Are your rules online? Four web rule essentials

Boley, Harold

National Research Council Canada
Conseil national de recherches Canada

Institute for Information Technology
Institut de technologie de l'information

NRC · CNRC

Are Your Rules Online? Four Web Rule Essentials *

Boley, H.
October 2007

* published at the International RuleML Symposium on Rule Interchange and Applications (RuleML-2007). October 25-26, 2007. Orlando, Florida, USA. NRC 49875.

Canada

Are Your Rules Online?
Four Web Rule Essentials

Harold Boley (harold.boley AT nrc.gc.ca)

Institute for Information Technology – e-Business
National Research Council of Canada
Fredericton, NB, E3B 9W4, Canada

Abstract. Four principal Web rule issues constitute our starting points: I1) Formal knowledge representation can act as content in its own right and/or as metadata for content. I2) Knowledge on the open Web is typically inconsistent but closed 'intranet' reasoning can exploit local consistency. I3) Scalability of reasoning calls for representation layering on top of quite inexpressive languages. I4) Rule representation should stay compatible with relevant Web standards.
To address these, four corresponding essentials of Web rules have emerged: E1) Combine formal and informal knowledge in a Rule Wiki, where the formal parts can be taken as code (or as metadata) for the informal parts, and the informal parts as documentation (or as content) for the formal parts. This can be supported by tools for Controlled Natural Language: mapping a subset of, e.g., English into rules and back. E2) Represent distributed knowledge via a module construct, supporting local consistency, permitting scoped negation as failure, and reducing the search space of scoped queries. Modules are embedded into an 'Entails' element: prove whether a query is entailed by a module. E3) Develop a dual layering of assertional and terminological knowledge as well as their blends. To permit the specification of terminologies independent of assertions, the CARIN principle is adopted: a terminological predicate is not permitted in the head of a rule. E4) Differentiate the Web notion of URIs as in URLs, for access, vs. URNs, for naming. A URI can then be used: URL-like, for module import, where it is an error if dereferencing does not yield a valid knowledge document; URN-like, as an identifier, where dereferencing is not intended; or, as a name whose dereferencing can access its (partial) definition.

Introduction

Web rules constitute an effort towards novel Web sites with machine-interpretable rule representations for automated reasoning.

This research builds on our previous work in Web rule foundations (e.g., POSL[1], Datalog^{DL} [MLB+07b], ALC$^{u}_{P}$ [MLB07a]), standards (e.g., RuleML [Bol06], SWRL [HPSB+04], RIF [BK07]), engines (e.g., OO jDREW [BBH+05]),

[1] http://www.ruleml.org/submission/ruleml-shortation.html

as well as use cases (e.g., AgentMatcher [BBY04], NBBizKB [MB05], FindXpRT [LBBM06], Rule Responder [PKB07], and Ontology Integration [BHV06]). Its objective is to devise complementary rule representation and reasoning techniques for Business Rules, the Semantic Web, Web Services, as well as other Web (and Web 2.0) areas.

Previous research led to the following four principal Web rule issues used here as starting points.

I1 The Web increasingly develops 'Semantic Subwebs' containing knowledge documents (knowledge bases, schemas, etc.). A formal knowledge representation can act as content that is queried and retrieved in its own right, as metadata that helps to retrieve other formal or informal content, or as a combination of both [Bol99].

I2 The open Web as well as some closed 'intranets' contain knowledge documents. Open Web knowledge (knowledge distributed over an open-ended set of Web documents) in expressively rich representations is typically inconsistent whereas closed intranet knowledge (knowledge maintained in closed sets of Web documents) is typically *paraconsistent*, i.e. each intranet is kept (locally) consistent, although their union may be (globally) inconsistent. Thus, while open Web reasoning cannot directly use classical 2-valued logic, closed intranet reasoning can exploit it locally. Beyond benefiting from an implicit module notion through documents, locality can be achieved via an explicit module construct.

I3 Besides the consistency issue I2, the trade-offs between representation expressiveness and reasoning tractability imply that scalability of reasoning to the open Web is still unresolved for higher expressive classes. This calls for representation layering on top of quite inexpressive languages. W3C's Semantic Web stacks, old and new [KdBBF05], have made a start, and introduced RDF as the fundamental knowledge layer. However, the (official) XML syntax of RDF is somewhat complicated (although its triple syntax is extremely simple and its N3-like syntaxes are quite popular), and the (official) semantics of RDF [Hay04] is rather complicated [Sch05]. Yet, simple RDF statements without blank nodes (existentials) in assertions, queried without property variables (existentials for predicates), are a candidate for the least expressive (binary-)fact layer. Binary Datalog rules, similar to relational views over 2-column tables, can then be added to derive new facts from conjunctions of other facts, much like relational joins. Finally, an irreflexive `subClassOf` fragment of RDF Schema can be employed to define order-sorted types for constants and rule variables.

I4 Since Web rules are layered on, and side-by-side with, existing Web languages, it is important to represent rules so that compatibility with relevant Web standards (e.g., XML, RDF, OWL) is preserved. The selection of Web standards can be hard, for instance when considering which, if any, query and transformation languages should be included (e.g., XQuery, XSLT, SPARQL, OWL-QL)[2]. Likewise, the (syntactic and semantic) levels and degrees of compatibility with each of the selected languages need to be determined. Therefore, essential E4 will focus on what is unique to Web languages, namely 'webizing' [BL01],

[2] http://www.dajobe.org/talks/200603-sparql-stanford

basically permitting the Web rule language to use URIs for global constants, in ways compatible with URIs in existing (Semantic) Web languages.

To address these issues, the following sections will consider four corresponding essentials of Web rules. For a tutorial-style survey on Web rules see [BKPP07].

E1 Combining Logic Rules with Controlled English

We propose to combine formal and informal knowledge in a **Rule Wiki**, where *clauses* (rules and, as a special case, facts) are given dual representations, in a natural language and in a logic language. The formal parts can be taken as code (or as metadata) for the informal parts, and the informal parts as documentation (or as content) for the formal parts. On the level of a single rule, both representations can be kept separate (our assumption in the following) or can be intertwined. This combination is analogous to Literate Programming [Knu84] and Javadoc. It may be supported by tools mapping Controlled Natural Languages (e.g., Controlled English) into rules and back. For instance, two English-to-rule tools based on Attempto [FHK+05] are TRANSLATOR [Hir06] and AceRules [Kuh07]. Related tools have also been developed for AceWiki[3] and "Semantics of Business Vocabulary and Business Rules" (SBVR) [BHC05].

A classical Wiki permits the authoring of informal-knowledge documents using natural-language-enriching markup simpler than HTML. Extending this concept, a Rule Wiki permits formal-knowledge authoring using logic-language-enriching markup simpler than HTML or XML, combining this with informal-knowledge authoring. The formal-knowledge language can employ a human-readable syntax such as POSL[1], integrating the Prolog and F-logic syntaxes.

Let us consider an example. In a logistics use case, the ternary relation `reciship` can represent reciprocal shippings of unspecified cargos at a total cost between two sites. A Datalog rule infers this conclusion from three premises, two `shipment` atoms and an `add` atom. The `shipment` relation is based on slotted facts, and the `add` relation, on a SWRL built-in. Below is a Controlled English representation using MediaWiki[4] markup, where commutative premise conjuncts, the default in POSL, become "`*`"-unordered list bullets ("`#`"-ordered lists could be used for sequential conjunctions), predicate/built-in and slot names (both of which could be hyperlinked to their definitions, cf. section E4) are bold-faced with triple apostrophes, and variable names are italicized with double apostrophes; the logical connectives `if`, `and`, etc. are marked up with double brackets as internal links (possibly redirecting via interwiki links) to their explanatory pages. For the proposed **Rule MediaWiki**, this is followed by a POSL representation, which will also be employed and developed in subsequent sections. The POSL rule markup exemplifies the use of Prolog's "`:-`" infix between the conclusion and premises, top-level commas for separating conjuncts, intra-atom commas for ordered arguments and semicolons for unordered arguments,

[3] http://attempto.ifi.uzh.ch/acewiki
[4] http://en.wikipedia.org/wiki/MediaWiki

F-logic's infix "->" for slots, as well as the prefix "?" for named variables and a stand-alone "?" for anonymous variables:

```
A reciprocal shipping, '''reciship''', at total amount, ''cost'',
takes place between sites ''A'' and ''B'' [[if]]
* a '''shipment''' has a '''cargo''', a '''price''' of ''cost1'', a
'''source''' of ''A'', and a destination, '''dest''', of ''B'' [[and]]
* a '''shipment''' has a '''cargo''', a '''price''' of ''cost2'', a
'''source''' of ''B'', and a destination, '''dest''', of ''A'' [[and]]
* ''cost'' is the addition, '''add''', of ''cost1'' and ''cost2''.

 reciship(?cost,?A,?B) :-
   shipment(cargo->?;price->?cost1;source->?A;dest->?B),
   shipment(cargo->?;price->?cost2;source->?B;dest->?A),
   add(?cost,?cost1,?cost2).
```

Fig. 1 shows the actual rendering of the MediaWiki part, followed by the proposed rendering of its POSL extension, employing corresponding fonts in Controlled English and POSL.

A reciprocal shipping, **reciship**, at total amount, *cost*, takes place between sites *A* and *B* if
- a **shipment** has a **cargo**, a **price** of *cost1*, a **source** of *A*, and a destination, **dest**, of *B* and
- a **shipment** has a **cargo**, a **price** of *cost2*, a **source** of *B*, and a destination, **dest**, of *A* and
- *cost* is the addition, **add**, of *cost1* and *cost2*.

```
reciship(?cost,?A,?B) :-
  shipment(cargo->?;price->?cost1;source->?A;dest->?B),
  shipment(cargo->?;price->?cost2;source->?B;dest->?A),
  add(?cost,?cost1,?cost2).
```

Fig. 1. A Rule Wiki example.

For Web interchange, POSL can be automatically converted to RuleML/XML, and vice versa, using our online translator[5] via Java Web Start. The result of serializing the POSL example in RuleML/XML is shown in appendix A.1.

E2 A Distributed Rule Module Construct

It is beneficial to represent distributed knowledge via a module construct, supporting local consistency, reducing the search space of scoped (module-restricted) queries and permitting scoped negation as failure (over closed worlds).

Such Web modules may be written and used 'in place' or defined at one place (URL) and accessed from other places. The semantics of modules should not depend on any needed URL-dereferencing.

[5] http://www.ruleml.org/posl/converter.jnlp

The granularity of modules may vary: (1) It is quite possible to specify multiple modules within one knowledge document, even allowing different modules specified in the premise conjuncts of a single rule. (2) It is very natural to specify one module per (URL-accessed) document. (3) It is somewhat unusual to specify multiple documents constituting one module.

Versions of contexts/modules have been studied in AI [McC93] and Prolog [HF06], and are used in the Semantic Web as TRIPLE models [SD02], F-logic's scoped formulas [YK03], N3 formulae [BL06], and RDF named graphs [SvESH07]. In RuleML and its OO jDREW implementation, modules take the form of a `Rulebase` element, which is assumed within a top-level `Assert`.

Initially, a flat set of modules can be studied on the basis of McCarthy's `ist` operator to prove whether a query is true in a module. In recent RuleML releases, modules have been embedded into an `Entails` element, which serves to prove whether a query or module is entailed by another module.

This can then be extended to a nested (cycle-free) inheritance system of modules, as surveyed in [BLM94]. We only need to consider a simplified kind of module inheritance here, since by default we do not assume a Prolog-like textual order in a module's set of assertional (fact and rule) or terminological (subclass-ontological) clauses and therefore do not need to merge clauses but can just take their union.

Another simplification occurs if a module consists only of clauses for a single predicate (as in Prolog dubbed a *procedure*). For such a module, an explicit syntactic grouping element might be introduced, intermediate between the levels of a clause and of a general module. The granularity of procedures relative to general modules may vary similarly to the above granularity of general modules relative to documents.

As an example let us consider two modules, which could reside in a single document or (as serialized in appendix A.2) each constitute a separate document. Both contain rules granting or denying discounts, which could be extended with rules about giveaways etc. The first, of the special form of a `discount` procedure, might be called the `loyalty` module, as it grants discount percentages to loyal customers for certain products (module contents will be enclosed using "{...}"):

```
{
  discount(?customer,?product,percent[5]) :-
    premium(?customer),
    regular(?product).

  discount(?customer,?product,percent[10]) :-
    premium(?customer),
    luxury(?product).
}
```

Note that the percent sign and units of measurement are written as (uninterpreted) unary functions applied to a value. For such 'passive' functions, the arguments will be enclosed using "[...]", where the entire application forms an expression that acts as a complex term (in Prolog called a 'structure').

The second might be called the `legality` module, as it denies discounts to customers who used fraudulent payment methods or who payed more than 45 days after a delivery (a "-" prefix is used for the strong negation of atoms):

```
{
  -discount(?customer,?product1,?percent) :-
    payment(?customer,?product2,?amount,?method,?time),
    fraudulent(?customer,?method,?time).

  -discount(?customer,?product1,?percent) :-
    delivery(?customer,?product2,?amount,date[?y1,?m1,?d1]),
    payment(?customer,?product2,?amount,?method,date[?y2,?m2,?d2]),
    datediff(days[?delta],date[?y2,?m2,?d2],date[?y1,?m1,?d1]),
    greaterThan(?delta,45).
}
```

Note that dates are expressed here as complex terms of three arguments: the year, followed by the month, followed by the day.

These modules are locally consistent, although their union is inconsistent. For instance, according to the first rule of the `loyalty` module, premium customers would be granted 5 percent discount for a regular product, but, according to the first rule of the `legality` module, would be denied any discount on any product if they used a fraudulent payment method on a product. To deal with this, *prioritization* (cf. Courteous Logic Programs [Gro04] and Defeasible Logic [BAV04]) could be employed on the module level to let all rules of the `legality` module *override* all `loyalty` rules.

Using `Naf` as the primitive for negation-as-failure, and "`|-`" as the infix for the `Entails` element, a negated-as-failure *query* scoped by a *module* can be achieved via *module* `|- Naf(`*query*`)`. E.g., `loyalty |- Naf(giveaway(John,Mercedes))` is true, since the `loyalty` module does not entail that John obtains a Mercedes giveaway (or any other giveaway).

Scoping is also helpful for positive queries, since the search space can be reduced if the modules to be searched are known beforehand. For example, in the `loyalty` module, the queries in the premises of the `discount` rules can be sped up by restricting them to `customer` and `product` modules:

```
{
  discount(?customer,?product,percent[5]) :-
    customer |- premium(?customer),
    product |- regular(?product).

  discount(?customer,?product,percent[10]) :-
    customer |- premium(?customer),
    product |- luxury(?product).
}
```

Moreover, to check whether a rulebase *KB* obeys integrity constraints in *IC*, we use the approach of [Rei88] to check whether module *KB* entails module *IC*.

E3 Assertional-Terminological Expressiveness Layering

Quite an effort has been made to develop a dual expressiveness layering of assertional and terminological knowledge as well as their blends [ADG+05,KdBBF05].

To retain decidability of querying, the *assertional bottom layer* usually consists of Datalog (function-free) assertions, perhaps restricted to unary/binary predicates. For the *terminological bottom layer*, an irreflexive version of RDF Schema's `subClassOf` can be employed, which could later be extended towards the ρDF [MPG07] fragment of RDF. The two layers can be blended through a hybrid combination (ρDF classes used as types for Datalog constants and variables, and `subClassOf` defined with order-sorted semantics) or a homogeneous integration (ρDF classes used as unary predicates in the body of Datalog rules, and `subClassOf` defined as special rules with Herbrand-model semantics).

The higher layers can develop Datalog into Horn (as in OO jDREW's hybrid implementation) and FOL (First-Order Logic) assertions, ρDF into ALC and SHIQ terminologies with classes and properties, and appropriate blends [Ros06], e.g. as advancements of our hybrid Datalog^{DL} [MLB+07b] or homogeneous ALC^{u}_{P} [MLB07a]. For certain purposes, especially in the early modeling phases, the assertional layers can move even beyond FOL, including towards higher-order and modal logics, as started as part of the RuleML family [Bol06].

To permit the specification of terminologies independent of assertions, a hybrid approach is proposed here adopting the CARIN [LR98] principle as a working hypothesis: A terminological predicate is not permitted in the head of a rule. Intuitively, terminological classes cannot be (re)defined by assertional clauses, because a terminology establishes more stable 'background' knowledge extended by assertions that constitute more volatile 'foreground' knowledge.

Such a hybrid lower layer can use sort restrictions as simple terminological queries in Datalog rule bodies, which in higher layers are extended to terminological queries involving properties, ALC expressions, etc. In the spirit of [KdBBF05], this should lead to a more realistic Semantic Web architecture with simplified foundations and better computational properties. Our fine-grained bottom-up approach also complements the recent differentiation of OWL-Lite into OWL 1.1 Tractable Fragments [GCG+06].

The following example uses classes of a `subClassOf` terminology as variable sorts of slightly extended Datalog rules, namely of Horn logic rules employing (unary) functions only for measurement units.

The terminology forms a DAG that introduces `Vehicle`-rooted classes and exemplifies multiple inheritance of `MiniVan` from `Van` and `PassengerVehicle` (the "`>`" infix is used between a superclass and a subclass):

```
Vehicle > Van
Vehicle > PassengerVehicle
Van > MiniVan
PassengerVehicle > MiniVan
PassengerVehicle > Car
```

The rules (serialized in appendix A.3) specify registration fees for vehicles. The first rule specifies a vehicle variable sorted by the `Van` class, while the second refers to the `Car` class (the ":" infix is used between a variable and its sort):

```
registration(?V:Van,CAD[?R:Decimal]) :-
  emission(?V,CO2[?E]),
  weight(?V,kg[?W]),
  emiweight(CAD[?R],CO2[?E],kg[?W]).

registration(?V:Car,CAD[?R:Decimal]) :-
  emission(?V,CO2[?E]),
  speed(?V,kmh[?S]),
  emispeed(CAD[?R],CO2[?E],kmh[?S]).
```

A `registration` query for a given vehicle class will thus unify only with correspondingly sorted rule conclusions, hence directly branch into the appropriate rule premises (the `emiweight` and `emispeed` premises compute the fees from the emissions as well as the weights and speeds for `Van`s and `Car`s, respectively). Section E4 will show URI-'webized' versions of these terminological classes.

E4 URIs for Access, Naming, Or Both

There have been attempts to differentiate the Web notion of URIs into two subnotions, as discussed in [Hal06]: URLs (Uniform Resource Locators), for access, and URNs (Uniform Resource Names), for naming. This distinction is independent from the recent IRI (Internationalized) versions of URIs. In the context of Web knowledge representation, especially for Web rules as explored in POSL, RuleML, and RIF, three central URI uses are emerging, given here in the order of further needed research (orthogonal to research in URI normalization [Bol03]).

First, a URI can be used, URL/access-style, for module import (transitive import for nested modules), where it is an error if dereferencing the URI does not yield a knowledge base valid with respect to the expected representation language.

Second, a URI can be used, URN/naming-style, as the identifier of an individual constant in the representation language, where URI dereferencing is not intended as part of the formal knowledge representation. If dereferencing is attempted as part of the metadata about the informal knowledge representation, it should retrieve a 'homepage' of the individual; cf. section E1.

Third, a URI can be used, naming-style, as the identifier of a class, property, relation, or function, and at the same time, access-style, where dereferencing yields (a "`#`"-anchor into) a knowledge base formally defining that identifier (albeit perhaps partially only, as for an RDF Schema knowledge base just giving the superclasses of a class).

Here are examples for the three URI uses in connection with rules.

First, the `loyalty` module of section E2 could be imported into the current rulebase using the URL/access-style URI `http://modeg.org#loyalty`.

Second, the URI `http://en.wikipedia.org/wiki/Pluto` can be used URN/naming-style to refer to a celestial body originally considered a planet, as in this rule (serialized in appendix A.4) specifying its years of planethood (a URI is enclosed in a pair of angular brackets, <...>):

```
planet(<http://en.wikipedia.org/wiki/Pluto>,AD[?year]) :-
  lessThanOrEqual(1930,?year),
  lessThanOrEqual(?year,2006).
```

As part of the formal rule knowledge, the Pluto URI is employed only for naming. The rule can also be employed as metadata about informal knowledge through ('semantic search engine') queries like `planet(?which,2005)`, because one of its solutions will bind `?which` to the URI, whose dereferencing ('clicking') will then retrieve Pluto's Wikipedia entry.

Third, referring to the terminology in section E3, for certain formal purposes a URI like `http://termeg.org#MiniVan` is needed just to provide a name; for other formal purposes, also to provide a total or partial definition found by using that same URI access-style (say, the partial definition of being `rdfs:subClassOf` both `http://termeg.org#Van` and `http://termeg.org#PassengerVehicle`).

Conclusion

Four Web rule issues I1-4 led to corresponding essentials E1-4. These Web rule essentials are variously interrelated. For instance, a Rule Wiki (E1) for assertional knowledge can be extended with terminological knowledge (E3), both of which can be kept in distributed modules (E2) accessed by URIs (E4), where terminological classes may "`#`"-extend their module URIs for global naming (and for access to their definitions).

While the focus of this paper was on declarative rules for knowledge derivation, the four essentials can be *transferred* to (re)active rules for knowledge update, which have been increasingly studied in Web languages such as Reaction RuleML [PKB07] and Prova [Pas07]: The new event and action parts of these rules can also be beneficially combined with Controlled English (E1), modules are even more important for containing side-effects (E2), terminologies can be directly added to formalize both event and action vocabularies (E3), and all kinds of URIs are also crucial to (re)active Web rules and Web Services (E4).

Taken together, the Web rule essentials constitute a diamond-like system, $E2^1_4 3$, with URIs (E4) at the bottom, modules (E2) and assertional-terminological layers (E3) on the same level in the middle, and Controlled English (E1) at the top. The $E2^1_4 3$ system can be *augmented* by other design principles such as support for (Semantic) Web Services, APIs to (Web) databases, and interfaces with Web 2.0 tools. This paper is a kind of progress report at understanding $E2^1_4 3$, and further research will be needed to accelerate the evolution of Web rules as a natural extension to other kinds of Web information.

Acknowledgements

Thanks go to David Hirtle for developing the RuleML XML Schema Definitions (XSDs) and for suggesting to take advantage of wiki lists to separate each rule premise, which is now part of the Rule Wiki proposal in section E1. Further thanks go to Gregory Sherman for various Rule Wiki suggestions. Also, thanks to Benjamin Craig for extending OO jDREW and, along with Tshering Dema, for validating the examples in appendix A. Moreover, NSERC is thanked for its support through a discovery grant.

References

[ADG+05] Grigoris Antoniou, Carlos Viegas Damasio, Benjamin Grosof, Ian Horrocks, Michael Kifer, Jan Maluszynski, and Peter F. Patel-Schneider. Combining Rules and Ontologies – A Survey. Deliverables I3-D3, REWERSE, http://rewerse.net/deliverables/m12/i3-d3.pdf, March 2005.

[BAV04] Nick Bassiliades, Grigoris Antoniou, and Ioannis P. Vlahavas. A Defeasible Logic Reasoner for the Semantic Web. In Grigoris Antoniou and Harold Boley, editors, *RuleML*, volume 3323 of *Lecture Notes in Computer Science*, pages 49–64. Springer, 2004.

[BBH+05] Marcel Ball, Harold Boley, David Hirtle, Jing Mei, and Bruce Spencer. The OO jDREW Reference Implementation of RuleML. In Asaf Adi, Suzette Stoutenburg, and Said Tabet, editors, *RuleML*, volume 3791 of *Lecture Notes in Computer Science*, pages 218–223. Springer, 2005.

[BBY04] Virendra C. Bhavsar, Harold Boley, and Lu Yang. A Weighted-Tree Similarity Algorithm for Multi-Agent Systems in e-Business Environments. In *Proc. Business Agents and the Semantic Web (BASeWEB) Workshop*. Also in: Computational Intelligence, November 2004.

[BHC05] Donald E. Baisley, John Hall, and Donald Chapin. Semantic Formulations in SBVR. In *Rule Languages for Interoperability*. W3C, 2005.

[BHV06] Yevgen Biletskiy, David Hirtle, and Olga Vorochek. Toward the Identification and Elimination of Semantic Conflicts for Integration of Ontologies. In *Canadian Semantic Web. Semantic Web and Beyond: Computing for Human Experience*, pages 135–142. Springer, 2006.

[BK07] Harold Boley and Michael Kifer. RIF Core Design. W3C Working Draft, W3C, March 2007. http://www.w3.org/TR/2007/WD-rif-core-20070330/.

[BKPP07] Harold Boley, Michael Kifer, Paula-Lavinia Pătrânjan, and Axel Polleres. Rule Interchange on the Web. In *Reasoning Web 2007*, number 4636, pages 269–309. Springer, September 2007.

[BL01] Tim Berners-Lee. Webizing Existing Systems. World Wide Web Consortium, personal notes on: Design Issues – Architectural and Philosophical Points, http://www.w3.org/DesignIssues/Webize.html, May 2001.

[BL06] Tim Berners-Lee. Notation 3. World Wide Web Consortium, personal notes on: Design Issues – Architectural and Philosophical Points, http://www.w3.org/DesignIssues/Notation3, March 2006.

[BLM94] Michele Bugliesi, Evelina Lamma, and Paola Mello. Modularity in Logic Programming. *Journal of Logic Programming*, 19/20:443–502, 1994.

[Bol99] Harold Boley. ONTOFILE: Exterior and Interior Ontologies of File/HTTP URLs. In Hannu Jaakkola, Hannu Kangassalo, and Eiji Kawaguchi, editors, *Information Modelling and Knowledge Bases X*. IOS Press, Amsterdam, "Frontiers in Artificial Intelligence and Applications", Spring 1999.

[Bol03] Harold Boley. Object-Oriented RuleML: User-Level Roles, URI-Grounded Clauses, and Order-Sorted Terms. In *Proc. Rules and Rule Markup Languages for the Semantic Web (RuleML-2003)*, pages 1–16. LNCS 2876, Springer, October 2003.

[Bol06] Harold Boley. The RuleML Family of Web Rule Languages. In José Júlio Alferes, James Bailey, Wolfgang May, and Uta Schwertel, editors, *PPSWR*, volume 4187 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 2006.

[FHK+05] Norbert E. Fuchs, Stefan Höfler, Kaarel Kaljurand, Fabio Rinaldi, and Gerold Schneider. Attempto Controlled English: A Knowledge Representation Language Readable by Humans and Machines. In Norbert Eisinger and Jan Maluszynski, editors, *Reasoning Web*, volume 3564 of *Lecture Notes in Computer Science*, pages 213–250. Springer, 2005.

[GCG+06] Bernardo Cuenca Grau, Diego Calvanese, Giuseppe De Giacomo, Ian Horrocks, Carsten Lutz, Boris Motik, Bijan Parsia, and Peter F. Patel-Schneider. OWL 1.1 Web Ontology Language Tractable Fragments. W3C Member Submission, http://www.w3.org/Submission/owl11-tractable/, December 2006.

[Gro04] Benjamin N. Grosof. Representing e-Commerce Rules via Situated Courteous Logic Programs in RuleML. *Electronic Commerce Research and Applications*, 3(1):2–20, 2004.

[Hal06] Harry Halpin. Identity, Reference, and Meaning on the Web. WWW2006 Workshop on Identity, Reference, and the Web, http://www.ibiblio.org/hhalpin/irw2006/, May 2006.

[Hay04] Patrick Hayes. RDF Semantics. W3C Recommendation, February 2004.

[HF06] Rémy Haemmerlé and François Fages. Modules for Prolog Revisited. Rapport de recherche No. 5869, INRIA, Rocquencourt, http://hal.inria.fr/inria-00070157/en/, March 2006.

[Hir06] David Hirtle. TRANSLATOR: A TRANSlator from LAnguage TO Rules. In *Canadian Symposium on Text Analysis (CaSTA)*, pages 127–139, Fredericton, Canada, October 2006.

[HPSB+04] Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, and Mike Dean. Semantic Web Rule Language (SWRL). W3C Member Submission, http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/, May 2004.

[KdBBF05] Michael Kifer, Jos de Bruijn, Harold Boley, and Dieter Fensel. A Realistic Architecture for the Semantic Web. In Asaf Adi, Suzette Stoutenburg, and Said Tabet, editors, *RuleML*, volume 3791 of *Lecture Notes in Computer Science*, pages 17–29. Springer, 2005.

[Knu84] Donald E. Knuth. Literate programming. *The Computer Journal*, 27(2):97–111, 1984.

[Kuh07] Tobias Kuhn. AceRules: Executing Rules in Controlled Natural Language. In Massimo Marchiori, Jeff Z. Pan, and Christian de Sainte Marie, editors, *RR*, volume 4524 of *Lecture Notes in Computer Science*, pages 299–308. Springer, 2007.

[LBBM06] Jie Li, Harold Boley, Virendrakumar C. Bhavsar, and Jing Mei. Expert Finding for eCollaboration Using FOAF with RuleML Rules. In *Montreal Conference of eTechnologies 2006*, pages 53–65, 2006.

[LR98] Alon A. Levy and Marie-Christine Rousset. CARIN: A Representation Language Combining Horn Rules and Description Logics. *Artificial Intelligence*, 104(1–2):165–209, 1998.

[MB05] Anna Maclachlan and Harold Boley. Semantic Web Rules for Business Information. In *Proc. International Conference on Web Technologies, Applications, and Services (WTAS 2005), Calgary, Canada*. IASTED, July 2005.

[McC93] John McCarthy. Notes on Formalizing Context. In *IJCAI*, pages 555–562, 1993.

[MLB07a] Jing Mei, Zuoquan Lin, and Harold Boley. ALC^{u}_{P}: An Integration of Description Logic and General Rules. In Massimo Marchiori, Jeff Z. Pan, and Christian de Sainte Marie, editors, *RR*, volume 4524 of *Lecture Notes in Computer Science*, pages 163–177. Springer, 2007.

[MLB+07b] Jing Mei, Zuoquan Lin, Harold Boley, Jie Li, and Virendrakumar C. Bhavsar. The $\mathrm{Datalog}^{DL}$ Combination of Deduction Rules and Description Logics. *Computational Intelligence*, 23(3):356–372, 2007.

[MPG07] Sergio Muñoz, Jorge Pérez, and Claudio Gutiérrez. Minimal Deductive Systems for RDF. In Enrico Franconi, Michael Kifer, and Wolfgang May, editors, *ESWC*, volume 4519 of *Lecture Notes in Computer Science*, pages 53–67. Springer, 2007.

[Pas07] Adrian Paschke. *Rule-Based Service Level Agreements – Knowledge Representation for Automated e-Contract, SLA and Policy Management*. IDEA Verlag GmbH, Munich, forthcoming 2007.

[PKB07] Adrian Paschke, Alexander Kozlenkov, and Harold Boley. A Homogenous Reaction Rule Language for Complex Event Processing. In *Proc. 2nd International Workshop on Event Drive Architecture and Event Processing Systems (EDA-PS 2007)*. Vienna, Austria, September 2007.

[Rei88] Raymond Reiter. On Integrity Constraints. In Moshe Y. Vardi, editor, *Proceedings of the Second Conference on Theoretical Aspects of Reasoning about Knowledge*, pages 97–111, San Francisco, 1988. Morgan Kaufmann.

[Ros06] Riccardo Rosati. The Limits and Possibilities of Combining Description Logics and Datalog. In T. Eiter, E. Franconi, R. Hodgson, and Susie Stephens, editors, *RuleML*, pages 3–4. IEEE Computer Society, 2006.

[Sch05] Klaus Schild. On the Model Theory of RDF. In S. Bab und T. Noll, editor, *Models and Human Reasoning – Eine Festschrift für Bernd Mahr*, pages 189–206. Wissensch. & Technik Verlag, Berlin, 2005.

[SD02] Michael Sintek and Stefan Decker. TRIPLE – A Query, Inference, and Transformation Language for the Semantic Web. In *1st International Semantic Web Conference (ISWC2002)*. Sardinia, Italy, June 2002.

[SvESH07] Michael Sintek, Ludger van Elst, Simon Scerri, and Siegfried Handschuh. Distributed Knowledge Representation on the Social Semantic Desktop: Named Graphs, Views and Roles in NRL. In Enrico Franconi, Michael Kifer, and Wolfgang May, editors, *ESWC*, volume 4519 of *Lecture Notes in Computer Science*, pages 594–608. Springer, 2007.

[YK03] Guizhen Yang and Michael Kifer. Reasoning about Anonymous Resources and Meta Statements on the Semantic Web. In Stefano Spaccapietra, Salvatore T. March, and Karl Aberer, editors, *J. Data Semantics I*, volume 2800 of *Lecture Notes in Computer Science*, pages 69–97. Springer, 2003.

A Serializations in RuleML/XML

This appendix serializes the paper's central POSL rules in RuleML/XML 0.91. Extended versions are maintained online.[6]

XML `schemaLocation` attributes point to the XSDs of the most specific existing RuleML sublanguages that still validate the instances.

POSL's "*conclusion* `:-` *premises* `.`" in our RuleML/XML basically becomes (primes indicate recursive transforms):

`<Implies>` *premises'* *conclusion'* `</Implies>`

A.1 The Example from E1

```
<?xml version="1.0" encoding="UTF-8"?>
<RuleML
xmlns="http://www.ruleml.org/0.91/xsd"
xmlns:xsi=
"http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation=
"http://www.ruleml.org/0.91/xsd
http://www.ruleml.org/0.91/xsd/datalog.xsd"
>
   <Assert mapClosure="universal">
      <Implies>
         <And>
            <Atom>
               <Rel>shipment</Rel>
               <slot>
                  <Ind>cargo</Ind>
                  <Var/>
               </slot>
               <slot>
                  <Ind>price</Ind>
                  <Var>cost1</Var>
               </slot>
               <slot>
                  <Ind>source</Ind>
                  <Var>A</Var>
               </slot>
               <slot>
                  <Ind>dest</Ind>
                  <Var>B</Var>
               </slot>
            </Atom>
            <Atom>
               <Rel>shipment</Rel>
               <slot>
                  <Ind>cargo</Ind>
                  <Var/>
               </slot>
               <slot>
                  <Ind>price</Ind>
                  <Var>cost2</Var>
               </slot>
               <slot>
                  <Ind>source</Ind>
                  <Var>B</Var>
               </slot>
               <slot>
                  <Ind>dest</Ind>
                  <Var>A</Var>
               </slot>
            </Atom>
            <Atom>
               <Rel>add</Rel>
               <Var>cost</Var>
               <Var>cost1</Var>
               <Var>cost2</Var>
            </Atom>
         </And>
         <Atom>
            <Rel>recishiip</Rel>
            <Var>cost</Var>
            <Var>A</Var>
            <Var>B</Var>
         </Atom>
      </Implies>
   </Assert>
</RuleML>
```

A.2 Two Examples from E2

The `loyalty` module:

```
<?xml version="1.0" encoding="UTF-8"?>
<RuleML
xmlns="http://www.ruleml.org/0.91/xsd"
xmlns:xsi=
"http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation=
"http://www.ruleml.org/0.91/xsd
http://www.ruleml.org/0.91/xsd/hornlog.xsd"
>
   <Assert mapClosure="universal">
      <Implies>
         <And>
            <Atom>
               <Rel>premium</Rel>
               <Var>customer</Var>
            </Atom>
            <Atom>
               <Rel>regular</Rel>
               <Var>product</Var>
            </Atom>
         </And>
         <Atom>
            <Rel>discount</Rel>
            <Var>customer</Var>
            <Var>product</Var>
            <Expr>
               <Fun>percent</Fun>
               <Data>5</Data>
            </Expr>
         </Atom>
      </Implies>
      <Implies>
         <And>
```

[6] http://www.ruleml.org/usecases/essentials

```
            <Atom>
               <Rel>premium</Rel>
               <Var>customer</Var>
            </Atom>
            <Atom>
               <Rel>luxury</Rel>
               <Var>product</Var>
            </Atom>
         </And>
         <Atom>
            <Rel>discount</Rel>
            <Var>customer</Var>
            <Var>product</Var>
            <Expr>
               <Fun>percent</Fun>
               <Data>10</Data>
            </Expr>
         </Atom>
      </Implies>
   </Assert>
</RuleML>
```

The `legality` module:

```
<?xml version="1.0" encoding="UTF-8"?>
<RuleML
xmlns="http://www.ruleml.org/0.91/xsd"
xmlns:xsi=
"http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation=
"http://www.ruleml.org/0.91/xsd
http://www.ruleml.org/0.91/xsd/folog.xsd"
>
   <Assert mapClosure="universal">
      <Implies>
         <And>
            <Atom>
               <Rel>payment</Rel>
               <Var>customer</Var>
               <Var>product2</Var>
               <Var>amount</Var>
               <Var>method</Var>
               <Var>time</Var>
            </Atom>
            <Atom>
               <Rel>fraudulent</Rel>
               <Var>customer</Var>
               <Var>method</Var>
               <Var>time</Var>
            </Atom>
         </And>
         <Neg>
            <Atom>
               <Rel>discount</Rel>
               <Var>customer</Var>
               <Var>product1</Var>
               <Var>percent</Var>
            </Atom>
         </Neg>
      </Implies>
      <Implies>
         <And>
            <Atom>
               <Rel>delivery</Rel>
               <Var>customer</Var>
               <Var>product2</Var>
               <Var>amount</Var>
               <Expr>
                  <Fun>date</Fun>
                  <Var>y1</Var>
                  <Var>m1</Var>
                  <Var>d1</Var>
               </Expr>
            </Atom>
            <Atom>
               <Rel>payment</Rel>
               <Var>customer</Var>
               <Var>product2</Var>
               <Var>amount</Var>
               <Var>method</Var>
               <Expr>
                  <Fun>date</Fun>
                  <Var>y2</Var>
                  <Var>m2</Var>
                  <Var>d2</Var>
               </Expr>
            </Atom>
            <Atom>
               <Rel>datediff</Rel>
               <Expr>
                  <Fun>days</Fun>
                  <Var>delta</Var>
               </Expr>
               <Expr>
                  <Fun>date</Fun>
                  <Var>y2</Var>
                  <Var>m2</Var>
                  <Var>d2</Var>
               </Expr>
               <Expr>
                  <Fun>date</Fun>
                  <Var>y1</Var>
                  <Var>m1</Var>
                  <Var>d1</Var>
               </Expr>
            </Atom>
            <Atom>
               <Rel>greaterThan</Rel>
               <Var>delta</Var>
               <Data>45</Data>
            </Atom>
         </And>
         <Neg>
            <Atom>
               <Rel>discount</Rel>
               <Var>customer</Var>
               <Var>product1</Var>
               <Var>percent</Var>
            </Atom>
         </Neg>
      </Implies>
   </Assert>
</RuleML>
```

A.3 The Rule Example from E3

```
<?xml version="1.0" encoding="UTF-8"?>
<RuleML
xmlns="http://www.ruleml.org/0.91/xsd"
xmlns:xsi=
"http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation=
"http://www.ruleml.org/0.91/xsd
http://www.ruleml.org/0.91/xsd/hornlog.xsd"
>
```

```
<Assert mapClosure="universal">
   <Implies>
      <And>
         <Atom>
            <Rel>emission</Rel>
            <Var type="Real">V</Var>
            <Expr>
               <Fun>CO2</Fun>
               <Var>E</Var>
            </Expr>
         </Atom>
         <Atom>
            <Rel>weight</Rel>
            <Var type="Real">V</Var>
            <Expr>
               <Fun>kg</Fun>
               <Var>W</Var>
            </Expr>
         </Atom>
         <Atom>
            <Rel>emiweight</Rel>
            <Expr>
               <Fun>CAD</Fun>
               <Var type="Decimal">R</Var>
            </Expr>
            <Expr>
               <Fun>CO2</Fun>
               <Var>E</Var>
            </Expr>
            <Expr>
               <Fun>kg</Fun>
               <Var>W</Var>
            </Expr>
         </Atom>
      </And>
      <Atom>
         <Rel>registration</Rel>
         <Var type="Van">V</Var>
         <Expr>
            <Fun>CAD</Fun>
            <Var type="Decimal">R</Var>
         </Expr>
      </Atom>
   </Implies>
   <Implies>
      <And>
         <Atom>
            <Rel>emission</Rel>
            <Var type="Real">V</Var>
            <Expr>
               <Fun>CO2</Fun>
               <Var>E</Var>
            </Expr>
         </Atom>
         <Atom>
            <Rel>speed</Rel>
            <Var type="Real">V</Var>
            <Expr>
               <Fun>kmh</Fun>
               <Var>S</Var>
            </Expr>
         </Atom>
         <Atom>
            <Rel>emispeed</Rel>
            <Expr>
               <Fun>CAD</Fun>
               <Var type="Real">R</Var>
            </Expr>
            <Expr>
               <Fun>CO2</Fun>
               <Var>E</Var>
            </Expr>
            <Expr>
               <Fun>kmh</Fun>
               <Var>S</Var>
            </Expr>
         </Atom>
      </And>
      <Atom>
         <Rel>registration</Rel>
         <Var type="Car">V</Var>
         <Expr>
            <Fun>CAD</Fun>
            <Var type="Real">R</Var>
         </Expr>
      </Atom>
   </Implies>
</Assert>
</RuleML>
```

A.4 The Example from E4

```
<?xml version="1.0" encoding="UTF-8"?>
<RuleML
xmlns="http://www.ruleml.org/0.91/xsd"
xmlns:xsi=
"http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation=
"http://www.ruleml.org/0.91/xsd
http://www.ruleml.org/0.91/xsd/hornlog.xsd"
>
 <Assert mapClosure="universal">
  <Implies>
   <And>
    <Atom>
     <Rel>lessThanOrEqual</Rel>
     <Data>1930</Data>
     <Var>year</Var>
    </Atom>
    <Atom>
     <Rel>lessThanOrEqual</Rel>
     <Var>year</Var>
     <Data>2006</Data>
    </Atom>
   </And>
   <Atom>
    <Rel>planet</Rel>
    <Ind
     uri=
     "http://en.wikipedia.org/wiki/Pluto"/>
    <Expr>
     <Fun>AD</Fun>
     <Var>year</Var>
    </Expr>
   </Atom>
  </Implies>
 </Assert>
</RuleML>
```