally, logic-based dialects include languages based on first-order logic (often restricted to Horn logic (Horn, 1951)) and also non-first-order logics underlying the various logic programming languages, e.g., stable model semantics (Niemelä, 1999). The rules-with-actions dialects are designed for production rule systems, such as Jess, Drools and JRules, as well as for reactive (or ECA) rules (W3C, 2010f). Accordingly, there are two logic dialects: Basic Logic Dialect (RIF-BLD) (W3C, 2010b) and Production Rule Dialect (RIF-PRD) (W3C, 2010g). Besides that, there is a subset, the RIF Core Dialect (W3C, 2010c), shared by them and an extensibility framework, called the Framework for Logic Dialects (RIF-FLD) (W3C, 2010e) has been developed. The family of RIF dialects is showed by a Venn diagram in Figure 3.

From a theoretical perspective, RIF-BLD corresponds to the definite Horn rules language with equality and standard first-order semantics (W3C, 2010b). The compatibility of RIF-BLD/RDF and RIF-BLD/OWL languages makes RIF-BLD a Web-aware language. Thus, RIF-BLD is essential in integrating Semantic Web and an Agent Programming Technique. However, the rules in the agent plans which the agent uses to react to its environment are production rules. Therefore, RIF-PRD is also needed. The necessity of both RIF-BLD and RIF-PRD in integrating Semantic Web and an Agent Programming Technique leads us to RIF-Core. RIF-Core is intended to be the common core of all RIF dialects. It has been designed to be a useful common subset of RIF-BLD and RIF-PRD.

In the RIF-Core syntax, terms can be defined as the following: (The phrase base term refers to a simple or positional term, or terms of the form External(t), where t is a positional term)

1. **Constants and variables**: if \( t \in \text{Const} \) or \( t \in \text{Var} \) then \( t \) is a simple term
2. **Positional terms**: if \( t \in \text{Const} \) and \( t_1, \ldots, t_n, n \geq 0 \), are base terms then \( (t_1, \ldots, t_n) \) is a positional term. (Positional terms correspond to the usual terms and atomic formulas of classical first-order logic.)
3. **List term**: a closed ground list has the form \( \text{List}(t_1, \ldots, t_m) \), where \( m \geq 0 \) and \( t_1, \ldots, t_m \) are ground terms (no tail and no variable).
4. **Equality terms**: has the form \( t = s \), if \( t \) and \( s \) are base terms.
5. **Class membership terms**: has the form \( t \# s \), if \( t \) and \( s \) are base terms.
6. **Frame terms**: \([p_1 \rightarrow v_1, \ldots, p_n \rightarrow v_n] \) is a frame term if \( t, p_1, \ldots, p_n, v_1, \ldots, v_n, n \geq 0 \), are base terms. The term \( t \) is the object of the frame; the \( p_i \) are the property or attribute names; and the \( v_i \) are the property or attribute values.
7. **Externally defined terms**: if \( t \) is a positional term then \( \text{External}(t) \) is an externally defined term.

Based on the terms described above, the definition of main formulas in the RIF-Core can be given as the following:

---

Copyright © 2011, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.
Definition 1. Any positional term of the form $p(...)$, where $p$ is a predicate symbol, is an atomic formula. Equality, membership, and frame terms, as well as an externally defined term of the form $\text{External}(\phi)$, where $\phi$ is an atomic formula, are also atomic formulas.

Definition 2. A formula can have several different forms and is defined as follows:
1. Atomic: If $\phi$ is an atomic formula then it is also a formula.
2. A condition formula: either an atomic formula or a formula that has one of the following forms:
   - **Conjunction**: If $\phi_1$, ..., $\phi_n$, $n \geq 0$, are condition formulas then so is $\text{And}(\phi_1, ..., \phi_n)$.
   - **Disjunction**: If $\phi_1$, ..., $\phi_n$, $n \geq 0$, are condition formulas then so is $\text{Or}(\phi_1, ..., \phi_n)$.
   - **Existentials**: If $\phi$ is a condition formula and $?V_1, ..., ?V_n$, $n > 0$, are distinct variables then $\text{Exists } ?V_1, ..., ?V_n(\phi)$ is an existential formula.
3. **Rule implication**: $\phi:- \psi$ is a rule implication if:
   - $\phi$ is an atomic formula or a conjunction of atomic formulas,
   - $\psi$ is a condition formula, and
   - none of the atomic formulas in $\phi$ is an externally defined term.
4. **Universal rule**: If $\phi$ is a rule implication and $?V_1, ..., ?V_n$, $n > 0$, are distinct variables then $\text{Forall } ?V_1, ..., ?V_n(\phi)$ is a formula called a universal rule.
5. **Universal fact**: If $\phi$ is an atomic formula and $?V_1, ..., ?V_n$, $n > 0$, are distinct variables then $\text{Forall } ?V_1, ..., ?V_n(\phi)$ is a formula, called a universal fact. Universal facts are often considered to be rules without premises.
6. **Group**: If $\phi_1$, ..., $\phi_n$ are RIF-Core rules, universal facts, variable-free rule implications, variable-free atomic formulas, or group formulas then $\text{Group}(\phi_1, ..., \phi_n)$ is a group formula.

We consider that the normative (mathematical) presentation of RIF-Core in W3C’s recommendation is not strict, as the description of RIF-Core is based on exclusion of components in RIF-BLD and RIF-PRD Syntax. Thus, RIF-Core is explained as two specializations instead of a whole and well-defined presentation language. W3C also comments that the presentation syntax is not intended to be a concrete syntax for RIF-Core (W3C, 2010c). The above lists of terms and formulas are an incomplete summarization of RIF-Core Syntax. It is notable that similar notations that are defined as terms in RIF-BLD are defined as formulas in RIF-PRD. Since they are all involved in the RIF-Core, we did not clearly distinguish terms from formulas.

Production rules have a ‘condition’ part, and an ‘action’ part. The condition rules are covered by RIF-Core and its extension RIF-BLD. The action part, which is essential in agent planning, is, however, not contained in RIF-Core. An action can assert facts, modify facts, and retract facts. In general, an action is different from the conclusion of a logic rule, which contains only a logical statement. The conclusion of rules interchanged using RIF-Core can be interpreted, according to RIF-PRD operational semantics, as actions that assert facts in the knowledge base. We consider that the other way around is not valid. Therefore, our discussion should not be limited to the RIF-Core language, but also concern RIF-PRD with a focus on action rules.

Actions in RIF-PRD include atomic actions or compound actions. An atomic action is a construct that represents an atomic transaction in the following forms:

1. **Assert fact**: If $\phi$ is a positional atom, a frame or a membership atomic formula in the RIF-PRD condition language, then $\text{Assert}(\phi)$ is an atomic action.
2. **Retract fact**: If $\phi$ is a positional atom or a frame in the RIF-PRD condition language, then $\text{Retract}(\phi)$ is an atomic action.
3. **Retract all slot values**: If \( o \) and \( s \) are terms in the RIF-PRD condition language, then \( \text{Retract}(o \ s) \) is an atomic action.

4. **Retract object**: If \( t \) is a term in the RIF-PRD condition language, then \( \text{Retract}(t) \) is an atomic action. \( t \) is called the target of the action.

5. **Execute**: if \( \varphi \) is a positional atom in the RIF-PRD condition language, then \( \text{Execute}(\varphi) \) is an atomic action.

A **compound action** is a construct that can be replaced equivalently by a pre-defined, and fixed sequence of atomic actions. There is only one compound action in RIF-PRD, which is defined as follows:

1. **Modify fact**: if \( \varphi \) is a frame in the RIF-PRD condition language: \( \varphi = o[s->v] \); then \( \text{Modify}(\varphi) \) is a compound action, defined by the sequence: \( \text{Retract}(o \ s) \) and \( \text{Assert}(\varphi) \).

The advantage of integrating Semantic Web and Agent Programming technology is to allow the agent system to use existing resources in Semantic Web. RIF is a facilitator of such an integration as it functions as a standard in knowledge interchange.

### INTEGRATING RIF RULES AND BDI AGENT PROGRAMMING

For integrating RIF and BDI agent programming, three steps are proposed. First of all, it must be identified from a semantic perspective what corresponding context should be translated between two systems. Then, it is necessary to distinguish the directions of the translation: which is translated from which system to which. Finally, from a syntactical perspective it should be determined whether such a translation is feasible and which limitations or preconditions exist.

The starting point to investigate this is the basic agent model provided by AgentSpeak. In the AgentSpeak language an agent contains a set of beliefs and a set of plans. In the interaction with the environment, the agent will sense changes in the environment and update its base of belief. Then, an appropriate plan is determined from its plan depository via reasoning. Actions will subsequently be carried out according to the selected plan. The action will impact the environment and will result in further interaction between the agent and the environment. This is conceptualized in Figure 4, where the arrows visualize the information flow (data or messages); the rectangles represent the agent behaviours (either percept or action); and the parallelograms refer to the depository of belief or plan.

By the combination with Semantic Web technology, the reasoning will not be just limited to the inside of the agent, but it is also possible to be done by external components. On the one hand, just like doing the reasoning internally, the perception of the environment needs to be transferred to the component which performs the reasoning functions. Therefore, the agent’s beliefs are the target information in translation. On the other hand, the result of reasoning should be carried out by the agent itself. Agents will behave according to their plans. Therefore, the plans of an agent are another target of the translation. Through the above analysis, the direction of the translation can also be distinguished as agent beliefs that should be translated into RIF facts, while RIF rules should be translated into agent plans. The cycle of external reasoning with the direction of translation is conceptualized as an architecture in Figure 5. Based on certain technology which is compatible with RIF, the cycle can be extended.

For the final step, mapping between the AgentSpeak syntax and the RIF syntax is required. The syntaxes of each system both contain terms and formulas, where the latter is composited by the former. Although the implication of the word “term” can be different in AgentSpeak and RIF-Core (e.g., RIF-Core consider “\( a = b \)” as a term, but AgentSpeak considers “\( = \)” as an operator), most terms in both syntax are matchable due to their similar-
In the mapping of terms, the unmatchable items, namely class membership term and externally defined term indicate the disability of directly translating these terms from RIF to AgentSpeak. As membership is used to describe objects and class hierarchies, it can be concluded that the rules related to hierarchies relationship are difficult to translate from RIF-Core format to AgentSpeak format. At the same time, the list term in AgentSpeak has more variety than the list term in RIF-Core. Hence, some lists in AgentSpeak are impossible to be directly translated into RIF-Core lists.

Positional terms and frame terms in RIF-Core can be matched to the structure terms in AgentSpeak, as the structure terms are able to represent complex data and consequently fulfilling the functions of positional terms and frame terms. Nevertheless, the inability of having a direct translation does not mean that it will be impossible to have a translation at all. The RIF-Core allows external terms which are used for representing built-in functions and predicates as well as “procedurally attached” terms or predicates, which might exist in various rule-based systems, but are not specified by RIF (W3C, 2010b). AgentSpeak in Jason implementation can also accept external defined literals. This requires customized definitions of the literals in the simulated environment which is programmed to support Jason agents. Therefore, the translation of externally defined terms in RIF-Core to AgentSpeak is possible but not directly and is more complicated. This will be demonstrated by Example 2 in the next section.

Based on the above analysis on terms, the mapping on formulas can be done, as a formula can be considered as a combination of terms and maybe operators. To summarize this, we use a logic symbol $\iff$ (logical equivalence) to represent equivalence of the two syntaxes. So, if $p \iff q$, it means presentation $p$ is function-
ally equivalent to presentation $q$. We put RIF syntax which is introduced in section 2.2 on the left of the equivalence, and AgentSpeak syntax which is introduced in section 2.1 on the right. This results in the following list of equivalence principles to facilitate the translation:

$t \in \text{Const}$: $\Leftrightarrow x \in \text{constant}$

$t \in \text{Var}$: $\Leftrightarrow x \in \text{variable}$

$t(t_1, ..., t_n)$: $\Leftrightarrow \text{at}(t_1, ..., t_n)$ (n$\geq$0)

$\text{List}(t_1, ..., t_n)$: $\Leftrightarrow \text{list}(t_1, ..., t_n)$ (n$\geq$0)

Equality term $\Leftrightarrow$ combination of AgentSpeak terms and the operator “=”

Positional term $\Leftrightarrow$ (structure term)

Class membership term $\Leftrightarrow$ unmatchable

Frame term $\Leftrightarrow$ (structure term)

Externally defined term $\Leftrightarrow$ requires customized environment support

The above equivalence principles show that many of the terms and formulas can be matched between RIF and AgentSpeak. Nevertheless, some formulas in RIF are not applicable in AgentSpeak. For example, membership terms (and also subclass terms, which are excluded in RIF-Core but used in both RIF-BLD and RIF-PRD) are used to describe objects and class hierarchies. AgentSpeak is not an object-oriented language, and as such those terms are out of scope in the mapping. At the same time, it can be concluded that RIF-BLD is extensible. It might not be able to translate all syntax into the AgentSpeak language. Therefore, focusing on RIF-Core is reasonable.

The limitations of our approach are also obvious. Firstly, agent behaviors contain operations, such as sending messages, that go beyond logical reasoning. The integration with the RIF standard can be considered as an enhancement of reasoning capabilities but is not a replacement of agent actions. The use might be limited, unless they are used by agents. Secondly, the compatibility between the systems requires a more elaborated evaluation. As RIF is extensible, there is no guarantee that future extensions are compatible with BDI agents. Last but not least, the RIF dialects have differences in syntax and applications. Selecting which dialect to use for the integration requires further consideration of the pros and cons. For

<table>
<thead>
<tr>
<th>Terms</th>
<th>AgentSpeak</th>
<th>RIF-Core</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constant::number</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>Constant::string</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>Variable</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>Structure::atom</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>Structure::list</td>
<td>√</td>
<td>only closed ground list</td>
</tr>
<tr>
<td>Equality term</td>
<td>√ (combination of AgentSpeak terms and the operator “=”)</td>
<td>√</td>
</tr>
<tr>
<td>Positional term</td>
<td>√ (structure term)</td>
<td>√</td>
</tr>
<tr>
<td>Class membership term</td>
<td>unmatchable</td>
<td>√</td>
</tr>
<tr>
<td>Frame term</td>
<td>√ (structure term)</td>
<td>√</td>
</tr>
<tr>
<td>Externally defined term</td>
<td>requires customized environment support</td>
<td>√</td>
</tr>
</tbody>
</table>
example, agents can use negation logic for their beliefs. RIF-PRD supports negation, but RIF-Core does not. However, RIF-Core holds the connection with RIF-BLD and might provide better compatibility with other Semantic Web technologies (currently, the interoperability with RDF, RDFS and OWL is provided).

**RULE TRANSLATION ILLUSTRATIONS AND DISCUSSION**

To demonstrate the feasibility of the rule translation, we will use examples of RIF presentations provided by W3C recommendation documents in this section. Those examples refer to RIF-Core, RIF-BLD, and RIF-PRD respectively. The examples will be translated into the AgentSpeak language (Jason implementation) based on the equivalence principles presented in the previous section.

The first example is using the RIF-Core syntax to present the interactions between a book seller and a buyer (W3C, 2010c). This example provides a reasoning to derive the buy relationship from the sell relationship. In natural language, it can be represented as “a buyer buys an item from a seller if the seller sells the item to the buyer”. Moreover, this example includes a fact that “John sells LeRif to Mary”. The two phrases can be represented in RIF-Core presentation syntax as follows.

In this example, “?Buyer”, “?Item” and “?Seller” are variables and “John”, “LeRif” and “Mary” are constants, while “sell” and “buy” are predicates. At the beginning of the example, there are three shorthand notations for internationalized resource identifiers (IRIs) obtained by concatenation of the prefix definition and suffix. For example, if bks is a prefix that expands into http://example.com/books#, then bks:LeRif is an abbreviation for “http://example.com/books#LeRif”^^rif:iri. Those prefix and suffix directives are defined in RIF-Interchange Format (W3C, 2010d). RIF-DTB is supported by but not involved in RIF dialects. We consider the use of those symbols has an intention on documentation. A limitation is that those documentation symbols cannot be translated into the AgentSpeak language, because they are meaningless for the AgentSpeak language. The AgentSpeak representation of this example is given as follows:

```
buy(Buyer, Item, Seller):-
sell(Seller, Item, Buyer) +sell(john, lerif, mary)
```

From this example, we can observe that not just the built-ins symbols (e.g., Prefix) but also set related syntax (e.g., Group formula) and the head of the universal rule (the “For all” part) are hardly consistent with other content after translation, as no comparable syntax can be found in the AgentSpeak language. The missing of RIF document symbols or group formulas or the head of the universal rule is not harmful for the AgentSpeak agent since they are not required by an agent implementation. However, if the translation direction is the other way around, namely from AgentSpeak to RIF, the complementarity of the missing elements is important. Some elements, such as IRI is an integral part of a RIF document. Therefore, when translating agent beliefs into RIF facts, missing syntax should be complemented. This requires additional research on AgentSpeak syntax and RIF-DTB. A potential solution of this problem is building namespace management into Jason (Madden & Logan, 2009). In this way, Jason can have a syntax comparable with the documentation symbols in RIF.

The second example shows a business rule that “if an item is perishable and it is delivered to John more than 10 days after the scheduled delivery date then the item will be rejected by him” (W3C, 2010b). For space reasons, we will skip the head of the document where some prefixes are presented, as they have been explained and discussed in the first example.
The RIF-BLD representation of this example is given as follows.

In the above code, the external built-in functions are nested and the (intermediate) result (sets) from functions are assigned to variables by equality. The AgentSpeak presentation of this example is given in Box 2.

In the translation of this example, we consider the environment functions, which are external to the AgentSpeak agent but can be perceived by the agent, as the target when translating the External formula in RIF. However, it is very arguable that whether this can still be called a translation. When using the simulated environment coded in Java to support customized literals in Jason, it does not just result in some AgentSpeak code, but also Java code. On the one hand, involving other languages and relying on specific functions of the implementation tool (Jason) is already out of the scope in our discussion of the equivalence principles. On the other hand, the openness of RIF enables the use of externally defined terms. At the same time, the openness of Jason also allows the use of the Java environment to enable externally defined terms. Since a RIF document can involve a content that is not described by its own syntax, the equivalence principles are not applicable for this content. This example shows that a “translation” of externally defined terms can still be provided, but not based on the equivalence principles.

In the third example, we focus on action rules in RIF-PRD. The context of this example can be found in (W3C, 2010g). As Modify action is a compound of the Assert and Retract actions, we only introduce a segment which contains Assert, Retract and Execute actions. The RIF-PRD presentation is given as follows:

The translation of action rules to AgentSpeak is significantly easier, as they have higher comparability. The AgentSpeak presentation is given as follows.

+customer(voucher(Voucher))
-customer(voucher(_))
.print(“Hello, world!”)

In the translation, the namespace format like “ex1:voucher” is not acceptable for AgentSpeak grammar, and therefore the prefix and its colon is removed.

The three illustrations demonstrate the translation from RIF-Core, RIF-BLD and RIF-PRD representations to AgentSpeak language. They reveal the feasibility of integrating RIF dialects and BDI agent programming. Under certain restrictions, RIF can be used as the intermediating language between Semantic Web languages and the BDI language.

CONCLUSIONS AND FUTURE RESEARCH

This paper shows the use of RIF to exchange rules between the Semantic Web and agents. An approach for integrating RIF and BDI agent programming was presented. This approach consists of an architecture integrating RIF and
DBI agents and equivalence principles for rule translation. Three examples are provided to demonstrate the feasibility of rule translation. The integration is expected to enhance the agent reasoning capability by using existing resources in RIF compatible technologies. This can be used to enhance agents reasoning capabilities by letting agents access Semantic

Example 2.

    cpt:reject(<John> ?item):-
    And(cpt:perishable(?item)
        cpt:delivered(?item ?deliverydate <John>)
        cpt:scheduled(?item ?scheduledate)
        ?diffduration = External(func:subtract-dateTimes(?deliverydate ?scheduledate))
        ?diffdays = External(func:days-from-duration(?diffduration))
        External(pred:numeric-greater-than(?diffdays 10)))
)

Box 2.

reject(john, Item) :-
    perishable(Item) &
    delivered(Item, Deliverydate)&
    scheduled(Item, Scheduledate)&
    Diffduration = subtract-dateTimes(Deliverydate, Scheduledate)&
    Diffdays = days-from-duration(Diffduration)&
    numeric-greater-than(Diffdays, 10).

//implement customized environment in java
import jason.asSyntax.*; import jason.environment.*;
public class DeliveryEnvir extends jason.environment.Environment {
    Literal x = Literal.parseLiteral("subtract-dateTimes(deliverydate, scheduledate)" );
    Literal y = Literal.parseLiteral("days-from-duration (diffduration) ");
    Literal z = Literal.parseLiteral("numeric-greater-than (source,target) ");
    @Override
    Public void init(String[] args){
        addPercept(x);
        addPercept(y);
        addPercept(z);
    }
    @Override
    public Boolean executeAction (String ag, Structure act) {
        //add functions refer to x, y and z
    }
}
...
Web resources. Such an integration can benefit the business rule implementation of creating flexible business processes.

The results indicate that large parts of RIF and BDI syntaxes can be translated into each other. Ideally, software components can be developed to automatically translate between the two languages. However, the study also shows that a complete translation is often not possible as both languages have different syntactical elements. The main unmatchable syntaxes are about class membership term, externally defined term and documentation formulas. This means that not all the terms and formulas can be translated in a straightforward manner. Therefore, an elaborated translation approach with precise syntax translation which complements the missing parts is required. This research has already indicated the difficulty in translating unmatchable terms and formulas. This study only explored the integration of RIF and BDI agents, but other agent frameworks (e.g., Brahms agents) might also be integrated with RIF. There will be less implementation dependencies, if the approach can be extended to various agent frameworks.

In further research, we will involve OWL technology into the software agents via RIF as an intermediate. Ontology models related to Dutch migration law and regulations will be built, and the ontology will be represented using OWL. Based on a case study, agent simulations will be employed to simulate different scenarios. The agent’s capability of reasoning is expected to be enhanced by using the ontology models, which ensures that all agents use the same data source in this way avoiding inconsistency. In case the law is changed, new rules can be derived easily from the new law in the ontology model and used by the agents, in this way bringing higher levels of flexibility. If this can be proved to be feasible, the integration of Semantic Web and software agent technologies will not just contribute to Internet applications which mentioned by Berners-Lee et al., but also to knowledge intensive organizations such as government departments and financial companies. As knowledge intensive organizations have the demand and intension to use Semantic technologies to represent their knowledge and Business Rule technology to operate their daily tasks, this approach can ensure consistency in their rules and support more easy updating and changing of the rules.

ACKNOWLEDGMENT

This work is supported by the AGILE project (Advanced Governance of Information services through Legal Engineering, details can be found on the web page http://www.jacquard.nl/?m=426).

REFERENCES


Yiwei Gong is a PhD candidate in the Information and Communication Technology section of the Faculty of Technology, Policy, and Management at Delft University of Technology, the Netherlands. His research interests include business process management systems, agent-based simulation and information architecture. Gong received his MSc in computer science from the Delft University of Technology.

Sietse Overbeek is an assistant professor in the Information and Communication Technology section of the Faculty of Technology, Policy, and Management at Delft University of Technology, the Netherlands. His research interests include conceptual and formal modeling of information systems, ontology modeling, and service-oriented architectures. Overbeek received his PhD in computer science from the Radboud University Nijmegen, the Netherlands. He is a member of the ACM.

Marijn Janssen is an associate professor in the Information and Communication Technology section of the Faculty of Technology, Policy, and Management at Delft University of Technology, the Netherlands. His research interests include business engineering, information integration, agent-based and service-oriented architectures, and designing the coordination of networked public and private organizations. He is also the director of the interdisciplinary SEPAM Master program. Janssen received his PhD in information systems from the Delft University of Technology.