4th International Conference on Web Reasoning and Rule Systems – RR 2010

Brixen, Italy

22-24 September 2010

Co-located RR 2010 Workshop

BuRO 2010: 1st International Workshop on Business Models, Business Rules and Ontologies

21 September 2010

Proceedings

Editors:

T. Eiter, A. El Ghali, S. Fernàndez, S. Heymans, T. Krennwallner, F. Lévy

Preface

This volume contains the papers as well as the invited talks presented at the 1st international workshop on *Business Models, Business Rules and Ontologies (BuRO 2010)* held on the 21st of September 2010 in Brixen, Italy, co-located with the 4^{th} International Conference on Web Reasoning and Rule Systems (RR 2010).

It is a challenge in a business to enable the right people to interact in their own way with the right part of their business application. We distinguish between three views on the business organization: (1) the view of the business analyst using a formal and validated business model; (2) the view of the knowledge engineer via ontologies and rules, and (3) the view of the IT department via an operationalization in applications. We can glue these views together via an end-to-end point solution: (1) conceptualization and where possible acquisition of business models and their transformation into ontologies and rules; (2) their management and maintenance, and (3) the transparent operationalization in IT applications.

The vision at the heart of the Semantic Web is of high relevance in a business setting as well. The proposed workshop addresses the different issues that arise in a business that wishes to have a transparent and where possible and useful a semi-automatic transfer of knowledge present in business documents expressing, e.g., policies, to an IT operationalization. Moreover, the workshop tackles these issues from an holistic perspective, raising awareness for the overall picture, instead of focusing on stand-alone issues. E.g., although OWL is well-investigated it is unclear how business knowledge expressed in SBVR can be mapped to it. Another example is the W3C's RIF effort: although based on well-investigated rule paradigms, it is less well-connected to upper business layers: how to go from a formal business model to RIF rules and how to interact with derived ontologies?

During the ONTORULE project which shares a similar vision on a business, it has been recognized that this holistic view goes beyond the results attainable within the project and that much more discussion and exchange is needed. As such the workshop wants to create awareness with researchers in stand-alone fields like ontology acquisition, business modeling, integration of ontologies and rules, implementations of rule/ontology engines, that there is a bigger picture that can and should be used to extract requirements on the one hand and to provide output that is fine-tuned for other fields on the other hand.

Some topics of interest are:

- the acquisition of ontologies and rules from unstructured text via Natural Language Processing (NLP) techniques
- the development of a complete, formal and validated business model, taking all possible inputs into account (people and documents, structured and

unstructured, some of which as output from an NLP phase), using the Semantics of Business Vocabulary and Business Rules (SBVR)
– transformation from structured business representations, from SBVR, to RDF/OWL and/or rules
– the management and maintenance of business models, ontologies and rules, e.g., consistency maintenance and the integration of rules and ontologies (semantics, algorithms)
– implementations of such management systems
– use cases and field reports

October 2010 The Editors

Workshop Organization

Organizing Committee

Thomas Eiter, TU Vienna, Austria
Adil El Ghali, IBM, France
Sergio Fernàndez, Fundación CTIC, Spain
Stijn Heymans, TU Vienna, Austria
Thomas Krennwallner, TU Vienna, Austria
François Lévy, Université Paris 13, France

Programme Committee

Patrick Albert, IBM, France
Darko Anicic, FZI, Germany
Christopher Brewster, Aston University, UK
Jordi Cabot, Universit de Nantes, France
Jean Charlet, INSERM, France
Michael Erdmann, ontoprise GmbH, Germany
Bernardo Cuenca Grau, University of Oxford, UK
Stephan Grimm, FZI, Germany
Pascal Hitzler, Wright State University, USA
Giovambattista Ianni, Universita della Calabria, Italy
Thomas Krennwallner, Vienna University of Technology, Austria
Markus Kroetzsch, AIFB, University of Karlsruhe, Germany
Yue Ma, LIPN, Univ. Paris 13, France
Diana Maynard, University of Sheffield, UK
Adeline Nazarenko, Univ. Paris 13, France
Sjir Nijssen, PNA University
Adrian Paschke, Free University Berlin, Germany
Maria Theresa Pazienza, Univ. Tor Vergata, Roma, Italy
Luis Polo, Fundacion CTIC, Spain
Edna Ruckhaus, Universidad Simon Bolivar Caracas, Venezuela
Sebastian Rudolph, AIFB, University of Karlsruhe, Germany
Jos de Bruijn, Vienna University of Technology, Austria

Table of Contents

Full Papers

Invited Talks

DReW: a Reasoner for Datalog-rewritable Description Logics and DL-Programs*

Guohui Xiao, Stijn Heymans, and Thomas Eiter

Institute of Information Systems 184/3
Vienna University of Technology
Favoritenstraße 9–11, A–1040 Vienna, Austria
{xiao,heymans,eiter}@kr.tuwien.ac.at

Abstract. Nonmonotonic dl-programs provide a loose integration of Description Logic (DL) ontologies and Logic Programming (LP) rules with negation, where a rule engine can query an ontology with a native DL reasoner. However, even for tractable dl-programs, the overhead of an external DL reasoner might be considerable. Datalog-*rewritable* DL ontologies, such as $\mathcal{EDL}^+$, can be rewritten to Datalog programs, such that dl-programs can be reduced to Datalog$^\neg$, i.e, Datalog with negation, under well-founded semantics. We developed the reasoner DReW that uses the Datalog-rewriting technique. DReW can as such answer conjunctive queries over $\mathcal{EDL}^+$ ontologies, as well as reason on dl-programs over $\mathcal{EDL}^+$ ontologies under well-founded semantics. The preliminary but encouraging experimental results show that DReW can efficiently handle large knowledge bases.

1 Introduction

As the envisioned basis of future information systems, the Semantic Web is a fertile ground for deploying AI techniques, and in turn raises new research problems in AI. As a prominent example, the combination of rules with Description Logics (DLs), which is central to the Semantic Web architecture, has received high attention over the past years, with approaches like *Description Logic Programs* [10], *DL-safe rules* [21], *r-hybrid KBs* [24], $\mathcal{DL}$+*log* [25], *MKNF KBs* [20], *Description Logic Rules and ELP* [15, 16], and *dl-programs* [5].

Nonmonotonic dl-programs provide a loose integration of Description Logic (DL) ontologies and Logic Programming (LP) rules with negation, where a rule engine can query an ontology using a native DL reasoner. For dl-programs over tractable DL ontologies under well-founded semantics, the reasoning problem is tractable [6]. However, even for tractable dl-programs, the overhead of an external DL reasoner might be considerable.

In [13], Datalog-rewritability was proposed to remedy the overload of calling external DL reasoners. A Datalog-rewritable ontology can be polynomially rewritten to a Datalog program. Moreover, dl-programs over such Datalog-rewritable ontologies

* This work is partially supported by the Austrian Science Fund (FWF) projects P20305 and P20840, and by the EC FP7 project OntoRule (IST-2009-231875).

can then be reduced to Datalog$^\neg$— Datalog with negation — programs. A particular Datalog-rewritable DL, called $\mathcal{DL}^+$, was also proposed in [13]. Reasoning in $\mathcal{DL}^+$ is tractable, under both data and combined complexity. Despite its low complexity, $\mathcal{DL}^+$ is still expressive enough in ontology applications sucha s role equvalences and transitive roles.

Based on the concept of Datalog-rewriting, we developed a new reasoner DReW (Datalog ReWriter)[1], which rewrites $\mathcal{DL}^+$ontologies (dl-programs over $\mathcal{DL}^+$ ontologies) to Datalog (Datalog$^\neg$) programs, and calls an underlying rule-based reasoner, currently DLV, to perform the actual reasoning. For $\mathcal{DL}^+$ ontologies, DReW does instance checking as well as answering of conjunctive queries (CQs). For dl-programs over $\mathcal{DL}^+$ ontologies, DReW computes the well-founded model.

The evaluation of the DReW reasoner goes along two axes: as a pure DL reasoner and as a reasoner for dl-programs. Furthermore, we show that several real-word ontologies fall to a large extent in the $\mathcal{DL}^+$ fragment. We compare CQs over the LUBM [11] benchmark with Pellet, KAON2 and RacerPro. For dl-programs, we compare DReW with DLVHEX over LUBM ontologies with dl-rules. The preliminary but encouraging experimental results show that DReW can efficiently handle large knowledge bases.

The remainder of the paper is organized as follows: in Section 2, we recall Datalog, Datalog$^\neg$, Description Logics, and dl-programs under well-founded semantics. In Section 3, we show how Datalog-rewritability can be used to reason on DL ontologies and dl-programs, and we present our new reasoner DReW. Section 4 describes the evaluation of DReW. We compare our work with Horn-$\mathcal{SROIQ}$ [23], ELP [15, 16], and KAON2 [22] in Section 5. Finally, we conclude our work in Section 6.

2 Preliminaries

2.1 Datalog and Datalog$^\neg$

Constants, variables, terms, and atoms are defined as usual. We assume that a binary inequality predicate $\neq$ is available; atoms not using $\neq$ are *normal*. A Datalog$^\neg$ rule r has the form

$$h \leftarrow b_1, \ldots, b_k, not\ c_1, \ldots, not\ c_m \tag{1}$$

where the *body* $b_1, \ldots, b_k, c_1, \ldots, c_m$ are atoms and the *head* h is a normal atom. We call $B^-(r) = \{c_1, \ldots, c_m\}$ the *negative body* of r. If $B^-(r) = \emptyset$, then r is a Datalog rule. A finite set of Datalog$^\neg$ (Datalog) rules is a Datalog$^\neg$ (Datalog) *program*. *Ground* terms, atoms, and programs are defined as usual. A *fact* is a ground rule (1) with $k = m = 0$.

The *Herbrand Domain* $\mathcal{H}_P$ of a program P is the set of constants from P. The *Herbrand Base* $\mathcal{B}_P$ of P is the set of normal ground atoms with predicates and constants from P. An *interpretation* of P is any set $I \subseteq \mathcal{B}_P$. For a ground normal atom a, we write $I \models a$ if $a \in I$; for a ground atom $c_1 \neq c_2$, we write $I \models c_1 \neq c_2$ if c_1 and c_2 are different; for a ground *negation as failure atom* $l = not\ a$, we write $I \models l$ if $I \not\models a$. For a set of ground (negation as failure) atoms α, $I \models \alpha$ if $I \models l$ for all $l \in \alpha$. A ground rule $r : h \leftarrow \alpha$ is *satisfied* w.r.t. I, denoted $I \models r$, if $I \models h$ whenever $I \models \alpha$.

[1] http://www.kr.tuwien.ac.at/research/systems/drew

An interpretation I of a ground program P is a *model* of P, if $I \models r$ for every $r \in P$; in addition, I is *minimal*, if P has no model $J \subset I$. For a non-ground P, I is a (minimal) model of P iff it is a (minimal) model of $gr(P)$, the *grounding* of P with the constants of P defined as usual. Each Datalog program P has some minimal model, which in fact is unique; we denote it with $MM(P)$. We write $P \models a$ if $MM(P) \models a$.

We recall the *well-founded semantics* [8] for Datalog$^\neg$. Let I be an interpretation for a Datalog$^\neg$ program P. The *GL-reduct* [9] P^I of a program P is the set of Datalog rules $h \leftarrow b_1, \ldots, b_k$ such that $r : h \leftarrow b_1, \ldots, b_k, not\ c_1, \ldots, not\ c_m \in gr(P)$ and $I \not\models c_i$, for all $i, 1 \leq i \leq m$.

Using the γ operator [3], one can define the well-founded semantics as follows. Let $\gamma_P(I) = MM(P^I)$ and $\gamma_P^2(I) = \gamma_P(\gamma_P(I))$, i.e., applying the γ operator twice; as γ_P is anti-monotone, γ_P^2 is monotone. The set of *well-founded* atoms of P, denoted $WFS(P)$, is exactly the least fixed point of γ_P^2. We denote with $P \models^{wf} a$ that $a \in WFS(P)$.

For a Datalog (Datalog$^\neg$) program P and an atom a, deciding $P \models a$ ($P \models^{wf} a$) is *data complete* (P is fixed except for facts) for PTIME and *(combined) complete* (P is arbitrary) for EXPTIME [4].

2.2 Description Logics

For space constraints, we assume the reader is familiar with DLs and adopt the usual conventions, see [2]. We highlight some points below.

A *DL knowledge base (KB)* $\Sigma = \langle \mathcal{T}, \mathcal{A} \rangle$ consists of a finite set $\mathcal{T}$ (called *TBox*) of *terminological* and *role axioms* $\alpha \sqsubseteq \beta$, where α and β are concept (respectively role) expressions, and a finite set $\mathcal{A}$ (called *ABox*) of assertions $A(o_1)$ and $R(o_1, o_2)$ where A is a concept name, R is a role name, and o_1, o_2 are individuals (i.e., constants). We also view Σ as the set $\mathcal{T} \cup \mathcal{A}$.

For particular classes of DL KBs Σ, we assume that (1) Σ is defined over a (finite) set $\mathcal{P}_o$ of concept and role names; we call the constants appearing in Σ the *Herbrand domain of* Σ, denoted with $\Delta_{\mathcal{H}(\Sigma)}$; (2) Σ can be extended with arbitrary assertions, i.e., for any ABox $\mathcal{A}'$ (over $\mathcal{P}_o$), $\Sigma \cup \mathcal{A}'$ is an admissible DL KB, and (3) Σ defines a ground entailment relation $\models$ such that $\Sigma \models Q(\mathbf{e})$ is defined for *dl-queries* $Q(\mathbf{e})$, $\mathbf{e}$ ground terms, which indicates that all models of Σ satisfy $Q(\mathbf{e})$. Here, a *dl-query* $Q(\mathbf{t})$ is either of the form (a) $C(t)$, where C is a concept and t is a term; or (b) $R(t_1, t_2)$, where R is a role and t_1, t_2 are terms.

The relation $\Sigma \models Q(\mathbf{e})$ is defined relative to the models of Σ, which are the interpretations $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ that satisfy all axioms and assertions of Σ, where $\Delta^\mathcal{I} \neq \emptyset$ is the domain and $\cdot^\mathcal{I}$ is an interpretation function for concept- and role names as well as individuals. Note that we do not allow for subsumption checking in this paper. We are mainly interested in query answering.

We will assume that the *unique names assumption* (UNA) holds in interpretations $\mathcal{I}$, i.e., $o_1^\mathcal{I} \neq o_2^\mathcal{I}$ for distinct o_1 and o_2, and moreover for simplicity that $o^\mathcal{I} = o$ for individuals o (in particular $\{o\}^\mathcal{I} = \{o^\mathcal{I}\}$ for *nominals*) appearing in the KB. UNA is required due to the translation to datalog later where we have UNA.

Example 1. Take the DL KB Σ:

$$\begin{aligned}(\geq 2\ PapToRev.\top) &\sqsubseteq Over\\ Over &\sqsubseteq \forall Super^{+}.Over\\ \{(a,b)\} \sqcup \{(b,c)\} &\sqsubseteq Super\end{aligned}$$

where $Super^{+}$ is the transitive closure of the role $Super$. The first two axioms indicate that someone who has more than two papers to review is overloaded, and that an overloaded person causes all the supervised persons to be overloaded as well (otherwise the manager delegates badly). The final axiom — equivalent to the assertions $Super(a,b)$ and $Super(b,c)$ — defines the supervision hierarchy.

The particular Description Logic that the reasoner DReW is able to handle is $\mathcal{LDL}^{+}$, as introduced in [13].

$\mathcal{LDL}^{+}$ is designed by syntactic restrictions on the expressions that occur in axioms, distinguishing between occurrence in the "body" α and the "head" β of an axiom $\alpha \sqsubseteq \beta$. We define

- *b-roles* (b for *body*) E, F to be *role names* P, *role inverses* E^{-}, *role conjunctions* $E \sqcap F$, *role disjunctions* $E \sqcup F$, *role sequences* $E \circ F$, *transitive closures* E^{+}, *role nominals* $\{(o_1, o_2)\}$, and *role top* $\top^2$, where o_1, o_2 are individuals, and $\top^2$ is the universal role;
- *h-roles* (h for *head*) E, F to be *role names* P, *role inverses* E^{-}, *role conjunctions* $E \sqcap F$, and *role top* $\top^2$.

Furthermore, let *basic concepts* C, D be concept names A, the top symbol $\top$, and *conjunctions* $C \sqcap D$; then we define

- *b-concepts* C, D as *concept names* A, *conjunctions* $C \sqcap D$, *disjunctions* $C \sqcup D$, *exists restrictions* $\exists E.C$, *atleast restrictions* $\geq n\, E.C$, *nominals* $\{o\}$, and the *top* symbol $\top$, where E is a b-role as above, and o is an individual.
- *h-concepts* (h for *head*) as *basic concepts* B or *value restrictions* $\forall E.B$ where B is a basic concept and E a b-role.

Now an $\mathcal{LDL}^{+}$ *KB* is a pair $\Sigma = \langle \mathcal{T}, \mathcal{A} \rangle$ of a finite TBox $\mathcal{T}$ and a finite ABox $\mathcal{A}$, where

- $\mathcal{T}$ is a set of *terminological axioms* $B \sqsubseteq H$, where B is a b-concept and H is an h-concept, and *role axioms* $S \sqsubseteq T$, where S is a b-role and T is an h-role, and
- $\mathcal{A}$ is a set of assertions of the form $C(o)$ and $E(o_1, o_2)$ where C is an h-concept and E an h-role.

Example 2. Reconsider the DL KB Σ from Example 1. It is easily checked that Σ amounts to an $\mathcal{LDL}^{+}$ KB.

As established in [13], one can check entailment of an atom w.r.t an $\mathcal{LDL}^{+}$ KB by calculating a minimal model w.r.t. to the Herbrand domain of the KB (the individuals appearing in the KB) and checking membership of the atom in that minimal model. This core result allows us to reduce entailment of atoms w.r.t. an $\mathcal{LDL}^{+}$ KB to checking entailment w.r.t. a Datalog program.

2.3 DL-Programs under Well-Founded Semantics

We introduce dl-programs under well-founded semantics (WFS) [6].

Informally, a dl-program consists of a DL KB Σ over $\mathcal{P}_o$ and a $\mathsf{Datalog}^\neg$ program P over a set of predicates $\mathcal{P}_p$ distinct from $\mathcal{P}_o$, which may contain queries to Σ. Roughly, such queries ask whether a certain ground atom logically follows from Σ. Note that the Herbrand domains of Σ and P are not necessarily distinct.

Syntax. A *dl-atom* $a(\mathbf{t})$ has the form

$$\mathrm{DL}[S_1 \uplus p_1, \ldots, S_m \uplus p_m; Q](\mathbf{t}) \;\; m \geq 0, \tag{2}$$

where each S_i is either a concept or a role name from $\mathcal{P}_o$, p_i is a unary, resp. binary, predicate symbol from $\mathcal{P}_p$, and $Q(\mathbf{t})$ is a dl-query. We call the list $S_1 \uplus p_1, \ldots, S_m \uplus p_m$ the *input signature* and $p_1, \ldots, p_m$ the *input predicate symbols*. Intuitively, $\uplus$ increases S_i by the extension of p_i prior to the evaluation of query $Q(\mathbf{t})$.[2]

A *dl-rule* r has the form (1), where any atom b_i, c_j may be a dl-atom. A *dl-program* $\mathcal{KB} = (\Sigma, P)$ consists of a DL KB Σ and a finite set of dl-rules P — *$\mathcal{KB}$ is a dl-program over $\mathcal{DL}$*, if Σ is a $\mathcal{DL}$ KB.

Semantics. We define the *Herbrand base* $\mathcal{B}_{\mathcal{KB}}$ of a dl-program $\mathcal{KB} = (\Sigma, P)$ as the set of ground atoms with predicate symbols from P (i.e., from $\mathcal{P}_p$) and constants from the Herbrand domains of Σ and P. An *interpretation* of $\mathcal{KB}$ is any subset $I \subseteq \mathcal{B}_{\mathcal{KB}}$. It satisfies a ground atom a under Σ, denoted $I \models_\Sigma a$,

– in case a is a non-dl-atom, iff $I \models a$, and
– in case a is a dl-atom of form (2), iff $\Sigma \cup \tau^I(a) \models Q(\mathbf{c})$,

where $\tau^I(a)$, the *extension of a under I*, is $\tau^I(a) = \bigcup_{i=1}^m A_i(I)$ with $A_i(I) = \{S_i(\mathbf{e}) \mid p_i(\mathbf{e}) \in I\}$. Satisfaction of ground dl-rules r under Σ is then as usual (see $\mathsf{Datalog}^\neg$) and denoted with $I \models_\Sigma r$. I is a model of $\mathcal{KB}$, denoted $I \models \mathcal{KB}$, iff $I \models_\Sigma r$ for all $r \in gr(P)$.

We define the well-founded semantics for dl-programs as in [6] using the γ^2 operator. For I and $\mathcal{KB} = (\Sigma, P)$, let $\mathcal{KB}^I = (\Sigma, sP^I_\Sigma)$, the *reduct of $\mathcal{KB}$ wrt. I*, be the dl-program where sP^I_Σ results from $gr(P)$ by deleting (1) every dl-rule r where $I \models_\Sigma a$ for some $a \in B^-(r)$, and (2) from the remaining dl-rules r the negative body $B^-(r)$. Note that sP^I_Σ may still contain positive dl-atoms. As shown in [6], $\mathcal{KB}^I$ has a single minimal model, denoted $MM(\mathcal{KB}^I)$.

Now the operator $\gamma_{\mathcal{KB}}$ on interpretations I of $\mathcal{KB}$ is defined by $\gamma_{\mathcal{KB}}(I) = MM(\mathcal{KB}^I)$. As $\gamma_{\mathcal{KB}}$ is anti-monotone, $\gamma^2_{\mathcal{KB}}(I) = \gamma_{\mathcal{KB}}(\gamma_{\mathcal{KB}}(I))$ is monotone and has a least fixpoint. This fixpoint is the set of *well-founded* atoms of $\mathcal{KB}$, denoted $WFS(\mathcal{KB})$; we denote with $\mathcal{KB} \models^{wf} a$ that $a \in WFS(\mathcal{KB})$.

[2] Modifiers that were included in the original dl-program, $\cup, \cap$, may be expressed by $\uplus$ in strong enough DLs and similarly for subsumption expressions $C \sqsubseteq D$. However, Datalog-rewritability precludes such constructs.

Example 3. Take $\mathcal{KB} = (\Sigma, P)$ where Σ as in Example 1 and P:

$$\begin{array}{lrcl} r_1: & good(X) & \leftarrow & \mathrm{DL}[;\, Super](X,\, Y), \\ & & & not\ \mathrm{DL}[PapToRev \uplus paper;\, Over](Y); \\ r_2: & over(X) & \leftarrow & not\ good(X); \\ r_3: & paper(b, p_1) & \leftarrow & ; \\ r_4: & paper(b, p_2) & \leftarrow & . \end{array}$$

Note that the first dl-atom has no input signature. Intuitively, r_1 indicates that if X is supervising Y and Y is not overloaded, then X is a good manager and r_2 indicates that if X is a not a good manager then X is overloaded. Then, $KB \models^{wf} over(a)$.

Deciding $(\Sigma, P) \models^{wf} a$ is combined complete for ExpTime (PTime$^{\mathrm{NExp}}$) for Σ in $\mathcal{SHIF}(\mathbf{D})$ ($\mathcal{SHOIN}(\mathbf{D})$) and data complete for PTime$^{\mathrm{NP}}$ for Σ in $\mathcal{SHIF}(\mathbf{D})$ and $\mathcal{SHOIN}(\mathbf{D})$ [6]; here data complete means that only the constants in Σ and P, the ABox $\mathcal{A}$, and the facts in P may vary.

3 DReW: Reasoning with Description Logics and DL-Programs using Datalog

As seen in the previous section, one can check entailment of an atom w.r.t. an $\mathcal{LDL}^+$ KB by calculating a minimal model w.r.t. to the Herbrand domain of the KB [13]. Thus, if we manage to write a Datalog program that has a minimal model corresponding to that minimal model of the KB, we obtain a procedure to check entailment of an atom w.r.t. an $\mathcal{LDL}^+$ using standard Datalog engines.

This is exactly the approach DReW takes for conjunctive query answering w.r.t. $\mathcal{LDL}^+$ knowledge bases, and by extension, for reasoning with DL-Programs that have as an underlying DL $\mathcal{LDL}^+$.

We illustrate the approach by means of Example 3. Take the DL part Σ of the particular dl-program. How to rewrite Σ to a Datalog program P such that CQs against Σ can be solved by posing the query to P?

Take $(\geq 2\ PapToRev.\top) \sqsubseteq Over$ from Σ. This corresponds trivially to a rule

$$Over(X) \leftarrow (\geq 2\ PapToRev.\top)(X)$$

where $(\geq 2\ PapToRev.\top)$ is treated as a predicate. In order to make the DL semantics of this number restriction explicit in the program, we add its definition:

$$\begin{array}{rcl} (\geq 2\ PapToRev.\top)(X) & \leftarrow & PapToRev(X,\, Y_1), PapToRev(X,\, Y_2), \\ & & \top(Y_1), \top(Y_2),\, Y_1 \neq Y_2 \end{array}$$

where again $\top$ is treated as a predicate. Note that if there are two different domain elements y_1 and y_2 such that $PapToRev(x, y_1)$ and $PapToRev(x, y_2)$ are true in the model of the program, then the literal $(\geq 2\ PapToRev.\top)(x)$ will be true as well. Vice versa, due to minimality of models of a program, the latter literal will only be true in a minimal model, if there are such two successors y_1 and y_2.

For the axiom $Over \sqsubseteq \forall Super^+.Over$, we have a rule

$$Over(Y) \leftarrow Super^+(X, Y), Over(X)$$

The transitive closure of $Super$ is defined by the traditional Datalog recursive rules to define transitive closure of a predicate:

$$\begin{aligned} Super^+(X, Y) &\leftarrow Super(X, Y) \\ Super^+(X, Y) &\leftarrow Super(X, Z), Super^+(Z, Y) \end{aligned}$$

Note that in contrast with the Horn fragment of Description Logic Programs [10], the translation uses recursive rules and thus full capabilities of Datalog reasoning.

The axiom $\{(a, b)\} \sqcup \{(b, c)\} \sqsubseteq Super$ results in rules

$$\begin{aligned} Super(X, Y) &\leftarrow \{(a, b)\}(X, Y) \\ Super(X, Y) &\leftarrow \{(b, c)\}(X, Y) \end{aligned}$$

Together with the rules for nominals,

$$\begin{aligned} \{(a, b)\}(a, b) &\leftarrow \\ \{(b, c)\}(b, c) &\leftarrow \end{aligned}$$

this ensures that both (a, b) and (b, c) are in the extension of $Super$ as intended by the axiom. Finally, we have rules to ensure the proper handling of inverses,

$$\begin{aligned} Super(X, Y) &\leftarrow Super^-(Y, X) \\ PapToRev(X, Y) &\leftarrow PapToRev^-(Y, X) \end{aligned}$$

and the rules that introduce the Herbrand domain of the DL KB via $\top$ and $\top^2$ predicates:

$$\begin{aligned} \top(a) &\leftarrow \\ \top(b) &\leftarrow \\ \top(c) &\leftarrow \\ \top^2(X, Y) &\leftarrow \top(X), \top(Y) \end{aligned}$$

The formal translation of any $\mathcal{LDL}^+$ knowledge base Σ to a Datalog program $\Phi_{\mathcal{LDL}^+}(\Sigma)$ can be found in [13]. In particular, we can reduce ground entailment of a ground atom w.r.t. $\mathcal{LDL}^+$ knowledge bases to checking whether the atom is present in the minimal model of the translation in Datalog [13, Proposition 10].

We can use this translation of $\mathcal{LDL}^+$ knowledge bases to Datalog to translate reasoning with dl-programs over $\mathcal{LDL}^+$ under well-founded semantics to Datalog$^\neg$ under well-founded semantics. In other words, for a dl-program $\mathcal{KB} = (\Sigma, P)$ where Σ is an $\mathcal{LDL}^+$ knowledge base and P is a Datalog$^\neg$ program, there is a Datalog$^\neg$ program $\Psi(\mathcal{KB})$ that is equivalent w.r.t. checking ground entailment.

Take the dl-program $\mathcal{KB} = (\Sigma, P)$ where $\Sigma = \{\ C \sqsubseteq D\ \}$ and

$$P \triangleq \{\ p(a) \leftarrow;\quad s(a) \leftarrow;\quad s(b) \leftarrow;$$
$$q \leftarrow DL[C \uplus s; D](a), not\ DL[C \uplus p; D](b)\ \}.$$

Intuitively, the dl-atom $DL[C \uplus s; D](a)$ extends C in Σ with the extension of s (i.e., with a and b) and queries Σ then for $D(a)$ which is indeed entailed from Σ once $C(a)$ holds; similarly $DL[C \uplus p; D](b)$ extends C with the extension of p (i.e., a), and queries Σ for $D(b)$ which is not entailed from $C \sqsubseteq D$ if only $C(a)$ is assumed to hold in Σ.

Note that each dl-atom sends up a different input/hypothesis to Σ and that entailments for each different input might be different. To this purpose, we copy Σ to new disjoint equivalent versions for each dl-atom, i.e., for each distinct dl-atom λ, we define a new knowledge base Σ_λ that results from replacing all concept and role names by a λ-subscripted version. Thus, for the set $\Lambda_P = \{\lambda_1 \triangleq C \uplus s, \lambda_2 \triangleq C \uplus p\}$ of dl-atoms, we have $\Sigma_{\lambda_1} = \{\ C_{\lambda_1} \sqsubseteq D_{\lambda_1}\ \}$ and $\Sigma_{\lambda_2} = \{\ C_{\lambda_2} \sqsubseteq D_{\lambda_2}\ \}$.

We translate these disjoint knowledge bases to a Datalog program using $\Phi_{\mathcal{LDL}^+}$ as described above, resulting in the rules $D_{\lambda_1}(X) \leftarrow C_{\lambda_1}(X)$ and $D_{\lambda_2}(X) \leftarrow C_{\lambda_2}(X)$.

The inputs in the dl-atoms Λ_P can then be encoded as rules $\rho(\Lambda_P)$:

$$C_{\lambda_1}(X) \ \leftarrow\ s(X)$$
$$C_{\lambda_2}(X) \ \leftarrow\ p(X)$$

Remains to replace the original dl-rules with rules not containing dl-atoms: P^{ord} results from replacing each dl-atom $DL[\lambda; Q](\mathbf{t})$ in P with a new atom $Q_\lambda(\mathbf{t})$, such that P^{ord} is the Datalog$^\neg$ program

$$P^{ord} \triangleq \{\ p(a) \leftarrow;\quad s(a) \leftarrow;\quad s(b) \leftarrow;$$
$$q \leftarrow D_{\lambda_1}(a), not\ D_{\lambda_2}(b)\qquad\}.$$

One sees that indeed $\mathcal{KB} \models q$ and $\Phi_{\mathcal{LDL}^+}(\Sigma_{\lambda_1}) \cup \Phi_{\mathcal{LDL}^+}(\Sigma_{\lambda_2}) \cup P^{ord} \cup \rho(\Lambda_P) \models q$, effectively reducing reasoning w.r.t. the dl-program to a Datalog$^\neg$ program.

Such a translation of the loose coupling via dl-programs is not limited to the use of $\mathcal{LDL}^+$. In [13], we showed that such a translation works for so-called Datalog-rewriteable DLs, roughly, DLs for which entailment can be reduced to Datalog reasoning. An example of such DLs are the (individual-free) Horn-DL fragments of OWL 1 and 2 [23].

Figure 1 shows a schematic overview of the component of DReW responsible for reducing entailment from DLs to Datalog. The extension for dl-programs is a straightforward elaboration of this. Taking as input a conjunctive query and an ontology in OWL 2 syntax extended for the complex role expressions of $\mathcal{LDL}^+$, DReW checks whether the ontology is in the $\mathcal{LDL}^+$ fragment. If it is, we translate the ontology according to the format suitable for the specified Datalog reasoner (DLV in our case).

DReW is written in Java using an extension of the OWL API 3.0.0[3] for parsing $\mathcal{LDL}^+$ ontologies. The underlying Datalog engine we used is the latest version of DLV

[3] http://owlapi.sourceforge.net/

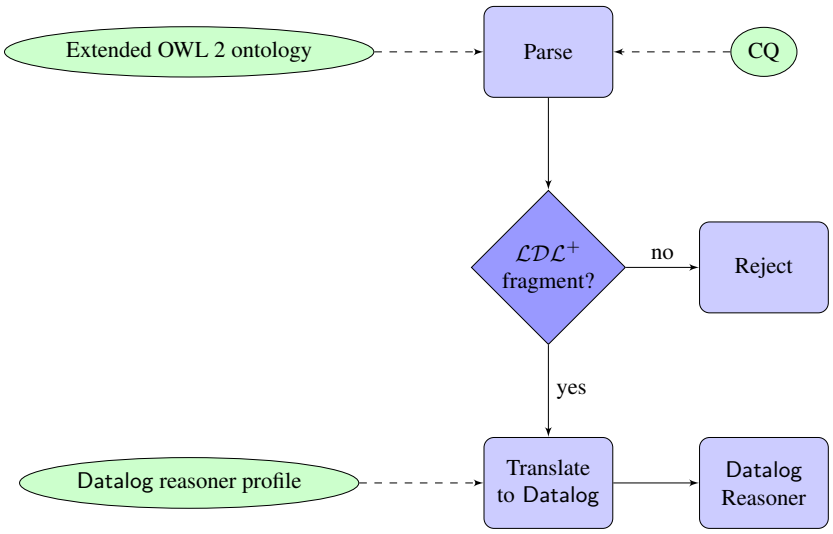

Fig. 1. DReW Control Flow — DL Component

(*dl-magic-snapshot-2009-11-26*) [4] which supports magic sets and well-founded semantics.

4 Evaluation

In this section, we evaluate the DReW reasoner. We do so along two axes: as a pure Description Logic reasoner and as a reasoner for dl-programs.

All experiments were performed on a laptop running Ubuntu 10.04 with a 1.83G CPU and 2G of memory; the memory of the Java Virtual Machine was set to 1G.

4.1 Reasoning with Description Logics

We first analyze to what extent common ontologies fall in the $\mathcal{LDL}^+$ fragment. Next, we analyze the performance of conjunctive query answering with DReW compared to standard DL reasoners on those ontologies.

Expressiveness of $\mathcal{LDL}^+$ To assess the expressiveness of $\mathcal{LDL}^+$, we select several ontologies and show that they fall to a large extent in the $\mathcal{LDL}^+$profile. We picked the ontologies that are used in Motik's thesis for testing the DL reasoner KAON2 [22]; they can be downloaded from the KAON 2 site[5].

The results of this experiment are listed in Table 1. Note that for Galen over 40% of the axioms are not in the $\mathcal{LDL}^+$ profile, but that for Dolce and Wine over 80% of

[4] http://www.dbai.tuwien.ac.at/proj/dlv/magic/

[5] http://kaon2.semanticweb.org/download/test_ontologies.zip

Ontology	Axioms	Inds	Concepts	Object Props	$\mathcal{ELH}^+$?	Violated Axioms	Violated %
Galen	4,356	0	2,747	261	no	1,881	0.43
Dolce	1,185	2	125	251	no	162	0.14
Wine	773	162	142	13	no	137	0.18
Vicodi	53,876	16,942	194	10	yes	0	0
Semintec	65,459	17,941	60	16	no	113	$1.73 \cdot 10^{-3}$
LUBM	8,612	1,555	43	25	no	8	$9.29 \cdot 10^{-4}$

Table 1. $\mathcal{ELH}^+$profile checking

axioms are in $\mathcal{ELH}^+$. Only Vicodi is fully in $\mathcal{ELH}^+$and over 99% of Semintec and LUBM axioms are in $\mathcal{ELH}^+$. Most of the violations are due to existential quantifiers occurring on the right side of axioms.

Conjunctive Query Evaluation To evaluate the performance of CQs over ontologies using DReW, we compare it with 3 state-of-the-art DL reasoners: KAON2, RacerPro, and Pellet. We did not consider other DL reasoners, such as HermiT or Fact++ as they cannot handle CQs; we did not consider REQUIEM and QuOnto as they can not handle $\mathcal{ELH}^+$ ontologies.

Reasoning in KAON2[6] [22] is implemented using novel algorithms that reduce a $\mathcal{SHIQ}(\mathbf{D})$ knowledge base to a disjunctive Datalog program based on resolution techniques. Pellet[7] [26] fully supports OWL 2[19]. In contrast with KAON2, it is a reasoner based on tableaux algorithms. RacerPro[8] [12] is a tableaux-based reasoner as well and implements the Description Logic $\mathcal{SHIQ}$. All 3 reasoners support conjunctive query answering.

We specifically tested CQs on The Lehigh University Benchmark (LUBM) [11]. LUBM is developed to facilitate the evaluation of Semantic Web repositories in a standard and systematic way. The benchmark is intended to evaluate the performance of those repositories with respect to extensional queries over a large data set that commits to a single realistic ontology. It consists of a university domain ontology, customizable and repeatable synthetic data, a set of test queries, and several performance metrics. The queries we evaluated are as in
http://swat.cse.lehigh.edu/projects/lubm/query.htm, referring to numbers 1-14.

As we indicated in Table 1, LUBM is not fully in $\mathcal{ELH}^+$: there are 8 violated axioms, e.g.,

$$Person \sqcap \exists headOf.Department \equiv Chair \ .$$

For our experiments and to have an $\mathcal{ELH}^+$ conformant fragment of LUBM, we replace such equivalence axioms by subsumption axioms, e.g., by

$$Person \sqcap \exists headOf.Department \sqsubseteq Chair \ .$$

In general, such a conversion changes the semantics of the ontology. However, in our considered test of the benchmark queries, the query results are exactly the same as

[6] http://kaon2.semanticweb.org/
[7] http://clarkparsia.com/pellet/
[8] http://www.racer-systems.com

Query	DReW	KAON2	RacerPro	Pellet
1	3.13	2.84	3.78	4.55
2	2.23	2.39	4.24	4.54
3	2.29	2.35	3.68	4.54
4	2.25	2.61	26.05	4.63
5	2.29	2.60	5.12	4.52
6	2.24	2.56	5.05	4.51
7	2.21	2.63	3.39	4.44
8	2.28	2.65	27.13	4.62
9	2.22	2.67	4.80	4.54
10	2.22	2.42	3.85	4.53
11	2.23	2.31	4.39	4.49
12	2.27	2.55	4.08	4.63
13	2.31	2.58	4.44	4.42
14	2.26	2.35	5.30	4.52

Table 2. Conjunctive Queries on LUBM (in secs.)

Ontology	Inds	DReW	KAON2	RacerPro	Pellet
LUBM0	904	1.61	2.27	4.51	3.53
LUBM1	1,555	2.27	2.54	7.52	4.53
LUBM2	2,753	5.07	3.72	9.38	7.57

Table 3. Conjunctive Queries on LUBM with Different Number of Individuals

on the original LUBM. It is part of future research to investigate how DReW can deal with partial $\mathcal{LDL}^+$ ontologies in answering queries as faithfully as possible.

The results of evaluating the 14 CQs on LUBM are shown in Table 2. From the table, we see that DReW outperforms RacerPro and Pellet in all the queries and that it is slightly better than KAON2 for most of the queries. Note the out-of-the-normal times for RacerPro on query 4 and query 8; we assume they are caused by the use of data properties.

As DReW and KAON2 have evaluation times close to each other, we also evaluate CQs on LUBM ontologies with a different numbers of individuals. The result is summarized in Table 3.

LUBM1 is the original LUBM ontology. By removing and adding individuals, we get LUBM0 and LUBM2. The number under each reasoner is the average time for answering the 14 queries. In all the LUBMs, DReW is better than RacerPro and Pellet. However, compared with KAON2, we also see that DReW is not so good at dealing with large number of individuals. We assume that the reason is the use of DLV as the underlying Datalog engine. Since there is no public API for DLV, we have to use it as a standalone process. When the translated ontology is big, the communication of processes costs significant time.

Query	DReW	DLVHEX+DL-Plugin	dl-atoms	Factor
0	2.81	4.31	1	1.53
1	2.63	3.04	1	1.16
2	2.60	3.88	1	1.49
3	2.59	4.04	1	1.56
4	2.75	3.51	1	1.27
5	3.00	5.10	1	1.70
6	4.69	19.59	6	4.17
7	3.20	8.38	2	2.62

Table 4. Reasoning on dl-programs

4.2 Reasoning with DL-Programs

DReW is designed for reasoning over dl-programs under well-founded semantics. The only reasoner available for comparison is DLVHEX [9] [7]. DLVHEX is a prototype implementation for computing the stable models of so-called HEX-programs – an extension of dl-programs for reasoning with external sources (not necessarily DL knowledge bases) under the answer set semantics. By using the *Description Logic Plugin* [10], which interfaces to OWL ontologies via a Description Logic reasoner (currently RacerPro), DLVHEX can reason on dl-programs under the answer set semantics.

Note that for Datalog programs (i.e., without negation), the well-founded semantics coincides with the answer set semantics. We thus evaluate both reasoners on LUBM, which is negation free. We manually generate serveral dl-program over LUBM to evaluate reasoning over Datalog-rewriteable ontologies.

All the test results are shown in Table 4. We see that DReW outperforms DLVHEX for all the tests. As the number of dl-atoms increases, the advantage of DReW becomes more clear, confirming our hypothesis that translating dl-programs to Datalog programs reduces the overload of calling external DL reasoners as is the case in DLVHEX.

5 Related Work

Horn Fragments of Description Logics Horn-$\mathcal{SHOIQ}$ and Horn-$\mathcal{SROIQ}$ are Horn fragments of OWL 1 and OWL 2 [23] respectively. Reasoning in Horn-$\mathcal{SHOIQ}$ is EXPTIME-complete, and reasoning in Horn-$\mathcal{SROIQ}$ is 2-EXPTIME-complete. Despite their high expressiveness, both Horn-$\mathcal{SHOIQ}$ and Horn-$\mathcal{SROIQ}$ have polynomial data complexity and as shown in [23] can be translated to Datalog.

However, $\mathcal{SROIQ}$ ($\mathcal{SHOIQ}$) is not Datalog-rewritable (in the sense of [13]) as the modularity property does not hold in general. In particular, $\Phi(\langle\emptyset, \mathcal{A}\rangle) \neq \mathcal{A}$. We do have that it is Datalog-rewritable if we restrict to individual-free knowledge bases. In essence, this means that we can use DReW to reason on dl-programs over Horn-$\mathcal{SHOIQ}$ and Horn-$\mathcal{SROIQ}$ KBs provided the latter do not contain individuals.

[9] http://www.kr.tuwien.ac.at/research/systems/dlvhex

[10] http://www.kr.tuwien.ac.at/research/systems/dlvhex/dlplugin.html

ELP ELP [15, 16] is a decidable fragment of the Semantic Web Rule Language (SWRL) that admits reasoning in polynomial time. ELP is based on the tractable DL $\mathcal{EL}^{++}$ and encompasses an extended notion of the DL rules[15]. Also ELP extends $\mathcal{EL}^{++}$ with a number of features introduced by OWL 2, such as disjoint roles, local reflexivity, certain range restrictions, and the universal role. A reasoning algorithm is based on a translation of ELP to Datalog in a in a tractable fashion.

There are several differences between ELP and dl-programs over $\mathcal{LDL}^{+}$: (1) ELP is a tightly-coupled combination of ontologies and rules, while dl-programs are loosely coupled; (2) in the rule part of a dl-program one can use default negation well-founded semantics, which can not be expressed in ELP. Indeed, ELPs have a first-order semantics compared to the minimal model semantics of dl-programs.

KAON2 KAON2 does not implement the tableaux calculus. Rather, reasoning in KAON2 is implemented by novel algorithms which reduce a $\mathcal{SHIQ}$ knowledge base to a disjunctive Datalog program of exponential size. The translation is not modular in the above-mentioned sense if the ABox is non-empty. However, with an empty ABox, also the KAON2 rewriting can be used in the context of our dl-programs to Datalog$^{\neg}$ reduction.

Hybrid MKNF KBs under WFS Well-founded Semantics is also used in other combinations. In [14, 1], WFS for the tightly-coupled Hybrid MKNF KBs was proposed. When the underlying DL of MKNF KBs is tractable, the data complexity of MKNF under WFS is in PTIME.

6 Conclusions and Outlook

We presented the class of Datalog-rewritable DLs and showed that reasoning with dl-programs over such DLs can be reduced to Datalog$^{\neg}$ under well-founded semantics. This reduction avoids the overhead that is normally associated with the calling of a native DL reasoner. The $\mathcal{LDL}^{+}$ DL is such a particular Datalog-rewritable DL. We developed a new reasoner, DReW, which can efficiently reason over $\mathcal{LDL}^{+}$ DL ontologies and dl-programs over $\mathcal{LDL}^{+}$ ontologies.

We plan to extend $\mathcal{LDL}^{+}$ with more DL constructors as in [17], while keeping the Datalog-rewritability. For example, disjoint classes in OWL 2 Profiles [18] can be added. Furthermore, currently, we only use DLV as the underlying rule-based reasoner. We plan to experiment with different rule engines, e.g., XSB. Finally, Datalog-rewritable DLs are a natural candidate for tightly-coupling approaches as well.

References

1. J. J. Alferes, M. Knorr, and T. Swift. Queries to hybrid mknf knowledge bases through oracular tabling. In *International Semantic Web Conference*, LNCS 5823:1-16, 2009.
2. F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook*. CUP, 2003.
3. C. Baral and V. S. Subrahmanian. Dualities between alternative semantics for logic programming and nonmonotonic reasoning. *JAR*, 10(3):399–420, 1993.

4. E. Dantsin, T. Eiter, G. Gottlob, and A. Voronkov. Complexity and expressive power of logic programming. *ACM Computing Surveys*, 33(3):374–425, 2001.
5. T. Eiter, G. Ianni, T. Lukasiewicz, R. Schindlauer, and H. Tompits. Combining answer set programming with description logics for the Semantic Web. *Artificial Intelligence*, 172(12-13):1495–1539, 2008.
6. T. Eiter, T. Lukasiewicz, R. Schindlauer, and H. Tompits. Well-founded semantics for description logic programs in the Semantic Web. In *Proc. RuleML*, pages 81–97, 2004. Full paper *ACM TOCL*, (to appear).
7. T. Eiter, G. Ianni, R. Schindlauer, and H. Tompits. Effective integration of declarative rules with external evaluations for semantic-web reasoning. In *ESWC*, LNCS 4011:273-287, 2006.
8. A. V. Gelder, K. Ross, and J. S. Schlipf. The well-founded semantics for general logic programs. *JACM*, 38(3):620–650, 1991.
9. M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *Proc. ICLP*, pages 1070–1080. The MIT Press, 1988.
10. B. N. Grosof, I. Horrocks, R. Volz, and S. Decker. Description logic programs: Combining logic programs with description logic. In *Proc. WWW 2003*, pages 48–57. ACM.
11. Y. Guo, Z. Pan, and J. Heflin. An evaluation of knowledge base systems for large owl datasets. Technical report, CSE Department, Lehigh University, 2004.
12. V. Haarslev and R. Möller. Racer system description. In *IJCAR*, LNCS 2083:701-706, 2001.
13. S. Heymans, T. Eiter, and G. Xiao. Tractable reasoning with DL-programs over datalog-rewritable description logics. In *roc. of 19th European Conference on Artificial Intelligence (ECAI)*. IOS Press, 2010.
14. M. Knorr, J. J. Alferes, and P. Hitzler. A coherent well-founded model for hybrid mknf knowledge bases. In *ECAI*, volume 178 of *Frontiers in Artificial Intelligence and Applications*, pages 99–103. IOS Press, 2008.
15. M. Krötzsch, S. Rudolph, and P. Hitzler. Description logic rules. In *Proc. ECAI*, pages 80–84. IOS Press, 2008.
16. M. Krötzsch, S. Rudolph, and P. Hitzler. ELP: Tractable rules for OWL 2. In *Proc. ISWC 2008*, pages 649–664.
17. M. Krötzsch and S. Rudolph. A matter of principles: Towards the largest dlp possible. In *Description Logics*, volume 477 of *CEUR Workshop Proceedings*, 2009.
18. B. Motik, B. C. Grau, I. Horrocks, Z. Wu, A. Fokoue, and C. Lutz, editors. *OWL 2 Web Ontology Profiles*. 2008. W3C Rec. 27 Oct. 2009.
19. B. Motik, P. F. Patel-Schneider, and B. Parsia, editors. *OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax*. 2008. W3C Working Draft April 2009.
20. B. Motik and R. Rosati. A faithful integration of description logics with logic programming. In *Proc. IJCAI*, pages 477–482, 2007.
21. B. Motik, U. Sattler, and R. Studer. Query answering for OWL-DL with rules. *Journal of Web Semantics*, 3(1):41–60, July 2005.
22. B. Motik. *Reasoning in Description Logics using Resolution and Deductive Databases*. PhD thesis, University of Karlsruhe, Karlsruhe, Germany, January 2006.
23. M. Ortiz, S. Rudolph, and M. Simkus. Worst-case optimal reasoning for the horn-dl fragments of owl 1 and 2. In *KR*, pages 269–279. AAAI Press, May 2010.
24. R. Rosati. On the decidability and complexity of integrating ontologies and rules. *Journal of Web Semantics*, 3(1):41–60, 2005.
25. R. Rosati. DL+log: Tight integration of description logics and disjunctive datalog. In *Proc. KR*, pages 68–78, 2006.
26. E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz. Pellet: A practical OWL-DL reasoner. *J. Web Sem.*, 5(2):51–53, 2007.

Representational Analysis of Business Process and Business Rule Languages

Vid Prezel[1], Dragan Gašević[1], Milan Milanović[2]

[1] School of Computing and Information Systems
Athabasca University, Athabasca, AB, Canada
vprezel@gmail.com, dgasevic@acm.org
[2]GOOD OLD AI Research Network, University of Belgrade, Serbia
milan@milanovic.org

Abstract. We conducted a representational analysis of a set of business process and business rule modeling languages by using the well-known Bunge-Wand-Weber (BWW) representation model. Our paper comparatively assesses their ontological deficiencies and explores different possibilities of combining process modeling languages with rule modeling languages in order to achieve the highest ontological completeness. We demonstrate that a combination of BPMN and R2ML (rBPMN) offers the highest ontological completeness among the languages studied.

Keywords: Business process modeling, business rules, representational analysis, formal ontology

1 Introduction

With the growing complexity of today's information systems, business process modeling has gained a lot of attention by both academic and industry communities. In fact, research on business process modeling involves different areas such as software engineering, service-oriented architectures, business process management, formal reasoning, and the Semantic Web. To analyze business process modeling languages, typically, different types of usage patterns are leveraged to estimate suitability of a business process modeling language in addressing certain tasks [6]. For example, workflow patterns are used for general analysis of languages for commonly-used control-flows or service orchestrations, while server-interaction patterns are used to evaluate the support for modeling service choreographies.

Recently, aiming to support modeling and development of more agile business processes, the research community started investigating the integration of more declarative formalism with business process modeling languages. In fact, relations of business process modeling with business rules are one of the most interesting research directions [26]. On the one hand, rules are seen as an alternative to model and/or implement business processes. On the other hand, rules are seen as complements to the existing business process modeling languages (so-called hybrid languages). To evaluate both groups of contributions, traditional business process modeling methods based on different types of patterns (workflow and service interaction) have again

been used [6]. While pattern-based evaluation is useful for indicating a level of support for solving different types of tasks, pattern-based analysis is not suitable to evaluate a general level of representational support to model information systems in general.

Borrowing from the formal ontology research, the research community has proposed the use of formal ontological models such as the Bunge-Wand-Weber (BWW) model [13]. In fact, the research community has developed methodologies that allow for representational analysis of modeling languages (so-called coverage analysis) of combinations of modeling languages (so-called overlap analysis) [23]. Adopting principles of these methodologies, works presented in [21-22] conducted a representational analysis of business process and rule languages. However, those efforts analyzed an older version of the Business Process Modeling Notation (BPMN). Moreover, that work did not include any existing proposals for a hybrid (rule-enhanced) business process modeling language. Adopting the methodology proposed in [23] and trying to take more recent business process modeling languages into consideration, this paper aims to provide:

- Representational analysis of the current and under-development versions (1.2 and 2.0) of the BPMN business process modeling language;
- Representational analysis of rule-based and hybrid modeling languages and their comparison with the current versions of the BPMN language;
- Representational analysis of pairing the current versions of the BPMN language with different business rule languages.

2 Background

This section describes a selected set of rule and process modeling languages, basic elements of the followed representational analysis, and related work on conducting representational analysis by following similar methodological principles.

2.1 Business process and business rules languages

We analyze the five modeling languages: BPMN [1-2], PRR [3], SWRL [4], R2ML [5], and rBPMN [6]. While we are aware that this is not a complete set of languages and we are working on the analysis of a few others (OCL, SBVR, and RIF), this selection might already provide some useful indicators comparing to the results reported in [21]. The choice of BPMN was due to its popularity and adoption by a wide range of process modeling tool vendors and organizations [1]. In the representational analysis conducted by Recker et al. [7], it was concluded that there is no representation for states in BPMN v1.0. So, one of our goals was to check what is the support for state modeling in the more recent versions of BPMN. In our analysis, we also wanted to include languages of four types of rules according to Wagner et al's classification cited in the PPR standard [3]. This coverage was assured by the inclusion of the following three rule languages: SWRL – supports integrity rules; PRR – production rules; and R2ML – integrity, derivation, production and reaction rules. We selected SWRL [4], as it is a widely-used (integrity) rule language for the Semantic Web; PRR [3], as it is

the OMG's standard for modeling production rules; and R2ML [5], as it can represent all types of rules. Lastly, we chose rBPMN, because it represents a hybrid language, incorporating process (BPMN) and rule languages (R2ML).

2.1.1 BPMN

The Business Process Modeling Notation (BPMN) [1] is a graphical notation and a language for modeling business processes. It was developed by Business Process Management Initiative (BPMI) and is based on other notations such as IDEF, UML, LOVeM, RosettaNet, and Event-driven Process Chains [7]. The major goal that led to the development of BPMN was to introduce a business process modeling notation that is acceptable and usable not only by process developers responsible for technology implementation, but also by business analysts and business managers responsible for the design and management of business processes. The other goal that led to the development of BPMN was to allow BPMN instances to be the source of an executable process, which means that there would be a mapping from one or more BPMN notation instances to an execution level instance. This allows BPMN to map directly to languages that were designed for the execution of business processes. Since the merger of BPMI with OMG in 2005, BPMN is now maintained by OMG. The first version BPMN 1.0 was released to public in May 2004 and adopted by OMG in 2006. The current version is BPMN 1.2, with BPMN 2.0 Beta1 in a finalization phase. According to OMG's website [1], there are currently sixty-two implementations and four planned implementations in practice.

2.1.2 Production Rule Representation (PRR)

PRR [3] was defined by vendors of business rules engines such as ILOG, Fair Isaac, LibRT, IBM, Pega, Corticon, TIBCO, academic community (RuleML.org), and UML tool vendors [3]. The current version (1.0 from December 2009) is now an adopted OMG standard, and a formal model for production rules. It uses a UML style for rule representation. PRR includes two types of rules: Forward chaining inference rules and sequentially processed procedural rules. Forward chaining rules (e.g., Rete-model) are used for common production rule engines, which makes it dependant on the types of rules executed by rule engines. Sequentially processed procedural rules are used for tools that extract simple business logic as non inference production rules [3]. The PRR is defined at two levels. A core structure of PPR (referred to as PRR Core) includes general rule and production rule model. PRR OCL structure includes an extended OCL normative expression language to allow compatibility with non UML representations. The high level structure of PRR is similar to R2ML, however in PRR there is no consideration of correspondences to (Semantic) Web rule languages [8].

2.1.3 Semantic Web Rule Language (SWRL)

Semantic Web Rule Language (SWRL) is a submission to the W3C trying to combine the rules (RuleML) and ontologies (OWL-DL and Lite). Rules in SWRL are expressed in terms of OWL constructs such as individuals, properties, literals, and classes. Rules are written as antecedent-consequent pairs [9]: The antecedent as the rule body, and the consequent as the rule head. This means whenever the conditions specified in the body are true, than the conditions specified in the head must also be

true. The head and body consist of zero or more atoms. If there are multiple atoms, they are treated as a conjunction and could be transformed into separate rules with atomic heads or consequents. If the body has zero atoms, it is satisfied by every interpretation, and thus, the head must also be satisfied by every interpretation. The same rule applies if the head has zero atoms. It is not satisfied by any interpretation, therefore the body is also not satisfied by any interpretation [4]. While SWRL is not standardized, it is a widely-used (or more modestly saying – widely-considered) language supported by a few commonly-used reasoners.

2.1.4 R2ML

A business rule is a statement that aims to influence or guide behavior and information in an organization [24]. REWERSE I1 Rule Markup Language (R2ML) is a general rule markup language [5]. It is originally designed to support rule interchange (thus, markup in its name), but it is also a comprehensive rule modeling language with a UML-based graphical concrete syntax. It can represent four types of rules, namely, integrity, derivation, reaction, and production. R2ML is built by using model-driven engineering principles, which include: metamodel, an XML-based textual concrete syntax, and a graphical concrete syntax (so-called URML [25]). A complete reference of R2ML can be found in [5]. All R2ML rule definitions are inherited from the Rule concept (class in meta-model). Each type of rule is defined over the R2ML vocabulary, where elements of the vocabulary are used in logical formulas (e.g., LogicalFormula – with no free variables) through the use of Atoms and Terms. An important aspect of R2ML is that it distinguishes between object and data atoms.

2.1.5 Rule-Based BPMN (rBPMN)

The rBPMN language is a product of integration of BPMN and R2ML, and it is defined by weaving the elements of the BPMN and R2ML abstract syntaxes (metamodels) [10]. The main element in the rBPMN language is a *RuleGateway*, which was added in the *Process* package of the BPMN metamodel (current submission for the BPMN 2.0 metamodel) and which actually relates to R2ML Rules. In this way, an R2ML Rule (i.e., reaction, derivation, production or integrity rule) can be placed into a process as a Gateway, but at the same time, the rule does not break the R2ML Rule syntax and semantics. rBPMN has been designed to support a rule-enhanced process-oriented modeling of service orchestrations and choreographies. More details about this language can be found in [10].

2.2 Representation Theory

According to Bunge [11], real-world systems in any domain can be explained by using an ontology. This can be done by defining structure, properties, and interaction between things of a domain under study [11]. Using a language L to describe topics in a domain D, an ontology provides a catalog of things (represented as concepts, relations, and predicates of language L) assumed to exist in domain D [12]. Any real world system can be explained in such a way (i.e., by an ontology). The application of ontology for the purpose of representation is known as representational analysis. Wand and Weber [13-15], adopted Bunge's ontology [11], and developed a theory

that consists of state tracking, decomposition, and representation models. The last one, the representation model is used in information systems domain, and is now known as the Bunge-Wand-Weber (BWW) representation model. The BWW model defines a set of constructs necessary to provide a complete representation of all things and their interactions in a real world. For a detailed description of BWW representation model and constructs please see, for example [13].

2.3 Related work

The Bunge-Wand-Weber (BWW) representation model has been so far the most widely-used for the ontological analysis of language grammars for business system analysis. In 2005, Green and Rosemann evaluated over twenty research projects that used the BWW model in the area of conceptual modeling [16]. Several of those studies focused on process modeling. Keen et al. [17] for example, evaluated flowcharts and flow diagrams in order to determine their ontological completeness. Green et al. [18] analyzed event-driven process chain notation using the BWW model, and different modeling standards for enterprise system interoperability [19], to determine their ontological completeness and clarity. In terms of evaluating business process and business rule modeling languages specifically, zur Muehlen et al., conducted a BWW representational analysis for several different languages particularly relevant to compliance management [20-21]. Recker et al. conducted a representational analysis with a focus on BPMN 1.0 [7] and process modeling [27]. Opdahl et al. conducted an ontological evaluation of UML [28]. Our work complements the work of [20-22] by analyzing the current version of BPMN, and compares it with an additional set of rule languages aiming to determine a maximum ontological completeness.

3 Methodology

For our analysis, we selected five business process and business rule modeling languages. First, we obtained clear definitions of each of the languages from language specifications and their meta-models. We then selected relevant language constructs[1]. To analyze the selected languages, we followed a reference methodology for conducting ontological analysis given in [23]. To do so, we examined the BWW representation model constructs [13-15] and defined relevant ontology constructs which we than used in our reference model[2]. We than started the process of identifying correspond-

[1] Relevant constructs are the major language constructs relevant for presenting concrete problem domains. For example, for process languages (all BPMN versions) all constructs that were included in their visual notations (as per their specification) were considered relevant as this was also useful for evaluation of visual notation itself. For example, Opdahl et al selected 67 (out of 216) constructs when evaluating UML [28]. We performed similar selection, but for the languages in our scope.

[2] As indicated in the related work section, there are lots of BWW-based studies (and interpretations of the BWW model), and most of them use different numbers of *relevant* BWW constructs. In our case, we selected 28 main BWW constructs based on the original work from [13]. The original BWW work leaves some room for interpretation, and thus different ana-

ing constructs in the modeling language. Based on [13], [22], [18], [20], we divided our reference model into four main clusters: Thing, State, Event, and System. We then defined the subgroups of each cluster and relevant BWW constructs. We performed a representational analysis and compared each of the language constructs with constructs of our reference model and vice versa.

We first mapped the core set of the language constructs for BPMN 1.2 and noted the results. We then proceeded with the extended set of BPMN 1.2. This was then followed by mapping of the basic and extended sets of constructs for BPMN 2.0, Beta 1. After the mappings were completed for both versions (1.2 and 2.0) of BPMN, we marked results to determine ontological clarity. We looked for any differences between corresponding language constructs and BWW representation model constructs, as this provides us with an indication of a representational deficiency. We followed similar steps of our representational analysis for our selection of business rule languages. Finally, we compared the findings for rule languages with those for BPMN.

3.1 Ontological completeness

When analyzing the results, we looked at the extent to which a language has construct deficit comparing to the BWW representation model [13-15]. As such, this approach can be used as a measure of ontological completeness. Ontological completeness determines whether users of a given modeling language are able to represent all relevant real world scenarios when modeling with the given modeling language.

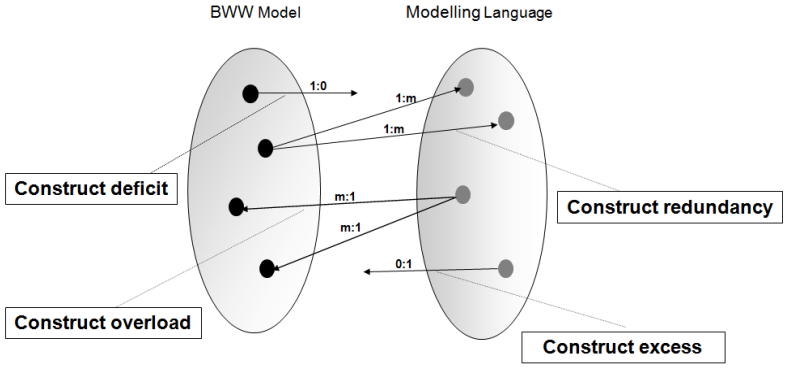

Fig. 1. Ontological completeness and clarity

Ontological clarity of a modeling language is determined by the extent to which language constructs are deemed to be overloaded, redundant, excessive or in deficit [23]. These metrics are illustrated in Fig. 1. Construct overload results when there are many to one (m:1) relationship mappings among constructs of a modeling language and the BWW model (i.e., one element of the modeling language can be used to

lyses might results in different number of BWW constructs. For example, zur Muehlen and Indulska used 29 main BWW constructs [21]. Their study lists the 'Acts on' BWW construct as a separate construct in the Event cluster, which is not the case in our study, as we could not find it in the original BWW model nor was the rationale for its inclusion given in [21].

represent many constructs of the BWW model); while redundancy occurs when there is one to many (1:m) relationship mappings (i.e., many elements of the modeling language can be used to represent one element of the BWW model). Construct excess represents zero to 1 (0:1) mapping, where at least one language construct does not map to any construct in the BWW ontological model. Construct deficit occurs when at least one construct in the BWW model does not map to any construct in an analyzed language. This can be described as a one to zero (1:0) mapping relationship.

3.2 Overlap analysis

In a scenario when none of the studied languages provides a complete representation capability, overlap analysis [19] is performed. This analysis combines a maximum ontological completeness and a minimum ontological overlap. This is useful for evaluating hybrid languages that already include combination of business process and business rule languages. It is also useful for exploring other language combinations or integrations which might offer a minimum overlap and a maximum completeness. With the overlap analysis, we are able to determine a symmetric difference and intersection of analyzed modeling languages [21]. By a symmetric difference, we determine a number of BWW constructs that are represented with no overlap for a given hybrid language or a language combination. With an intersection, we look at the number of concepts which can be represented additionally by a known hybrid language or by a newly proposed combination of languages. We are also able to determine a relative complement [21] for rule and process combinations of languages. This means how many BWW constructs were contributed by a rule modeling language to a process modeling language and vice versa.

4 Data Collection and Analysis

4.1 Representation Analysis of BPMN, versions 1.2 and 2.0

We started with the identification of the core constructs in BPMN 1.2 from the language specification. We then proceeded with the extended set of BPMN 1.2, which was followed by BPMN 2.0 Beta 1 core and extended sets. We performed a complete representational analysis of the core and extended sets of constructs for both versions of BPMN. We were interested in evaluating possible deficiencies in the above mentioned languages sets. As described in section 3.1, we examined four types of representational deficiencies: construct deficit, redundancy, overload, and excess. The lack of representation for particular BWW constructs means that users will have difficulties modeling certain scenarios in a real world domain. Table 1 shows our results for all four sets of BPMN constructs. The constructs columns show the number of constructs that exhibit a certain deficiency. The percentage columns indicate the percentage of constructs that reveal a particular deficiency. Both core versions of BPMN have 39.3% of deficit. The deficit is reduced in the extended sets of BPMN, with the extended set of BPMN 2.0 offering the lowest construct deficit of 32.1%.

Table 1. Construct Deficit, Redundancy, Overload, and Excess

	BPMN 1.2 Core		BPMN 2.0 Core		BPMN 1.2 Ext		BPMN 2.0 Ext	
	Constructs	Percentage	Constructs	Percentage	Constructs	Percentage	Constructs	Percentage
Deficit	11	39.3%	11	39.3%	10	35.7%	9	32.1%
Redundancy	11	39.3%	11	39.3%	16	57.1%	16	57.1%
Overload	5	17.9%	5	17.9%	29	103.6%	30	107.1%
Excess	4	14.3%	5	17.9%	16	57.1%	22	78.6%

In terms of construct redundancy both core sets offer lowest redundancy. This is due to the fact that core sets of elements include a smaller number of constructs overall, which is beneficial in terms of complexity. However, they offer a lower level of construct completeness and higher level of construct deficit. Both extended sets have a redundancy rate of 57.1%, comparing to 39.3% for the core sets of BPMN. Excessive redundancy can potentially cause some confusion to the languages users as to when to use a particular language construct. For example, all BPMN language sets include Pool and Lane language constructs. Both of these constructs map to BWW construct Thing which could cause confusion as to which construct to use, for example, to represent a department in an organization.

In terms of construct overload, both core sets of BPMN offer 17.9% overload, comparing to 103.6% for extended BPMN 1.2 set, and 107.1% for BPMN 2.0 set. This is again due to the number of constructs each sets offer and the number of constructs that actually map to the 28 selected BWW constructs. For example, core BPMN 2.0 has only 14 constructs comparing to 68 for the extended set. Out of 68 constructs, 30 constructs map to more than one BWW construct. As an example for BPMN 1.2, the Lane construct maps to system, subsystem, system composition, system environment, system decomposition, level structure, and thing. This means that users may be confused as to when to use this construct when modeling for example, a department in an organization, a seller/ buyer, or an application system.

In terms of excess, all of the evaluated language sets have excess, which means they have constructs that cannot be mapped to any BWW construct. For example, certain BPMN constructs such as Off-Page-Connector, Activity looping, and Association Flow, have no real world meaning from the BWW perspective and are included as excess. Those types of constructs may be useful for actual modeling activities, but not for capturing semantics of a real world domain. Large numbers of construct excess also contributes to the additional complexity. The core set of BPMN 1.2 has 14.3% excess, comparing to 17.9% for the core set of BPMN 2.0. The extended set of BPMN 1.2 has 57.1% excess, comparing to 78.6% for BPMN 2.0.

Based on the above analyses, we conclude that the extended set of BPMN 2.0 offers the lowest construct deficit, and therefore the highest ontological completeness. Ontological completeness is not 100%, as there are still nine BWW constructs that cannot be represented with the current version of BPMN. Despite that and its higher construct redundancy, overload, and excess, the extended set of BPMN 2.0 offers the most complete set of constructs to model scenarios in a real-world domain.

4.2 Representational Analysis of Rule Languages and Comparison with BPMN

Since we indentified that there are nine BWW constructs that cannot be represented with any of the constructs from the extended set of BPMN 2.0, we were further motivated to evaluate a few other languages that may offer representation for the remaining nine BWW constructs. BPMN 2.0 has almost no representation in the State cluster and no representation for a few of the other BWW constructs in the remaining three clusters. As this particular (State) cluster is important for modeling business rules, we further examined PRR, R2ML, SWRL business rule languages, and a hybrid language rBPMN, to determine if ontological completeness can be further improved. Table 2 shows the mappings of these languages compare to the extended set of BPMN 2.0.

Table 2. BPMN, PRR, R2ML, SWRL, rBPMN construct mapping

BWW Construct	BPMN 2.0 Ext	PRR 0.5	R2ML 0.5	SWRL 1.0	rBPMN
THING	+	-	+	+	+
PROPERTY	+	+	+	+	+
CLASS	+	+	+	+	+
KIND	+	-	-	-	+
STATE	-	-	+	-	+
CONCEIVABLE STATE SPACE	-	-	-	+	-
LAWFUL STATE SPACE	-	+	-	-	-
STATE LAW	-	+	+	-	+
STABLE STATE	-	-	-	-	-
UNSTABLE STATE	-	-	-	-	-
HISTORY	+	-	-	-	+
EVENT	+	-	+	-	+
CONCEIVABLE EVENT SPACE	-	-	+	-	+
LAWFUL EVENT SPACE	-	-	+	-	+
EXTERNAL EVENT	+	-	-	-	+
INTERNAL EVENT	+	-	-	-	+
WELL-DEFINED EVENT	+	-	-	-	+
POORLY-DEFINED EVENT	+	-	-	-	+
TRANSFORMATION	+	-	+	+	+
LAWFUL TRANSFORMATION	+	+	+	+	+
COUPLING	+	-	-	-	+
SYSTEM	+	+	-	-	+
SYSTEM ENVIRONMENT	+	-	-	-	+
SYSTEM COMPOSITION	+	+	-	-	+
SYSTEM DECOMPOSITION	+	-	-	-	+
SYSTEM STRUCTURE	-	-	-	-	-
SUBSYSTEM	+	-	-	-	+
LEVEL STRUCTURE	+	-	-	-	+
	19/28	7/28	10/28	6/28	23/28
	67.9%	25.0%	35.7%	21.4%	82.1%
EXCESS	+	+	+	+	+

For PRR 0.5 and SWRL 1.0, we included mapping results from zur Muehlen et al. [21], as they performed a similar comparison for these two particular language versions. We performed the mapping by following the methodology described in Section 3. From the overall ontological completeness perspective, as described earlier,

BPMN offers 67.9% completeness, PRR 25%, R2ML 35.7%, SWRL 21.4%, and rBPMN 82.1%. rBPMN offers the highest completeness, that is, combing BPMN and R2ML is beneficial from the overall ontological completeness perspective.

In terms of construct deficit, rBPMN has the lowest deficit of 17.9%. As indicated in Table 3, there are only five out of 28 BWW constructs that cannot be represented with rBPMN, which gives a construct deficit rate of 17.9%.

Table 3. Construct Deficit

Construct Deficit	**BPMN 2.0 Ext**	**PRR 0.5**	**R2ML 0.5**	**SWRL 1.0**	**rBPMN**
# of Constructs	9	21	18	22	5
Percentage	32.1%	75.0%	64.3%	78.6%	17.9%

The extended set of BPMN 2.0 also offers a relatively low construct deficit of 32.1%. R2ML has a construct deficit of 64.3%, followed by PRR with 75% and SWRL with 78.6%. Fig. 2 illustrates this comparison. The y axis is the number of BWW constructs that cannot be represented by a given language listed on the x axis.

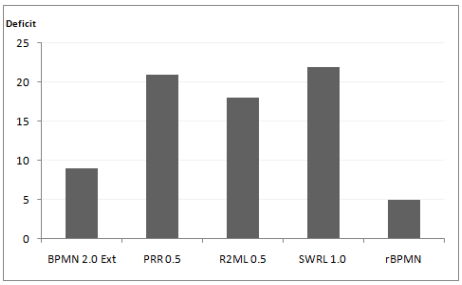

Fig. 2. Construct Deficit

In terms of construct excess, and as demonstrated in Table 2, all languages have construct excess, which means there is at least one language construct that does not map to any of the BWW constructs. From the cluster by cluster perspective, we notice that the cluster Thing is best represented with BPMN and rBPMN, which both offer a complete (100%) representation in this cluster. SWRL and R2ML have 75% representation, and PRR 50% representation. In the State cluster, BPMN and SWRL have 14.3% representation which is the lowest of all the analyzed languages. rBPMN offers the best representation of 42.9% followed by PRR and R2ML with 28.6% representation of the BWW state constructs. In the Event cluster, rBPMN is the only language that offers a complete (100%) representation of all event constructs, followed by BPMN, R2ML, SWRL, and PRR. In the System cluster, rBPMN and BPMN offer the highest representation of 85.7%, followed by PRR. Table 4 shows this comparison.

Overall, from the cluster-by-cluster perspective, rBPMN offers the highest representation in all four clusters, Thing, State, Event, and System among the five languages in comparison. Fig. 3 illustrates the results graphically. Y axis represents a percentage rate of representation for each of the four clusters per language studied.

Table 4. BWW Cluster Representation

Cluster	BPMN 2.0 Ext	PRR 0.5	R2ML 0.5	SWRL 1.0	rBPMN
Thing	100.0%	50.0%	75.0%	75.0%	100.0%
State	14.3%	28.6%	28.6%	14.3%	42.9%
Event	80.0%	10.0%	50.0%	20.0%	100.0%
System	85.7%	28.6%	0.0%	0.0%	85.7%

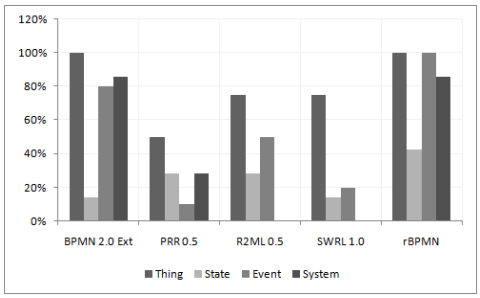

Fig. 3. Cluster by Cluster Comparison

4.3 Representational Analysis of Paired Business Process and Rule Languages

Based on the results described so far, we can conclude that rBPMN offers the highest degree of ontological completeness. Although rBPMN does not offer a complete representation capability, we can already conclude that combining business process and business rule modeling languages is beneficial – as demonstrated with rBMPN. Since rBPMN combines business process modeling language and business rule modeling language, namely BPMN 2.0 and R2ML 0.5, and we discovered that rBPMN offers better completeness than each of the two languages alone, we were interested to evaluate if this language pair also offers the best completeness among any given combination of business process and business rule modeling languages that we studied.

We further evaluated the following three language pairs: BPMN + PRR, BPMN + SWRL, and rBPMN (BPMN + R2ML). As per [21], we performed the overlap analysis and calculated symmetric difference, intersection, and relative compliment. With symmetric difference, we can determine the number of BWW constructs that are represented with no overlap for a given language combination (PΔR). With intersection, we look at the number of BWW constructs which can be represented additionally (with overlap) by a particular language pair (P$\cap$R). To determine a relative complement, we look at how many non-overlapping BWW constructs were contributed by a business process modeling language to a business rule process modeling language (P\R) and how many non-overlapping constructs were contributed by the business rule language to the business process language (R\P). Table 5 shows the results.

Based on the results in Table 5, BPMN+R2ML (rBPMN) offers 17 distinctively represented BWW constructs free of overlap. This is the highest number of the three language combinations. The BPMN+PRR combination has 16, while BPMN+SWRL has 15 BWW constructs. The second column in Table 5 displays the intersection.

rBPMN has 6 constructs that can be additionally represented by both languages, comparing to 5 for BPMN+PRR and BPMN+SWRL. In terms of relative complement P\R and R\P, rBPMN represents 13 non-overlapping BWW constructs contributed by BPMN, and 4 constructs represented by R2ML. Fig. 4 illustrates this comparison.

Table 5. Overlap Analysis

Language Pair	PΔR	P∩R	P\R	R\P
BPMN 2.0 + PRR 0.5	16	5	14	2
BPMN 2.0 + SWRL 1.0	15	5	14	1
BPMN 2.0 + R2ML 0.5 (rBPMN)	17	6	13	4

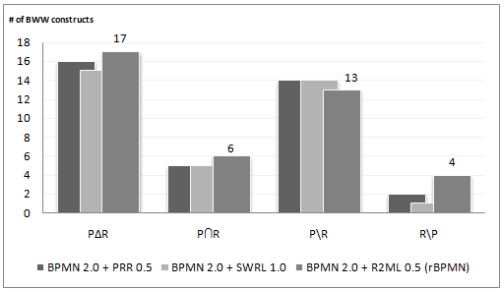

Fig. 4. Overlap Analysis

Based on our analysis, the language combination of BPMN and R2ML (rBPMN) is the most desirable language combination not only because it represents the highest number of distinct non overlapping constructs, but also the highest number of constructs that can be additionally represented with the overlap. In fact, this particular language combination is the best choice also because R2ML contributes the highest number of BWW constructs to BPMN comparing to any other language pair studied.

We cannot compare the results of our analysis for the best performing language combination (i.e., rBPMN) to the results of work presented in [21]. From the ontological completeness perspective, zur Muehlen and Indulska's best hypothetical pair, BPMN 1.0 and Simple Rule Markup Language (SRML) represents 23 constructs out of 29 used in their reference BWW model (79%). In our case, rBPMN represents 23 out of 28 included in our model (82%). zur Muehlen and Indulska's analysis also contained the "Acts on" concept in the BWW model, which we did not as already explained in footnote 2. If we consider only the 28 concepts covered in the BWW model that we used, it appears that the BPMN 1.0 + SRML combination covers 22 concepts. Two BWW concepts, "Conceivable State Space" and "Lawful State Space" are only represented in BPMN 1.0+SMRL, and three BWW concepts "History", "Conceivable Event Space', and "Lawful Event Space" are only represented by rBPMN. Out of last three, two are contributed by R2ML and one by BPMN 2.0.

The best performing combination of languages (R2ML + BPMN) analyzed in our study might have important implications for tool developers comparing to the case of BPMN 1.0 and SRML. Both constitutive languages of rBPMN are developed by using model-driven engineering principles. That fact already offers mechanisms for a

solid basis of type safety and static semantic analysis needed for an effective language use. If BPMN were combined with a rule language which is designed to serve as a markup and only represented in an XML format, then there would a need to invest additional efforts in tool development to also overcome issues of different language definition mechanisms (e.g., Ecore and XML Scheme).

5 Conclusion and Future Work

A limitation of our study is that we have not included an analysis of the Rule Interchange Format (RIF) and Object Constraint Language (OCL), as they are important standards. Our on-going work takes these languages into consideration. Considering the types of language constructs supported in OCL and RIF, we can hypothesize that there will hardly be better coverage support comparing to rBPMN. This is due to the fact that RIF and OCL do not have a full support for state modeling, similar to the R2ML language. Yet, a thorough representation is to be conducted to investigate all other characteristics of RIF and OCL once it is combined with BPMN and/or other process modeling language. It will also be very important to analyze other rule standards such as Semantics for Business Vocabularies and Rules (SBVR). Furthermore, our future work will more thoroughly compare the results of our analysis with the results of representational analysis reported in previous reports [21] and [22].

From our analysis (see Table 2) as well as from the previous work [21], it appears that modeling state space is the major source of incompleteness w.r.t. the BWW model of the current process and rule modeling languages. While this might be a problem for developing some types of systems, this might not be the major point of concern for others such as service-oriented systems where statelessness is one of the main premises. Still, to have a complete support, the future languages might consider support for state space modeling. Some types of rules such as reactive and production might be a very good basis for this support in future research.

References

1. OMG, Business Process Modeling Notation, Final Adopted Specification, Online, Available: http://www.omg.org/docs/formal/09-01-03.pdf, (2009).
2. OMG, Business Process Model and Notation (BPMN), v2.0 - FTF Beta 1, Online, Available: http://www.omg.org/cgi-bin/doc?dtc/09-08-14.pdf, (2009).
3. OMG, Production Rule Representation (PRR), Specification, formal/2009-12-01, Online, Available: http://www.omg.org/spec/PRR/1.0/PDF/ (2009).
4. H. Ian, P. Peter F., B. Harold, T. Said, G. Benjamin, and D. Mike: SWRL: A Semantic Web Rule Language, Combining OWL and RuleML, W3C Member Submission, (2004).
5. Rewerse Working Group I1, "R2ML - The REWERSE I1 Rule Markup Language," Online, Available: http://oxygen.informatik.tu-cottbus.de/rewerse-i1/?q=node/6.
6. Milanović, M., Gašević, D.: Towards a Language for Rule-enhanced Business Process Modeling, IEEE Int'l Enterprise Distributed Object Computing Conf., pp. 64-73. (2009).
7. Recker, J., Indulska, M., Rosemann, M., Green, P.: How good is bpmn really? Insights from theory and practice, 14th European Conference on Information Systems, pp. 1-12, (2006).

8. Nemuraite, L., Ceponiene, L., Vedrickas, G.: Representation of Business Rules in UML&OCL Models for Developing Information Systems, Proc. of 1st IFIP WG 8.1 Working Conf. Practice of Enterprise Modeling, pp. 182-196, (2008).
9. O`Connor, M., Knublauch, H., Tu, S., Musen, M.: Writing Rules for the Semantic Web Using SWRL and Jess, 8th Int'l Protégé Conference, Protege with Rules Workshop, (2005).
10. Milanović, M., Gašević, D., Wagner, G.: Combining Rules and Activities for Modeling Service-Based Business Processes, In Proc. EDOC 2008 International Workshop on Models and Model-driven Methods for Enterprise Computing, (2008).
11. Bunge, M.,: Treatise on Basic Philosophy: Volume 3: Ontology I: The Furniture of the World, Springer, (1977).
12. Gašević, D., Djurić, D., Devedzić, V.: Model Driven Architecture and Ontology Development, Springer, (2006).
13. Wand Y., Weber, R.: On the ontological expressiveness of information systems analysis and design grammars, Science And Technology, vol. 3, pp. 217-237, (1993).
14. Wand, Y., Weber, R.: On the deep structure of information systems, Information Systems Journal, vol. 5, pp. 203-223, (1995).
15. Weber, R.: Ontological Foundations of Information Systems, Melbourne, Australia: Coopers & Lybrand and the Accounting Association of Australia and New Zealand, (1997).
16. Green, P., Rosemann, M.: Ontological analysis of business systems analysis techniques: Experiences and proposals for an enhanced methodology, Business Systems Analysis with Ontologies, P. Green and M. Rosemann, Idea Group Publishing, pp. 1-27, (2005).
17. Keen, C., Lakos, C.: An Analysis of the Design Constructs Required in Process Modelling, the 1996 Int'l Conference on Software Engineering: Education and Practice , (1996).
18. Green, P, Rosemann, M., Indulska, M: Ontological Evaluation of Enterprise Systems Interoperability Using ebXML, IEEE Trans. on Knowledge and Data Eng., vol. 17, (2005).
19. Green, P., Rosemann, M., Indulska, M., Manning, C.: Candidate interoperability standards: An ontological overlap analysis, Data & Knowledge Eng., vol. 62, pp. 274-291, (2007).
20. zur Muehlen, M., Indulska, M., Kamp, G.: Business Process and Business Rule Modeling Languages for Compliance Management: A Representational Analysis, In Tutorials, At the 26th international Conference on Conceptual Modeling, pp. 127-132, (2007).
21. zur Muehlen, M., Indulska, M.: Modeling languages for business processes and business rules: A representational analysis, Information Systems, vol. 35, pp. 379-390, (2009).
22. Recker, J. Indulska, M., Rosemann, M., Green, P.: Do Process Modelling Techniques Get Better? A Comparative Ontological Analysis of BPMN, 16th Australasian Conference on Information Systems, Sydney, Australia, (2005).
23. Rosemann, M., Green, P., Indulska, M.: A Reference Methodology for Conducting Ontological Analyses, 23rd International Conference on Conceptual Modeling, S.Z. H. Lu, W. Chu, P. Atzeni, Shanghai, China: Springer Verlag Berlin Heidelberg, pp. 110-121, (2004).
24. Steinke, G., Nikolette, C.: Business rules as the basis of an organization's information system, Industrial management + Data Systems, vol. 103, p. 52, (2003).
25. UML-based Rule Modeling Language, Online, Available: http://oxygen.informatik.tu-cottbus.de/rewerse-i1/?q=node/7.
26. Graml, T., Bracht, R., Spies, M.: Patterns of business rules to enable agile business processes. Enterprise IS, vol. 2, no, 4, pp. 385-402 (2008).
27. Recker, J., Rosemann, M., Indulska, M., Green, P.: Business process modeling : a comparative analysis. J.the Association for Information Systems, 10(4). pp. 333-363 (2009).
28. Opdahl, A. L., Henderson-Sellers, B.: Ontological Evaluation of the UML Using the Bunge–Wand–Weber Model. Weber Model, Soft. & Sys. Modeling, 1(1), pp. 43-67 (2002).

A bridge between legislator and technologist - Formalization in SBVR for improved quality and understanding of legal rules

Åshild Johnsen*, Arne-Jørgen Berre#

*Department of Informatics University of Oslo, Norway, ashild.johnsen@finanstilsynet.no
#SINTEF, Norway, Arne.J.Berre@sintef.no

Abstract. The paper reports on how the two separate worlds of legislator and IT-technologist can be bridged through the formalization of legal rules with SBVR. The legislator can use SBVR to transform legal rules expressed in natural language to legal rules expressed in controlled natural language. During the transformation, impreciseness and inconsistence in the law formulation may be revealed and entail improved quality of the law formulation. The institutions implementing the legal rules can use SBVR documents published by the legislator to save time in their analyzing phase and even to automate the transformation from vocabulary and rules in SBVR to vocabulary and rules in a business rules management system (BRMS). The more institutions that are affected by the same legislation the more time and effort will be saved.

Keywords: SBVR, legal rules, legislator, interpreting, semantic ontology

1 Introduction

This paper reports on the conclusions of a study on formalization of legal rules with SBVR [Johnsen 2010]. Business rules are built on internal business rules, external business rules according to branch standards and legislation. Legislation consists of laws and regulations and represents legal rules the institutions are obligated to implement. Legal rules can be concrete, complex and detailed and be obvious candidates for automated proceeding by IT-solutions. Some laws and regulations affect several institutions. Each of them has to analyze and implement the legislation into their IT-systems. This paper is about using [OMG 2008] Semantics of Business Vocabulary and Business Rule (SBVR) on this kind of legislation.

The main idea of the study reported here is to let the legislator who is the author of the legislation, do the interpretation of the legislation. In other word, let the legislator use SBVR to transform from natural language to controlled natural language. SBVR represent a basic non-commercial standard for transforming natural language to

formal language, and you do not have to be an IT-technologist to use it. By transferring the interpretation from the institutions to the legislator, each of the institutions implementing the law would save time. In addition the SBVR modeling could reveal impreciseness in the law text that should be revised. The legislator is the one who knows the legal content of the law text and is in position to adjust it and in this way reduce need for later adjustments. The laws and regulations should still exist in natural language, but be supplemented with the SBVR formulations, as a kind of approved guidance.

Section 2 describes the case, section 3 summarizes findings from the case using examples in SBVR and section 4 is a suggestion for future work.

2 Case description

To examine whether laws could be formed in a way making them more consistent and easier to implement in IT-solutions, and whether SBVR could be used to do it, SBVR was tried to model three cases from the law and the findings evaluated.

All cases were selected from Norwegian Law [LovData] the "National Insurance Act of 28 February 1997":

Case 1: § 2 "Membership in the National Insurance" modeled into a SBVR document with vocabulary and a rule set containing 17 rules.

Case 2: § 3-34 - § 3-26 with supplementary regulation § 1 - § 4 "Supplement for spouse and supplement for children" modeled into a SBVR document with vocabulary and a rule set containing 17 rules.

Case 3: § 19-8 - § 19-9 "Lowest pensionlevel" modeled into a SBVR document with vocabulary and a rule set containing 7 rules.

The cases pay special attention on legislation regulating the insurance- and pension field which is a branch with complicated rules which applies to both governmental and private institutions. However the same principles could be applied to legislation in other rule-intensive areas as custom services, tax services, welfare administration, public road services and so on.

Legal rules convenient for the kind of reformulation described in this paper, have two characteristics:

- It contains rules to decide if a person fill the requirement to receive a benefit or pay a tax

- It contains rules to calculate size of the benefit or the payment

Extracting rules from legislation is often tedious work, whoever does it. This is familiar to IT people. It is often about structuring details, so also here. The case descriptions had to be quite detailed to show how the law content could be converted to SBVR compliant vocabulary and rules. The findings in section 3 are illustrated with examples from the first of the three cases which concerns membership in the National Insurance.

To our knowledge SBVR has not been used on texts written in Norwegian before, so the modeling started with defining SBVR Structured Norwegian. In this paper, however, the examples are translated into SBVR Structured English. Source is used to connect the rule to corresponding subsection of the act. One sentence in the act, may give several rules.

3 Evaluation of findings from using SBVR to transform legal rules

Citations from the law are in frames and refer to the "National Insurance Act of 28 February 1997". The examples are picked among more similar examples.

As an introduction to the findings, I briefly draw up the main principle for membership in the National Insurance in Norway: To be entitled to receive benefits in accordance with the National Insurance Act, one has to be or to have been a member in the national insurance. Some benefits require that you are a member at the time a situation, which can invoke the benefits occur. For other benefits it is enough to have been a member of the National Insurance. The size of the benefits is often proportional to the duration of the membership.

The findings can be grouped in three divisions:

3.1 Findings related to clarify and improve the law text

3.1.1 Reveal unambiguous formulations

Legal rules can be understood in different ways and be a basis for alternative rules. Example:

§ 2–1. People who are Norwegian residents, subsection 2

"Norwegian resident is defined as one who is staying in Norway, when the stay is intended to last or has surpassed 12 months. A person who moves to Norway is considered a resident from the date of arrival."

The formulation "is intended to last" is imprecise. The formulation considers something that could happen in the future. Should a stay that was intended to last at least 12 months, but in the end lasted less than 12 months, be a period in a membership in the National Insurance or not? The formulation is ambiguous, and two alternative rules could be possible:

Rule alternative 1: The residence in Norway doesn't give any temporary membership in the National Insurance:
If a person *moves to* **Norway** and *has* residence permit and the resident *is intended to last* at least **12** months and the resident *lasts* less than **12** months, then the resident *do* not *gives* a member period *in* membership in the national insurance
Source: § 2-1 subsection 2

Rule alternative 2: The residence in Norway gives a membership in the National Insurance even if the stay lasted less than 12 months
If a person *moves to* **Norway** and *has* residence permit and the resident *is intended to last* at least **12** months and the resident *lasts* less than **12** months, then the resident *gives* a member period *in* membership in the national insurance *with* start date *equal* date of arrival and end date *equal* date of departure
Source: § 2-1 subsection 2

If the legal rules were supplemented with rules formulated with SBVR, one of the alternatives would be chosen and thereby eliminating any doubt about what was the correct interpretation.

3.1.2 Reveal and clarify imprecise formulations

Parts of the law text need to be made more precise before it is possible to deduce rules from it. Often definitions in the vocabulary contribute to the necessary preciseness. Example:

§ 2–1. People who are Norwegian residents, subsection 4 second part

"However, this does not apply if the person concerned has stayed or will be staying abroad for more than six months per year for two or more consecutive years."

It is difficult to understand the exact meaning of "per year for two or more consecutive years" which could be interpreted in two different ways:

a) one year starts at date of departure and lasts 12 months from this date
b) one year is a calendar year

In the vocabulary both year and calendar year were defined:

year

Definition:	period *with* duration **12** months
Supporting fact types:	year *has* start date
	year *has* end date
	year *has* duration 12 months

calendar year

Definition:	year in accordance with the Christian era
Supporting fact types:	calendar year *starts* **01 January**
	calendar year *ends* **31 December**
	calendar year *has* duration **12** months

In the rule beneath year as defined in the vocabulary is chosen and a necessary preciseness is achieved:

If a member of the national insurance *on* actual date *has stayed* at least **6** months abroad each year *for* **2** or more consecutive years, then the person *is* not member of the national insurance *from* actual date

Source: § 2-1 subsection 4 second part

3.1.3 Identify certain kind of formulations to avoid

Parts of the law text embrace two different time aspects i.e. past and future in the same formulation. This is an example of a kind of formulation that should be avoided in laws dealing with detailed regulations because it makes exact formulation difficult. Formulations like these should be placed on some sort of 'black-list' collecting no-wanted kinds of formulations.

Example (same law text as in example A.2):

§ 2–1. People who are Norwegian residents, subsection 4 second part

"However, this does not apply if the person concerned has stayed or will be staying abroad for more than six months per year for two or more consecutive years."

The law formulation hides two different circumstances/conditions: One about the future and one about the past.

Rule a. refers to the future: "However, this does not apply if the person concerned **will be staying** abroad for more than six months a year for two or more consecutive years.":

Rule a.
If a member of the national insurance *moves from* **Norway** and *will be staying* at least **6** months abroad each year *for* **2** or more consecutive years, then the person *is* not member of the national insurance *from* date of departure

Source: § 2-1 subsection 4 second part

Rule b. refers to the past: "However, this does not apply if the person concerned **has stayed** abroad for more than six months a year for two or more consecutive years.":
Rule b.
If a member of the national insurance *on* actual date *has stayed* at least **6** months abroad each year *for* **2** or more consecutive years, then the person *is* not member of the national insurance *from* actual date

Source: § 2-1 subsection 4 second part

Rules a. and b. give different results. § 2-1 subsection 4 second part could possibly be interpreted as that you could stay as a member of the National Insurance for two more years if you do not plan in advance the duration of your stay abroad.

3.1.4 Contribute to reformulation of the law text itself
Generating vocabulary and rules has, besides the findings in 3.1.1 and 3.1.2 which were clarified through the supplementary SBVR-formulations, enforced amplifications that have entailed suggestions to reformulations of the law text itself. This was maybe the most surprising finding. In all three cases the transforming to SBVR enforced alternative, often more precise, formulation of the law text itself. Example:

§ 2-5 deals with persons abroad who according to their kind operations abroad, still could be member of the National Insurance. Focus is on the text in italic.

§ 2-5, second subsection third part

"The spouse of a person mentioned in first section letter c to f, *must have been a member of the National Insurance for at least three of the last five calendar years."*

First, what is the meaning of "***at least** three of the last five calendar years*"? It could be interpreted as that member of the National Insurance is something you are a whole calendar year, but the rest of the law doesn't indicate that. Does it mean that it is sufficient having a membership that includes periods that are in connected with at least three of the last five years, but that the duration of the periods are not important? Instinctively I believe what is meant, is that the sum of the duration of the member periods should be at least three years in the course of the last five years. A more

precise formulation could be “in sum at least three years during the last five calendar years”.

Second, what is the meaning of *“at least three of the **last** five calendar years”*. Last five calendar years from when? § 2-5 is about residence abroad. I suppose it means actual date measured against date of departure for the residence abroad. It could have been formulated more precisely by ”at least three of the last five calendar years on the date of departure for the residence abroad”.

Third, what is the meaning of *“at least three of the last five **calendar** years”*. If the residence abroad starts 1 October, what is the last calendar year (or last five calendar years)? Is it 2009? Is it 2008? Is it the last twelve months from 1 October 2008 – 1 October 2009? In this case ”at least three of the last five years on the date of departure for the residence abroad” would have been a more precise formulation.

To summarize, the law text could be rephrased to the following text which is possible to deduce a rule from:

§ 2-5, second subsection, third part - rephrased

The spouse of a person mentioned in first section letter c to f*, must have been a member of the National Insurance for at least a total duration equal to three years during the last five years at the time the stay abroad is initiated.*

3.2 Findings related to SBVRs ability to express the law text in a way making it easier to transform the law text into programming code

3.2.1 Unambiguous definitions of terms

A SBVR compliant vocabulary supports definition of terms at any level. Depths in the language can be expressed by recursively defined terms in SBVR. Here is a simple example:

date
Definition: time that is to the precision of year-month-day

start date
Definition: date something starts

end date
Definition: date something ends

period
Definition: a time interval measured from a start date to an end date

Supporting fact types: period *has* start date
period *has* end date
period *has* duration
Necessity: start date *is before* end date

member period
Definition: period that *is included in* a membership in the national insurance

membership in the national insurance *includes* at least **1** member period
Concept type: partitive-fact-type

The concept member period inherits the attributes of period. member period recursively refers period which recursively refers start date which recursively refers date.

3.2.2 Rich language that ensures common understanding

Precise formulations are essential for common understanding of the same law. In addition to the definitions in the vocabulary, SBVR offer statements and rules, supplemented by fact types and synonymous formulations, all based on logical formulations. Example:

If a person *moves to* **Norway** and *has* residence permit and the resident *is intended to last* at least **12** months, then *the* person *is* member of the national insurance *from* date of arrival

Synonymous Form: It is not necessary that a person who *moves to* **Norway** and *has* residence permit and the resident *is intended to last less than* **12** months, *is* member of the national insurance
Source: § 2-1 subsection 2
Note: In this case a person can apply for voluntary membership in the national insurance in accordance with **National Insurance Act** § 2-7

3.3 Supporting different usage groups

3.3.1 Practicable to exchange between institutions

The 'SBVR-format' with the characteristic font, colour- and indent- combination, is easy to identify and is itself, without any transformation, suitable for exchanging. However one of the main purposes with SBVR was to exchange SBVR documents between institutions, and the SBVR standard includes a specification of transforming SBVR to xml-format for exchanging by XMI [OMG 2008].

3.3.2 Simple to use

You do not have to be an IT-technologist to be a SBVR modeler. To be able to do logic analysis is the most important. The legislator has the major advantage by the fact that he knows the intention with the legislation. This will make it easier to transform it into rules. Anyway, an introduction- and education program in SBVR would be helpful, if not essential.

3.3.3 Supporting tools

SBVR can be used without any tools. However using tools can ensure consistence in vocabulary and rules difficult to gain manually. There are SBVR-compliant editors as [RuleArts] FactXpress and RuleXpress. Through these tools, it will also be possible to interface with BRMS and gain an automated transformation from legal rules to business rules.

4 Conclusion

SBVR modeling can be a useful method both for legislator formulating the law text and for those who interpret and implement it. For legislator, formulating the law, modeling with SBVR can represent a quality assurance of the law formulation and reduce need for later adjustments. For the institutions implementing the law, the result of the SBVR modeling, the vocabulary and the rules, can contribute to a smoother interpretation and transforming of the law and reduce time spending in the analyze phase. The more institutions affected by the same legislation, the more time to be saved. The SBVR documents constitute in this way a semantic ontology, a common interpretation all recipients can adapt to their IT-solutions.

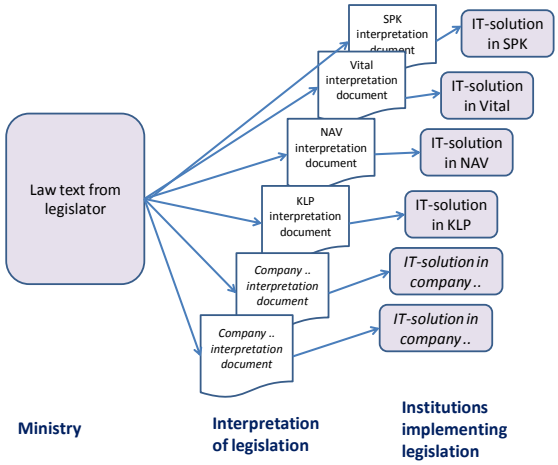

Fig. 1 The situation today – each company do their own interpretation

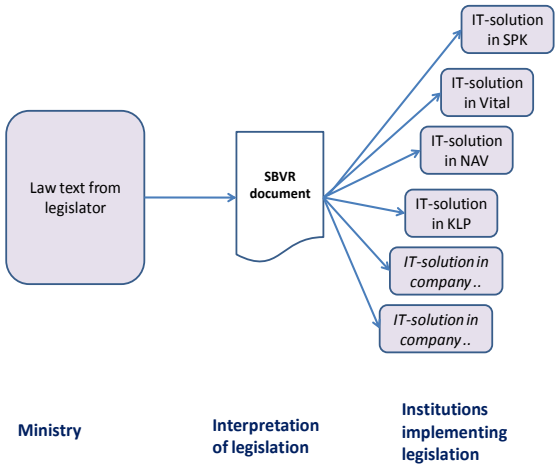

Fig. 2 Tomorrow - SBVR represents a common interpretation as a semantic ontology

5 Future work

A suggested next step is to try out this way of using SBVR on 'both side of the table' i.e. to form a business case where both legislator and institution (one or more) participate and evaluate their part in it.

The legislator has to:

- Choose either a new or an adjustment to a law text
- Based on the law text, model vocabulary an rules according to the SBVR specification
- Evaluate SBVR's effect on the formulation of the law text itself
- Evaluate time spending on modeling the law text with SBVR
- Publish the SBVR-documents in addition to the law

To carry out this part of the business case, an education program in SBVR specially prepared for the non-IT-technologist i.e. the legislator should be available.

The institution(s) have to:

- Evaluate effect on time spending in the analyze phase both in the business- and the IT-department

- Transform vocabulary and rules to a BRMS or other kind of IT-solution
- Evaluate effect on time spending in the implement phase

We are currently working with plans for a further exploration of this approach within Norwegian legislation, and are also considering opportunities for this within the European laws that affect Norway and all other countries in Europe.

References

[OMG 2008] OMG, "Semantics of Business Vocabulary and Business Rules (SBVR) Specification, Version 1.0, Volume 1", Object Management Group, OMG Document Number: formal/2008-01-02, January 2008. http://www.omg.org/spec/SBVR/1.0/

[RuleArts] "Product: FactXpress, RuleXpress" (FactXpress/RuleXpress) http://www.rulearts.com/ (accessed 2010)

[LovData] "LOV-1997-02-28-19 Folketrygdloven - ftrl. Lov om folketrygd (National Insurance Act of 28 February 1997). (1997-02-28) " http://www.lovdata.no/ (accessed 2010)

[Johnsen 2010] (Semantisk modellering av juridisk regelverk med bruk av SBVR - en brobygger mellom jus og IT), Åshild Johnsen, master thesis, Department of Informatics, University of Oslo, August 2010.

Adventures of Two Little OWLs in Rule Land

Markus Krötzsch, Oxford University Computing Laboratory

Abstract. Combining ontological and rule-based modelling can be an onerous task, from the choice of a suitable semantic framework (there are quite a few) to the selection of a chain of tools for supporting it (there are just a few). Typical solutions combine not only the advantages but also the difficulties of both domains, especially regarding computational complexity. For the recently introduced light-weight profiles of OWL 2, however, the situation is remarkably different. Here we find that existing rule-based systems can rather easily be adopted to support ontological inferencing using established algorithmic methods. This is well-known for OWL RL RL is for Rule Language after all but much less so for OWL EL.
In this talk, we take a closer look at this exciting grey area between light-weight ontologies and rules where both approaches are close enough to allow for an easy combination. We recall the features of OWL EL and RL, and explain how reasoning tasks in both languages can be answered by common rule systems with only a slight transformation of syntax. This approach uses rules as a computational formalism for implementing OWL reasoning without implying a semantic connection: even production rule systems could be used. Going further, we aim at a more intimate semantic combination of (logical) rules, OWL EL, and OWL RL, carefully tuned to allow efficient implementation in polynomial time. Further insights into matters of practical efficiency are gained from recent results on the worst-case space requirements of OWL EL inferencing, and from our experiences with the prototype implementation Orel.

Bio. Markus Krötzsch is a post-doctoral researcher at the Oxford University Computing Laboratory. He completed his PhD studies at the Institute of Applied Informatics and Formal Description Methods (AIFB) of the Karlsruhe Institute of Technology (KIT) in 2010. His research interest is the intelligent automatic processing of information, ranging from the foundations of formal knowledge representation to application areas like the Semantic Web. He is the lead developer of the successful Semantic Web application platform Semantic MediaWiki, co-editor of the W3C OWL 2 specification, chief maintainer of the semanticweb.org community portal, and co-author of the textbook Foundations of Semantic Web Technologies.

Combining Nonmonotonic Knowledge Bases for Modular and Distributed Knowledge-Based Information Systems

Thomas Krennwallner, Vienna University of Technology

Abstract. The developments in information technology during the last decade have been rapidly changing the possibilities for data and knowledge access. To respect this, several declarative knowledge representation formalisms have been extended with the capability to access data and knowledge sources that are external to a knowledge base. Such knowledge sources can come in various forms and may be as simple as a query interface to a database up to a full-fledged knowledge base.
In this talk we present two formalisms that that are centered around Answer Set Programming and have been designed with multiple knowledge bases in mind. One is modular nonmonotonic logic programs (MLP), which take up the issue of combining modules of logic programs into a coherent framework. The other formalism is multi-context systems (MCS), which are concerned with integrating knowledge from heterogeneous and possibly nonmonotonic knowledge bases (the contexts) using bridge rules, and combine them to a system with a semantics for contextual reasoning. We will argue that MLPs have the potential to host other formalisms that are relevant for the Semantic Web, like hybrid languages that combine ontologies and rules. MCS on the other hand are well-suited for distributed scenarios, where we can only assume an interface to contextualized knowledge bases—e.g., description logic or default theories—and do not get access to the actual content of the individual context. Heterogeneous nonmonotonic multi-context systems and modular nonmonotonic logic programs provide a basis for advanced knowledge-based information systems, which are targeted in ongoing research projects. They have been developed by the KBS group of the Vienna University of Technology in cooperation with external colleagues.
This work has been supported by the Austrian Science Fund (FWF) projects P20840 & P20841, the EC ICT Integrated Project Ontorule (FP7 231875), and the Vienna Science and Technology Fund (WWTF) project ICT08-020.

Bio. Thomas Krennwallner is a project assistant since June 2008 at the Institute of Information Systems at Vienna University of Technology (TU Wien), Austria, funded by the EU FP7 project "Ontorule" and the Austrian Science Fund project "Modular HEX-Programs." In 2007 and 2008, he was working as research intern at Digital Enterprise Research Institute Galway, Ireland, in the EU FP6 funded project "inContext." Between 1999 and 2004 he was a software developer in several companies. He has contributed to various software systems, most recently to DLVHEX, DMCS, GiaBATA, and XSPARQL. He is currently pursuing his PhD at the Knowledge-Based Systems Group at TU Wien, where he is developing extensions and algorithms for modular and distributed evaluation of HEX-programs, modular nonmonotonic logic programs, and heterogeneous nonmonotonic multi-context systems. He obtained a master's degree in Computational Intelligence in 2007 and a bachelor's degree in Software and Information Engineering in 2005, both at TU Wien.

Using OWL in Ontology-based data integration

Domenico Lembo, Sapienza Universita of Rome

Abstract. Data integration is the problem of providing a single interface and unified mechanisms to access data stored in several autonomous, possibly heterogeneous, information sources. This is a challenging task in many IT applications, such as enterprise information management and data warehousing, as well as in scenarios like e-science, e-government, and web data management. In the context of the Semantic Web, data integration has been often faced through the adoption of shared conceptualizations of the domain of interest referred to as ontologies, with the aim of posing the semantics of the application domain at the center of the scene. It is therefore interesting to analyze which are the implications of using ontologies in data integration, and in particular of adopting Semantic Web languages, such as OWL, within the traditional architecture for data integration. According to such architecture, a data integration system is composed by a global schema, which represents the interface towards the user, a source schema, which models all the sources to be integrated, and the mapping between the two.
In this talk, we consider data integration under this framework when the global schema is specified in OWL, and discuss the impact of this choice on computational complexity of query answering under different instantiations of the framework in terms of query language and form and interpretation of the mapping. As we will see, query answering in the resulting setting is in general computationally too complex, and some limitations on the expressive power of the various components of the framework has to be adopted in order to have efficient query answering. In particular, we will present OWL 2 QL, a tractable profile of OWL 2, and consider it as the ontology language used to express the global schema. OWL 2 QL essentially corresponds to a member of the DL-Lite family, a family of Description Logics designed to have a good trade-off between expressive power of the language and computational complexity of reasoning.
The results in this talk represent joint work with Diego Calvanese (Free University of Bozen/Bolzano), Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati (SAPIENZA University of Rome).

Bio. Domenico Lembo is assistant professor at the Department of Computer and System Sciences of the SAPIENZA University of Rome. His research interests concerns mainly information integration, Description Logics, Ontologies and the Semantic Web, inconsistency-tolerance in information systems. He authored more than 50 publications on the above topics in international journals and conferences. He is the author of several tutorials in the areas of data integration, ontologies, and the Semantic Web.

Incorporating Regulations, Business Rules and other Texts in the IT

François Lévy, Paris13 University

Abstract. Quite a lot of regulations available in texts impact everyday activities and are of interest in IT systems. Applicability of rights or duties to particulars (individuals or companies) is more easily answered with automated analysis of individual cases. Conformance of procedures to the many constraints that apply to organizations can benefit from automation (The conformance of organization procedures can be checked automatically, as soon as the relevant constraints are made explicit and formalized). This is the case for laws and regulations originating from official organizations as well as for internal regulations and business rules used in companies. Nevertheless, there is a critical bottleneck: analyzing regulatory texts to extract rules that the IT system will have to implement is still a challenging task. This is the topic of this talk.

A first part of the talk will focus on the general process leading from a source text written in Natural Language to a normalized set of rules and constraints that model the source policy and to its translation in a specialized formalism (either production rules, or a form of ontology *plus* logic programming for instance). The goal is to obtain a reformulation as close as possible of a controlled language, without loss of information.

After this general view, the various operations which participate in a progressive normalization of the source text will be inventoried and described, starting with the identification of relevant sentences. The challenge is to design tools to support an efficient computer aided transformation. We will present the rule editing environment that is currently being developped for that purpose, showing how ontology building, semantic annotation of the source text, semantic calculus, pattern-based analysis and index querying can help the task of human analysts.

Even if a formal evaluation of this aided modeling process is difficult to set up, we will consider the role of the natural language processing and knowledge engineering in the light of the life cycle of IT systems, showing the benefit of the backward traceability to source texts in different maintenance tasks.

Bio. François Lévy is full professor at Paris 13 university since 1993. He has been responsible of the french group "Natural Language semantics", working on logical representation of semantic and cognitive phenomena such as events, processes in narratives, and causality. He has also had some activities in default logic and in diagnosis.

The Entity-centric Organization

Heiko Stoermer, Fondazione Bruno Kessler, Trento, Italy

Abstract. Semantic technologies enable a shift from a schema-centric (or data centric) approach to data management in complex organizations to an entity-centric approach, where different data sources are viewed as potential providers of statements about relevant business entities (people, companies, products, locations, events, projects, etc.). We will argue that such a shift may enormously simplify the management of data and in particular their integration and exploration. This claim will be supported by a number of very concrete use cases where we have used this approach to solve very different issues and by showing why the entity-centric approach was better (along different dimensions) with respect to different and more traditional approaches.

Bio. Heiko Stoermer works as a researcher in the area of Entity-centric DIKM at Fondazione Bruno Kessler (FBK-irst), Trento, Italy. After studies in Linguistics he moved to the area of Computer Science, where he holds a German university degree. He gathered industry experience as a software consultant and project manager, and opted for an academic career in 2004 when he received a scholarship at the International Doctorate School in Trento. In 2008, he was awarded with a PhD for his work on Identity and Reference on the Semantic Web. His research interests include information integration, semantic interoperability and contextual knowledge representation. He has (co-) authored a number of scientific publications, acted as a reviewer for important international conferences, is co-organizer of several workshops, and steering committee member of the workshop series on Semantic Web Applications and Perspectives. Currently, he is devoting most of his time to his position as Technical Director of the European large-scale Integrated Project OKKAM, which he co-founded.