Rule Interchange on the Web

Harold Boley[1], Michael Kifer[2], Paula-Lavinia Pătrânjan[3], and Axel Polleres[4]

[1] University of New Brunswick, Faculty of Computer Science
Institute for Information Technology - e-Business, NRC
46 Dineen Drive, Fredericton, Canada
harold.boley@nrc-cnrc.gc.ca
http://www.cs.unb.ca/ boley/
[2] State University of New York at Stony Brook
Department of Computer Science
Stony Brook, New York 11794-4400, USA
kifer@cs.sunysb.edu
http://www.cs.sunysb.edu/ kifer/
[3] University of Munich, Institute for Informatics
Oettingenstr. 67, D-80538 München
paula.patranjan@ifi.lmu.de
http://www.pms.ifi.lmu.de
[4] Digital Enterprise Research Institute
National University of Ireland, Galway
axel.polleres@deri.org
http://www.polleres.net/

Abstract. Rules play an increasingly important role in a variety of Semantic Web applications as well as in traditional IT systems. As a universal medium for publishing information, the Web is envisioned to become the place for publishing, distributing, and exchanging rule-based knowledge. Realizing the importance and the promise of this vision, the W3C has created the Rule Interchange Format Working Group (RIF WG) and chartered it to develop an interchange format for rules in alignment with the existing standards in the Semantic Web architecture stack.

However, creating a generally accepted interchange format is by no means a trivial task. First, there are different understandings of what a "rule" is. Researchers and practitioners distinguish between deduction rules, normative rules, production rules, reactive rules, etc. Second, even within the same category of rules, systems use different (often incompatible) semantics and syntaxes. Third, existing Semantic Web standards, such as RDF and OWL, show incompatibilities with many kinds of rule languages at a conceptual level.

This article discusses the role that different kinds of rule languages and systems play on the Web, illustrates the problems and opportunities in exchanging rules through a standardized format, and provides a snapshot of the current work of the W3C RIF WG.

1 Introduction

Rule interchange on the Web has become an increasingly important issue during the last couple of years in both industry and academia. Offering a flexible, adaptive approach towards applications development on a high level of abstraction, declarative rules languages have already been developed by different communities and deployed in various application domains. Companies manage and specify their business logic in the form of rules [The00]. Rules are also being used for modeling security policies in cooperative systems [Bon05], and they are gaining popularity as a means of reasoning about Web data [BS04,EIP+06,Ros06].

To exploit the full potential of rule-based approaches, the business rules and the Semantic Web communities have started to develop solutions for reusing and integrating knowledge specified in different rule languages. The Rule Markup Initiative[5], which started in 2000, focused its efforts on defining a shared Rule Markup Language (RuleML) that tries to encompass different types of rules. The European Network of Excellence REWERSE [BS04] (Reasoning on the Web with Rules and Semantics) proposes R2ML [WGL06], the REWERSE I1 Rule Markup Language, which offers a solution for interchanging rules between heterogeneous systems combining RuleML, SWRL, and OCL. Moreover, three member submissions for Web rule languages, namely the Semantic Web Rule Language (SWRL) [HPSB+04], the Web Rules Language (WRL) [ABdB+05], and SWSL Rules [BBB+] (a rule language proposed by the Semantic Web Services Language Committee[6]) were submitted independently to the World Wide Web Consortium (W3C) as starting points for standardization in the rules area.

Finally, at the end of 2005, W3C launched the Rule Interchange Format Working Group (RIF WG) which has been chartered to standardize a common format for rule interchange on the Web.

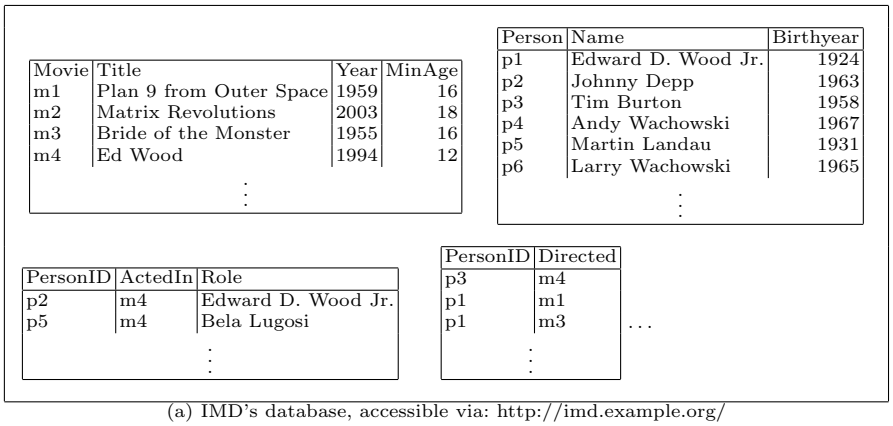
In this article, we start with a general discussion of problems related to rule interchange on the Web and outline the role of rules in realizing the Semantic Web vision. Then, we present the results of the W3C RIF WG achieved so far and outline future work to be done. In particular, we will discuss the first two public working drafts released by the Working Group—the *Use Cases and Requirements* [GHMe] document and the technical design of a core rule interchange format, the *RIF Core* [Be07].

1.1 Running Example: A Simple Movie Scenario

To illustrate the main ideas, we use an example where several movie databases interoperate through the Web.

Example 1. The fictitious International Movie Datastore (IMD) publishes its database through its Web site `http://imd.example.org/`. This database is shown in Figure 1(a). Likewise, John Doe Sr., who owns a DVD Rental Store, MoviShop, makes his movie rental services available on the Web. His site is supported by a similar database shown in Figure 1(b).

[5] Rule Markup Initiative, `http://www.ruleml.org`
[6] http://www.daml.org/services/swsl/



Movie	Title	Year	MinAge
m1	Plan 9 from Outer Space	1959	16
m2	Matrix Revolutions	2003	18
m3	Bride of the Monster	1955	16
m4	Ed Wood	1994	12

Person	Name	Birthyear
p1	Edward D. Wood Jr.	1924
p2	Johnny Depp	1963
p3	Tim Burton	1958
p4	Andy Wachowski	1967
p5	Martin Landau	1931
p6	Larry Wachowski	1965

PersonID	ActedIn	Role
p2	m4	Edward D. Wood Jr.
p5	m4	Bela Lugosi

PersonID	Directed
p3	m4
p1	m1
p1	m3

...

(a) IMD's database, accessible via: http://imd.example.org/

DVD#	Movie
dvd1	m1
dvd2	m1

Customer#	DVD#	RentalDate	ReturnDate
c1	dvd2	2007-09-01	2007-09-01
c2	dvd5	2007-01-01	

Customer#	Name	Age
c1	Joanna Doe	31
c2	Johnny Dough	12
c3	John Doe Jr.	16

((b) MoviShop's database

Fig. 1. IMD, a fictitious movie database, and MoviShop, a fictitious DVD rental store

By employing Web formats such as XML, RDF, and OWL, John can share data over the Web. John also uses Semantic Web technologies for implementing some of the services MoviShop is offering. For instance, MoviShop provides customers with information about the newest movies by importing data from `http://imd.example.org/`. It also publishes the information about available movies and links to other Web sources that provide reviews and recommendations (such as IMD). MoviShop also supports online ordering of movies, which can later be picked up in a branch office of MoviShop.

Recently John has become fascinated by the new business opportunities, which could arise if MoviShop could import and exchange not only data but also business rules. He is thus very excited about W3C's ongoing efforts towards a common rules interchange format.

1.2 Rules and Rule Types

Creating a format for rule interchange on the Web, which could help John conduct his business, is not a trivial task. First, there are different ideas about what *types* of rules are of interest: one might get different views depending on whether you talk to a production rule vendor, a database vendor, or a Semantic Web researcher. This is because the term "rule" is an umbrella for a number of related, but still quite different concepts.

A simple example of a rule is a CSS selector, i.e. a statement that defines how browsers should render elements on an HTML page, such as the list of available

movies in John's MoviShop. At the business level, John might be interested to enforce *constraint rules* such as "*credit cards accepted by MoviShop are VISA and MasterCard.*" Moreover, rules can define implicit data, e.g. "*Customers who have rented more than 100 movies are priority customers.*"

Integrity constraints on the structure of the data in databases, such as "*each movie has a single production year,*" or "*each MoviShop customer has a unique identification number,*" provide another example of rules.

Statements specifying parts of an application's dynamic behavior such as *'if an item is perishable and it is delivered more than 10 days after the scheduled delivery date then the item will be rejected'* or *'if a customer is blacklisted then deny the customer's request to booking a movie'* are also rules. The second rule is a good example for rules that could be interchanged between systems deployed by MoviShop and similar online shops.

The above examples show that rules may specify *constraints*, *(implicit) construction of new data*, *data transformations*, *updates on data* or, more general, *event-driven actions*. Rules are also used for *reasoning* with Web and Semantic Web data. From a high-level view, rules are statements that express static or dynamic relationships among different data items, the logic of applications, and more generally of business processes within enterprises.

Rules may differ significantly not only in their purpose (*e.g.*, transforming data), but also in their form. Consider, for example, the following rules:

```
(R1) IF movie ?M was produced before 1930
     THEN ?M is a black and white movie

(R2) ON request from customer ?C to book a movie
     IF customer ?C is blacklisted
     DO deny ?C's request
```

Rule *R1* has two components: The `IF` part searches for movies produced before 1930 and binds the variable M to such movies. The `THEN` part of the rule uses the retrieved information to construct new data—a view over the movie data. Rule *R2* has a different structure: The `ON` part waits for a request for booking a movie to come in, i.e. an event. The `IF` part of rule *R2* is similar to the `IF` part of rule *R1*—it checks a condition about the customer requesting a movie. The `DO` part specifies the action to be executed on a request from a blacklisted customer.

Different kinds of rules present different requirements to the implementor. *R2* needs a richer language and a more complex execution semantics than *R1*, since its `ON` part requires support for detecting incoming events and its `DO` part needs support for executing actions. However, both types of rules share the need to support conditions in the `IF` part. This commonality among different types of rules suggest a way of approaching the development of a rule interchange format by starting with common parts (such as the `IF` part) and extending this support to various features of different types of rules. To better understand the similarities among different types of rules, we divide them into three categories: *deduction rules, normative rules, and reactive rules* (see [TW01,BM05]).

Deduction rules are statements of how to derive knowledge from other knowledge by using logical inference. Deduction rules are also called derivation rules in the business rules community, constructive rules by logicians, and views in the database community.

One could say that deduction rules describe *static* dependencies among entities which might be used to infer additional implicit knowledge from what is explicitly stated in a knowledge base or database. Deductive rules are often specified as implications of the form `head` ← `body`: The `head` gives a specification of the data to be constructed/inferred and the `body` queries the underlying database(s). Both rule parts may and usually share variables. Variables get bound to data items in the `body` and these bindings are then used in the `head` to infer new data. In the previous example, rule *R1* is a deductive rule. Its body is introduced by the keyword `IF` and the head by `THEN`. The head and the body share the variable *M*, which is a placeholder for a movie object.

The actual syntax and the nature of the parts in a rule depend on the chosen rule language. Some languages divide the `body` into a query part, which selects data from one or more databases, and a condition part, which acts as a filter for the results obtained by evaluating the query part. Consider the rule *R1*, which constructs a view of black and white movies from a movie database at `http://imd.example.org/movies.xml`. The `body` of the rule searches for movies in a database and then filters out the movies produced before 1930. As a concrete example of a rule language, let us consider rule R1 expressed in Xcerpt[7], an XML rule language based on deductive rules:

```
CONSTRUCT
  black-and-white { all var Title }
FROM
  in { resource { "http://imd.example.org/movies.xml", XML},
       moviedb {{
          movie {{
             title { var Title },
             year { var Year }
          }}
       }}
  } WHERE var Year < 1930
END
```

Rule components specify patterns for the data to be constructed or queried. The rule head, introduced by `CONSTRUCT`, gathers *all* substitutions for the variable *Title*. The rule body, introduced by `FROM` gives an incomplete pattern for the *movies.xml* data and retrieves the movie titles and years as substitutions for the variables *Title* and *Year*. The `WHERE` part specifies the desired filter condition.

The given Xcerpt rule exemplifies a deductive rule with *selection and filtering* of data in the `body` and *grouping* of data items in the `head`. Data is retrieved from a single data source—an XML document—but more than one data source can be

[7] Xcerpt, `http://xcerpt.org`

queried within one deductive rule. The data source may be explicitly given, as in this case, or given implicitly (e.g. upon loading a ruleset). Depending on the rule language used to specify and evaluate deductive rules, other features may also be supported. Join queries over multiple data sources, which correlate pieces of data found in the data sources, may also be specified in a deductive rule's body. Other features, such as aggregation of data (e.g. by means of aggregate functions like `count`, which counts the substitutions for a given variable), or external function calls may or may not be allowed to be used in rules.

Other examples of rule languages based on deductive rules are SQL views, Datalog, Prolog, and most other logical rules languages.

Normative rules are rules that pose constraints on the data or on the logic of an application. Such rules ensure that changes made to a database do not cause inconsistencies and that the business logic of a company is obeyed. Normative rules are also called structural rules in the business rules community and integrity constraints in databases.

Normative rules describe disallowed inconsistencies rather than inferring new knowledge. A simple constraint is that each customer must have a unique identification number. Two different identification numbers for the same person are an indication of corrupted data. Similar examples arise in E-R models in the form of key declarations and cardinality restrictions. XML's DTDs also provide rudimentary capabilities to express such key constraints in the form of ID and IDREF attributes, and XML Schema provides rich support for key and foreign key constraints.

In some cases, deductive rules can be used to implement normative rules[8]. In other cases, reactive rules (discussed next) can be used to implement certain types of normative rules. The decision largely depends on the application and on the available support for different rule types. For example, in [FFLS99] constraints are used to describe the structure of Web sites and deductive rules are used as a specification and implementation mechanism. Deductive rules are also used to specify assertion-style constraints in databases. In addition, database implementors often use triggers—a form of reactive rules—as a technique for implementing integrity constraints.

Reactive rules offer means to describe reactive behavior and implement reactive systems, i.e. to automatically execute specified actions when events of interest occur and/or certain conditions become true. Reactive rules are also called active rules and dynamic rules. Unlike deduction rules, reactive rules talk about state changes. Events represent changes in the state of the world and rules specify further changes that must occur in reaction to the events (hence the name *reactive rules*). Reactive rules usually have the form of Event-Condition-Action (ECA) rules or production rules.

ECA rules are rules of the form `ON Event IF Condition DO Action`. This means that the `Action` should be executed if the `Event` occurs, provided that the `Condition` also holds. The `Event` part serves a twofold purpose: detecting

[8] More details on the proposed classification of rules can be found at `http://www.w3.org/2005/rules/wg/wiki/Classification_of_Rules`

events of interest and selecting data items from their representations (by binding variables as discussed above). Different kinds of events may be supported, ranging from low-level, data-driven events, such as the deletion of a subtree of an XML tree, to high-level more abstract events, such as the delay of a flight. Depending on the rule language, only single occurrences of events (*atomic events*) may be allowed to be specified and detected, or temporal combinations of events (*composite events*) may be permitted. The events mentioned previously are all atomic. The following is an example of a composite event: "*customer ?C requests booking a movie AND no response from ?C on bringing back movie M1.*"

The `Condition` part queries the state of the world—e.g. XML and RDF data in Web applications or a legacy database—and may be expressed in a query language such as XPath[9], Xcerpt[10], or SPARQL[11]. Like the `Event` part, the `Condition` also selects data items that can be further used in executing the `Action`. Different kinds of actions may be supported by reactive systems: updates to data (e.g., insertion of a subtree into an XML tree), changes to the rule set itself (e.g., removal of a rule), new events to be triggered, or procedure calls (e.g., calling an external function which sends an email message). Actions can also be combined to ease the programming of complex applications.

The following example gives a possible implementation of the rule *R2* previously mentioned in this section. We use XChange[12] here, a language which is based on ECA rules:

```
ON
  xchange:event {{
    xchange:sender { var S },
    order {{
      customer { var C }
    }}
  }}
FROM
  in { resource { "http://MoviShop.org/blacklisted.xml", XML },
    desc var C
  }
DO
  xchange:event {
    xchange:recipient { var S },
    message { "Your request can not be processed,
               since you are blacklisted" }
  }
END
```

[9] XPath, `http://www.w3.org/TR/xpath`
[10] Xcerpt, `http://xcerpt.org`
[11] http://www.w3.org/TR/rdf-sparql-query/
[12] XChange, `http://reactiveweb.org/xchange/intro.html`

XChange [BBEP05,BEP06b,PŏŠ05] integrates the query language Xcerpt for specifying the condition part (introduced by `FROM`) of reactive rules. XChange assumes that each reactive Web node has reactive rules in place and that the communication of Web nodes is realized through event messages—messages carrying information on the events that have occurred. XChange reactive rules react upon such event messages, which are labelled `xchange:event`. The `ON` part of the example reactive rule gives a pattern for incoming event messages that represent booking requests. The `FROM` part is an Xcerpt query; the `desc`endant construct in the example is used for searching at a variable depth in *blacklisted.xml* for the customer requesting the order. The `DO` part sends an event message to the customer announcing the reasons for denying the request.

Examples of other ECA rule languages and systems are the General Semantic Web ECA Framework,[13] [BFMS06,MSvL06] Prova,[14], ruleCore[15], and Active XQuery [BBCC02]. Another ECA rule language for XML data is proposed in [BPW02] and adapted for RDF data as the RDF Triggering Language (RDF-TL) [PPW03]. Reaction RuleML[16] was recently launched as part of the RuleML Initiative.[17] Its aim is the development of a markup language for the family of reactive rules.

Triggers are a form of ECA rules employed in database management systems: The E part specifies the database update operation to react to, the C part specifies the condition that needs to hold for executing the database update operations specified in the A part. Triggers combined with relational or object-oriented database systems give rise to so-called active database systems. Most of the active database systems support triggers where all three rule parts—E, C, and A—are explicitly specified; proposals also exist where the E or the C part is either missing or implicit. Examples of active database systems include Starburst [Wid96], Postgres [PS96], and the systems based on SQL:1999 [Mel02].

It is worthwhile to also mention other, less popular, variations of ECA rules. Event-Condition-Action-Alternative (ECAA) rules extend the ECA rules by specifying the action to be executed when the given condition doesn't hold. Note that an ECAA rule can be rewritten into two ECA rules, where the condition part of one rule is the negation of the other. An EC^nA^{n+1} rule is a generalization of an ECAA rule. Yet another form of ECA rules are the Event-Condition-Action-Postcondition (ECAP) rules, which extend ECA rules by specifying a (post)condition that needs to hold after the action has been executed. These kinds of restrictions make sense, for instance, when considering action execution frameworks supporting transactions.

Production rules are rules of the form `IF Condition DO Action`. They say that the `Action` part must be executed whenever a change to the underlying

[13] General Semantic Web ECA Framework, `http://www.dbis.informatik.uni-goettingen.de/eca/`
[14] Prova,`http://www.prova.ws`
[15] ruleCore,`www.rulecore.com`
[16] Reaction RuleML,`http://ibis.in.tum.de/research/ReactionRuleML/`
[17] RuleML Initiative, `http://www.ruleml.org`

database makes `Condition` true. The `Condition` queries the working memory, which contains the data on which the rules operate. Selected data is then used to execute the actions specified in the `Action` part. An action usually contains operations on the working memory (e.g., assert a new data item) but also statements à la procedural programming (e.g., assignment, loop).

As an example, consider a different incarnation of the reactive rule *R2*—this time using ILOG's commercial production rules system JRules.[18] Other production rules systems include OPS5,[19] JBoss Rules,[20] and Jess.[21]

```
rule denyBlacklistedCustomers {
     when {
       c: Customer (blacklisted == yes);
       m: MoviesCart (owner == c; value > 0);
     } then {
        out.println ("Customer "+c.name+" is blacklisted!");
        retract m;
     }
 }
```

The condition part of the rule (introduced by `when`) gives a pattern for instances of classes `Customer` and `MoviesCart`. It matches `Customer` instances whose attribute `blacklisted` is set to `yes` and whose `MoviesCart`'s attribute `value` is greater than zero (meaning that the customer wants to order at least one movie). The two classes used in the example rule may be Java or C classes or element types from an XML Schema. The action part (introduced by `then`) announces the reason of denying the request and updates the working memory removing the `MoviesCart` object for the respective `Customer` instance (by the statement `retract m`).

ECA and production rules are discussed in [BBB+07], where an in-depth presentation of both approaches to reactive behavior is given. The work also tries to reveal similarities and differences between the two approaches to programming reactive applications on the Web. The structure and semantics of the two kinds of rules indicate that the ECA rule approach is well suited for distributed applications which rely on event-based communication between components, while the production rule approach is more appropriate for coding the logic of stateful applications.

A considerable number of rule languages that can be employed for programming Semantic Web applications have been proposed, which we will discuss in more detail in Section 2. An overview of current rule languages and systems that were considered of interest by the participants of the W3C RIF WG can be found

[18] ILOG JRules, `http://www.ilog.com/products/jrules/`
[19] Official Production System 5, `http://www.cs.gordon.edu/local/courses/cs323/OPS5/ops5.html`
[20] JBoss Rules, `http://www.jboss.com/products/rules`
[21] Jess, `http://www.jessrules.com/`

at http://www.w3.org/2005/rules/wg/wiki/List_of_Rule_Systems. In this section we presented a classification of rules that can be a basis for discovering commonalities between rule languages. However, as the examples have shown, they also reveal considerable differences with respect to syntax, supported features, and semantics. Thus, to determine whether language constructs have the same effect in different rule languages, an analysis based on the language semantics is needed. A standard interchange format should be geared for exchanging rules with different structure, constructs, and semantics.

1.3 W3C RIF WG Charter

The charter of the W3C RIF WG[22] gives guidelines and requirements for the rule interchange format to be developed within the Working Group. This document and the particular interests of the participants of the W3C RIF WG will influence the ultimate shape RIF is going to take.

The W3C RIF WG is chartered to develop an exchange format for rules that should enable rules to be translated between different rule languages and, thus, to be used in different rule systems. This is to be achieved through two phases corresponding to the development of an extensible core interchange format and a set of extensions (called standard dialects). Each of these work phases is chartered for up to two years.

Phase I focuses essentially on Horn rules for a core rule interchange format. The RIF Core development should build a stable backbone and allow extensions in the second phase. For Phase II, the charter just gives starting points for possible extensions. For example, the core format might be extended in the direction of first-order rules, (possibly non-monotonic) logic programming rules, and particularly ECA and production rules, neither of which to be fully covered in the core.

The charter also emphasizes compatibility with Web and Semantic Web technologies such as XML, RDF, SPARQL, and OWL. It states that the primary normative syntax of a general rule interchange format must be based on XML. Moreover, it states that RIF must support interoperability with RDF data and be compatible with OWL and SPARQL.

1.4 Outline

The subsequent sections are structured as follows. In Section 2 we describe in more detail the role of rules and rule interchange in the Semantic Web architecture.

Then we turn to a discussion of the early working draft documents published by the RIF WG. The use cases for rule interchange described in the Second W3C Public Working Draft of *Use Cases and Requirements* are discussed in Section 3. The requirements for RIF that follow from these initial use cases are discussed in Section 4. The first public Working Draft on *RIF Core Design* [Be07] will be

[22] http://www.w3.org/2005/rules/wg/charter.html

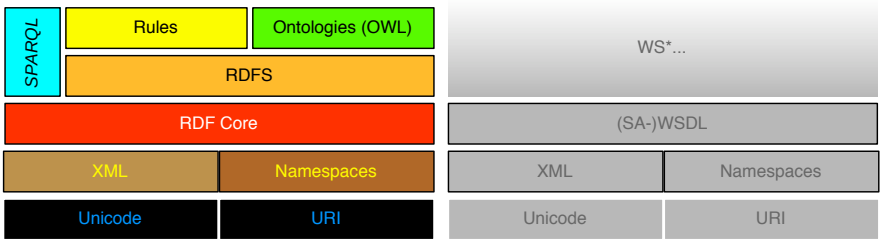


Fig. 2. The role of rules in the Semantic-Web layer cake

discussed in Section 5, where we present the current syntax and semantics of the RIF Core using an example.

A comprehensive classification system, called RIF Rule Arrangement Framework (RIFRAF), is currently under development and is being populated with existing rule languages and systems. The status of this study and impact on possible extensions to RIF Core are also briefly discussed in Section 5.

2 Rules in the Semantic Web architecture: Where do rules belong?

Recent versions of the often cited "Semantic Web Layer Cake" [BL05] position rules as a central component (see Figure 2).

We have already pointed out in the introduction that rule interchange, and thus RIF, will not be restricted to Semantic Web applications. It is expected, for example, that rules will affect Web Services standards, such as SA-WSDL,[23] which merge the Semantic Web and Web services worlds by allowing semantic annotations within WSDL documents. However, since the role of rules within the Web services layer is yet to be clearly defined by the standards bodies, we will focus on the "core" semantic Web architecture and discuss the applications of rules to the existing Semantic Web standards: XML, RDF, RDFS, OWL, and SPARQL. Note that the presented version of the Semantic Web architecture stack in Figure 2 leaves out layers such as "Proof," "Security," "Encryption," and "Trust." Rules are certainly going to play an important role within these layers as well, but here we will focus on the layers that already have recommendations endorsed by W3C.

2.1 URIs and Unicode

The Semantic Web is based on the idea of *Uniform Resource Identifiers (URIs)* as unique identifiers for documents that live on the Web, but also for real-world or abstract objects. Following this paradigm, RIF will support and require the use of

[23] http://www.w3.org/2002/ws/sawsdl/

URIs for objects, relations, and other components of rules. The main difference between URIs and URLs is that, for instance, `http://imd.example.org#m1` might be a URI that identifies a movie stored at IMD, but it does not necessarily need to identify any document on the Web. URIs may also have syntactic forms that are not URLs. In the near future, Semantic Web languages will be expected to use the rich *Unicode* character set in order to provide support for non-Latin letters in a uniform way. In order to be able to identify objects using non-Latin character sets, the IRI (International Resource Identifier) specification has been recently adopted by W3C. An IRI is just like a URI, but it might include, for example, Kanji or Hebrew characters. It has been decided that RIF will use IRIs to denote globally accessible objects.

2.2 XML and Namespaces

The *eXtensible Markup Language* (XML) has been chosen as the standard exchange syntax for data and messages on the Web and, consequently, will likely also be the basis for the exchange of rules on the Web.

Example 2 (Running example continued). In order to support movie enthusiasts and other people like John, our video store owner, IMD makes its movie data available in an XML document, as shown in Figure 3.

```
<?xml version="1.0"  encoding="UTF-8"?>
<moviedb xmlns="http://imd.example.org/ns/">
  <movie ID="m1">
    <title>Plan 9 from Outer Space</title>
    <directedBy IDref="p1"/>
    ...
    <year>1959</year>
    <age>16</age>
  </movie>

  ...

  <person ID="p1">
    <name>Edward D. Wood Jr.</name>
    <dateOfBirth>1924-10-10</dateOfBirth>
  </person>

  ...

</moviedb>
```

Fig. 3. IMD's XML dump, available at `http://imd.example.org/movies.xml`

In XML, URIs also serve to denote unique identifiers. For example, identifiers in `http://imd.example.org/movies.xml`, such as `http://imd.example.org/movies.xml#m1`, can be used to denote objects described in the document. Moreover, URIs serve to denote "scopes" of element or attribute names within XML

documents using XML's *namespace* mechanism. For instance, the XML document in Figure 3 has the default namespace `http://imd.example.org/ns/`. An element name `name` in the figure refers to "the `name` element associated with the URI `http://imd.example.org/ns/`," which is used to specify the name of a person. Namespaces are used to disambiguate references to element or attribute names that appear in the same document, but have different meaning and are typically defined externally. For instance, the `name` element that describes a person could have a different structure from the name element that is used for companies. The latter may be defined in an external document using some other namespace.

Sometimes, notably in RDF, however, the term "namespace" is used to refer to prefixes of full URLs. For instance, `<imd:age>` is treated as an abbreviation of the URI `http://imd.example.org/ns/age`, if `imd` is defined as a "macro" for `http://imd.example.org/ns/`. Reusing the term "namespace" in this situation is unfortunate and causes confusion. A more appropriate term, *compact URI* (or *curi*), has recently been adopted for the abbreviation schemes like `<imd:age>` above. RIF will support compact URIs as well.

Semantic Web languages are required to support an XML exchange syntax and so will RIF. Since XML is verbose and is hard for humans to write and understand, it is used mostly for machine-to-machine exchange. Semantic Web languages, such as RDF or OWL, also support more human-friendly *abstract syntaxes*, and RIF will provide such a human-oriented syntax as well.

XML comes with several accompanying standards: for querying XML documents (XPath and XQuery); for transforming XML documents to XML and other formats (XSLT); for specifying document structure (XML Schema); and so on. Although XPath, XQuery, and XSLT are the most common query and transformation tools for XML, many XML rule languages based on logic programming have been developed. These include eLog [BFG+01], Xcerpt [Sch04] and XChange [BEP06a], and Prolog systems with extensive XML support like Ciao Prolog [CH01] or SWI Prolog [WHvdM06]. Examples of Xcerpt and XChange were given in the introduction, and similar examples can be worked out for the other languages on the above list. These examples and overlapping expresive features of these rule languages suggest that at least some rules could be exchanged among dissimilar systems. However, it is not yet clear how far this can be taken and to what extent (query, transformation, or validation) rules on top of XML can be interchanged in general.

2.3 RDF and RDFS

The *Resource Description Framework* (RDF) is the basic data model for the Semantic Web. It is built upon one of the simplest structures for representing data—a directed labeled graph. An RDF graph is typically described by a set of triples of the form ⟨*Subject Predicate Object*⟩, also called *statements*, which represent the edges of the graph. In the RDF terminology, predicates are called *properties* and are identified by URIs. Subjects and Objects can be either URIs denoting real or abstract *resources*, datatype literals (e.g., 1.2, "abc"), or XML

literals (i.e. well-formed XML snippets).[24] Besides a normative RDF/XML syntax, several more readable syntaxes for RDF have been devised, we use here the more terse Turtle [Bec06] syntax.

Example 3 (Running example continued). IMD also exports its movie data in RDF, see Figure 4 (a) and additionally provides some structural information on its data as an RDFS hierarchy Figure 4(b).

```
@prefix imd: <http://imd.example.org/ns/>
@prefix foaf: <http://xmlns.com/foaf/0.1/>
@prefix bio: <http://purl.org/vocab/bio/0.1/>
@prefix rdf: <http://www.w3 ...rdf-syntax-ns#>

<imd:m1> <imd:title> "Plan 9 from Outer Space".
<imd:m1> <imd:directedBy> <imd:p1> .
<imd:m1> <imd:year> "1959" .
 ...

<imd:m29> <imd:year> "1929" .

 ...

<imd:p1> <foaf:name> "Edward D. Wood Jr.";
         <bio:event> _:p1Birth.

 _:p1Birth a <bio:Birth>;
          <bio:date> "1924-10-10".
          <bio:place> "Poughkeepsie, NY, USA".
 ...
```

(a) Movie data in RDF

```
@prefix imd: <http://imd.example.org/ns/>
@prefix foaf: <http://xmlns.com/foaf/0.1/>
@prefix bio: <http://purl.org/vocab/bio/0.1/>
@prefix rdf: <http://www.w3 ...rdf-syntax-ns#>
@prefix rdfs: <http://www.w3 ...rdf-schema#>


 ...


<imd:directedBy> <rdfs:domain> <imd:Movie>.
<imd:directedBy> <rdfs:range> <imd:Director>.
<imd:Director> <rdfs:subclassOf> <foaf:person>.



 ...
```

(b) Structural metadata in RDF Schema (RDFS)

Fig. 4. IMD's data in RDF(S)

John imports this metadata to process it for MoviShop, finding it more flexible and easier to combine with his own data than a fixed XML scheme. John's customers are often interested in additional information about movies, such as, whether old movies are color or black-and-white—information that is *not* explicitly provided by the exported IMD's metadata. In order to avoid labeling every movie explicitly as color or black-and-white, John wants his system to *automatically* infer that all movies produced before 1930 are black and white, and that the movies produced after, say, 1950 are likely to be color movies. Then he would have to label explicitly only the movies produced in-between.

```
(R1)  Every movie at http://movishop.example.org/ produced
      before 1930 is black and white.
(R1') Every movie at http://movishop.example.org/ produced
      after 1950 is color unless stated otherwise.
```

He finds out that there are several rule languages which he could use and again to process such rules, but not really any format which allows him to publish this rule.

[24] Strictly speaking, RDF does not allow literals in subject positions, but languages like SPARQL lift this restriction.

Rule Languages for RDF. Several rule engines can process rules of the form (R1) and some also rules of the form (R1').[25]

TRIPLE[26] allows to access RDF data and manipulate it by Horn rules defined in F-Logic [KLW95] syntax. The engine is based on XSB[27].

JENA[28] has its own rule language and execution engine, which allows forward (RETE-style [For82]) and backward chaining (tabled SLG resolution[SW94]) evaluation of rules on top of RDF.

cwm[29] is a simple forward chaining reasoner, which uses Notation3 (N3), a rule language that extends RDF's widely used turtle syntax [Bec06]. The semantics of N3 is not, however, defined formally, so the results of rule evaluation are implementation-dependent.

FLORA-2[30] is a very powerful rule-based system, which is capable of representing and efficiently executing logic programming style rules. It is based on F-logic, but in addition has support for higher-order modeling via HiLog [CKW93a] and for declarative state changes via Transaction Logic [BK98,BK94]. It uses XSB as its rule evaluation engine.

Finally, dlvhex[31] is a flexible framework for developing extensions to the declarative Logic Programming Engine DLV,[32] which supports RDF import and provides rules on top of RDF. DLV's [LPF+06] rules language is disjunctive Datalog [EGM97], i.e., it is based on logic programming with disjunctions in rule heads, negation as failure in rule bodies, and several other extensions—all based on the so-called answer set semantics (see [Bar03]).

As can be seen by this short list of the available engines, they cover a plethora of different languages and semantics. However, they all also support a common sublanguage, i.e., function-free Horn rules. Most of these systems would for instance allow to express and process rule `(R1)` from the above example, which can be written as a Horn rule as follows:

$$\forall D, M, Y.\ (triple(D, rdf:type, moviShop:Dvd) \wedge triple(D, moviShop:shows, M) \wedge triple(M, rdf:type, imd:Movie) \wedge triple(M, imd:year, Y)\ \wedge\ Y < 1930 \rightarrow triple(M, rdf:type, moviShop:BWMovie))$$

A common sublanguage that underlies most of the rule systems is the basic idea behind RIF Core, the core dialect of RIF. However, not all languages support all types of rules. For instance, even within the category of languages that are based on deduction rules not all the languages support rule (R1'). To support

[25] Rule (R1') is using so-called "default reasoning" and its treatment requires non-first-order logic. It can be handled by systems such as FLORA-2 and dlvhex, which are briefly described here.

[26] `http://triple.semanticweb.org/`

[27] `http://xsb.sourceforge.net/`

[28] `http://jena.sourceforge.net/`

[29] `http://www.w3.org/2000/10/swap/doc/cwm`

[30] `http://flora.sourceforge.net`

[31] `http://con.fusion.at/dlvhex/`

[32] `http://www.dlvsystem.com/`

exchange between languages that are more expressive than the core, RIF will provide dialects that extend the RIF Core dialect. For instance, rule (R1') could be supported by a logic programming dialect of RIF.

RDF Schema. RDF itself was not designed to express schema information—this job was given to a separate specification known as *RDF Schema* (RDFS). RDFS is a very simple ontology language for specifying taxonomies of resources and properties, as well as domain and range restrictions for properties, such as the ones shown in Figure 4(b). Our IMD site uses RDFS to express some of the axioms about the movie domain. For instance, an axiom might say that each subject of a triple with the `<imd:directedBy>` property is a member of the class `<imd:Movie>`.

These structural axioms can themselves be interpreted as rules. In fact the above-mentioned RDF engines often approximate RDF and RDFS semantics with Datalog rules [tH05,EIP+06].

For instance, the RDFS entailment rule (rdfs3) from [Hay04], which states

> *If an RDF graph contains triples (`P rdfs:range C`) and (`S P O`) then the triple `O rdf:type C` is entailed.*

can be written as a Horn rule as follows:

$$\forall S, P, O, C.triple(P, rdf : range, C) \wedge triple(S, P, O) \rightarrow triple(O, rdf : type, C)$$

The following examples show how this rule is represented in TRIPLE's F-Logic style, JENA's rule syntax, N3, FLORA-2, and dlvhex.

TRIPLE:

```
rdf:= 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'.
rdfs:= 'http://www.w3.org/2000/01/rdf-schema#'.
type := rdf:type.
range := rdfs:range.

FORALL O,C O[type->C] <- EXISTS  S,P (S[P->O] AND P[range->C]).
```

JENA:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.

[rdfs3: (?s ?p ?o) (?p rdfs:range ?c) -> (?o rdf:type ?c)]
```

N3:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix log: <http://www.w3.org/2000/10/swap/log#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

{ <#p> rdfs:range <#c>. <#s> <#p> <#o> . }
               log:implies { <#o> rdf:type <#c> }.
```

FLORA-2:

```
:- iriprefix rdf = 'http://www.w3.org/2000/01/rdf-schema#'.

?O[rdf#type->?C] :- ?S[?P->?O], ?P[rdf#range->?C].
```

dlvhex:

```
#namespace("rdf","http://www.w3.org/1999/02/22-rdf-syntax-ns#")
#namespace("rdfs","http://www.w3.org/2000/01/rdf-schema#")

triple(O,rdf:type,C) :-  triple(P,rdfs:range,C), triple(S,P,O).
triple(S,P,O) :-
       &rdf["http://UrlWithRdfData.example.org/data.rdf"](S,P,O).
```

As we can see, all these languages have syntactic differences, but they express the RDFS axiom in our example in similar ways. A common exchange format like RIF would enable interchange of such rules between the various systems.

RDF enhanced with rules may be useful in several other contexts: for defining mappings between RDF vocabularies, for specifying implicit metadata, for integrating information from different sources, and so on.

2.4 OWL

RDFS is good only for very simple ontologies, and a more expressive language based on Description Logic, was recommended by W3C. The *Web Ontology Language* [DSB+04] (OWL) adds several features to the simple class hierarchies of RDFS. First, it provides an algebra for constructing complex classes out of simpler ones. Second, it extends what one can do with properties by allowing transitive, symmetric, functional, and inverse properties. It also supports restrictions on property values and cardinality.

Still, OWL proved to be insufficient for many applications on the Semantic Web and the need to add rules became evident early on. Unfortunately, OWL is not easily combinable with most rule-based formalisms. For one thing, OWL allows disjunctive and existential information, while most rule systems do not. For another, OWL is entirely based on first-order logic, while many rule-based formalisms support so-called *default reasoning*, which is not first-order.

To illustrate, the following statement is easily within OWL's competence:

```
Every movie has at least one director.
```

Using the abstract syntax [PSHH04] of OWL, one can write it as follows:

```
Class( imd:movie partial
          restriction (imd:directedBy minCardinality 1) )
```

But representing the same as a logical rule

$$\forall M.\ (\ triple(M, rdf:type, imd:movie)\ \rightarrow\ \exists D.triple(M, imddirectedBy, D)\ )$$

requires existential quantification in the rule head, which places such a rule squarely outside of most rule systems.[33]

Even a simple extension of OWL by Horn rules, such as SWRL [HPSB+04], raises several non-trivial semantic issues and the language quickly looses its most attractive asset—decidability. Moreover, as already mentioned, many rule systems support so-called default negation as opposed to classical negation. Rule (R1') in Example 3 uses precisely this kind of negation. Such a rule cannot be represented in OWL or its rule-based extension SWRL and yet this kind of statements are commonplace.

Default negation goes beyond RIF Core, but it is expected to be supported by at least one of the future dialects of RIF, which will be developed as part of Phase II work. We refer the reader to [dBEPT06,EIP+06,Ros06] for further details on the issues concerning the integration of OWL and rules.

2.5 SPARQL

SPARQL [Pe06] is a forthcoming RDF query language standard, which is still under development by the W3C Data Access Working Group (DAWG).[34] Other RDF query languages with interesting features were proposed as well [FLB+06].

SPARQL is interesting in connection with rules and RIF for several reasons:

1. The RIF Working Group promises in its charter to "ensure that the rule language is compatible with the use of SPARQL as a language for querying of RDF datasets." This could be achieved by allowing SPARQL queries in rule bodies.
2. SPARQL's CONSTRUCT queries can be viewed as deductive rules, which create new RDF triples from RDF datasets [Pol07,SS07].
3. SPARQL queries can be represented as Datalog rules [Pol07].

SPARQL allows querying RDF *datasets* via simple and complex graph patterns. A graph pattern is a graph some of whose nodes and arcs are labeled with variables instead of resources. Besides graph patterns, SPARQL has several interesting features, such as optional graph patterns, filtering values, and unions of patterns.

Example 4 (Running example continued). Although John could choose any number of systems for his shop, he was concerned with being locked into one of the systems without a possibility to switch. So, he decided to try his luck with SPARQL, as this language is expected to get W3C's stamp of approval soon. Suppose that the RDF dataset from Figure 4(a) is accessible via URI `http://imd.example.org/movies.rdf` and that RDF data about the movies in John's MoviShop is accessible through `http://movishop.example.org/store.rdf`. Then he could ask a query about all movies produced before 1930 as follows:

[33] Existentials in rule heads are beyond Horn rules, though in many cases, rule systems can approximate such existential information using Skolem functions.

[34] `http://www.w3.org/2001/sw/DataAccess/`

```
@prefix imd: <http://imd.example.org/ns/>
@prefix moviShop: <http://movishop.example.org/ns/>
@prefix rdf: <http://www.w3 ...rdf-syntax-ns#>

SELECT ?M
FROM <http://imd.example.org/movies.rdf>
FROM <http://movishop.example.org/store.rdf>
WHERE { ?D rdf:type moviShop:Dvd . ?D movShop:shows ?M .
        ?M rdf:type imd:Movie . ?M imd:year ?Y .
       FILTER (?Y < 1930) }
```

Since this was easy, John feels glad that RIF WG has decided to ensure SPARQL compatibility.

Then suddenly John stumbled upon a brilliant idea: to express the above query as a rule using SPARQL's `CONSTRUCT` statement:

```
CONSTRUCT { ?M rdf:type moviShop:BWMovie }
FROM <http://imd.example.org/movies.rdf>
FROM <http://movishop.example.org/store.rdf>
WHERE { ?D rdf:type moviShop:Dvd . ?D movShop:shows ?M .
        ?M rdf:type imd:Movie . ?M imd:year ?Y .
       FILTER (?Y < 1930) }
```

However, and insider told John that the data access working group, which is in charge of SPARQL, is not planning to position SPARQL as a rules language and the semantics of the rules expressed using the CONSTRUCT statement is not fully defined. The insider also suggested that John look into the ways of translating complex SPARQL queries into Datalog rules and process them with one of the earlier mentioned rule engines [Pol07].

We thus see that rules play (or can play) an important role in several layers of the Semantic Web architecture. In turn, the Semantic Web architecture has influenced the design of RIF inspiring such design decisions as the use of IRIs, the compact URI scheme, XML, and the use of XML data types. The Semantic Web imposed a number of other requirements on RIF, which will be discussed further in Sections 4 and 5. However, in order to get a better idea of practical scenarios which defined these requirements, we will first discuss some of the use cases for rule interchange, which served as input to RIF design.

3 W3C Use Cases on Rule Interchange

Close to fifty use cases[35] for rule interchange on the Web have been submitted by the W3C RIF WG participants. These use cases depict a wide variety of scenarios where rules are useful or even indispensable. Scenarios range from life

[35] `http://www.w3.org/2005/rules/wg/wiki/Use_Cases`

sciences, to e-commerce to business rules and Semantic Web rules. Other scenarios involve fuzzy reasoning, automated trust establishment, rule-based service level agreement, etc.

The W3C RIF WG collected use cases for both phases of its chartered work, but the Use Cases document does not group them according to the work phases. Such a classification would be difficult in many cases, since different implementations of the same use case might be possible. Some of these implementations might require Phase II features but some might be realizable completely within Phase I.

Each use case presents requirements to be acknowledged by the rule interchange format to be developed by the working group. Some of these requirements are stated explicitly, others are implied.

The submitted use cases were analyzed and classified into eight categories.[36] Seven use cases were chosen and edited for publication as the First W3C Public Working Draft of *Use Cases and Requirements*,[37] which was released in March 2006.

The first draft did not include the requirements to RIF—these were included in the second draft along with two new use cases. In this paper we present four of the use cases from the second draft. The requirements to rule interchange are discussed in Section 4.

3.1 Negotiating eBusiness Contracts Across Rule Platforms

The first two use cases motivate the need for a rule interchange format towards facilitating automated, Web-based negotiations where rules are to be exchanged between involved parties. The first use case presented in this section shows the importance of such interchange for the reuse of electronic business documents (such as order requests and business policies) that are made available online.

Jane and Jack negotiate an electronic business contract on the supply of items by Jane's company. The negotiation process involves exchange of contract-related data and rules. Since the two companies may use different technologies, Jane and Jack agree upon the data model and use RIF for interchanging rules. The data is transmitted as XML documents using an agreed-upon format, together with the rules to run simulations, and results of these simulations are also represented as XML data.

A purchase order from Jack's company contains XML documents describing information on the desired goods, packaging, and delivery location and date. The associated rules describe delivery and payment policies. An example of such a rule is

```
If an item is perishable and it is
```

[36] Use case categories, `http://www.w3.org/2005/rules/wg/wiki/General_Use_Case_Categories`

[37] First W3C Public Working Draft of *Use Cases and Requirements* `http://www.w3.org/2005/rules/wg/ucr/draft-20060323.html`

```
delivered more than 10 days after the scheduled
delivery date then the item will be rejected.
```

Jane's company wants a relaxation of the delivery policy proposed by Jack. Thus, Jane proposes for acceptance and sends to Jack the following rule expressing the result of the negotiation carried out so far:

```
If an item is perishable and
it is delivered more than 7 days after the scheduled
delivery date but less than 14 days after the scheduled
delivery date then a discount of 18.7 percent will be
applied to this delivery.
```

Jack's company defines future requests for items in form of an appropriate XML schema document and a set of rules. This information is then published on their Web site, so as to give supply companies the possibility to respond electronically by sending XML cost sheets. Just like shown in the example scenario above, companies send rules expressed using RIF and Jack's company analyzes them for negotiation of electronic contracts.

Reactive rule-based systems (discussed already in Section 1) offer elegant means for implementing rules such as those exemplified in this use case. Rule-based negotiation frameworks or languages such as Protune [38] could be used for implementing this use case.

3.2 Negotiating eCommerce Transactions Through Disclosure of Buyer and Seller Policies and Preferences

This use case shows that a higher degree of interoperability can be gained by employing a rule interchange format within the process of establishing trust between the parties offering and those requesting a service in eCommerce scenarios. Automated trust establishment is possible when policies for every credential and every service can be codified, so as to minimize user intervention, the policies should be checked automatically whenever possible. The notion of policies is a quite general one referring to access control policies, privacy policies, business rules, etc. (see e.g. [BO05] for a more in-depth discussion of trust negotiation). Policies and credentials are themselves subject to access control. Thus, rule interchange is necessarily done during negotiation and disclosure of rules (in general) depends on the current level of trust that negotiating systems have achieved.

The interchange of rules is exemplified here with the negotiation between an online shop (eShop) and a customer (Alice) that wants to buy a device at eShop, a scenario that very similarly could apply to our running MoviShop example. Both Alice and eShop employ systems (agents) for establishing trust through negotiation with the goal of successful completion of the desired transaction. The negotiation is based on policies which describe which partners are trusted

[38] Protune, `http://www.l3s.de/~olmedilla/pub/2005/2005_policy_protune.pdf`

for what purposes, the credentials each system has, and which it will disclose if a certain level of trust is achieved.

Upon Alice's request to buy the desired device, eShop's agent sends parts of its policy back to Alice's agent. The two agents involved in the negotiation interchange a set of rules implementing the policies. Such an example rule is:

```
A buyer must provide credit card information together with
delivery information (address, postal code, city, country).
```

Alice's agent evaluates its own policies and the received ones for determining whether eShop's information request is consistent with her own policies. The agent uses policy rules such as:

```
Disclose Alice's credit card information only to
online shops belonging to the Better Business Bureau.
```

By disclosing the above given rule, Alice's agent asks eShop's agent to provide credentials stating that it belongs to the Better Business Bureau, Alice's most trusted source of information on online shops. Since eShop has such a credential and its policy states to release it to any potential customer, eShop's agent passes the credential to Alice's agent. Before disclosing credit card and delivery information to eShop, Alice's agent checks whether release of this information would not break Alice's denial constraints. These constraints are given by the following two rules:

```
Never disclose two different credit cards to the same
online shop.

For anonymity reasons, never provide both birth
date and postal code.
```

For this purchase, the birth date is not an issue and only information on one credit card is requested. Thus, Alice's constraints are respected. Alice's negotiation system therefore provides her credit card and delivery information to eShop. eShop checks that Alice is not in its client black list, then confirms the purchase transaction, generates an email notification to Alice that contains information about the purchase, and notifies eShops's delivery department.

3.3 Access to Business Rules of Supply Chain Partners

This use case shows that the existence of a rule interchange format would ease the integration of different business processes by offering business process designers a unified view over the used business rules while still allowing each involved company to work with their own technology of choice.

The focus of the use case is on the integration of supply chain business processes of multiple partners across company boundaries. Similar scenarios

can be encountered also within a single company that is organized into semi-independent business units. Each business unit might use different strategies (i.e., the logic behind the decisions taken in a process) and technologies.

Such situations as addressed by this use case occur usually when companies merge and their business processes need to be integrated. *Business processes* are "structured, measured sets of activities designed to produce a specific output for a particular customer or market" [Dav93]. A business process can span activities of a single company or of multiple companies. Often the strategies followed by an organization are specified by means of *business rules* [The00,Hal05]. In contrast to procedural code, business rules are easier to comprehend and use for business process designers, since they are declarative and high-level specifications.

For integrating supply chain business processes where multiple organizations are involved, part of the business processes' logic needs to be exposed (for determining the best strategy for integrating the processes) and, furthermore, business rules defined by different (partner) organizations need to be executed. There are two possible solutions for using such business rules and the decision of implementing one of them depends also on ownership constraints: Different rule sets could be merged into a single set of business rules, which can be then processed in a uniform manner on one side. The other possibility consists in accessing the other parties' rule sets only by invoking remote engines and (locally) processing their results only.

Consider an inspection of a damaged vehicle and the corresponding insurance adjustment process as examples of two processes that need to be integrated. Since the inspection of the vehicle is usually performed by independent inspectors, the inspection process can not be directly integrated into the adjustment process. The following business rules defines a decision point within the processes:

```
If inspector believes vehicle-is-repairable
then process-as-repair
otherwise process-as-total-loss.
```

The choice of sub-processes to be performed after the inspector's work depends on the decision taken using the above rule. In terms of business process modeling, the example given is an instance of the *exclusive choice pattern* [vdAtHKB03]; the insurance adjustment process branches into two alternative paths and exactly one of them can be chosen. Systems supporting *Event-Condition-Action-Alternative Action (ECAA)* offer an elegant solution for implementing such rules. As already noted in Section 1, for obtaining the same effect with a system supporting reactive rules of the *Event-Condition-Action* form, two ECA rules can be used where the condition of one rule is the negated condition of the other.

3.4 Managing Inter-Organizational Business Policies and Practices

This use case demonstrates the need for supporting annotation of rules in a rule interchange format, where rules or rulesets are labeled with tags carrying meta-information.

The use case addresses the need for interchanging rules that may not be directly executed by machines, i.e. rules that need input in form of a decision or confirmation from a person. The setting is somewhat similar to the use case of Section 3.3—multiple organizations and/or multiple units of a single organization—but the focus here is on the management of their business policies and practices. Another similarity consists in the fact that policies and practices are specified as business rules.

The scenario is based on the EU-Rent case study [EU-05], a specification of business requirements for a car rental company, promoted by the European Business Rules Conference [Eur05] and the Business Rules Group [Bus05]. EU-Rent operates in different EU countries. Thus, the company needs to comply with existing EU regulations and each of its branches needs to comply also with the regulations of the country it operates in.

CarWise and AutoLaw are consultancy companies that offer different services like, for instance, clarifying regulations on managing fleets of vehicles by negotiating with EU regulators and UK regulators, respectively. The outcome of such a service are interpreted regulations and rules that can be directly used by rule systems. CarWise and AutoLaw advise EU-Rent of rules that are to be used at European and UK level, respectively.

EU-Rent has a set of rules in place that implement the companies' policies and the EU regulations. Part of these rules are distributed to all EU-Rent branches. Thus, for example EU-Rent UK needs to integrate the received rules with the existing ones at the UK level. The rule set might change at the company level (case in which rules need to be propagated to the branches), but also at national (branches) level due to new national regulations. Changes of the rule set could be updates (modifications) or deletions (removal) of existing rules, or insertions of new rules into the rule set.

A concrete example rule that the EU-Rent corporate HQ could add is the following:

```
Each electronic compliance document must have its
required electronic signatures 48 hours before its
filing deadline.
```

Such kinds of (business) rules can be implemented by means of reactive rules. The event part of the reactive rule specifies the event *48 hours before the filing deadline*. The condition part queries for the existence of the electronic signatures. The action part reports an out-of-compliance situation.

All three different types of rules discussed in Section 1 are usually encountered in specifications of use cases for managing business policies and practices. The following example gives three (business) rules of the EU-Rent case study: the rule R1 is a deductive rule, R2 a normative rule, and R3 a reactive rule.

```
(R1) A customer who spends more than 1000 EUR per year
     is a Gold customer.
(R2) A customer can rent at most one car at a time.
```

```
(R3) A rental reservation must not be accepted if the
     customer is blacklisted.
```

Section 1 has also presented two possible implementations of rule *R3* using reactive rules. More generally, the article [BEPR06] analyzes the realization of business processes by means of ECA rules. With a focus on control flow, a concrete EU-Rent business process scenario is implemented using the reactive rule-based language XChange. This work has led to the introduction of a procedure notion in XChange which is absent in most other rule languages. The work has also shown that constructs for structuring rule sets are desirable in rule-based languages.

3.5 Other Use Cases

Apart from the four use cases, which we described in detail, let us briefly recap the other use cases specified in the RIF WG's use cases document. The complete use case descriptions, as published by the Working Group, can be found at `http://www.w3.org/TR/2006/WD-rif-ucr-20060710/`.

Collaborative Policy Development for Dynamic Spectrum Access This use case shows that by using a rule interchange format and deploying third-party systems, the flexibility in matching the goals of end-users of a service or device with the ones of providers and regulators of such services and devices can be increased.

The use case concentrates on examples from the dynamic spectrum access for wireless communication devices. It is assumed that the policies of a region and the protocols for dynamically accessing available spectrums are defined by rules. The goal is to have reconfigurable devices that can operate legally in various regulatory and service environments. One of the technical preconditions of making this possible relies on the format in which the rules are expressed and the ability of devices to understand and use these rules.

To be able to use the advantages of a rule interchange format in this setting, a third-party group should be formed, which is responsible for translating regional policies and protocols into the interchange format.

Ruleset Integration for Medical Decision Support This use case motivates the need for merging rule sets written in different rule languages. This allows for inferring data that could not be obtained without the merge. This is an important task of expert systems based on reasoning with rules. Complex decision making systems use different data bases, ontologies, and rules written in different rule languages.

The use case gives examples from the medical domain, where different data sources come into play, such as pharmaceutical, patient data bases, and medical ontologies.

Interchanging Rule Extensions to OWL The use case gives a concrete motivation for a rule interchange format that is compatible with OWL (as required also by the W3C RIF WG charter). This is the shortest use case in the Second W3C Public Working Draft of *Use Case and Requirements*, yet a considerable number of application domains such as medicine, biology, and e-Science, which partly adopted OWL already, could benefit from the combination of rules with ontologies. The domain of labeling brain cortex structures in MRI (Magnetic Resonance Imaging) images is chosen here for illustration purposes. A deductive rule is exemplified by use of which implicit knowledge, i.e. dependencies between ontology properties, is inferred which cannot be expressed directly in OWL itself.

Vocabulary Mapping for Data Integration Different application domains such as health care, travel planning, and IT management often need solutions to the problems raised by integrating information from multiple data sources with different data representation models. The idea of this use case relies on the reusability of rules that implement mappings between such data models.

The use case gives examples from IT systems management. The concrete example uses three different data sources, which are taken as basis for analyzing the flexibility of a division's business processes with respect to changes of their IT management contracts. In many cases, the simple solution of mapping the information of the three data sources to a single data format such as RDF does not offer satisfactory results. One of the problems lies in the different granularities of the contained data, which might be either too detailed or too coarse grained. Thus, deductive rules—possibly involving aggregation of fine-grained data—are used for defining simple and usable views over the data sources. The implemented deductive rules are then published so as to be reused across the company, where similar views are needed.

BPEL Orchestration of Web Services The use case exemplifies a commercial credit approval Web service implemented as a BPEL orchestration of two Web services, a credit history service and a rule-based decision service. A rule interchange format would allow the re-use of rules for evaluating credit histories. Moreover, rule editing and customization tools from different RIF compatible vendors would ease the rule specification task.

Three rule sets for credit evaluation are used, which are executed sequentially. The first two sets of rules calculate threshold values and a credit score, while the third set of rules compares these values and makes the decision to approve or deny the credit. The outcome of the decision system is in form of an XML document informing about the decision and the reason(s) for it. The need for querying and constructing XML data comes here into play, since for replying to the customer, the answer needs to be constructed based on the XML data received from the Web Service.

Publishing Rules for Interlinked Metadata The use case stresses the importance of specifying implicit knowledge in form of rules and publishing them

for re-use, so as to allow interlinking (meta-)data and rules from different Web sources. Semantic Web technologies such as RDF allow publishing metadata expressing the semantics of data in machine-readable form. Often, such (explicit) knowledge is supplemented by rules capturing implicit knowledge. Thus, a more concise representation of data is obtained, maintenance of meta-data is simplified, and storage requirements are reduced.

The role of a rule interchange format is exemplified by means of the movie database scenario presented in Section 1.1 that we have already used throughout this paper. The example rule *R1* given in Section 1.2, which constructs a view of black and white movies, captures such implicit knowledge that can be further used. However, also more complex rules implicitly linking and cross-referencing between several online sources or involving (scoped) negation are covered, e.g.

```
(R3)
   IF movie M is  listed at http://AlternativeMDB.example.org but not
      listed at http://imd.example.org
   THEN M is an independent movie
```

4 W3C Requirements on a Rule Interchange Format

As already mentioned in Section 3, each use case imposes certain requirements that should be taken into account when developing RIF. Consider again our running example shortly described in Section 1.1. John Doe's MoviShop uses data on movies, customers, etc., expressed in XML, RDF, RDFS, and OWL. Thus, to exchange rules over this data, RIF should offer support for XML, RDF, and OWL. More details on what *support* means in such a context are given by the corresponding requirements on data representation models discussed in this section. If RIF (core or some dialect of it) meets the requirements posed by MoviShop's rules and the rule language R used for implementing them, John needs a translator from R to RIF for interchanging (part of) its rules. For developing such translators, one of the general requirements for RIF concerns the precise syntax and semantics, which are to become starting points for correct translator implementations.

This section shortly discusses the requirements on RIF, which have been approved by the W3C RIF WG and published in the Second Public Working Draft of 'RIF Use Cases and Requirements'. The process of gathering and deciding upon requirements on RIF has taken into account three sources of requirements: Firstly, we have considered the (explicit and implicit) requirements posed by the RIF use cases in the above mentioned document; we have determined a set of requirements that are posed by each of these use cases—we call them *general requirements*. Secondly, each WG participant have had the possibility to propose requirements that she or he considered relevant for RIF. The third source of requirements is the so-called RIF Rulesystems Arrangement Framework [39]

[39] RIFRAF, `http://www.w3.org/2005/rules/wg/wiki/Rulesystem_Arrangement_Framework`

(RIFRAF), a framework that classifies rule systems based on a set of discriminators. RIFRAF is to be used also as basis for determining desired RIF dialects. The outcome of the work on RIF requirements is presented next by dividing the requirements into *general*, *Phase I*, and *Pase II requirements*. The focus is more on the general and Phase I requirements, since the Phase II requirements are ongoing work at moment of writing.

4.1 General Requirements

The following requirements on RIF have a general character in the sense that they express high-level conditions that RIF and the translators from and to RIF need to meet.

- *Implementability:* RIF must be implementable using well understood techniques, and should not require new research in e.g. algorithms or semantics in order to implement translators.
- *Semantic precision:* RIF core must have a clear and precise syntax and semantics. Each standard RIF dialect must have a clear and precise syntax and semantics that extends RIF core.
- *Extensible Format:* It must be possible to create new dialects of RIF and extend existing ones upwardly compatible.
- *Translators:* For every standard RIF dialect it must be possible to implement translators between rule languages covered by that dialect and RIF without changing the rule language.
- *Standard components:* RIF implementations must be able to use standard support technologies such as XML parsers and other parser generators, and should not require special purpose implementations when reuse is possible.

4.2 Phase I Requirements

The list of requirements given next refers to the RIF developed within Phase I. It consists of requirements on the core interchange format.

- *Compliance model:* RIF must define a compliance model that will identify required/optional features.
- *Default behavior:* RIF must specify at the appropriate level of detail the default behavior that is expected from a RIF compliant application that does not have the capability to process all or part of the rules described in a RIF document, or it must provide a way to specify such default behavior.
- *Different semantics:* RIF must cover rule languages having different semantics.
- *Embedded comments:* RIF must be able to pass comments.
- *Embedded metadata:* RIF must support metadata such as author and rule name.
- *Limited number of dialects:* RIF must have a standard core and a limited number of standard dialects based upon that core.

- *OWL data:* RIF must cover OWL knowledge bases as data where compatible with Phase I semantics.
- *RDF data:* RIF must cover RDF triples as data where compatible with Phase I semantics.
- *Rule language coverage:* RIF must cover the set of languages identified in the Rulesystem Arrangement Framework (RIFRAF). This requirement acts as an umbrella for a set of requirements, which are expected as outcome of the work on RIFRAF.
- *Dialect Identification:* RIF must have a standard way to specify the dialect of the interchanged rule set in a RIF document. As the rule interchange format developed within the W3C RIF WG—RIF—will come in form of a core (RIF Core) and a set of dialects extending the core interchange format, a mechanism is needed for specifying which RIF dialect is used for a set of rules to be interchanged. This plays a role for example in the case that incompatible RIF dialects exist.
- *XML syntax:* RIF must have an XML syntax as its primary normative syntax.
- *XML types:* RIF must support an appropriate set of scalar datatypes and associated operations as defined in XML Schema part 2 and associated specifications. This requirement is also stated in the W3C RIF WG charter.
- *Merge Rule Sets:* RIF should support the ability to merge rule sets. The big interest in a standardized interchange format for rules is also determined by the possibility of merging rule sets written in different rule languages through RIF. This requirement is also explicitly stated e.g. in the use case 'Ruleset Integration for Medical Decision Support'.
- *Identify Rule Sets:* RIF will support the identification of rule sets.

4.3 Phase II Requirements

The *Second Public Working Draft of 'RIF Use Cases and Requirements'* contains one single requirement for the second phase of RIF development, namely that *RIF must be able to accept XML elements as data*. The list of Phase II requirements will of course be extended in the near future. At moment of writing, the WG started the work on gathering other requirements for the RIF dialects to be developed within Phase II. Under discussion are requirements such as the full coverage of RDF and OWL, or the support for external calls (e.g. to a SPARQL query processor). For an elegant implementation of rule R2 of our running example, which is employed for denying requests from blacklisted MoviShop customers, a RIF dialect is needed where e.g. action specifications are allowed in the rules head. In other words, a RIF dialect for reactive rules or just a form of them (such as production or ECA rules) is desirable. Whether or not the RIF WG will develop such a RIF dialect depends largely on the interest of its participants, their willingness to work towards a *'reactive'* dialect, and its acceptance by the WG as a whole.

5 The Rule Interchange Format: Current Core and Possible Extensions

This section discusses the current work of the W3C RIF WG on the RIF Core and means for determining the RIF standard dialects that will extend this core interchange format.

In the first phase, the working group agreed on defining a deliberately inexpressive core language which covers features available in most common rule languages and devise a first proposal for a common exchange syntax for this language.

Going back to our classification of rules from the introduction, where we divided rule languages into deductive, normative, and reactive rules, let us briefly recap the common ingredients which are necessary to model these rules.

All the rules we mentioned were in some sense checking a *condition* on a (static or dynamic) knowledge base. In our examples from the introduction, we called this the `IF` part for deduction rules and production rules; queries and normative rules can, as a whole be viewed as checking a condition. This common feature of RIF rules is acknowledged in that RIF Core will provide a simple logic language to specify such conditions.

The *RIF Condition Language* (which at present is a working title for a simple language fragment to express these common conditions) is the fundamental layer shared by Logic Programming rules (based on the Horn subset of first-order logic), production (Condition-Action) rules, Event-Condition-Action rules, normative rules (integrity constraints), and queries.

This RIF Condition Language will thus provide means to exchange basic conditions, consisting of simple conjunctions and disjunctions of atomic formulas with existential variables, as well as a distinguished equality predicate. Starting from the requirement to support sorted constants and variables, this core dialect is developed as a general multisorted logic, whose sorts can be "webized", i.e. referenced by IRIs and aligned with relevant XML standards for typing, such as XML Schema datatypes, as well as, later on, OWL and RDFS classes. Other constructs of this language (constants, functions, predicates, etc.) can also be identified by IRIs.

As an example of an extension layer on top of the RIF Condition Language, the first working draft of RIF Core introduces the *RIF Horn Rule Language* as chartered for RIF Phase 1. Because of the underlying "Condition Logic with Equality and Sorts" we obtain a "Multi-Sorted Horn Logic with Equality". It will turn out that, given the former, only a small extra effort is required to obtain the latter: the main part of a Horn rule is its body, and this is exactly a condition in our sense and already sufficient for expressing simple rules. The Horn rule layer will allow simple inference of atomic formulae, given that the respective condition holds, thus it covers deductive rules and assert-only production rules. The definition of dynamic aspects, namely event and non-assert action parts, is currently under discussion.

Next steps will include extensions such as built-ins and negation, which are useful features not only for the Horn Rule dialect, but also for a potential Pro-

duction Rule dialect of RIF. Moreover, it is planned to define adequate (RDF) metadata for RIF rules and rulesets, to specify how RDF data can be processed by RIF rules, and to embed RIF rules into RDF statements.

Now let us have a closer look at the condition language as it is defined in the first working draft.

5.1 The RIF Core Condition Language – Syntax

The basis of the language in Phase 1 is formed by conjunctive conditions that can appear in the bodies of Horn-like rules with equality. Disjunctions of these conditions are also allowed because such generalized rules are known to reduce to the pure Horn case.

The first working draft of the RIF Core document [Be07] develops a syntax and semantics for such RIF conditions, which also supports a basic set of primitive data types (such as xsd:long, xsd:string, xsd:decimal, xsd:time, xsd:dateTime, taken from the respective XML Schema datatypes [Be04]).

In order to support a general approach, as well as possible future higher-order dialects based on HiLog [CKW93b] and Common Logic [(ed06], the RIF Core language does not separate symbols used to denote constants from symbols used as names for functions or predicates. Instead, all these symbols are drawn from the same universal domain. When desired, separation between the different kinds of symbols is introduced through the mechanism of *sorts*, which will also be used for "typing" arguments as mentioned before. In logic, the mechanism of sorts is used to classify symbols into separate subdomains. One can decide that certain sorts are disjoint (for example, decimal and dateTime) and others are not (for example, integer could be a subsort of the sort decimal). Control of what sorts can be used for predicate (or concept) names, for function symbols, and so on, shall, by the general mechanism introduced in RIF Core, be upon agreement between the rule exchanging parties, or the ones defining a specific RIF dialect.

Figure 5 shows a snapshot of the current RIF condition meta-model.

Based on this metamodel, two syntaxes are currently proposed for simple conditions. A preliminary XML syntax, as well as a more readable syntax which is similar in style to what in OWL is called the Abstract Syntax [PSHH04]. This latter syntax, which resembles other standard syntaxes for (variants of) first-order logic, is based on the following EBNF, used in examples of the first RIF Core working draft:

```
CONDITION   ::= CONJUNCTION | DISJUNCTION | EXISTENTIAL | ATOMIC
CONJUNCTION ::= 'And' '(' CONDITION* ')'
DISJUNCTION ::= 'Or' '(' CONDITION* ')'
EXISTENTIAL ::= 'Exists' Var+ '(' CONDITION ')'
ATOMIC      ::= Uniterm | Equal
Uniterm     ::= Const '(' TERM* ')'
Equal       ::= TERM '=' TERM
TERM        ::= Const | Var | Uniterm
```

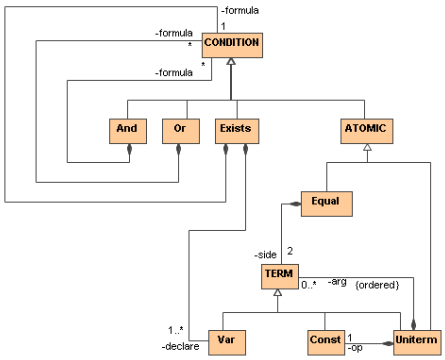


Fig. 5. The RIF Core condition meta-model

```
Const        ::= CONSTNAME | '"'CONSTNAME'"''^^'SORTNAME
Var          ::= '?'VARNAME | '?'VARNAME'^^'SORTNAME
```

The terminal and non-terminal symbols in this EBNF should be largely self-explanatory and we refer the reader to [Be07] for details. Note that the productions for constants (`Const`) and variables (`Var`) include optionally sorted versions. Most knowledge representation, programming and rule languages allow/require "typing" of constants and also of variables: take for example typed literals in RDF, or variables in common programming languages, which is accounted for by the *Multisorted RIF Logic*, to be discussed in more detail below. At this point we do not commit to any particular vocabulary for the names of constants, variables, or sorts.

Example 5 (Running example (cont'd)). For instance, in John's rule for classifying old movies as black-and-white, the condition part written in first-order logic looks as follows:

$$\exists D, Y.\ MoviShopDvd(D) \wedge shows(D, M) \wedge \\ IMDMovie(M) \wedge IMDYear(M, Y)\ \wedge\ before(Y, 1930)$$

In RIF's EBNF syntax, he could write this straightforwardly:

```
Exists ?D ?Y (
    And ( "moviShop:Dvd"( ?D ) "imd:shows"( ?D ?M )
          "imd:Movie"( ?M ) "imd:Year"( ?M ?Y )
          "op:date-less-than"( ?Y "1930-01-01T00:00:00Z"^^dateTime ) ) )
```

Note that the names of the predicates are IRIs and thus are "webized." In the future, builtin predicates, like `"op:date-less-than"` will be also standardized around XPath and XQuery functions [MMe07].

Note that in the above condition there is one free (i.e. non-quantified) variable. Free variables arise because we are dealing with formulas that might occur in a rule IF part. When this happens, the free variables in a condition formula shall also occur in the rule THEN part. We will see that such variables are

quantified universally outside the rule, and the scope of such quantification is the entire rule.

An XML syntax for this condition language is currently under discussion, where element names will be close to the metamodel/EBNF. Let us turn now to the model theory behind RIF's Condition Language.

5.2 The RIF Core Condition Language – Semantics

The first step in defining a model-theoretic semantics for a logic-based language is to define the notion of a semantic structure, also known as an interpretation. RIF takes here, in order to be general and extensible, an approach common to systems that not only cover classical first-order logic, but also uncertainty or inconsistency (which are clearly important in the open Web environment), i.e., multi-valued logics. Here, truth values are not only `f` ("false") and `t` ("true"), but a set of truth values TV, which has a total or partial order, called the truth order (denoted with $<_t$). For instance, in classical logic this order is simply `false` $<_t$ `true`, whereas logics dealing with uncertainty or inconsistency often are four-valued logics, e.g. with a partial order `f` $<_t$ `u` $<_t$ `t` and `f` $<_t$ `i` $<_t$ `t`, where `u` and `i` denote "unknown" and "inconsistent", respectively.[40]

Moreover, since RIF on the syntactic level does not distinguish between constants, functions, and predicates, a semantic structure, I, is defined as a tuple of mappings $\langle I_C, I_V, I_F, I_R\rangle$, which determines the truth value of every formula, as explained below. Here, I_C, I_V, I_F, and I_R denote the interpretation of the domain elements as, respectively, constants, variables, functions, and relations.

Definition 1 (Interpretation). *Let D be a non-empty set of elements called the domain of I, `Const` the set of constants, predicate names, and function symbols, and `Var` the set of variables.*

An interpretation $I = \langle I_C, I_V, I_F, I_R\rangle$ *consists of four mappings:*

- I_C : `Const` $\rightarrow D$
- I_V : `Var` $\rightarrow D$[41]
- I_F : *from* `Const` *to functions from* $D^* \rightarrow D$, *where* D^* *is a set of all tuples of any length over the domain* D
- I_R : *from* `Const` *to truth-valued mappings* $D^* \rightarrow TV$

Using these mappings, we can define a more general mapping, I, as follows:

- $I(k) = I_C(k)$ *if* $k \in$ `Const`
- $I(?v) = I_V(?v)$ *if* $?v \in$ `Var`
- $I(f(t_1 ... t_n)) = I_F(f)(I(t_1), \ldots\ldots, I(t_n))$

Finally, the mapping $I_{Truth} : \phi \rightarrow TV$ for conditions ϕ is defined inductively:

[40] Such logics can also be given another partial order $<_k$, called the knowledge order: `u` $<_k$ `t` $<_k$ `i` and `u` $<_k$ `f` $<_k$ `i`. See, e.g., [Fit02] for details.

[41] This is also often called variable assignment elsewhere in the literature.

- *Atomic formulas:* $I_{Truth}(r(t_1...t_n)) = I_R(r)(I(t_1), ..., I(t_n))$
- *Equality:* $I_{Truth}(t_1 = t_2) =$ `t` *iff* $I(t_1) = I(t_2)$; $I_{Truth}(t_1 = t_2) =$ `f` *otherwise.*
- *Conjunction:* $I_{Truth}($ `And ( ` $c_1 \ \ldots \ c_n$ `)` $) = min_t(I_{Truth}(c_1), \ldots,$ $I_{Truth}(c_n))$*, where* min_t *is minimum with respect to the truth order.*
- *Disjunction:* $I_{Truth}($ `Or ( ` $c_1 \ \ldots \ c_n$ `)` $) = max_t(I_{Truth}(c_1), \ldots, I_{Truth}(c_n))$*, where* max_t *is maximum with respect to the truth order.*
- *Quantification:* $I_{Truth}($ `Exists` $?v_1 \ \ldots \ ?v_n \ (c)) = max_t(I'_{Truth}(c))$ *where* max_t *is taken over all interpretations* $I' =< I_C, I'_V, I_F, I_R >$*, which agree with* I *everywhere except possibly in the interpretation* I'_V *of the variables* $?v_1, \ldots ..., ?v_n$*.*

Multisorted RIF Logic As mentioned earlier and also seen in the EBNF, one may attach *sorts* from a set of primitive sorts (defined for a RIF dialect) to constants and variables. The list of supported primitive sorts in RIF Core (which probably will be extended later on) is: `long`, `string`, `decimal`, `time`, and `dateTime`. Signatures for function and relation symbols specify their arity and argument (and value) sorts. The current syntax to declare function sorts is:

```
':- signature' '"'NAME'"' s1 '*' ...'*' sn '→' s ','
                          r1 '*' ...'*' rk '→' r ',' ...
```

Relation (or predicate) sorts are declared similarly:

```
':- signature' '"'NAME'"' s1 '*' ...'*' sn ','
                          r1 '*' ...'*' rk ',' ...
```

For instance, the sorts of the XPath/XQuery relation `op:date-less-than` used above could be defined by this signature:

```
:- signature "op:date-less-than" dateTime * dateTime
```

Interpretations of multi-sorted RIF dialects extend Definition 1 by new functions to assign primitive sorts and function sorts assign a set of allowed primitive sorts to the symbols of `Const`. The details of multi-sorted interpretations are currently being worked out, but they seem to be an important feature, as many languages (including RDF, Prolog, HiLog and F-Logic, Common Logic) support signature declarations and/or typed literals and variables.

5.3 RIF Core Horn Rules

As a first simple core format for a complete but minimal rules interchange language, the RIF WG defined a *RIF Core Rule Language* by extending the RIF Core Condition Language, where conditions become rule bodies. RIF Phase 1 covers only the expressivity of Horn Rules, i.e. rules with one positive derived atomic formula in the head (or THEN part).[42]. This simple rules dialect extends the EBNF syntax for Core Conditions by the following productions:

[42] Note, that the minor extensions such as allowing existentials and disjunctions in the body via RIF Core conditions do not increase the expressive power of the language above Horn.

```
Ruleset  ::= RULE*
RULE     ::= 'Forall' Var* CLAUSE
CLAUSE   ::= Implies | ATOMIC
Implies  ::= ATOMIC ':-' CONDITION
```

The symbol `:-` denotes the implication connective used in rules. The statement `ATOMIC :- CONDITION` should be informally read as if `CONDITION` is true then `ATOMIC` is also true. We deliberately avoid using the connective $\leftarrow$ here because in some RIF dialects, such as Logic Programming dialects and Production Rules dialects, the implication `:-` will have different meaning from the meaning of the first-order implication $\leftarrow$.

The upcoming envisioned RIF dialects will extend this core rule language by generalizing the positive RIF conditions in the bodies, and probably they will also allow more expressive rule heads.

RIF Core Horn Rules – Semantics In Section 5.2 above we already defined the notion of semantic structures and the truth value of a RIF condition in such a semantic structure (interpretation).

While semantic structures can be multivalued, rules are typically two-valued even in logics that support inconsistency and uncertainty: a rule is either satisfied in an Interpretation I (true) or not (false). We can define satisfaction of a rule '*head* `:-` *body*' in Interpretation I, denoted by $I \models$ *head* `:-` *body* simply as follows:

$$I \models \textit{head} \texttt{ :- } \textit{body} \text{ iff } I_{Truth}(\textit{head}) \geq_t I_{Truth}(\textit{body})$$

Note that, since in RIF Core we consider Horn clauses, where free variables are assumed to be universally quantified over the whole rule, strictly speaking, we need to refine this to: $I \models \textit{clause}$ iff $I' \models \textit{clause}$ for every I' that agrees with I everywhere except possibly on some variables free in *clause*. In this case, we also say that I is a model of the clause. I is a model of a rule set R iff it is a model of every rule in R.

The notion of a model is only the basic ingredient in the definition of a semantics of a rule set. In general, the semantics of a rule set R is the set of its *intended* models (see e.g. [Sho87]). There are different theories of what the intended sets of models are supposed to look like depending on the features of the particular rule sets.

For Horn rules, which we use in this section, the intended set of models of R is commonly agreed upon: it is the set of *all* models of R. However, in (future) rule dialects which allow constructs such as nonmonotonic negation (aka negation-as-failure) in rule bodies, only some of the models of a rule set are viewed as intended. This issue will be addressed in the appropriate dialects of RIF. The two most common theories of intended models are based on the so called well-founded models [GRS88] and stable models [GL88].

Future extensions of the presented RIF Core will need to enable the provider of a rule set to be interchanged to declare explicitly what notion of intended models are assumed in this rule set.

Example 6 (Running example (cont'd)). Finally, John Doe can publish and exchange his rule

```
(R1)
   IF movie M was produced before 1930
   THEN M is a black and white movie
```

which declares his definition of black and white movies using RIF Core:

```
"moviShop:BWMovie" ( ?M ) :-
  Exists ?D ?Y (
    And ( "moviShop:Dvd"( ?D ) "imd:shows"( ?D ?M )
          "imd:Movie"( ?M ) "imd:Year"( ?M ?Y )
          "op:date-less-than"( ?Y "1930-01-01T00:00:00Z"^^dateTime ) ) )
```

Not too bad, but John waits what's next and when he will be able to exchange more complex rules such as

```
(R2)
   IF movie M is  listed at http://altmd.example.org but not
      listed at http://imd.example.org
   THEN M is an independent movie
(R3)
   ON request from customer C to book a movie
   IF customer C is blacklisted
   DO deny C's request
```

and he is eagerly waiting for a complete RIF which will enable him to do so. The current core does not yet provide this feature, but is carefully designed to enable plugging in of different forms of negation or various models of events and actions in RIF dialects to be defined in the future.

5.4 What's next?

The core rules fragment we have seen so far will provide the basis for further dialects to cover richer features and express rules and rulesets beyond simple Horn. The current efforts towards a very general model theory, catering for multi-sorted and multi-valued logic extensions do not seem necessary for simple Horn rules, but will enable upward compatible extensions of this common basis.

Currently, the underlying metamodel, introduced at the beginning of this section, is also being extended towards a base ontology for describing the available features of existing rule languages and systems. The working group has analyzed this feature space and collected a list of discriminators (distinguishing features) in the so called RIF Rules Arrangement Framework[43] (RIFRAF). Aligning the RIF Core Metamodel with RIFRAF in a common ontology will help RIF users to classify their rule sets and features with respect to the upcoming family of possibly diverging RIF dialects. We point out here again, that it is not the goal of RIF to provide a one-for-all rule language which can cover all of these features. Distinct features of different rule systems are often simply incompatible.

[43] `http://www.w3.org/2005/rules/wg/wiki/Rulesystem_Arrangement_Framework`

Rather, the concept of RIF dialects will enable the exchange of rules within common fragments or variable feature sets between various parties in a modular fashion.

6 Conclusion

This paper has presented a snapshot of the current working drafts of the W3C RIF Working Group, which is working on the development of an interchange format for rules. The need for such a format has been motivated by the use cases of the Second Public Working Draft of the RIF WG, which we have described briefly. We illustrated the main ideas behind rule interchange using an example of a DVD rental store. The example illustrates that in order to adequately represent the services provided by MoviShop we need different types of rules: *deductive*, *normative*, and *reactive*. MoviShop's rules work with the data obtained from different data sources. These data may be expressed using XML, RDF, and/or OWL. The types of rules and the data (and metadata) representation models needed for the task pose a number of requirements to the interchange format for MoviShop's rules. These requirements drive the development of a core interchange format for rules—the *RIF Core*—and its extensions—*RIF dialects*.

As of this writing, the RIF WG has released the First Public Working Draft of `RIF Core`. It includes the *RIF Condition Language* and its extension to the *RIF Horn Rule Language*. The RIF Condition Language is expected to be used for expressing the part of rules that is common to deductive, normative, and reactive rules, namely the `IF` part, as shown via examples. The *RIF Horn Rule Language* is intended to allow exchanging Horn-style deductive rules. This core language is clearly not sufficient for interchanging many kinds of rules, but it offers the basis for future extensions most of which will be developed as RIF dialects.

Acknowledgments

This research has been partly funded by the European Commission and by the Swiss Federal Office for Education and Science within the EU FP6 Network of Excellence REWERSE[44] (IST-506779). This work was done while Michael Kifer was visiting DERI Innsbruck. His work was supported in part by the BIT Institute, NSF grants CCR-0311512 and IIS-0534419, and by US Army Research Office under a subcontract from BNL. The work of Axel Polleres was partly supported by the EU FP6 project inContext (IST-034718)[45] as well as by the Spanish MEC and Universidad Rey Juan Carlos under the project SWOS (URJC-CM-2006-CET-0300). We also thank NSERC for its support through the discovery grant of Harold Boley.

[44] `http://rewerse.net/`
[45] `http://www.in-context.eu/`

References

[ABdB+05] Jürgen Angele, Harold Boley, Jos de Bruijn, Dieter Fensel, Pascal Hitzler, Michael Kifer, Reto Krummenacher, Holger Lausen, Axel Polleres, and Rudi Studer. Web Rule Language (WRL), September 2005. W3C member submission.

[Bar03] Chitta Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press, 2003.

[BBB+] S. Battle, A. Bernstein, H. Boley, B. Grosof, M. Grüninger, R. Hull, M. Kifer, D. Martin, D. L. McGuinness, S. McIlraith, J. Su, and S. Tabet. Semantic web services framework (SWSF).

[BBB+07] Bruno Berstel, Pilippe Bonnard, François Bry, Michael Eckert, and Paula-Lavinia Pătrânjan. Reactive Rules on the Web. In *Reasoning Web – Third International Summer School 2007, Tutorial Lectures*, 2007. In preparation.

[BBCC02] Angela Bonifati, Daniele Braga, Alessandro Campi, and Stefano Ceri. Active XQuery. In *18th Int. Conf. on Data Engineering (ICDE2002)*, San Jose, California, 2002.

[BBEP05] James Bailey, François Bry, Michael Eckert, and Paula-Lavinia Pătrânjan. Flavours of XChange, a rule-based reactive language for the (Semantic) Web. In *Proc. Int. Conf. on Rules and Rule Markup Languages for the Semantic Web*, volume 3791 of *LNCS*. Springer, 2005.

[Be04] Paul V. Biron and Ashok Malhotra (eds.). XML Schema Part 2: Datatypes, Second Edition, October 2004. W3C Recommendation.

[Be07] Harold Boley and Michael Kifer (eds.). RIF Core Design, 2007. W3C Editor's Draft, in preparation.

[Bec06] Dave Beckett. Turtle - Terse RDF Triple Language, April 2006. Available at `http://www.dajobe.org/2004/01/turtle/`.

[BEP06a] F. Bry, M. Eckert, and P.-L. Patranjan. Reactivity on the web: Paradigms and applications of the language xchange. *Journal of Web Engineering*, 5(1):3–24, 2006.

[BEP06b] François Bry, Michael Eckert, and Paula-Lavinia Pătrânjan. Reactivity on the Web: Paradigms and applications of the language XChange. *J. of Web Engineering*, 5(1):3–24, 2006.

[BEPR06] François Bry, Michael Eckert, Paula-Lavinia Pătrânjan, and Inna Romanenko. Realizing business processes with eca rules: Benefits, challenges, limits. In *Proceedings of 4th Workshop on Principles and Practice of Semantic Web Reasoning , Budva, Montenegro (10th–11th June 2006)*, LNCS, 2006.

[BFG+01] Robert Baumgartner, Sergio Flesca, Georg Gottlob, Robert Nieuwenhuis, and Andrei Voronkov. The elog web extraction language. In *LPAR 2001 : Logic for Programming, Artificial Intelligence, and Reasoning*, volume 2250 of *LNCS*, Havana, Cuba, December 2001. Springer.

[BFMS06] Erik Behrends, Oliver Fritzen, Wolfgang May, and Franz Schenk. Combining ECA Rules with Process Algebras for the Semantic Web. In *Proceedings of Second International Conference on Rules and Rule Markup Languages for the Semantic Web, Athens, Georgia, USA (10th–11th November 2006)*, pages 29–38, 2006.

[BK94] A.J. Bonner and M. Kifer. Transaction logic programming (or a logic of declarative and procedural knowledge). Technical Report CSRI-270,

University of Toronto, April 1992. Revised: February 1994. http://www.cs.toronto.edu/~bonner/transaction-logic.html.

[BK98] A.J. Bonner and M. Kifer. A logic for programming database transactions. In J. Chomicki and G. Saake, editors, *Logics for Databases and Information Systems*, chapter 5, pages 117–166. Kluwer Academic Publishers, March 1998.

[BL05] Tim Berners-Lee. Web for Real People, April 2005. Keynote Speech at the 14th World Wide Web Conference (WWW2005). Slides available at http://www.w3.org/2005/Talks/0511-keynote-tbl/.

[BM05] François Bry and Massimo Marchiori. Ten theses on logic languages for the Semantic Web. In *Proc. Int. Workshop on Principles and Practice of Semantic Web Reasoning*, volume 3703 of *LNCS*. Springer, 2005.

[BO05] Piero A. Bonatti and Daniel Olmedilla. Driving and monitoring provisional trust negotiation with metapolicies. In *IEEE Int. Workshop on Policies for Distributed Systems and Networks*. IEEE Comp. Soc., 2005.

[Bon05] Piero A. Bonatti. Rule languages for security and privacy in cooperative systems. In *29th Annual International Computer Software and Applications Conference (COMPSAC 2005), 25-28 July 2005, Edinburgh, Scotland, UK*, pages 268–269, 2005.

[BPW02] James Bailey, Alexandra Poulovassilis, and Peter T. Wood. An event-condition-action language for XML. In *Proc. Int. World Wide Web Conf.*, pages 486–495. ACM, 2002.

[BS04] François Bry and Uta Schwertel. REWERSE – reasoning on the Web. *AgentLink News*, 15, 2004.

[Bus05] Business Rules Group. www.businessrulesgroup.org, 2005.

[CH01] D. Cabeza and M. Hermenegildo. Distributed www programming using (ciao-)prolog and the pillow library. *Theory and Practice of Logic Programming*, 1(3):251–282, May 2001.

[CKW93a] W. Chen, M. Kifer, and D.S. Warren. HiLog: A foundation for higher-order logic programming. *Journal of Logic Programming*, 15(3):187–230, February 1993.

[CKW93b] W. Chen, M. Kifer, and D.S. Warren. HILOG: A foundation for higher-order logic programming. *Journal of Logic Programming*, 15(3):187–230, 1993.

[Dav93] Thomas H. Davenport. *Process Innovation: Reengineering Work through Information Technology.* Havard Business School Press, 1993.

[dBEPT06] Jos de Bruijn, Thomas Eiter, Axel Polleres, and Hans Tompits. On representational issues about combinations of classical theories with nonmonotonic rules. In *Proceedings of the 1st International Conference on Knowledge Science, Engineering and Management (KSEM'06)*, LNCS, Gullin, China, August 2006. Springer. Invited paper.

[DSB+04] Mike Dean, Guus Schreiber, Sean Bechhofer, Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. OWL Web Ontology Language Reference, February 2004. W3C Recommendation.

[(ed06] Harry Delugach (ed.). Iso common logic, 2006. Available at http://philebus.tamu.edu/cl/.

[EGM97] Thomas Eiter, Georg Gottlob, and Heikki Mannila. Disjunctive datalog. *ACM Transactions on Database Systems*, 22(3):364–418, September 1997.

[EIP+06] Thomas Eiter, Giovambattista Ianni, Axel Polleres, Roman Schindlauer, and Hans Tompits. Reasoning with rules and ontologies. In Pedro Barahona, François Bry, Enrico Franconi, Ulrike Sattler, and Nicola Henze, editors, *Reasoning Web 2006*, volume 4126 of *LNCS*, pages 93–127. Springer, September 2006.

[EU-05] EU-Rent Case Study. `www.eurobizrules.org/ebrc2005/eurentcs/eurent.htm`, 2005.

[Eur05] European Business Rules Conference. `www.eurobizrules.org`, 2005.

[FFLS99] Mary F. Fernandez, Daniela Florescu, Alon Y. Levy, and Dan Suciu. Verifying integrity constraints on web sites. In *IJCAI*, pages 614–619, 1999.

[Fit02] Melvin Fitting. Fixpoint semantics for logic programming – a survey. *Theoretical Computer Science*, 278(1-2):25–51, 2002.

[FLB+06] Tim Furche, Benedikt Linse, François Bry, Dimitris Plexousakis, and Georg Gottlob. Rdf querying: Language constructs and evaluation methods compared. In *Reasoning Web 2006*, volume 4126 of *LNCS*, pages 1–52. Springer, September 2006.

[For82] Charles Forgy. RETE: A fast algorithm for the many pattern/many object pattern match problem. *Artificial Intelligence*, 19:17–37, 1982.

[GHMe] Allen Ginsberg, David Hirtle, Frank McCabe, and Paula-Lavinia Patranjan (eds.). RIF Core Design. W3C Working Draft.

[GL88] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In Robert A. Kowalski and Kenneth Bowen, editors, *5th Int'l Conf. on Logic Programming*, pages 1070–1080, Cambridge, Massachusetts, 1988. The MIT Press.

[GRS88] Allen Van Gelder, Kenneth Ross, and John S. Schlipf. Unfounded sets and well-founded semantics for general logic programs. In *7th ACM Symposium on Principles of Database Systems*, pages 221–230, Austin, Texas, 1988. ACM.

[Hal05] John Hall. Business rules boot camp. Tutorial at the European Business Rules Conference, 2005.

[Hay04] Patrick Hayes. RDF semantics. Technical report, W3C, February 2004. W3C Recommendation, `http://www.w3.org/TR/rdf-mt/`.

[HPSB+04] Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, and Mike Dean. SWRL: A semantic web rule language combining OWL and RuleML, May 2004. W3C Member Submission.

[KLW95] Michael Kifer, Georg Lausen, and James Wu. Logical foundations of object-oriented and frame-based languages. *Journal of the ACM*, 42(4):741–843, 1995.

[LPF+06] Nicola Leone, Gerald Pfeifer, Wolfgang Faber, Thomas Eiter, Georg Gottlob, Simona Perri, and Francesco Scarcello. The dlv system for knowledge representation and reasoning. *ACM Transactions on Computational Logic*, 7(3):499–562, July 2006.

[Mel02] Jim Melton. *Advanced SQL 1999: Understanding Object-Relational, and Other Advanced Features.* Elsevier Science Inc., New York, NY, USA, 2002.

[MMe07] Ashok Malhotra, Jim Melton, and Norman Walsh (eds.). XQuery 1.0 and XPath 2.0 Functions and Operators , January 2007. W3C Recommendation, available at `http://www.w3.org/TR/xpath-functions/`.

[MSvL06] Wolfgang May, Franz Schenk, and Elke von Lienen. Extending an OWL Web Node with Reactive Behavior. In *Proceedings of 4th Workshop on Principles and Practice of Semantic Web Reasoning, Budva, Montenegro (10th–11th June 2006)*, volume 4187 of *LNCS*, pages 134–148. REWERSE, 2006.

[Pe06] Eric Prud'hommeaux and Andy Seaborne (eds.). SPARQL Query Language for RDF, April 2006. W3C Candidate Recommendation, available at `http://www.w3.org/TR/rdf-sparql-query/`.

[Pol07] Axel Polleres. From SPARQL to rules (and back). In *Proceedings of the 16th World Wide Web Conference (WWW2007)*, Banff, Canada, May 2007. Accepted for publication, technical report version available at `http://www.polleres.net/publications/GIA-TR-2006-11-28.pdf`.

[PPW03] George Papamarkos, Alexandra Poulovassilis, and Peter T. Wood. Event-condition-action rule languages for the Semantic Web. In *Proc. Int. Workshop on Semantic Web and Databases (co-located with VLDB)*, pages 309–327, 2003.

[PS96] S. Potamianos and M. Stonebraker. The postgres rule system. In J. Widom and S. Ceri, editors, *Active Database Systems - Triggers and Rules for Advanced Database Processing*, pages 44–61. Springer, Berlin,, 1996.

[PSHH04] Peter F. Patel-Schneider, Patrick Hayes, and Ian Horrocks. OWL Web Ontology Language Semantics and Abstract Syntax, February 2004. W3C Recommendation.

[Pŏ5] Paula-Lavinia Pătrânjan. *The Language XChange: A Declarative Approach to Reactivity on the Web*. Dissertation/Ph.D. thesis, Institute of Computer Science, LMU, Munich, 2005. PhD Thesis, Institute for Informatics, University of Munich, 2005.

[Ros06] Riccardo Rosati. Integrating Ontologies and Rules: Semantic and Computational Issues. In Pedro Barahona, François Bry, Enrico Franconi, Ulrike Sattler, and Nicola Henze, editors, *Reasoning Web, Second International Summer School 2006, Lissabon, Portugal, September 25-29, 2006, Tutorial Lectures*, volume 4126 of *LNCS*, pages 128–151. Springer, September 2006.

[Sch04] Sebastian Schaffert. *Xcerpt: A Rule-Based Query and Transformation Language for the Web*. PhD thesis, University of Munich, October 2004.

[Sho87] Yoav Shoham. Nonmonotonic logics: Meaning and utility. In *Proceedings of the 10th International Joint Conference on Artificial Intelligence (IJCAI-87)*, pages 388–393. Morgan Kaufmann, 1987.

[SS07] Simon Schenk and Steffen Staab. Networked rdf graphs. Technical Report Arbeitsberichte des Fachbereichs Informatik 3/2007, Institut fr Informatik, Universitt Koblenz-Landau, 2007.

[SW94] Terrance Swift and David S. Warren. Efficiently implementing slg resolution, January 1994.

[tH05] Herman J. ter Horst. Completeness, decidability and complexity of entailment for rdf schema and a semantic extension involving the owl vocabulary. *Journal of Web Semantics*, 3(2), July 2005.

[The00] The Business Rules Group. Defining business rules – what are they really? Available at `www.businessrulesgroup.org`, 2000.

[TW01] Kuldar Taveter and Gerd Wagner. Agent-oriented enterprise modeling based on business rules. In *ER*, pages 527–540, 2001.

[vdAtHKB03] Wil M. P. van der Aalst, Arthur H. M. ter Hofstede, Bartek Kiepuszewski, and Alistair P. Barros. Workflow patterns. *Distributed and Parallel Databases*, 14(1), 2003.

[WGL06] Gerd Wagner, Adrian Giurca, and Sergey Lukichev. A Usable Interchange Format for Rich Syntax Rules Integrating OCL, RuleML and SWRL. In Pascal Hitzler, Holger Wache, and Thomas Eiter, editors, *RoW2006 Reasoning on the Web Workshop at WWW2006*, may 2006.

[WHvdM06] J. Wielemaker, Z. Huang, and L. van der Meij. Swi-prolog and the web. manuascript, `http://hcs.science.uva.nl/projects/SWI-Prolog/articles/TPLP-plweb.pdf`, 2006.

[Wid96] Jennifer Widom. The starburst active database rule system. *Knowledge and Data Engineering*, 8(4):583–595, 1996.