Rules and Ontologies for the Semantic Web*

Thomas Eiter[1], Giovambattista Ianni[2],
Thomas Krennwallner[1], and Axel Polleres[3]

[1] Institut für Informationssysteme, Technische Universität Wien
Favoritenstraße 9-11, A-1040 Vienna, Austria
{eiter,tkren}@kr.tuwien.ac.at
[2] Department of Mathematics, Universitá della Calabria, Rende, Italy
ianni@mat.unical.it
[3] Digital Enterprise Research Institute, National University of Ireland, Galway
{firstname.lastname}@deri.org

Abstract. Rules and ontologies play a key role in the layered architecture of the Semantic Web, as they are used to ascribe meaning to, and to reason about, data on the Web. While the Ontology Layer of the Semantic Web is quite developed, and the Web Ontology Language (OWL) is a W3C recommendation since a couple of years already, the rules layer is far less developed and an active area of research; a number of initiatives and proposals have been made so far, but no standard as been released yet. Many implementations of rule engines are around which deal with Semantic Web data in one or another way. This article gives a comprehensive, although not exhaustive, overview of such systems, describes their supported languages, and sets them in relation with theoretical approaches for combining rules and ontologies as foreseen in the Semantic Web architecture. In the course of this, we identify desired properties and common features of rule languages and evaluate existing systems against their support. Furthermore, we review technical problems underlying the integration of rules and ontologies, and classify representative proposals for theoretical integration approaches into different categories.

1 Introduction

The issue of having rules on top or aside ontologies written in OWL is an important milestone on the World Wide Web Consortium's (W3C) agenda for completing the Semantic Web architecture. Despite arising theoretical issues, due to the complementary nature of existing ontology and rules languages a plethora of rule based systems have been developed over the last years, driven by the need for rule-based integration of constantly growing Semantic Web data; currently, the W3C is designing of a unifying exchange format – the Rule Interchange Format (RIF) [9] – for the various existing languages. This article aims at giving a snapshot overview of existing languages and systems implementations, of their features and of the theoretical approaches they build upon.

* This research has been partially supported by the European FP6 projects inContext (IST-034718) and REWERSE (IST-2003-506779), by the Austrian Science Fund (FWF) projects P17212, P20840, and P20841, and by the Science Foundation Ireland under Grant No. SFI/02/CE1/I131.

Given the mature state of the RDF and the OWL standards, the building of a rule language is not just a cheap add-on to the standards created so far. During research on Semantic Web technologies the demand for combined formalisms, which integrate ontology and rule languages, emerged as a consequence to supply advanced reasoning capabilities in this setup. Ontology languages are good for describing knowledge adhering to the Open World Assumption, i.e., the encoded knowledge is considered incomplete and conclusions, which cannot be derived from an ontology, are treated agnostically. But under this assumption, one might not get certain rational conclusions, which are reasonable to infer even under incomplete knowledge. To weaken the conservative stance of the Open World Assumption, rule languages, which are proponents of the Closed World Assumption, have been conceived as partners for ontologies. This assumption maintains, hence the name, a closed view of the world; everything which is not derivable from such kind of knowledge base is assumed to be false. This allows for reasoning in problem domains which have to deal with default knowledge, i.e., knowledge that "usually holds" like "birds typically fly.", unless there is evidence of the contrary.

Ontology languages on their own cannot fulfill all the prescribed requirements; rule languages should close at least some of the known obstacles. But such a combination of rules and ontologies, which integrates well with current W3C standards, is not a simple task due to various reasons shown later.

We direct our attention here to rule-based approaches for the Semantic Web, in view of rule systems operating upon RDF data, and ontology languages for the Web, in particular RDF Schema, OWL, and its dialects. This article takes a view on these approaches from the perspective of integrating knowledge gathered from the Semantic Web under several aspects. In particular, we consider modelling features that are needed for practical use cases, and also their mutual relationships. We then discuss several implemented systems and evaluate their support of these features. Finally we give an overview of semantic problems that rise with introducing rules, particularly when they should be combined with expressive ontology languages like OWL. We discuss directions on how these problems might be overcome; furthermore, issues for further research are pointed out.

When we talk about rule-based approaches here, the focus will be on deductive rules languages approaches with a two-valued semantics; probabilistic, fuzzy, dynamic (event-condition-action rules, production rules) approaches, etc., will not be considered. For students interested in these areas, we point to previous editions of the Reasoning Web Summer school and other contributions in the present volume where these topics have been presented in more depth [9, 37, 88, 93].

The remainder of this article is organized as follows. The next section provides some preliminaries, including RDF/RDF Schema and Description Logics as well as OWL. In Section 3, we then turn to rule-based aggregation and integration of Semantic Web data, where – based on practical use cases – we discuss several features that are interesting to compare different available rule systems. After that, we examine in Section 4 several languages and systems with respect to these criteria. The second part of the article addresses then combinations of rules and ontologies using a dedicated approach and semantics. In Section 5, first general issues that come up in combining logic-based rules and ontologies are revisited in more detail; after that, three different generic settings for

the combination are considered that allow to group existing approaches into different categories. Example instances of approaches falling into each of these settings are discussed in Section 6. We conclude the article in Section 7 with a short summary and a brief discussion of issues for research.

2 Preliminaries

The Semantic Web architecture [8] defines at its bottom a simple, and at the same time extremely flexible data model, the *Resource Description Framework* (RDF) [101, 48]. Based on RDF data, which can be used to annotate Web pages and export data from legacy sources, ontologies and rules represent the two main components in the Semantic Web vision – the heart of the Semantic Web –, which shall enable to integrate and make new inferences from existing data. While there are already standard languages for ontologies recommended by W3C, viz. *RDF Schema* (RDFS) [12] and the *Web Ontology Language* (OWL) [27] (which are becoming increasingly used), there is no standard for a rules language available yet. Many rule languages and systems have been proposed, and they offer varying features to reason over Semantic Web data. To mitigate the situation, the *Rule Interchange Format* (RIF) working group of W3C [9, 10] is currently developing a standard exchange format for rules on the Web that takes languages features into account, but is less concerned with a committed semantics. Before we turn our attention to the various rules languages and systems, let us briefly review the basics of RDF, RDFS, and OWL.

In this section, we chose as motivating problem domain a publication scenario, in which we express knowledge about authors and their co-authors, the publications they made, etc. To this end, we start with RDF graphs of the authors of this chapter which encode information like relationships to persons and bibliographic information (see Figure 2–5). We will increase the expressiveness of the represented knowledge by using a description logic ontology given in Example 2. Later on, in Section 3, we will extend the context of our problem domain and look for suitable reviewers of unpublished articles based on given RDF(S) and OWL data using RIF rules.

Along the path of this scenario, we will define the notions used and provide helpful pointers to the interested reader.

2.1 RDF and RDF Schema

The *Resource Description Framework* (RDF) defines the data model for the Semantic Web. Driven by the goal of least possible commitment to a particular data schema, the simplest possible structure for representing information was chosen – labeled, directed graphs. An RDF dataset (that is, a *RDF graph*) can be viewed as a set of the edges of such a graph, commonly represented by *triples* (or *statements*) of the form:

$$Subject\ Predicate\ Object$$

where

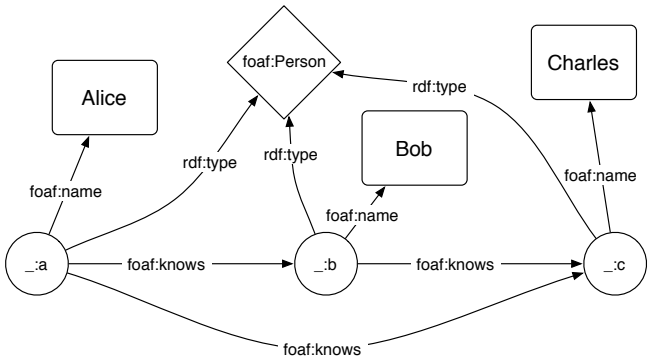

Fig. 1. A simple RDF graph

- the edge links $Subject$, which is a *resource* identified by a URI or a *blank node*, to $Object$, which is either another resource, a blank node, a *datatype literal*, or an *XML literal*;
- $Predicate$, in RDF terminology referred to as *property*, is the edge label.

The next example will clarify the main concepts of RDF.

Example 1. Take a scenario in which three persons named Alice, Bob, and Charles, have certain relationships among each other: Alice knows both Bob and Charles, Bob just knows Charles, and Charles knows nobody. The graphical representation of this simple example showing the relationships between these persons is given in Figure 1. Note that we encode the information using the so called FOAF (friend-of-a-friend) RDF vocabulary [42].

A subgraph of Figure 1 states that "a person called called Bob knows a person called Charles." This can be given by several RDF triples:
`_:b rdf:type foaf:Person`, `_:b foaf:name "Bob"`, `_:b foaf:knows _:c`, `_:c rdf:type foaf:Person`, and `_:c foaf:name "Charles"`.

For instance, the triple

```
_:b foaf:name "Bob"
```

expresses that "someone has the name Bob." `_:b` is a blank node and can be seen as an anonymous identifier. In fact, the name for a blank node is meaningful only in the context of a given RDF graph; conceptually, blank node names can be uniformly substituted inside a RDF graph without changing the meaning of the encoded knowledge. The semantics of blank nodes will be sketched later on.

RDF information can be represented in different formats. One of the most common is the RDF/XML syntax.[4] Our graphical representation above can be given as RDF/XML document:

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:foaf="http://xmlns.com/foaf/0.1/">
```

[4] `http://www.w3.org/TR/rdf-syntax-grammar/`

```
<foaf:Person rdf:nodeID="a">
 <foaf:name>Alice</foaf:name>
 <foaf:knows>
  <foaf:Person rdf:nodeID="b">
   <foaf:name>Bob</foaf:name>
   <foaf:knows>
    <foaf:Person rdf:nodeID="c">
     <foaf:name>Charles</foaf:name>
    </foaf:Person>
   </foaf:knows>
  </foaf:Person>
 </foaf:knows>
 <foaf:knows rdf:nodeID="c"/>
</foaf:Person>
</rdf:RDF>
```

Unfortunately, this XML representation is hard to deal with, and, on top of that, the same RDF graph can look very different in distinct RDF/XML documents due to ambiguous variants of this format. From a didactic point of view, the much simpler Turtle[5] representation of our RDF graph is preferable:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix foaf: <http://xmlns.com/foaf/0.1/>.
_:a rdf:type foaf:Person .
_:a foaf:name "Alice" .
_:a foaf:knows _:b .
_:a foaf:knows _:c .
_:b rdf:type foaf:Person .
_:b foaf:name "Bob" .
_:b foaf:knows _:c .
_:c rdf:type foaf:Person .
_:c foaf:name "Charles" .
```

The above encoding explicitly states triples carrying the same information as the RDF/XML example before. We will make heavy usage of a Turtle shortcut notation throughout this chapter, like

```
_:a rdf:type foaf:Person ;
    foaf:name "Alice" ;
    foaf:knows _:b ;
    foaf:knows _:c .
```

which is a condensed version of the first four triples stated before.

A constantly growing number of RDF graphs – typically in RDF/XML format – is already accessible on the Web. Other common notations, more or less human readable, are N-Triples,[6] Notation 3,[7] and Turtle. We will adopt Turtle in the following, since it is also a fundamental part of SPARQL, which will be described later on.

[5] http://www.w3.org/TeamSubmission/turtle/

[6] http://www.w3.org/2001/sw/RDFCore/ntriples/

[7] http://www.w3.org/DesignIssues/Notation3.html

```
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .
@prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix foaf:  <http://xmlns.com/foaf/0.1/> .
<http://www.mat.unical.it/~ianni/foaf.rdf> a foaf:PersonalProfileDocument.
<http://www.mat.unical.it/~ianni/foaf.rdf> foaf:maker _:me .
<http://www.mat.unical.it/~ianni/foaf.rdf> foaf:primaryTopic _:me .
_:me a foaf:Person .
_:me foaf:name "Giovambattista Ianni" .
_:me foaf:homepage <http://www.gibbi.com> .
_:me foaf:phone <tel:+39-0984-496430> .
_:me foaf:knows [ a foaf:Person ;
         foaf:name "Axel Polleres" ;
         rdfs:seeAlso <http://www.polleres.net/foaf.rdf>].
_:me foaf:knows [ a foaf:Person ;
         foaf:name "Wolfgang Faber" ;
         rdfs:seeAlso <http://www.kr.tuwien.ac.at/staff/faber/foaf.rdf>].
_:me foaf:knows [ a foaf:Person ;
         foaf:name "Francesco Calimeri" ;
         rdfs:seeAlso <http://www.mat.unical.it/kali/foaf.rdf>].
_:me foaf:knows [ a foaf:Person .
         foaf:name "Roman Schindlauer" .
         rdfs:seeAlso <http://www.kr.tuwien.ac.at/staff/roman/foaf.rdf>].
```

Fig. 2. Giovambattista Ianni's personal FOAF file.

Figures 2–5 show some information about the authors of this article extracted from RDF data that are available on the Web. RDF defines a special property `rdf:type`,[8] abbreviated in Turtle syntax by the "`a`" letter. It allows the specification of "IS-A" relations, such as, for instance,

```
<http://www.mat.unical.it/~ianni/foaf.rdf> a foaf:PersonalProfileDocument .
```

in Figure 2 links the resource `<http://www.mat.unical.it/~ianni/foaf.rdf>` to the resource `foaf:PersonalProfileDocument` via `rdf:type`.

Qualified names like `foaf:Person` are shortcuts for full URIs like `http://xmlns.com/foaf/0.1/Person`, making usage of *namespace prefixes* from XML, for ease of legibility. For instance, `:me` is a shortcut for `http://www.polleres.net/foaf.rdf#me` in the graph of Figure 4. If we compare this graph with Figure 2, we see that there is no obligation to give identifiers to entities on the Semantic Web: while the graph in Figure 2 uses a blank node to refer to the entities Giovambattista Ianni and Axel Polleres, the graph in Figure 4 assigns a URI to the entity Axel Polleres.

Types supported for RDF property values are URIs, or the two basic types, viz. `rdf:Literal` and `rdf:XMLLiteral`. Under the latter, a basic set of XML schema datatypes are supported.

SPARQL Query Language for RDF. SPARQL is the W3C standard language for querying RDF data.[9] A query in this language can be roughly divided in two parts: (i) the retrieval part (*body*) and (ii) the result construction part (*head*).

Figure 6 shows a schematic overview of the building blocks that SPARQL queries consist of. We do not go into details of SPARQL here (see [82, 76, 77] for formal details).

The first part of a SPARQL query (the prologue part **P**) consists of namespace prefix declarations, which are used in the `where` part in the body to shortcut IRI literals.

The body part of a SPARQL query (**DWM**) offers the following features. An RDF *dataset* (**D**), i.e., the set of source RDF graphs used as input data, is specified using

[8] short for the full URI `http://www.w3.org/1999/02/22-rdf-syntax-ns#type`

[9] `http://www.w3.org/TR/rdf-sparql-query/`

```
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .
@prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix foaf:  <http://xmlns.com/foaf/0.1/> .
@prefix owl:   <http://www.w3.org/2002/07/owl#> .
@prefix :      <http://www.postsubmeta.net/> .
:foaf a foaf:PersonalProfileDocument .
:foaf foaf:maker <http://www.postsubmeta.net/foaf#TK> .
:foaf foaf:primaryTopic <http://www.postsubmeta.net/foaf#TK> .
:foaf owl:sameAs <http://www.postsubmeta.net/foaf.rdf> .
<http://www.postsubmeta.net/foaf#TK> a foaf:Person ;
      foaf:name "Thomas Krennwallner" ;
      foaf:homepage <http://www.postsubmeta.net/> ;
      rdfs:seeAlso <http://www.postsubmeta.net/foaf> ;
      owl:sameAs <http://www.postsubmeta.net/foaf.rdf#TK> ;
      foaf:knows [ a foaf:Person ; foaf:name "Roman Schindlauer" ;
                   rdfs:seeAlso <http://www.kr.tuwien.ac.at/staff/roman/foaf.rdf> ] ;
      foaf:knows [ a foaf:Person ; foaf:name "Giovambattista Ianni" ;
                   rdfs:seeAlso <http://www.gibbi.com/foaf.rdf> ] ;
      foaf:knows [ a foaf:Person ; foaf:name "Axel Polleres" ;
                   rdfs:seeAlso <http://www.polleres.net/foaf.rdf> ] ;
      foaf:knows [ a foaf:Person ; foaf:name "Francesco Calimeri" ;
                   rdfs:seeAlso <http://www.mat.unical.it/kali/foaf.rdf> ] ;
      foaf:knows [ a foaf:Person ; foaf:name "Wolfgang Faber" ;
                   rdfs:seeAlso <http://www.kr.tuwien.ac.at/staff/faber/foaf.rdf> ] ;
      foaf:knows [ a foaf:Person ; foaf:name "Alessandra Martello" ] .
      foaf:knows [ a foaf:Person ; foaf:name "Thomas Eiter" ] .
```

Fig. 3. Thomas Krennwallner's personal FOAF file.

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix owl: <http://www.w3.org/2002/07/owl#>.
@prefix foaf: <http://xmlns.com/foaf/0.1/>.
@prefix :  <http://www.polleres.net/foaf.rdf#>.
<http://www.polleres.net/foaf.rdf> foaf:maker :me;
                                   foaf:primaryTopic :me.
:me a foaf:Person; foaf:name "Axel Polleres";
 foaf:givenname "Axel"; foaf:surname "Polleres";
 foaf:phone <tel:+35391495723>, <fax:+35391495541>;
 foaf:workplaceHomepage <http://www.deri.ie/> .
 owl:sameAs
  <http://dblp.l3s.de/d2r/resource/authors/Axel_Polleres>.
...
:me foaf:knows
   <http://www.harth.org/~andreas/foaf.rdf#ah>.
<http://www.harth.org/~andreas/foaf.rdf#ah>
 a foaf:Person;   foaf:name "Andreas Harth";
 rdfs:seeAlso <http://www.harth.org/~andreas/foaf.rdf>.
<http://www.polleres.net/foaf.rdf#me> foaf:knows _:b1
_:b1 a foaf:Person; foaf:name "John Breslin";
 rdfs:seeAlso <http://www.johnbreslin.com/foaf/foaf.rdf>.
<http://www.polleres.net/foaf.rdf#me> foaf:knows _:b2.
_:b2 a foaf:Person; foaf:name "Giovambattista Ianni";
 rdfs:seeAlso> <http://www.gibbi.com/foaf.rdf> .
...
```

Fig. 4. Axel Polleres' personal FOAF file.

multiple `from` or `from named` clauses. Merging multiple RDF sources specified in consecutive `from` clauses is a crucial feature of SPARQL, which complements the lack of this possibility in plain RDF format. The `where` part (**W**) allows to match parts of the RDF dataset at hand, by specifying a graph *pattern* possibly involving variables (variable symbols are prefixed with a `?` sign). This pattern is given in a Turtle-based syntax, in the simplest case by a list of consecutive triple patterns, i.e., triples containing variables, IRIs, blank nodes, and RDF literals. More involved patterns allow unions of graph patterns, optional matching of parts of a graph, matching of named graphs selected in `from named` clauses, etc. Finally, variable bindings matching the `where` pattern in the source graphs can be ordered, but also other solution modifiers (**M**) such as `limit` and `offset` are allowed to restrict the number of solutions considered in the result.

In the head portion, SPARQL allows to specify one of four query forms. Each one is associated to a specific result format representing a view over the solutions of the

```
@prefix xsd:  <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .
@prefix foaf:  <http://xmlns.com/foaf/0.1/> .
@prefix dcterms:  <http://purl.org/dc/terms/> .
@prefix dc:    <http://purl.org/dc/elements/1.1/> .
@prefix swrc:  <http://swrc.ontoware.org/ontology#> .
<http://dblp.L3S.de/d2r/resource/authors/Thomas_Eiter>
  a foaf:Agent ;
  foaf:name "Thomas Eiter" .
...
<http://dblp.L3S.de/d2r/resource/publications/conf/foiks/2002>
  a swrc:Proceedings ; a foaf:Document;
  swrc:editor <http://dblp.L3S.de/d2r/resource/authors/Thomas_Eiter>.
...
<http://dblp.L3S.de/d2r/resource/publications/conf/icdt/2005>
  a swrc:Proceedings ; a foaf:Document;
  swrc:editor <http://dblp.L3S.de/d2r/resource/authors/Thomas_Eiter>.
...
<http://dblp.L3S.de/d2r/resource/publications/conf/webi/EiterIST06>
   dc:creator <http://dblp.L3S.de/d2r/resource/authors/Thomas_Eiter>,
      <http://dblp.L3S.de/d2r/resource/authors/Giovambattista_Ianni>,
      <http://dblp.L3S.de/d2r/resource/authors/Hans_Tompits>,
      <http://dblp.L3S.de/d2r/resource/authors/Roman_Schindlauer>;
   a foaf:Document; dcterms:issued "2006"^^xsd:gYear ;
   dcterms:bibliographicCitation
      <http://dblp.uni-trier.de/rec/bibtex/conf/webi/EiterIST06>.
   dcterms:partOf
      <http://dblp.L3S.de/d2r/resource/publications/conf/webi/2006>.
...
<http://dblp.L3S.de/d2r/resource/publications/conf/webi/2006>
 rdfs:label
 "2006 IEEE/WIC/ACM Int.l Conference on Web Intelligence"^^xsd:string;
 swrc:series <http://dblp.L3S.de/d2r/resource/conferences/webi>.
...
<http://dblp.L3S.de/d2r/resource/authors/Giovambattista_Ianni>
      foaf:name "Giovambattista Ianni" .
<http://dblp.L3S.de/d2r/resource/authors/Hans_Tompits>
      foaf:name "Hans Tompits" .
<http://dblp.L3S.de/d2r/resource/authors/Roman_Schindlauer>
      foaf:name "Roman Schindlauer" .
...
```

Fig. 5. An RDF graph about Thomas Eiter extracted from DBLP.

pattern matching mechanism. The three most-used query forms (**CSA**) are `construct`, `select`, and `ask`; the `describe` query form can be used to get RDF graphs describing resources, but no formal semantics is defined for this operator and the output depends on the used SPARQL implementation, hence we omit a discussion here. The `construct` query form takes a *template* as parameter, which consists of a list of triple patterns in Turtle syntax, possibly involving variables that carry over bindings from the `where` part. This operator can be used to translate between different RDF formats. The `select` query form is used to retrieve the bindings for variables mentioned in the graph pattern of the `where` part. The `ask` query form returns true, if there is a binding for the supplied graph pattern, or false otherwise.

An example for a simple SPARQL `select` query is

```
prefix foaf: <http://xmlns.com/foaf/0.1/>
select ?name
from <http://www.mat.unical.it/~ianni/foaf.rdf>
from <http://www.postsubmeta.net/foaf.rdf>
from <http://www.polleres.net/foaf.rdf>
where {
   ?person foaf:knows ?friend .
   ?friend foaf:name ?name .
}
```

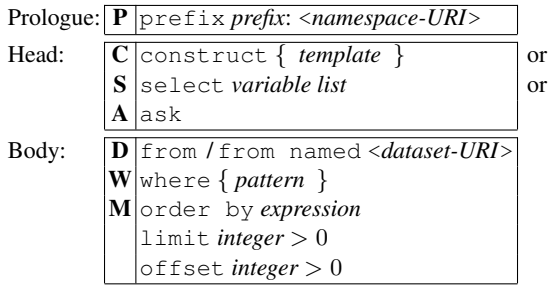

Fig. 6. A schematic overview of SPARQL

which retrieves all `?name`s of `?friend`s known by `?person`s over the combined RDF graphs shown in Figure 2–4.

Usage on the Web. RDF graphs are gaining wide popularity. Driven by efforts such as the Linked Data initiative, [10] RDF datasets are becoming available in several ways, making the current amount of available RDF data substantial. Available datasets can be categorized as:

- *Data exposed directly in RDF format and publicly accessible from the web.*
  This direct way is preferred when advertised data has relatively small size. For instance, a lot of RDF data is available on the Web in personal graphs using the FOAF vocabulary mentioned before. The personal FOAF files of some of the authors are shown in Figures 2–4.[11]
- *Data available as SPARQL endpoint.*
  The SPARQL query language, as shown above, allows to query RDF data using SPARQL endpoints, which are standardized Web Services that answer SPARQL queries by following the SPARQL protocol [18].
  For example, the widely known DBLP online citation index is accessible in two ways: as plain and huge RDF document and via a SPARQL endpoint.[12] By means of the SPARQL endpoint, only the interesting portion of the data is returned to the user, which means that both client and server save time and network bandwidth. DBLP contains a huge amount of information about scientific publications and their authors (see Figure 5 for an example).
- *Data available by means of conversion services, also called wrappers.*
  Converters from popular data formats of heterogeneous provenance such as iCAL (calendar and agenda description format) and RSS (Web feeds) are available, as well as adapters from, e.g., Amazon and eBay Web services.[13] In this context, W3C's Gleaning Resource Descriptions from Dialects of Languages [19] (GRDDL)

[10] `http://linkeddata.org/`

[11] Data accessible from the Web at `http://www.gibbi.com/foaf.rdf`, `http://www.postsubmeta.net/foaf`, and `http://www.polleres.net/foaf.rdf`, resp.

[12] `http://dblp.l3s.de/d2r/`

[13] see `http://esw.w3.org/topic/ConverterToRdf` for a comprehensive list

working group has the goal to complement the concrete RDF/XML syntax with a mechanism to relate to other XML dialects (especially XHTML or "microformats") [19]. GRDDL focuses on extracting RDF from XML. To this end, the working group recently published a finished Recommendation which specifies how XML documents or XML Schema namespace documents can reference transformations that are then processed by a GRDDL-aware application to extract RDF from the respective source file.

An excerpt of the RDF data available about Thomas Eiter from DBLP is shown in Figure 5. This graph contains information such as the papers Thomas Eiter authored, links to co-authors of these papers, etc. The property `dc:creator` belongs to the Dublin Core vocabulary [71] and is used to denote the authorship relation. For instance, the statement

```
<http://dblp.L3S.de/d2r/resource/publications/conf/webi/EiterIST06>
      dc:creator <http://dblp.L3S.de/d2r/resource/authors/Thomas_Eiter> .
```

says that the article with URI

`http://dblp.L3S.de/d2r/resource/publications/conf/webi/EiterIST06`

was created by the person with URI

`http://dblp.L3S.de/d2r/resource/authors/Thomas_Eiter`

Semantics and logical characterization. RDF graphs may contain anonymous, existential nodes, also called "blank" nodes, in order to express incomplete information about the identity of the subject or the object of a statement. An RDF graph can be equally viewed as a first-order formula, where we use a special predicate $triple$ to denote statements made in the graph and where blank nodes are viewed as existentially quantified variable. For instance, the graph from Figure 2 corresponds to the following formula:

$$
\begin{aligned}
&\exists me, b1, b2, b3, b4 \\
&(triple(\texttt{foaf.rdf}, \texttt{rdf:type}, \texttt{foaf:PersonalProfileDocument}) \\
&\wedge triple(\texttt{foaf.rdf}, \texttt{foaf:maker}, me) \\
&\wedge triple(\texttt{foaf.rdf}, \texttt{foaf:primaryTopic}, me) \\
&\wedge triple(me, \texttt{rdf:type}, \texttt{foaf:Person}) \\
&\wedge triple(me, \texttt{foaf:name}, \texttt{"Giovambattista Ianni"}) \\
&\wedge triple(me, \texttt{foaf:homepage}, \texttt{http://www.gibbi.com}) \\
&\wedge triple(me, \texttt{foaf:phone}, \texttt{tel:+39-0984-496430}) \\
&\wedge triple(me, \texttt{foaf:knows}, b1) \wedge triple(b1, \texttt{rdf:type}, \texttt{foaf:Person}) \\
&\wedge triple(b1, \texttt{foaf:name}, \texttt{"Axel Polleres"}) \\
&\wedge triple(b1, \texttt{rdfs:seeAlso}, \texttt{http://www.polleres...}) \\
&\wedge triple(me, \texttt{foaf:knows}, b2) \wedge triple(b1, \texttt{rdf:type}, \texttt{foaf:Person}) \\
&\wedge triple(b2, \texttt{foaf:name}, \texttt{"Wolfgang Faber"}) \\
&\wedge triple(b2, \texttt{rdfs:seeAlso}, \texttt{http://www.kr.tuwien...}) \\
&\wedge triple(me, \texttt{foaf:knows}, b3) \wedge triple(b1, \texttt{rdf:type}, \texttt{foaf:Person}) \\
&\wedge triple(b3, \texttt{foaf:name}, \texttt{"Francesco Calimeri"}) \\
&\wedge triple(b3, \texttt{rdfs:seeAlso}, \texttt{http://www.mat.unical...}) \\
&\wedge triple(me, \texttt{foaf:knows}, b4) \wedge triple(b1, \texttt{rdf:type}, \texttt{foaf:Person}) \\
&\wedge triple(b4, \texttt{foaf:name}, \texttt{"Roman Schindlauer"}) \\
&\wedge triple(b4, \texttt{rdfs:seeAlso}, \texttt{http://www.kr.tuwien...})) \ .
\end{aligned}
\tag{1}
$$

There are different alternative logical representations conceivable for formula (1). For instance, Frame Logic (F-Logic) [56] has often been proposed as an adequate

representation for RDF graphs [105, 26]. F-Logic extends classical first-order logic by concepts from object-oriented programming like objects and class inheritance, and allows to reason about complex objects, which are built from simpler ones. The name Frame Logic stems from the similarity to frame-based languages, which deal with objects and classes and relationships between themselves. F-Logic has a special representation for the class membership relation (`rdf:type`) denoted as "`:`", or "`#`", and frames are expressed in square brackets denoting slots with "$\rightarrow$". As a frame logic formula, the graph from Figure 2 would look as follows:

$$\begin{array}{l}\exists me, b1, b2, b3, b4\ (\texttt{foaf.rdf\#foaf:PersonalProfileDocument}\\ \wedge\ \texttt{foaf.rdf}[\texttt{foaf:maker} \rightarrow me]\\ \wedge\ \texttt{foaf.rdf}[\texttt{foaf:primaryTopic} \rightarrow me]\\ \wedge\ me\#\texttt{foaf:Person} \wedge \cdots)\ .\end{array} \tag{2}$$

Alternatively, the OWL community tends to favor a representation using unary and binary predicates for RDF properties, where unary predicates are used for the `rdf:type` predicate and binary predicates for all other predicates. In that representation, the graph from Figure 2 would look as follows:

$$\begin{array}{l}\exists me, b1, b2, b3, b4\ (\texttt{foaf:PersonalProfileDocument}(\texttt{foaf.rdf})\\ \wedge\ \texttt{foaf:maker}(\texttt{foaf.rdf}, me)\\ \wedge\ \texttt{foaf:primaryTopic}(\texttt{foaf.rdf}, me)\\ \wedge\ \texttt{foaf:Person}(me) \wedge \cdots)\ .\end{array} \tag{3}$$

The semantics of an RDF graph can be essentially viewed as corresponding to the first-order representation chosen in Figure 4 plus entailment of several axiomatic triples. For instance, the triple X `rdf:type rdf:Property` is an axiom for all X which occur in the predicate position of any other triple. In particular, this also makes, for instance, `rdf:type rdf:type rdf:Property` an axiom.

The semantics of RDF involves some more peculiarities in the handling of XML literals, RDF containers, and lists. We refer the interested reader to [37, 48] for more details.

RDF Schema (RDFS) *RDF Schema* (RDFS) is a semantic extension of basic RDF. In a nutshell, by giving special meaning to the properties `rdfs:subClassOf` and `rdfs:subPropertyOf`, to `rdfs:domain` and `rdfs:range`, as well as to several types (like `rdfs:Class`, `rdfs:Resource`, `rdfs:Literal`, `rdfs:Datatype`, etc.), RDFS allows to express simple taxonomies and hierarchies among properties and resources, as well as domain and range restrictions for properties.

The axiomatization of RDFS can to a large extent be approximated by a set of sentences of first-order logic (FOL), as shown in Table 1, plus the axiomatic triples from [48, Sections 3.1 and 4.1].[14] Note that our choice of using a ternary predicate *triple* in favor of a binary representation helped us to avoid higher-order-like rules such as

$$\forall S, P, O\,(P(S, O) \supset \texttt{rdf:type}\,(P, \texttt{rdf:Property}\,))$$

in this axiomatization. Roughly speaking, a triple t is true in a RDF graph G under RDFS semantics if the theory constructed as the union of

[14] We use '$\supset$' for material implication to avoid confusion with '$\leftarrow$' as commonly used in logic programming.

Table 1. Semantics of RDFS

$\forall S, P, O\,(triple(S, P, O) \supset triple(S, \texttt{rdf:type}, \texttt{rdfs:Resource}))$
$\forall S, P, O\,(triple(S, P, O) \supset triple(P, \texttt{rdf:type}, \texttt{rdf:Property}))$
$\forall S, P, O\,(triple(S, P, O) \supset triple(O, \texttt{rdf:type}, \texttt{rdfs:Resource}))$
$\forall S, P, O\,(triple(S, P, O) \wedge triple(P, \texttt{rdfs:domain}, C) \supset triple(S, \texttt{rdf:type}, C))$
$\forall S, P, O, C\,(triple(S, P, O) \wedge triple(P, \texttt{rdfs:range}, C) \supset triple(O, \texttt{rdf:type}, C))$
$\forall C\,(triple(C, \texttt{rdf:type}, \texttt{rdfs:Class}) \supset triple(C, \texttt{rdfs:subClassOf}, \texttt{rdfs:Resource}))$
$\forall C_1, C_2, C_3\,(triple(C_1, \texttt{rdfs:subClassOf}, C_2) \wedge$
$\quad triple(C_2, \texttt{rdfs:subClassOf}, C_3) \supset triple(C_1, \texttt{rdfs:subClassOf}, C_3))$
$\forall S, C_1, C_2\,(triple(S, \texttt{rdf:type}, C_1) \wedge triple(C_1, \texttt{rdfs:subClassOf}, C_2) \supset triple(S, \texttt{rdf:type}, C_2))$
$\forall S, C\,(triple(S, \texttt{rdf:type}, C) \supset triple(C, \texttt{rdf:type}, \texttt{rdfs:Class}))$
$\forall C\,(triple(C, \texttt{rdf:type}, \texttt{rdfs:Class}) \supset triple(C, \texttt{rdfs:subClassOf}, C))$
$\forall P_1, P_2, P_3\,(triple(P_1, \texttt{rdfs:subPropertyOf}, P_2) \wedge$
$\quad triple(P_2, \texttt{rdfs:subPropertyOf}, P_3) \supset triple(P_1, \texttt{rdfs:subPropertyOf}, P_3))$
$\forall S, P_1, P_2, O\,(triple(S, P_1, O) \wedge triple(P_1, \texttt{rdfs:subPropertyOf}, P_2) \supset triple(S, P_2, O))$
$\forall P\,(triple(P, \texttt{rdf:type}, \texttt{rdf:Property}) \supset triple(P, \texttt{rdfs:subPropertyOf}, P))$

- axiomatic triples,
- entailment clauses (as in Table 1), and
- the encoding of G as an existentially quantified conjunction of atoms (as in Figure 1)

entails t. Again, we do not elaborate upon peculiarities and additional rules or axioms in the context of RDF containers, XML literals, etc. here. A thorough formalization of RDF(S) semantics can be found in [66].

2.2 Description Logics and the OWL Web Ontology Language

The next layer in the Semantic Web stack serves to formally define domain models as shared conceptualizations, also often called ontologies [46], on top of the RDF/RDFS data model. In order to formally specify such domain models, the W3C has chosen a language which is close to a syntactic variant of an expressive but still decidable Description Logic (DL) [4], namely $\mathcal{SHOIN}(\mathbf{D})$. More precisely, the OWL DL variant coincides with this DL, at the cost of imposing several restrictions on the usage of RDF(S). These restrictions (e.g., disallowing that a resource is used both as a class and an instance) are lifted in OWL Full which combines the description logic flavor of OWL DL and the syntactic freedom of RDF(S). For an in-depth discussion of the peculiarities of OWL Full, we refer the interested reader to the language specification [27] and restrict our observations to OWL DL here.

While RDFS itself may already be viewed as a simple ontology language, OWL adds several features beyond RDFS' simple capabilities to define hierarchies (`rdfs:subPropertyOf`, `rdfs:subClassOf`) among properties and classes.

As for properties, OWL allows to specify transitive, symmetric, functional, inverse, and inverse functional properties. The correspondences of respective OWL properties and classes with respective description logics and first-order logic axioms can be found

Table 2. Expressing OWL DL Property axioms to DL and FOL

OWL property axioms as RDF triples	DL syntax	FOL short representation
$\langle P$ `rdfs:domain` $C\rangle$	$\top \sqsubseteq \forall P^-.C$	$\forall x, y.P(x, y) \supset C(x)$
$\langle P$ `rdfs:range` $C\rangle$	$\top \sqsubseteq \forall P.C$	$\forall x, y.P(x, y) \supset C(y)$
$\langle P$ `owl:inverseOf` $P_0\rangle$	$P \equiv P_0^-$	$\forall x, y.P(x, y) \equiv P_0(y, x)$
$\langle P$ `rdf:type owl:SymmetricProperty` $\rangle$	$P \equiv P^-$	$\forall x, y.P(x, y) \equiv P(y, x)$
$\langle P$ `rdf:type owl:FunctionalProperty` $\rangle$	$\top \sqsubseteq \leqslant 1P$	$\forall x, y, z.P(x, y) \wedge P(x, z) \supset y = z$
$\langle P$ `rdf:type owl:InverseFunctionalProperty` $\rangle$	$\top \sqsubseteq \leqslant 1P^-$	$\forall x, y, z.P(x, y) \wedge P(z, y) \supset x = z$
$\langle P$ `rdf:type owl:TransitiveProperty` $\rangle$	$P^+ \sqsubseteq P$	$\forall x, y, z.P(x, y) \wedge P(y, z) \supset P(x, z)$

Table 3. Mapping of OWL DL Complex Class Descriptions to DL and FOL

OWL complex class descriptions*	DL syntax	FOL short representation
`owl:Thing`	$\top$	$x = x$
`owl:Nothing`	$\bot$	$\neg x = x$
`owl:intersectionOf` $(C_1 \ldots C_n)$	$C_1 \sqcap \cdots \sqcap C_n$	$C_1(x) \wedge \cdots \wedge C_n(x)$
`owl:unionOf` $(C_1 \ldots C_n)$	$C_1 \sqcup \cdots \sqcup C_n$	$C_1(x) \vee \cdots \vee C_n(x)$
`owl:complementOf` (C)	$\neg C$	$\neg C(x)$
`owl:oneOf` $(o_1 \ldots o_n)$	$\{o_1, \ldots, o_n\}$	$x = o_1 \vee \cdots \vee x = o_n$
`owl:restriction` $(P$ `owl:someValuesFrom` $(C))$	$\exists P.C$	$\exists y.P(x, y) \wedge C(y)$
`owl:restriction` $(P$ `owl:allValuesFrom` $(C))$	$\forall P.C$	$\forall y.P(x, y) \supset C(y)$
`owl:restriction` $(P$ `owl:value` $(o))$	$\exists P.\{o\}$	$P(x, o)$
`owl:restriction` $(P$ `owl:minCardinality` $(n))$	$\geqslant nP$	$\exists y_1 \ldots y_n. \bigwedge_{k=1}^{n} P(x, y_k) \wedge \bigwedge_{i<j} y_j \neq y_j$
`owl:restriction` $(P$ `owl:maxCardinality` $(n))$	$\leqslant nP$	$\forall y_1 \ldots y_{n+1}. \bigwedge_{k=1}^{n+1} P(x, y_k) \supset \bigvee_{i<j} y_i = y_j$

*For reasons of legibility, we use a variant of the OWL abstract syntax [73] in this table.

in Table 2. Note that we switch to the binary representation $P(S, O)$ of triples here, since in description logics (and thus in OWL DL), predicate names and resources are assumed to be disjoint.

Moreover, OWL allows the specifications of complex class descriptions to be used in `rdfs:subClassOf` statements. Complex descriptions may involve class definitions in terms of union or intersection of other classes, as well as restrictions on properties. Table 3 gives an overview of the expressive possibilities of OWL for class descriptions and its semantic correspondences with description logics and first-order logics.[15] Such class descriptions can be related to each other using `rdfs:subClassOf`, `owl:equivalentClass`, and `owl:disjointWith` keywords, which allow us to express description-logic axioms of the form $C_1 \sqsubseteq C_2$, $C_1 \equiv C_2$, and $C_1 \sqcap C_2 \sqsubseteq \bot$, respectively, in OWL.

Finally, OWL allows to express explicit equality or inequality relations between individuals by means of the `owl:sameAs` and `owl:differentFrom` properties, e.g., the triples

```
<http://www.polleres.net/foaf.rdf#me> owl:sameAs
```

[15] We use a simplified notion for the first-order logic translation here—actually, the translation needs to be applied recursively for any complex DL term. For a formal specification of the correspondence between DL expressions and first-order logic, cf. [4].

```
        <http://dblp.l3s.de/d2r/page/authors/Axel_Polleres> .
```

and

```
<http://polleres.net/foaf.rdf#me> owl:differentFrom
        <http://www.gibbi.com/foaf.rdf#me> .
```

boil down to

```
http://www.polleres.net/foaf.rdf#me =
        http://dblp.l3s.de/d2r/page/authors/Axel_Polleres
∧ http://polleres.net/foaf.rdf#me ≠
        http://www.gibbi.com/foaf.rdf#me.
```

For details on the description logics notions used in the Tables 2 and 3, we refer the interested reader to, e.g., [4]. For our purposes, basic understanding of the corresponding definitions in terms of first-order logic will be sufficient. What makes description logics the formalism of choice is the fact that they resemble *decidable fragments* of first-order logic, i.e., queries for entailment of subclass relationships or class membership of a particular individual are effectively computable. At the moment of writing, the next iteration of OWL (version 2) has the status of a member submission at W3C and is further developed by the recently relaunched OWL working group.[16] If accepted in the present form, OWL2 will, based on the decidable description logic $\mathcal{SROIQ}$ [51], support additional features such as acyclic role composition, qualified number restrictions, possibility to declare (for simple roles) symmetry, reflexivity, or disjointness axioms.

Example 2 (Ontologies in Description Logics). We take a simple ontology about publications available online at `http://asptut.gibbi.com/sandbox/reviewers.rdf` as an example to illustrate some of the conceptualizations therein in their corresponding DL syntax:

$$\exists ex{:}title.\top \sqsubseteq ex{:}Paper \tag{i}$$

$$\exists ex{:}title^{-}.\top \sqsubseteq xsd{:}string \tag{ii}$$

$$ex{:}isAuthorOf^{-} \equiv dc{:}creator \tag{iii}$$

$$ex{:}Publication \equiv ex{:}Paper \sqcap \exists ex{:}publishedIn.\top \tag{iv}$$

$$\top \sqsubseteq \leqslant 1\ ex{:}publishedIn^{-} \tag{v}$$

$$ex{:}Senior \equiv foaf{:}Person \sqcap \geqslant 10\ ex{:}isAuthorOf \sqcap \exists ex{:}isAuthorOf.ex{:}Publication \tag{vi}$$

$$ex{:}Club100 \equiv foaf{:}Person \sqcap \geqslant 100\ ex{:}isAuthorOf \tag{vii}$$

This knowledge base expresses the following information: $ex{:}title$ is a datatype property on $ex{:}Papers$ that takes strings as values (axioms (i) and (ii)). Furthermore, the property $ex{:}isAuthorOf$ is the inverse of the property $dc{:}creator$ (axiom (iii)). Next, the ontology defines in (iv) a class $ex{:}Publication$ which consists of all the papers which have been published, and in (v), we state that $ex{:}publishedIn$ to be an inverse functional

[16] `http://www.w3.org/2007/OWL/wiki/OWL_Working_Group`

property (i.e., every paper is published in at most one venue). A $ex{:}Senior$ researcher (vi) is defined as a person who has at least ten papers, some of which are published. Finally, the class $ex{:}Club100$ is defined as the persons having authored more than 100 papers.

3 Rule-based aggregation and integration of Semantic Web data

The main use case we want to address in this article is rule-based aggregation and integration of Semantic data. In other words, we will focus on how it would be possible to reach the goal of combining existing data from the Web, by exploiting rule-based technologies and available Semantic Web rules languages and engines.

To give a condensed introduction into rule-based languages, consider this rule specimen from non-monotonic logic programming: a disjunctive rule is of form

$$a_1 \vee \cdots \vee a_l \leftarrow b_1, \ldots, b_k, not\ b_{k+1}, \ldots, not\ b_m \ , \tag{4}$$

where $l \geq 0$, $m \geq k \geq 0$, and all a_i and b_j are literals, i.e., possibly negated atoms. The disjunction $a_1 \vee \cdots \vee a_l$ is called the head of a rule, while the conjunction $b_1, \ldots, b_k, not\ b_{k+1}, \ldots, not\ b_m$ is the body of a rule. Each expression $not\ b_j$ is a *negation as failure (NAF) literal*, which is true by default, i.e., if we cannot infer that b_i is true. In the usual semantics of such languages, the head of a rule is true if the body is true, i.e., we can infer new knowledge from other knowledge. As an example, $male(X) \vee female(X) \leftarrow person(X)$ and $author(X) \leftarrow isAuthorOf(X, Y)$, $not\ unpublished(Y)$ are valid rules. For a more detailed explanation of the syntax and the semantics of rule-based languages see, e.g., [37, 13].

3.1 Common formats for Rule Interchange on the Web

Since all available rule languages use fairly differing syntaxes, we will illustrate rules using a simplified version of the *Rule Interchange Format Basic Logic Dialect* (RIF-BLD) presentation syntax [10]. RIF-BLD is basically a syntactic variant of Horn rules, which most available rule systems can process. RIF allows frames as in F-Logic notation and the use of URIs as object identifiers,[17] where URIs are enclosed in angle brackets as in Turtle. Likewise, (typed) literals as in Turtle notation are allowed, i.e., for instance we write the RDF triple

```
<http://dblp.L3S.de/d2r/resource/publications/conf/webi/EiterIST06>
   dcterms:issued "2006"^^xsd:gYear.
```

from Figure 5 as a RIF frame

```
<http://dblp.L3S.de/d2r/resource/publications/conf/webi/EiterIST06>[
 dcterms:issued -> "2006"^^xsd:gYear]
```

RIF uses the Prolog style "`:-`" for separating rule head (consequent) and body (antecedent). We start our illustration with a simple example use case for rule based integration.

[17] Strictly speaking, RIF allows IRIs (International Resource Identifier) [32]. IRIs are a generalization of URIs, allowing for example Kanji, Chinese, Arabic or Hebrew characters.

Example 3 (Reviewer Selection). Let us assume that we have FOAF and DBLP information about the authors of the present article available as given above. Based on that information, we want to find more information about which are suitable reviewers for this article, and on persons, which, having conflicts of interests, can not be instead elected as reviewers. In order to do so, we want to use an available Semantic Web rules engine which we wish to feed with information shown next:[18]

– The namespace declarations:

```
Prefix(xsd http://www.w3.org/2001/XMLSchema#)
Prefix(rdfs http://www.w3.org/2000/01/rdf-schema#)
Prefix(owl http://www.w3.org/2002/07/owl#)
Prefix(foaf http://xmlns.com/foaf/0.1/)
Prefix(ex http://www.example.org/)
```

– The set of conflicting reviewers, that is, either persons having the same names as individuals the authors know personally, according to their FOAF files:

```
Forall ?P ?A ?P1 ?N
 ( ?P#ex:ConflictingReviewer :-  And(
      <http://dblp.l3s.de/d2r/page/publications/conf/rweb/EiterIKP08>
         [dc:creator -> ?A]
      ?A[foaf:knows -> ?P1]
      ?P1[foaf:name -> ?N]
      ?P[foaf:name -> ?N]
      ?P#foaf:Person
     )
 )
```
(5)

– or, persons having the same names as people that, according to DBLP, co-authored papers with the authors of the paper in question.

```
Forall ?P ?A ?Pub ?P1 ?N
 ( ?P#ex:ConflictingReviewer :- And(
     <http://dblp.l3s.de/d2r/page/publications/conf/rweb/EiterIKP08>
         [dc:creator -> ?A]
     ?Pub[dc:creator -> ?A]
     ?Pub[dc:creator -> ?P1]
     ?P1[ foaf:name -> ?N]
     ?P[ foaf:name -> ?N]
     ?P#foaf:Person
    )
 )
```
(6)

– People with the same names as people who have published in the same conferences or journals as the authors are, instead, possible reviewers.

```
Forall ?P ?A ?Pub ?ConfOrJournal ?P1 ?N
 ( ?P#ex:CandidateReviewer :- And(
     <http://dblp.l3s.de/d2r/page/publications/conf/rweb/EiterIKP08>
         [dc:creator -> ?A]
     ?Pub[dc:creator -> ?A]
     ?Pub[dcterms:partOf  -> ?ConfOrJournal]
     ?Pub1[dcterms:partOf -> ?ConfOrJournal]
     ?Pub1[dc:creator -> ?P1]
     ?P1[ foaf:name -> ?N]
     ?P[ foaf:name -> ?N]
     ?P#foaf:Person
    )
 )
```
(7)

[18] We will assume the URI of the present work is `http://dblp.l3s.de/d2r/page/publications/conf/rweb/EiterIKP08`

In principle, any rule system which (i) provides access to RDF data, such as Figures 2–5, via import facilities, and (ii) uses the Frame style RDF representation analog to (2) would be capable of processing the rules (5)–(7) and computing conflicting reviewers.

We could easily transform these rules to the alternative RDF representation styles in (1) or in (3) above for other rules systems which support them. We will show these representations later on, when discussing concrete rules systems and their supported syntaxes.

In the following, we will discuss the different features which are (or, should be) present in available rules systems. Small illustrating examples extending the basic reviewer selection scenario from above will be exploited. We will focus on the following aspects:

- RDF data import
- RDF schema support
- OWL support
- Modules, context, and named graphs
- Blank nodes and function symbols
- Built-in predicates and functions
- Defaults and negation as failure
- Advanced features including unstratified negation, constraints, and disjunction

3.2 RDF Data Import

The first and most basic feature for processing Semantic Web data, which we have already mentioned in the previous section, is an import or access facility for RDF data from one or more RDF graphs (or RDF data extracted by a GRDDL [19] transformation from an HTML or XML source). Many available rules systems provide such import facilities, either

- by import directives or mapping definitions, external to the rules language, to access RDF graphs accessible on the Web; or
- by special built-in predicates as part of the rule language to import RDF graphs.

As a special case, we expect that many future rules systems for the Semantic Web will – as opposed to direct import of whole RDF graphs – allow access to RDF stores via a SPARQL [82, 18] endpoint, i.e., providing import directives or built-predicates to dispatch SPARQL queries. More details on how different existing rules systems support this feature are given in Section 4 below.

3.3 RDF Schema Support

The next feature which we would expect from a reasonable rules language/system operating on Semantic Web data is obviously that ontological statements from RDF Schema are taken into account.

Let us have a closer look at the rules above, and assume that we execute them just on the "raw" RDF data available on the Web. Without taking additional RDFS inferences

into account, we would not be able to find out that Thomas Eiter is a conflicting reviewer, since for `http://dblp.L3S.de/d2r/resource/authors/Thomas_Eiter` only the membership in the class `foaf:Agent` is known in the data in Figure 5, but all the rules above have membership in the class `foaf:Person` in its prerequisites for inferring that somebody is a conflicting reviewer.

Fortunately, we have – in our own knowledge base – some knowledge which relates the DC [71], SWRC [94], and FOAF [42] ontologies referred to in Figure 5. As we have seen above, RDFS supports taxonomies on classes and properties as well as domain and range restrictions on properties. Let us assume that our own knowledge base contains the following statements relating SWRC and FOAF:

```
foaf:maker rdfs:subPropertyOf dc:creator .
swrc:editor rdfs:domain foaf:Document .
swrc:editor rdfs:range foaf:Person .
swrc:Person rdfs:subClassOf foaf:Person .
```
(8)

From this and the rules in Table 1, we can conclude that Thomas is indeed a conflicting reviewer. From the DBLP data we can conclude truth of the body condition in rule (6) for binding the variable `?Pub` to `http://dblp.l3s.de/d2r/page/publications/conf/rweb/EiterIKP08`, all the variables `?A`, `?P1`, and `?P` to `http://dblp.L3S.de/d2r/resource/authors/Thomas_Eiter`, and the variable `?N` to `"Thomas Eiter"`. These bindings make all atoms of the condition except the last one – `?P#foaf:Person` – true. However, the inference of the necessary RDF statement

```
<http://dblp.L3S.de/d2r/resource/authors/Thomas_Eiter> a foaf:Person .
```

follows from the RDF statement

```
<http://dblp.L3S.de/d2r/resource/publications/conf/foiks/2002>
    swrc:editor <http://dblp.L3S.de/d2r/resource/authors/Thomas_Eiter>.
```

plus the third statement of (8) and the fifth RDFS inference rule in Table 1. In fact, we can write this and all other RDFS inference rules from Table 1 similarly in RIF syntax:

```
Forall ?S ?P ?O ?C ( ?O#?C :- And( ?S[ ?P -> ?O] ?P[ rdfs:range -> ?C] ))
```
(9)

So, basically for any rule engine that is capable of processing rules (5)–(7), we can equally encode all the RDFS inference rules analogous to (9), and we would be able to compute all conflicting reviewers when taking in addition the RDFS inferences into account.

3.4 OWL Support

Note that we did a little shortcut in the previous example by making in the third statement of (8) explicit that the *swrc:editor* had `foaf:Person` in its range, adding a respective RDFS statement. However, in fact the SWRC ontology is specified in OWL and states this in a different way. Among others, the SWRC ontology contains the following statements, which is not expressible in RDFS alone:

$$swrc{:}Proceedings \sqsubseteq \forall swrc{:}editor.swrc{:}Person;$$

this particular axiom can still be translated into a rule:

```
Forall ?P ?Proc( ?P#swrc:Person :- And( ?Proc#swrc:Proceedings
                                        ?Proc[ swrc:editor -> ?P] ) ).
```
(10)

From above rule, we can still derive that `http://dblp.L3S.de/d2r/resource/authors/Thomas_Eiter` is a `foaf:Person`, namely by

```
<http://dblp.L3S.de/d2r/resource/publications/conf/foiks/2002>
  a swrc:Proceedings ;
  swrc:editor <http://dblp.L3S.de/d2r/resource/authors/Thomas_Eiter>.
```

plus the last statement of (8) and the respective RDFS inference rule. So, by translating to rules the OWL axioms from the SWRC ontology and all other involved ontologies, i.e., the DC ontology and the FOAF ontology, then adding the resulting rulebase to imported data, we could still compute all conflicting reviewers within a rules engine. Faithful preservation of semantics when translating OWL to rules is however a known problem. A premier fragment of OWL which can be translated into rules quite since (since it has a one-to-one correspondence with Horn rules), is described e.g. in [96] or [21, Section 9.3].

Unfortunately, not all OWL axioms can be translated to rules. To illustrate this, let us have a look into the axioms in the Reviewer ontology from above: it is not difficult to translate the rules (i)–(ii) in Example 2 to rules similar to (10), looking at the equivalent FOL representation for OWL statements in Tables 2 and 3. The remaining three rules are equivalences; each equivalence $A \equiv B$ can be "decomposed" into two axioms $A \sqsubseteq B$ and $B \sqsupseteq A$, which are then translated to rules. As for the axiom (iv), this is easy for the $\sqsubseteq$-axiom:

```
Forall ?P ?X ( ?P#ex:Publication :- And( ?P#Paper
                                        ?P[ ex:pulishedIn -> ?X] ) )
```
(11)

However, for the $\sqsupseteq$-axiom we end up with a rule which is not Horn:

```
Forall ?P ( And( ?P#ex:Paper Exists ?X( ?P[ ex:pulishedIn -> ?X] ) )
            :- ?P#Publication )
```
(12)

In fact, rule (12) is not admissible in RIF-BLD syntax. Likewise the axioms (v) and (vi) from Example 2 are not translatable to rules. However, we can easily imagine situations in which we would need inferences both in OWL and also over rules in order to aggregate Semantic Web data for our reviewer selection scenario.

Suppose we have collected a list of experts from the Semantic Web or Knowledge Representation areas all of which have over a hundred publications which are possibly candidates to review the paper at hand, i.e., we know that they are all members of the $ex{:}Club100$[19] defined above which we could state in an RDF graph as follows:

```
<http://dblp.l3s.de/d2r/page/authors/Stefan_Decker> a ex:Club100.
<http://dblp.l3s.de/d2r/page/authors/Dieter_Fensel> a ex:Club100.
<http://dblp.l3s.de/d2r/page/authors/Georg_Gottlob> a ex:Club100.
<http://dblp.l3s.de/d2r/page/authors/Ian_Horrocks> a ex:Club100.
<http://dblp.l3s.de/d2r/page/authors/Michael_Gelfond> a ex:Club100.
<http://dblp.l3s.de/d2r/page/authors/Michael_Kifer> a ex:Club100.
<http://dblp.l3s.de/d2r/page/authors/Vladimir_Lifschitz> a ex:Club100.
```
(13)

[19] Although this may not necessarily reflect current and actual content of DBLP, we assume the set of authors given has more than 100 certified publications each.

We want to find candidate reviewers from this list and the remaining information from DBLP based on the following additional rules. Firstly, we want to add that those documents having a $dcterms{:}bibliographicCitation$ count as publications:

```
Forall ?X ?C ( ?X#ex:Publication :-  And(
                    ?X#foaf:Document
                    ?X[ dcterms:bibliographicCitation -> ?C ] ) )
```
(14)

Secondly, we want to state that any senior researcher, i.e., any member of the class $ex{:}Senior$, is a candidate reviewer:

```
Forall ?P ( ?P#ex:CandidateReviewer :- ?P#ex:Senior )
```
(15)

Note that the above rule can be stated just as well as part of our ontology using the DL axiom $ex{:}CandidateReviewer \sqsubseteq ex{:}Senior$. Regarding (15), obviously, without OWL reasoning support in our Semantic Web Rules engine we cannot come to the conclusion that our designated $ex{:}Club100$ members from (13) are indeed candidate reviewers. An OWL reasoner, like Pellet [92] or Racer [47], that supports the inference of such a rule engine would allow to infer class membership of, e.g., `http://dblp.l3s.de/d2r/page/authors/Vladimir_Lifschitz` in the $ex{:}Senior$ class and thus making him a $ex{:}CandidateReviewer$ by the following rationale:

1. Each of Vladimir's publications in DBLP would by rule (14) trigger class membership of the respective publication in the class $ex{:}Publication$.
2. By Vladimir being member of the $ex{:}Club100$ class and the ontology axioms (vii) from Example 2 we know that Vladimir has more than 100 fillers for the property $ex{:}isAuthorOf$, and thus obviously is also an author of more than 10 papers.
3. Now, class membership for Vladimir in the $ex{:}Senior$ class is established by rule (vi) from Example 2.
4. Finally, rule (15) establishes that Vladimir is indeed a candidate Reviewer.

This inference chain needs both rules and ontological inferences from the OWL ontology.

In the next rule (16), we want to state that for each $ex{:}Publication\ ?X$, which according to our knowledge base is $dcterms{:}partOf$ another entry $?Y$, we can also assert that $?X$ was $ex{:}publishedIn\ ?Y$:

```
Forall ?X ?Y ?P ( ?X[ex:publishedIn -> ?Y] :- And (
                        ?X#ex:Publication ?X[dc:partOf -> ?Y]  ) )
```
(16)

Like (15), the above rule can be expressed using DL axioms as part of an ontology, but is far less legible than the simple rule above.

As we already discussed, not all OWL axioms are expressible in Horn rules; on the other hand, also not all rules, even not all Horn rules, are expressible in OWL. Take, for instance, the next rule, which is a variant of the uncle rule in [52]:

```
Forall ?A ?E ( ?A[ex:editedBy -> ?E] :- Exists ?C ( And (
                    ?A[dc:partOf -> ?C] ?C[swrc:editor -> ?E] ) ) )
```
(17)

This rule simply states that every article $?A$ has an editor $?E$, if $?E$ is the editor of $?A$'s collection $?C$. This property is not expressible in OWL alone. One formalism that is capable of expressing it is SWRL, which adds a form of rules to OWL and will be

described in section 4.1; this already shows the increase in expressivity obtained by combining rules and ontologies. See also the discussion in [67].

We note at this point that in the general case, a combination of such rules and ontologies poses several problems, such as defining the right semantics or ensuring decidability, in particular for rule systems that allow to take OWL reasoning into account. This is discussed in more detail in Section 5.

3.5 Modules, Context, and Named Graphs

As mentioned in Section 3.2, a flexible rules system should enable access to one or more RDF graphs. However, we did not yet discuss facilities to refer to data coming from different RDF graphs within rules or across several rules. For instance, we could simplify rule (14) from above. Instead of stating that the documents having a $dcterms{:}bibliographicCitation$ count as publications, we could simply say that *all documents listed at DBLP* count as publications. Given that an RDF graph containing all documents listed at DBLP is accessible at the URL `http://dblp.l3s.de/d2r/all/Publications`, we could reformulate the rule (14) for extracting $ex{:}Publication$s as follows:

```
Forall ?X ( ?X#ex:Publication :-
            ?X#foaf:Document @ <http://dblp.l3s.de/d2r/all/Publications> )
```
(18)

Here, we used the '@' symbol to denote the *module* [55, 91], or the *context* [78] to which a particular statement belongs. This module mechanism is not (yet) part of the standard RIF syntax; we borrowed this syntax from systems like $\mathcal{F}$lora-2 [55] or Triple [91] for the moment. Often a context is associated with the physical URL where a certain statement can be found, but there are also more general definitions of named RDF Graphs [17], where the graph name or context does not necessarily corresponds to a Web-accessible URL. Note that named graphs are also present in SPARQL via the GRAPH keyword.

3.6 Blank Nodes and Function Symbols

As mentioned in Section 2.1, blank nodes are used in RDF to denote unknown nodes, akin to existential variables in first-order logic. If we want to write rules that create new statements including such blank nodes we run into similar problems as in rule (12), since a rule creating blank nodes boils down to a rule with existential variables in the head. In fact, rule (12) could be viewed as a rule creating a new blank node $\_{:}X_{?P}$ for each binding for $?P$. Although hardly any rule system supports existentials in rule heads, rule systems which support function symbols can typically work around this by creating new identifiers using a Skolem function (see [13, Section 4.1.5] for details about Skolemization). That is, each existential variable X in the head of a Horn rule can be replaced by a term $f_X(Y_1, \ldots, Y_n)$ using a new function symbol f_X whose parameters are all variables Y_i that have an unbound occurrence inside the scope of the existential variable. For example, the Skolemized version of rule (12) is

```
Forall ?P ( And( ?P#ex:Paper ?P[ ex:publishedIn -> f_X(?P)] )
            :- ?P#Publication )
```
(19)

Here f_X is a fresh function symbol, not occurring elsewhere in the rule set to be processed, and P is its single parameter.

The RIF BLD syntax – and most existing rule systems – do not allow conjunctions but only atomic formulas in the rule head, but following the transformations defined by Lloyd and Topor [62] we can equivalently rewrite rule (19) to two Horn rules as follows:

```
Forall ?P ( ?P#ex:Paper :- ?P#Publication )
Forall ?P ( ?P[ ex:pulishedIn -> f_X(?P)] :- ?P#Publication )
```
(20)

Rule systems supporting complex terms with function symbols (such as for instance all Prolog systems), can use this method to emulate rules such as (12). However, function symbols also cause problems with respect to decidability and termination; many existing rule systems therefore simply disallow them. There exists however a substantial effort for including and implementing function symbols in rule languages under a fully declarative framework, such as the Answer Set semantics [11, 15, 90].

3.7 Built-in Predicates and Functions

Many rule systems and languages support a range of built-in functions and predicates for string manipulations, arithmetics and alike, up to flexible APIs for adding procedural attachments to rules which allow to implement and invoke arbitrary external functions from rules. By "built-in" functions and predicates we mean here functions and predicates with a fixed, semantics, that is "built in" in the rules system.

An example for a kind of standard list of functions and predicates is provided by the XQuery/XPath Functions and Operators [65] by W3C, which encompass – besides standard arithmetics – a number of useful manipulations for XML and Web data manipulation.

A built-in predicate could for instance be used to extract a substring from a URI. The following variant of rule (18) checks – instead of the data source of a triple – directly its Document URI to determine whether an article corresponds to one listed at DBLP:

```
Forall ?X ?A ( ?X#ex:Publication :- And(
   ?X#foaf:Document
   ?X[ dc:creator-> ?A ]
   fn:startsWith(?X,"http://dblp.l3s.de/d2r/") ) )
```
(21)

Another example, now for a built-in function, (see [80]) is the mapping from vCard/RDF (`http://www.w3.org/TR/vcard-rdf`) to FOAF. Here we want to combine from vCard the given name and the family name to a `foaf:name` by string concatenation using a built-in function directly in the rule head:

```
Forall ?X ?N ?F ?G ( ?X[ foaf:name -> fn:concat(?F," ",?G) ] :-
   And( ?X [vCard:N -> ?N]
        ?N[ vCard:Given-> ?G ]
        ?N[ vCard:Family-> ?F ] ) )
```
(22)

Some rules languages do not support built-in functions but only predicates. Note that built-in functions can be "emulated" by respective built-in predicates. For instance, if a rule system doesn't offer the XPath function `fn:concat` directly, but a ternary built-in predicate $CONCAT(X, Y, Z)$ having fixed interpretation such that $CONCAT(x, y, z)$

is true whenever z is the concatenation of strings x and y. Using this, we can emulate rule (22) as follows:

```
Forall ?X ?N ?F ?G ?F1 ?F2 ( ?X[ foaf:name -> ?F2 ] :-
   And( ?X [vCard:N -> ?N]
        ?N[ vCard:Given-> ?G ]
        ?N[ vCard:Family-> ?F ]
        CONCAT(?F," ",?F1)
        CONCAT(?F1,?G,?F2) )
```
(23)

Furthermore, many rules systems restrict the use of variables in built-in predicates and functions in the sense that variables occurring in built-ins must be *bound*, i.e., they must occur also in some non-built-in body atom. This is similar to the notion of variable safety [98] in Datalog rules.

We note here that some subtle issues arise with introducing built-ins in Semantic Web rules languages. For instance, it is not entirely clear whether a string-function like `fn:startsWith` in (21) can be applied to an IRI bound to the variable $?A$. That is, it is debatable what it means to convert IRIs – which are actually only a syntactic (and atomic) representation of a constant (an RDF resource in this case), but have no "syntactic" meaning by themselves – to a string. This and other issues which are handled differently in existing rule-based approaches are currently under discussion in the RIF working group.

3.8 Defaults and Negation as Failure

A common extension in many rules languages is negation in rule bodies. For instance, after having established who are conflicting reviewers in rules (5)–(7), one may want to extend rule (15) by stating that candidate reviewers are exactly those senior researchers *not* in conflict:

```
Forall ?P ( ?P#ex:CandidateReviewer :-
                And( ?P#ex:Senior Not( ?P#ex:ConflictingReviewer) ) )
```
(24)

Note that, when integrating data from open sources such as the Web, we have to take care about what "*not* in conflict" means here. Particularly, most rules systems that support rules like (24), would read Not there as nonmonotonic or weak negation, or negation as failure. That means, the rule would fire for any $?P$ for whom we could prove that $?P$ is a senior researcher, but we cannot prove that $?P$ is a conflicting reviewer. These rules are called *nonmonotonic*, since additional information might lead to retraction of a previously made inference, e.g., if we add new RDF statements stating that a senior researcher has published papers with one of the authors.

This is different from classical logic, which always behaves monotonically. If we try to formulate rule (24) as an OWL DL axiom, we could write:

$$ex{:}Senior \sqcap \neg ex{:}ConflictingReviewer \sqsubseteq ex{:}CandidateReviewer$$

or in a rule:

```
Forall ?P ( ?P#ex:CandidateReviewer :-
                And( ?P#ex:Senior Neg( ?P#ex:ConflictingReviewer) ) )
```
(25)

However, such a rule would only fire for individuals $?P$ about which we have *explicit* knowledge that they are not conflicting reviewers, which is also sometimes called *strong*

negation. Such explicit knowledge about negated facts is typically not available on the Web and, as opposed to rule (25), rule (24) rather expresses a default assumption stating that "unless we know that $?P$ is a conflicting reviewer, we assume that $?P$ is a possible candidate."

3.9 Advanced features: Unstratified Negation, Constraints, and Disjunction

Rules involving negation, as those shown in the previous section, are particularly tricky for rules systems if such negation occurs in recursive rules, i.e., if negative rules depend on each other. Imagine we add the following rule, that states that if some candidate reviewer was not chosen, she is an available reviewer.

```
Forall ?P ( ?P#ex:AvailableReviewer :-
            And( ?P#ex:CandidateReviewer Not( ?P#ex:AssignedReviewer ))) (26)
```

Likewise, one could state it the other way around, i.e., if some candidate reviewer was not available, she is an chosen reviewer.

```
Forall ?P ( ?P#ex:AssignedReviewer :-
            And( ?P#ex:CandidateReviewer Not( ?P#ex:AvailableReviewer ))) (27)
```

Many rules systems, in particular Prolog-based systems have difficulties with rules that involve cyclic (or unstratified, see [13, Section 5.3.1]) negation; for any candidate reviewer, it is not clear which of the two rules should fire: without further discrimination, both rules should fire, but upon firing one, the other should be blocked.

Sections 5.3.2–5.3.5 of [13] illustrate several possible semantics for such unstratified rule sets, including the stable model semantics (now more widely known as answer set semantics) [43] and the well-founded semantics [100]:

- Given a candidate reviewer x who is a $ex{:}CandidateReviewer$, the stable model semantics would allow for two possible stable models (*answer sets*). In one of them, the fact $x\#ex{:}AssignedReviewer$ holds, but not $x\#ex{:}AvailableReviewer$, in the other stable model it is the other way round.
- The well-founded semantics, which is a 3-valued semantics, would take an agnostic view here, with only a single model, but assigning *unknown* as a third truth value to both $x\#ex{:}AssignedReviewer$ and $x\#ex{:}AvailableReviewer$.

There are rule systems supporting either of these semantics; we refer to Section 4 below.

We remark here that the multiple-model view of the stable model semantics, as a opposed to a canonical model semantics, can be profitably used for declarative problem solving when multiple solutions exist. The idea is that a problem is represented by a non-monotonic logic program such that its stable models correspond to the solutions of the problem, which then can be computed using a logic programming engine; this paradigm is often referred to as *Answer Set Programming* (ASP). For example, consider in our scenario the problem "give me all possible sets of reviewer assignments." A rule set including both rules (26) and (27) would have exactly these sets as answers (i.e., stable models) under the stable model semantics.

Constraints Constraints are special rules that have an empty head, and lead to the inference of a contradiction if their body is true. In the multi-model approach of the stable model semantics, constraints are customary to filter out unwanted models which correspond to "wrong" assignments with respect to candidate solutions.

For instance, if we add the constraint

```
Forall ?P1 ?P2 ( :- And( ?P1#ex:AssignedReviewer ?P2#ex:AssignedReviewer
                         ?P1 != ?P2 ) )                                    (28)
```

to the rules (26) and (27), all assignments where two or more reviewers are assigned would be excluded from possible answers. The following rule and constraint guarantee that at least one reviewer is assigned:

```
Forall ?P ( someoneAssigned :- ?P#ex:AssignedReviewer )
:- Not( someoneAssigned )                                                  (29)
```

In combination, the rules (26)–(29) guarantee that exactly one candidate reviewer is assigned. Constraints are supported by several rules systems for the stable model semantics.

Disjunctive Rules One useful extension are disjunctive rules, i.e., rules which do not only permit atomic formulas in the head but also a disjunction of atoms. Disjunction enables us to write (26) and (27) more concisely in just one rule, which reads very natural:

```
Forall ?P ( Or( ?P#ex:AssignedReviewer
                ?P#ex:AvailableReviewer ) :- ?P#ex:CandidateReviewer  )    (30)
```

This disjunction has the following semantics: for each $?P\#ex{:}CandidateReviewer$, either $?P\#ex{:}AssignedReviewer$ is made true or $?P\#ex{:}AvailableReviewer$; this is different from classical logic, according to which we would just know that at least one of the two is true.

Although we used a "RIF style" syntax here, both negation as failure as well as disjunction in rule heads is beyond the current version of RIF BLD. For more details and examples on Answer Set Programming as well as particular rules systems, see [13, 6, 36].

4 Languages and Systems

In this section, we present languages and tools for reasoning with RDF data. This kind of data will be classified in two categories: RDF(S) and OWL, which have been defined in Section 2.

One important use case for combining rules and ontologies is ontology alignment, or in general data integration from different data sources. In OWL ontologies, you can import additional ontologies using $owl{:}import$ statements, but this feature can be seen as splitting ontologies into partitions rather than integrating ontologies from different sources. RDF(S) has no built-in support for integrating other RDF data; this task is outsourced to SPARQL [82]: to merge different RDF data sources, one can specify every RDF graph in a `from` clause of a SPARQL query. Typically, rule systems can

easily reference data from multiple sources and provide even more expressive reasoning support than SPARQL alone.

Several languages and systems exist which support accessing and querying RDF data and allow to combine data sources under several aspects. In the following, we will look into this in more detail and outline the features of prominent languages and systems. We classify the languages into four categories (SWRL, RDF Stores, Logic Programming, and Hybrid Combinations), which should make their flavour more visible. Please note that we can only describe a fragment of available tools, hence the next sections display an inherent incomplete list of rule systems with ontology support.

4.1 SWRL – OWL Reasoners with Rules Support

The *Semantic Web Rules Language* (SWRL) is an ontology language that integrates OWL with a rule layer [52] built on top of it. SWRL's goal of enhancing description logics with rules is aimed at overcoming some known expressive limitation in ontology languages, which can be easily fixed by adding rules to the ontology. The addition of rules is also the main cause why reasoning in SWRL is in general undecidable, but decidable fragments are known, like DL-safe rules [70]. This language is a non-hybrid coupling approach of rules and ontologies; see also Section 5 for fundamental issues of amalgamating rules and ontology reasoning.

SWRL supports a rich set of built-ins inspired by XQuery and XPath2 functions [65]. Since SWRL is an extension of the OWL ontology language, it is restricted to unary and binary DL-predicates. Furthermore, it does not support nonmonotonic inference. Also, combining OWL data from outside ontologies is only possible through $owl{:}import$ constructs.

A SWRL ontology is composed of ordinary OWL axioms and SWRL rules. The rules constitute of antecedents and consequents, which both consist of lists of atoms. Atoms may be OWL class expressions, property definitions, or built-ins.

Usually, SWRL rules are part of an OWL ontology encoded in XML or in abstract syntax. The next example might serve as an illustration for the SWRL abstract syntax, which is just a different way for representing rule (17):

```
Implies( Antecedent( dc:partOf(I-variable(A) I-variable(C))
                     swrc:editor(I-variable(C) I-variable(E)) )
         Consequent( ex:editedBy(I-variable(A) I-variable(E))) )
```

DL reasoners now increasingly support SWRL. For instance, state of the art engines like KAON2[20] and Pellet [92] facilitate the DL-safe fragment of SWRL, while RacerPro [47] supports a SWRL-like syntax with a slightly different semantics (for instance, closed world reasoning is supported in RacerPro's variant of SWRL).

4.2 RDF Stores with Rules Support

RDF stores (or triple stores) are frameworks for managing, accessing, and processing RDF data. These kind of systems do not employ standardized languages. Instead, they

[20] `http://kaon2.semanticweb.org/`

provide their own proprietary rules implementations. These implementations are not as expressive as SWRL, in favor of manageable computational properties.

In the following, we will briefly show three of the most common proponents of this category and address different aspects on how rules are managed. One of such aspects, mentioned quite often, is the handling of forward- and backward-chaining rules. For instance, the well-known RETE algorithm is the backbone of many forward-chaining systems. Depth-first backtracking traversal of rule sets based on SLD-Resolution [61] is the most widely known representative of backward-chaining algorithms for rule processing, deployed in most Prolog systems.

For a more in-depth explanation of the differences between forward- and backward-chaining, we refer the interested reader to [13].

In the following, we will show how to write some of the rules modelling conflict of interest for reviewers from Example 3 in systems with rule support.

Jena The Jena[21] Semantic Web framework comes with both forward- and backward-chaining rule support, where the former implementation uses the RETE algorithm, and the latter a standard logic programming style engine. Both engines can be tied together and run in a hybrid mode where rules to be processed in a forward-chaining fashion are syntactically distinguished from those to be processed by backward-chaining.

As an example for a rule expressed in Jena's forward-chaining syntax, we show next the translation of rule (5). Recall that this rule expresses that a conflicting reviewer is a person which knows an author of this paper:

```
[ conflict1:
  (http://dblp.l3s.de/d2r/page/publications/conf/rweb/EiterIKP08 dc:creator ?A),
  (?A foaf:knows ?P1), (?P1 foaf:name ?N), (?P foaf:name ?N),
  (?P rdf:type foaf:Person)
  ->
  (?P rdf:type ex:ConflictingReviewer) ]
```

In this example, *conflict1* is simply a name for the rule, and the atoms in the antecedent of a rule might be either triple patterns of the form $(Subject\ Predicate\ Object)$ or built-ins of the form $builtin(Subject\ Predicate\ Object)$. The terms in subject, predicate, or object position could be RDF terms in the style of [82] or variables prefixed with a "?" symbol.

Similarly, the above rule could be executed using the backward chaining inference engine. In this reasoning mode, the same rule is just written with the consequent first:

```
[ conflict1back: (?P rdf:type ex:ConflictingReviewer)  <-
  (http://dblp.l3s.de/d2r/page/publications/conf/rweb/EiterIKP08 dc:creator ?A),
  (?A foaf:knows ?P1), (?P1 foaf:name ?N), (?P foaf:name ?N),
  (?P rdf:type foaf:Person) ]
```

Jena comes with support for custom rules, i.e., rules which are used to define a semantics using the predefined RDF(S) or OWL semantics.

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

Sesame/OWLIM The Sesame[22] project maintains a reasoning and storage framework for querying and persistently storing RDF data. By means of the OWLIM[23] forward-chaining engine, Sesame can be turned into a reasoning platform which supports the ontology languages RDFS, as well as the non-standard OWL fragments OWL DLP [45], and OWL-Horst [96]. OWL DLP is a fragment of OWL DL expressible entirely in function-free Horn rules. The OWL fragment defined by Herman ter Horst (thus sometimes referred to as OWL-Horst) adds more, yet incomplete, support for the fragment of OWL Full translatable to rules.[24]

To give a glimpse on how OWLIM rules look like, we render rule (6) from Section 3, which expresses that a conflicting reviewer is a person who co-authored a paper with an author of the article in question:

```
Id: conflict2
<http://dblp.l3s.de/d2r/page/publications/conf/rweb/EiterIKP08> <dc:creator> A
Pub <dc:creator> A
Pub <dc:creator> P1
P1 <foaf:name> N
P <foaf:name> N
P <rdf:type> <foaf:Person>
-------------------------
?P#ex:ConflictingReviewer
```

Moreover, OWLIM supports constraints on the variable bindings in each triple, i.e., the user can filter certain matches. Another feature are custom rule-sets ($Axioms$), which allows users of this system to define their own semantics and control the complexity of reasoning.

Oracle 11*g* The Oracle $11g$ RDF database[25] provides full RDF(S) support and comes with a reasoning engine for a subset of OWL DL, more specifically, OWLPrime [104]. It includes support for forward-chaining rules and extends its SQL dialect with new constructs for querying RDF inside of Oracle's relational DBMS, i.e., it features a rule system built entirely on top of the existing Oracle DBMS infrastructure. Like Jena and Sesame/OWLIM, Oracle $11g$ facilitates adding inference rules on top of the built-in rules for implementing user-defined semantics based on RDF.

For instance, our rule (17) can be defined as new element in the rulebase store of the RDF database:

```
INSERT INTO mdsys.semr_user_rulebase VALUES ('editedby_rule',
'(?x <http://purl.org/dc/elements/1.1/partOf> ?y)
  (?y <http://swrc.ontoware.org/ontology#editor> ?z)',
NULL, '(?x <http://www.example.org/editedBy> ?z)', null);
```

Getting the extension of the $ex{:}editedBy$ predicate can be done using the following extended SQL query:

```
SELECT s,o FROM table(SEM_MATCH('(?s <http://www.example.org/editedBy> ?o)',
           SEM_MODELS('OWLTST'),
           SEM_RULEBASES('OWLPRIME','USER_RULEBASE'),  null, null ));
```

[22] http://www.openrdf.org/

[23] http://www.ontotext.com/owlim/

[24] Different other rule-expressible fragments of OWL exist in the literature, e.g., (i) the *intentional OWL* fragment defined in Jos de Bruijn's thesis [21, Section 9.3] which does a rigid analysis of ter Horst's work and tries to fix some of the problems therein, (ii) the OWL$^-$ fragment [25] which is a slight extension of OWL DLP, or (iii) OWLPrime [104] discussed below.

[25] http://www.oracle.com/technology/tech/semantic_technologies/

which retrieves all $ex{:}editedBy$-related pairs using OWLPrime plus our user-rulebase as entailment regime.

4.3 Logic Programming Engines with RDF Support

Logic programming has a long tradition in rule-based knowledge representation. Here programs are composed of sets of rules in the form of (4). Inferencing with rules is in logic programming is mostly performed using reasoning engines such as backward-chaining Prolog systems. Other systems implementing logic programming paradigms such as Answer Set Programming[6, 36] often rely on a forward-chaining Datalog engine underneath.

Among systems following the logic programming spirit, we next present such representatives which at least have support for importing RDF data (from possibly different locations), and thus allow to partially address our use cases outlined above.

Prolog Systems with RDF libraries SWI-Prolog[26] is a Prolog engine with many features. It can import RDF using the Semantic Web Library [103] for SWI-Prolog and reason about this data using Prolog-style backward chaining. RDFS and query support works by using standard Prolog rules. For example, the fifth axiom in our RDFS axiomatization (see Table 1) can be specified as

```
triple(O, rdf:type, C) :- rdf(S, P, O), rdf(P, rdfs:range, C).
```

As any common Prolog system, SWI Prolog only supports stratified negation as failure, denoted in Prolog by '\+'.

Rule (24) could (assuming all the relevant data is in the graph `data.rdf`) be expressed in SWI-Prolog as follows.

```
triple(P, rdf:type, ex:CandidateReviewer) :-
                          rdf(P, rdf:type, ex:Senior),
                          \+ (rdf( P, rdf:type, ex:ConflictingReviewer)).
?- rdf_assert(S,P,O), triple(S,P,O).
```

$\mathcal{F}$LORA-2 The $\mathcal{F}$LORA-2 system is an F-Logic reasoner with built-in support for RDF(S).[27] Negation as failure is supported under well-founded semantics. $\mathcal{F}$LORA-2 is implemented on top of the XSB Prolog engine.[28] Reconsidering the fifth axiom in our RDFS axiomatization (Table 1), it can be specified as

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

which is very close to RIF's presentation syntax. Note that also the KAON2 system mentioned above has limited support for F-Logic.

[26] `http://www.swi-prolog.org/`
[27] `http://flora.sourceforge.net/`
[28] `http://xsb.sourceforge.net/`

cwm Finally, an example for a rule-based RDF engine in spirit of logic programming is cwm.[29] It is built for *Notation3* (N3),[30] which is an RDF notation enhanced with support for modelling formulae and rules. An example is our rule (7), which can be expressed in N3 as

```
@forAll P, A, P1, N .
{ <http://dblp.l3s.de/d2r/page/publications/conf/rweb/EiterIKP08> dc:creator A .
  Pub dc:creator A .
  Pub dcterms:partOf ConfOrJournal .
  Pub1 dcterms:partOf ConfOrJournal .
  Pub1 dc:creator P1 .
  P1 foaf:name N .
  P foaf:name N .
  P rdf:type foaf:Person .
 } log:implies { P rdf:type ex:ConflictingReviewer } .
```

N3 is based on a proprietary forward-chaining engine implemented in Python. It also supports also a rich set of built-ins. Interestingly, N3/cwm also support for stratified negation as failure with the `log:notIncludes` directive. Rule (24) could (assuming all the relevant data is in the graph `data.rdf`) be expressed in N3 as follows.

```
@forAll :P.
{ <data.rdf>.log:semantics.log:conclusion
           log:notIncludes { :P a ex:ConflictingReviewer };
           log:includes    { :P a ex:Senior.  } }
     log:implies {:P a ex:CandidateReviewer}.
```

4.4 Systems for Hybrid Combinations

In anticipation of Section 6, we show here systems which apply some of the more complex approaches to combine rules and ontologies introduced there. These systems are typically very expressive, and combine full DL reasoning with some form of logic programming.

Hybrid Rules HD-rules,[31] which realizes *Hybrid rules under well founded semantics* as defined in [31, 30]. The system is implemented using XSB[32] and a DL reasoner of choice capable of handling the DIG format.

dl-Programs The software prototype NLP-DL[33] implements dl-programs as described in [40, 35, 41], under stable model and well-founded semantics, by integrating the ASP reasoner DLV [58] and RACER [47]. An example is given in the next section, and further details will be shown in Section 6.1.

---

[29] `http://www.w3.org/2000/10/swap/doc/cwm.html`
[30] `http://www.w3.org/DesignIssues/Notation3`
[31] `http://www.ida.liu.se/hswrl/`
[32] `http://xsb.sourceforge.net/`
[33] `http://www.kr.tuwien.ac.at/research/systems/semweblp/`

HEX-programs HEX-programs, proposed in [38, 39], are an extension of nonmonotonic logic programs under the answer set semantics [43] with support for higher-order and external atoms. External atoms are a very generic form of built-ins. They generalize the semantics of dl-programs by providing a special notion of *external atom* which enables access to DL reasoners and, above that, ensures the possibility of integrating generic external software modules.

dlvhex[34] is an implementation of a large fragment of HEX-programs. It has been used for a variety of applications such as ontology merging, bio-ontologies, e-government, web querying, and policy management.

HEX-programs combine many approaches into a single extensible language for RDF and DL reasoning, among others. Remarkably, external atoms allow a bidirectional data flow between external sources and HEX-programs, i.e., inferences can be fed as input to the outside data source.

An example for RDF support is the rdf external atom of dlvhex, which is of the form $\&rdf[U](S, P, O)$. Through such an atom, RDF triples (S, P, O) from URL U can be accessed:

```
triple(S,P,O) :- &rdf[<http://...>](S,P,O).
triple(S,"rdf:type","ex:ConflictingReviewer") :-
    triple("http://dblp.l3s.de/d2r/page/publications/conf/rweb/EiterIKP08",
           "dc:creator:, A),
    triple(Pub, "dc:creator", A), triple(Pub, "dc:creator", P1),
    triple(P1, "foaf:name", N), triple(P, "foaf:name", N),
    triple(P, "rdf:type", "foaf:Person").
```

By the notion of DL external atoms, HEX-programs are able to query external description logics reasoners; the dlvhex system is able to accommodate dl-atoms in the style of Section 6.1, which give a more concise syntax:

```
publishedIn(X,Y) :- DL[ex:Publication](X), DL[dc:partOf](X,Y).
```

Table 4 summarizes features of the previously introduced rules languages. As we focus here on Semantic Web rule languages, it is no wonder that almost all support RDF(S), and those systems, which do not have support for this language, have OWL support instead. The OWL column shows then which systems promote the description logics part of OWL, i.e., OWL Lite and OWL DL; we did not consider OWL Full systems. The next column reveals tools with module support, which can be found in $\mathcal{F}$LORA-2 and SWI-Prolog. Similarly, function symbols are only present in those two systems and HD-rules, since they are based on Prolog engines. In contrast, built-in predicates or functions are not available in OWLIM and Pellet. Typically, all built-in-aware systems provide an API, which allows the system users to specify their own built-ins, but only dlvhex provides a declarative semantics for this feature. Higher-order predicates (HO), that is the possibility of making a variable quantify over predicate names, are only supported in two systems, whereas dlvhex and NLP-DL are the only engines with constraint rules. As shown in Table 4, support for (unstratified) negation as failure (NAF) is typical for descendants of logic programming systems and hybrid combination approaches. Our last category, disjunctive rules, are only present in dlvhex and KAON2 due to their heritage of disjunctive Datalog.

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

Table 4. Overview of rule systems features

System (Language)	RDF(S)	OWL	Modules	Functions	Built-ins	HO	Constraints	NAF	$\vee$
cwm (N3)	+	−	−	−	+	−	−	+	−
dlvhex (HEX)	+	+	−	−	+	+	+	+	+
$\mathcal{F}$LORA-2 (F-Logic)	+	−	+	+	+	+	−	+	−
HD-rules (Hybrid rules)	−	+	−	+	−	−	−	+	−
Jena (Jena Rules)	+	+	−	−	+	−	−	−	−
KAON2 (SWRL)	+	+	−	−	+	−	−	−	+
NLP-DL (dl-programs)	−	+	−	−	−	−	+	+	−
Oracle $11g$ (OWLPrime)	+	+$\sim$	−	−	+	−	−	−	−
OWLIM (OWL Horst)	+	+$\sim$	−	−	−	−	−	−	−
Pellet (SWRL)	+	+	−	−	−	−	−	−	−
RacerPro (SWRL)	+	+	−	−	+	−	−	−	−
SWI-Prolog (RDF(S))	+	−	+	+	+	−	−	+	−

Legenda: HO = Higher Order predicates, + = yes, − = no, +$\sim$ = yes, with some proviso

5 Combining Rules with Ontologies

Whereas we focused on practical features and implemented systems so far, in this section we examine the general issues that come up when combining logic-programming based (nonmonotonic) rules and (monotonic) ontology languages from a more theoretical perspective. After discussing the semantic discrepancies which are the source of difficulties when integrating logic programs with FOL – namely the Description Logics fragment corresponding to OWL DL – we classify the integration approaches in three categories. Eventually, we will present some representative approaches for each category in more detail.

5.1 The issue of combining Rules with Description Logics

The combination and extension of terminological concepts defined in a DL theory by means of rules is nowadays acknowledged as an important tool enriching knowledge representation capabilities of traditional ontology languages such as OWL. As a prototypical example, one cannot define the role *uncleOf*, given the roles *brotherOf* and *fatherOf* in OWL DL (see e.g. [52]). OWL DL does not feature a role composition construct or, more generally, a mechanism for defining axiomatic rules. Such aspects are covered by extensions of OWL DL: for instance OWL2, based on $\mathcal{SROIQ}$, adds the possibility of constructing roles by composition, while SWRL adds the possibility to declare arbitrary Horn clauses, which however leads to undecidability of crucial reasoning tasks such as subsumption in OWL. More troubles arise when rules governed by nonmonotonic semantics should be introduced in a monotonic context, like a description logic knowledge base [22].

As well-known, the core of logic programming, i.e., definite positive programs (positive Datalog programs), has a direct correspondence with the Horn subset of classical FOL. To wit, a rule of the form

$$a_1 \vee \cdots \vee a_l \leftarrow b_1, \ldots, b_k, not\, b_{k+1}, \ldots, not\, b_m, \tag{31}$$

which is definite (i.e., when $l = 1$) and not-free (i.e., when $m = k$) can be read as a first-order sentence

$$(\forall)\, b_1 \wedge \ldots \wedge b_k \supset a_1 \tag{32}$$

where $(\forall)$ denotes the universal quantification of all variables. This subset of FOL allows for a sound and complete decision procedure for entailment of ground atomic formulae.

Several attempts to embrace such definite rules within a homogeneous (which can be classified as *non-hybrid coupling*) semantic framework based on classical first-order semantics have been made; most noticeable is SWRL, which is submitted to the W3C (see Section 4.1 and [52]). SWRL embeds rules and terminological knowledge bases under the same first-order semantics, but is restricted to (monotonic) Horn rules. This approach has a smooth and homogeneous semantics, but still suffers from undecidability problems; this can be addressed by introducing appropriate syntactic restrictions to the rules, such as *DL-safety* [70]. DL-safe Horn rules can be combined with Description Logics still retaining decidability.

Among non-hybrid approaches, also DLP [45] is noticeable. DLP, in contrast to SWRL, restricts the syntax of the supported OWL DL fragment to those axioms expressible in Horn rules, while allowing arbitrary Horn rules to be added while still staying within the Horn fragment.

As opposed to these non-hybrid approaches we will now mainly concentrate on the possibility of combining nonmonotonic rule sets under traditional logic programming semantics with a (monotonic) Description Logics knowledge base, which we refer to as the so-called *hybrid approaches*.

While equivalence theorems between Horn Clausal Logic and function-free positive Datalog under minimal model semantics are well known traditional results, the latter diverts crucially as soon as nonmonotonic constructs are introduced. Hybrid approaches have thus to take the great semantic and philosophical differences among the two worlds into account.

We will in the following denote a *hybrid knowledge base* $\mathcal{KB} = \langle \mathcal{T}, P \rangle$ as the combination of:

- a first-order theory $\mathcal{T}$ (the *classical component*), expressed in a FO language with signature $\Sigma_{\mathcal{T}}$; and
- a logic program P (the *rules component*), formulated with a signature Σ_P.

The combined signature of $\mathcal{KB}$ is $\Sigma_{\mathcal{KB}} = \Sigma_{\mathcal{T}} \cup \Sigma_P$. Predicates in $\Sigma_{\mathcal{T}}$ (resp. Σ_P) are termed *classical (resp. rule) predicates*.

5.2 Logic programming versus First-Order Logic

We summarize next some of the crucial differences among logic programming (into which ASP [43] and Frame Logic under nonmonotonic semantics [56] fall), and Classical Logic (into which OWL DL and, in general, Description Logics, fall).

Closed vs. Open World Assumption and single vs. multiple models. A logic program is seen as a description of a single world, over which knowledge is complete. Incomplete knowledge about a proposition is simply resolved by turning it into falsity. Indeed, logic programming embraces Reiter's *Closed World Assumption* (CWA) [83]: If a theory $\mathcal{T}$ does not logically entail a ground atom A, then conclude $\neg A$.

On the other hand, a set of FOL sentences (or DL axioms) is intended as a description of possible worlds (interpretation), in which all the sentences must hold. Conclusions about propositions which cannot be proven to be true in all the possible worlds are kept open. Under *Open World Assumption* (OWA) incomplete information is treated agnostically (i.e., under a theory $\mathcal{T}$ it might be that neither $\mathcal{T} \models A$ nor $\mathcal{T} \not\models A$ holds for a proposition A).

The OWA is often reasonable in the Semantic Web context. However, taking the agnostic stance of OWA may be not helpful for drawing rational conclusions under incomplete information. Indeed, one can see the Web as a set of knowledge sources. A locally scoped closed world assumption might be preferred when, for instance, one has complete knowledge over a given source. In such cases a mix of CWA and OWA may be appropriate, cf. [20, 79].

It is worth noting that the issue of OWA vs. CWA is strictly related, but not equivalent, to the multiple models approach taken in FOL versus the single model approach taken in logic programming. Indeed, Answer Set Programming is a representative of a logic programming paradigm where the closed world assumption is combined with the possibility to control the modelling of multiple worlds. Also, there are fragments of first-order which can be seen as the description of a single, canonical model (e.g., Horn logic or DL-Lite [16]).

Negation as failure vs. classical negation. *Negation as failure (NAF)* is the traditional operator for inferring negative knowledge from incomplete information, and is peculiar of logic programming. The behavior of NAF compared the classical negation is noticeably different. For instance, consider the logic program

$$\begin{array}{rl} P: & person(X) \leftarrow author(X). \\ & nonAuthor(X) \leftarrow not\ author(X). \\ & person(joe\_doe). \end{array}$$

From P, we can conclude the fact $nonAuthor(joe\_doe)$. Now consider the first-order counterpart of P:

$$\begin{array}{rl} \mathcal{T}: & \forall X.\,(Author(X) \supset Person(X)) \wedge \\ & \forall X.\,(\neg Author(X) \supset NonAuthor(X)) \wedge \\ & Person(joe\_doe). \end{array}$$

From $\mathcal{T}$, we cannot conclude $NonAuthor(joe\_doe)$.

Strong negation vs. classical negation. Several logic programming formalisms feature the possibility to avoid negation as failure and use the so-called *strong negation*. For instance, the seminal paper about Answer Set Programming [43] introduces a language

comprising both negation as failure and strong negation. Strong negation is often seen as a "surrogate" of classic negation, but it must not be mispelled as equivalent to the latter, due to some crucial semantic differences.

For instance, given the logic program

$$P : person(X) \leftarrow author(X).$$
$$-person(joe\_doe).$$

where "$-$" is used for denoting strong negation, we cannot $-auther(joe\_doe)$ from P; on the other hand, from the corresponding first-order theory:

$$\mathcal{T} : \forall X.\, (Author(X) \supset Person(X)) \wedge$$
$$\neg Person(joe\_doe).$$

we can conclude $\neg Author(joe\_doe)$ from $\mathcal{T}$.

This discrepancy can be traced to the different setting in which the two types of negation live: strong negation can be seen as negation under OWA but in a single model setting. In a single model (in the sense of logic programming) knowledge about strongly negated atoms might be incomplete. For instance, it might be that in a stable model M neither an atomic proposition A nor its strong negation $-A$ is known (i.e., evaluates to true).

On the other hand, classical negation inherits its behavior from a scenario where the OWA is obtained by quantifying the truth of possible answers over multiple interpretations. The uncertainty of an assertion A is given by the fact that there might be interpretations in which A holds, and others in which A is false. In a single first-order interpretation, classical negation is interpreted under a complete knowledge assumption, and thus either A or $\neg A$ evaluates to true.

However, like in the example above, FOL semantics allows to determine that there is no interpretation in which $Author(joe\_doe)$ can hold, hence we can infer that $T \models \neg Author(joe\_doe)$, while the same conclusion does not hold using strong negation in a logic programming setting. Note that adding to P the rule

$$-author(X) \leftarrow -person(X).$$

is in general not enough to enforce a similar behavior, since logic programming lacks the *tertium non datur* property for strong negation; to enforce it, a rule $-p(X) \vee p(X) \leftarrow$ for every predicate p would need to be added.

Treatment of equality. Logic programming formalisms, including ASP, typically employ a Unique Name Assumption (UNA), i.e., different ground terms denote different objects, and do not support real equality reasoning, i.e., the possibility to infer knowledge about (in)equality of names. This does not comply necessarily with the view in classical logic, and thus with RDF and OWL, where no such assumption is made. While equality "$=$" and inequality "$\neq$" predicates are allowed in rule bodies, they represent syntactic equality and (default) negation thereof only. This shall not be confused with OWL's `owl:sameAs` and `owl:differentFrom` directives. Following up the example from Section 2.2, consider the following rule base:

$knowsOtherPeople(X) \leftarrow knows(X, Y), X \neq Y;$
$knows$(“`http://polleres.net/foaf.rdf#me`”,
“`http://www.polleres.net/foaf.rdf#me`”).

Under standard ASP semantics, “$\neq$” amounts to “$not =$”. Hence,

$knowsOtherPeople$(“`http://polleres.net/foaf.rdf#me`”)

would be entailed, while the same would not hold in similarly modelled OWL knowledge bases. Enabling reasoning with equality has usually a very high computational cost. Indeed, common DL reasoners like FACT++ [97] or RACER [47] also do not support full equality reasoning and nominals.

Existential quantification. The inability of expressing existence of individuals in logic programming is also matter of semantic discrepancy. Consider the theory

$$\mathcal{T} : \forall X \exists Y. (Person(X) \supset hasNationality(X, Y))$$

which, in DL Syntax, is equivalent to $Person \sqsubseteq \exists hasNationality$. This can be rendered as an equi-satisfiable Horn clause, by skolemizing the Y in the head (see above):

$$\mathcal{T} : \forall X. (Person(X) \supset hasNationality(X, f_Y(X)))$$

This clause can be rendered as a rule in logic programming, but not in standard Datalog, where function symbols are not allowed. However, most implemented systems can not actually evaluate a logic program equivalent to the clause above, since corresponding models are infinite. Elimination of functions symbols from logic programs or their evaluation in a decidable setting is indeed matter of continuous research (see, e.g., [11, 7, 90] and references therein).

Decidability. Finally, the probably largest obstacle towards combining the description logics world of OWL and the logic-programming world stems from the fact that these two worlds face undecidability issues from two completely different angles.

Indeed, decidability of logic programming (and in particular of its answer set programming dialects) follows from the fact that it is based on function-free Horn logic where ground entailment can be determined by checking finite subsets of the Herbrand base, i.e., decidability and termination of evaluation strategies is guaranteed by the finiteness of the domain. However, this is not so for description logics. Decidability of reasoning tasks such as satisfiability, class subsumption, or class membership in description logics is often strictly dependent on the combination of constructs which are allowed in the terminological language, living in a infinite domain.

For description logics, it is often possible to prove decidability of reasoning by means of the so called *tree-model property*. This property expresses that a DL knowledge base has a model iff it has a (possibly infinite) tree shaped model whose branching factor is bounded by the size of the knowledge base [4], such that the model gets, loosely speaking, repetitive after a finite number of steps. It is worth noting, however, that the DL $\mathcal{SHOIN}$ has not the tree-model property, and also not the finite-model property [53].

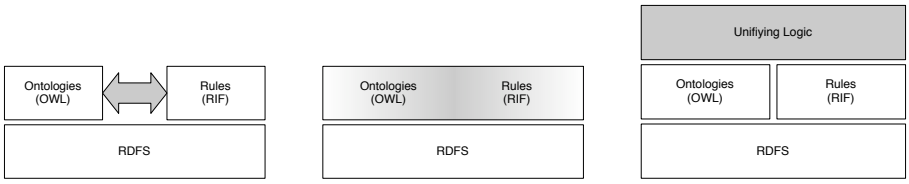

Fig. 7. Different combination categories for rules and ontologies

Unfortunately, it is difficult to combine two decidable fragments coming from the two worlds. As shown already in [59], the naive combination of even a very simple DL with an arbitrary Horn logic program is undecidable. Levy & Rousset [59] highlighted recursion and *unsafety* of rules as culprits for undecidability, and suggested *role-safety* as a remedy: at least one of variables X, Y in a role atom $R(X, Y)$ in a rule r must occur in a rule predicate in r that does not occur in any rule head of the program. As we will see later, most of the hybrid approaches indeed provide a notion of safety as a key tool for ensuring decidability.

5.3 Taxonomy of hybrid approaches

We can group hybrid rule formalisms into three main categories:

- Loose coupling (strict semantic separation);
- Tight integration; and
- Full integration.

We summarize next the peculiarities of the three categories. The reader can find further interesting material and discussion in [22, 72, 3, 88].

Loose coupling. Languages that are classified under the loose coupling category are denoted by a high level of semantic separation between P and $\mathcal{T}$.

Roughly speaking, the rule base P and the first-order theory $\mathcal{T}$ are treated as separate and independent components. An interface mechanism is then defined that allows the exchange of knowledge between the two sides. The particular design of the interfacing mechanism (*safe interfacing*) guarantees decidability of the combined knowledge base, although the flow of knowledge between the two sides is restricted, and in some cases, unidirectional (e.g., the rules component can import data from the classical component, but not vice versa). In important note is that loose coupling approaches are better suited for practical implementation on top of existing reasoners for the two sides.

As representatives of loose coupling frameworks, we mention nonmonotonic dl-programs [40, 35], defeasible logic coupled with description logic bases [102], and probabilistic dl-programs [63].

Tight semantic integration. With respect to loose coupling approaches, formalisms categorized under the tight semantic integration scenario tend to integrate FOL statements with the logic program to a larger extent, while keeping the vocabularies of the first-order predicates and the logic programming predicates distinct.

In general, a tightly integrated language is built on the notion of an integrated model which satisfies both the rules part P and the first-order part $\mathcal{T}$ of the knowledge base. Such a model can be often seen as $M = (M_o, M_l)$, that is, it is composed of two separate models M_o and M_l that share the same domain. M_o should satisfy the first-order theory, while M_l should satisfy the corresponding program. Depending on the semantics of the language at hand, there are different ways to define "agreement" of M_o and M_l on the overall knowledge base, thus defining a "safe interaction" method between the two worlds; see, e.g., [22] for more discussion. Representative of this category are CARIN [59], r-hybrid KBs, r^+-hybrid KBs, and $\mathcal{DL}$+*log* [85–87, 89].

Full integration. Full integration approaches are mostly distinct by the absence of separation between the two vocabularies at hand: the two universes are treated to a large extent in a homogeneous way; this, however, does not exclude to ascribe a certain intended role to a particular predicate (to be a rule predicate, or a classical predicate), which has to be done by proper axiomatization within the formalism.

Representative examples are Hybrid MKNF knowledge bases [69], first-order Autoepistemic Logic [23] and Open Answer Set Programs [49]. Terminological Default Logic [5], and Description Logics of Minimal Knowledge [29] can be viewed as related precursors.

In their work on g-hybrid knowledge bases [50], Heymans et al. show that actually tight integration approaches, such as r-hybrid KBs [86] from above, can partially be embedded into the above-mentioned Open Answer Set Programs. Likewise, in [24] de Bruijn et al. show that a non-classical logic can embrace several tight-coupling approaches by an elegant embedding into Quantified Equilibrium Logic [74, 75]. These two proposed approaches can be seen as frameworks unifying classical logic with disjunctive logic programs under *open* answer set programming in a common logical framework and thus may – despite keeping up the separation between classical and rules predicates – be viewed among the full integration approaches.

6 Sample Combination Approaches

In this section, we briefly review some concrete approaches for combining rules and ontologies that were mentioned in the previous section, one representative for each of the general kinds of integration, viz. non-monotonic dl-programs as an example for loose coupling, [33, 34], $\mathcal{DL}$+*log* [89], as an example for tight coupling, and Hybrid MKNF knowledge bases [69] as an example for full integration. After that, we compare these approaches in Section 6.4 with respect to several criteria.

6.1 Loose Coupling: Non-monotonic dl-Programs

dl-programs extend (function-free) answer set programs with *queries to DL knowledge bases* through dl-*atoms* [40, 35], which may be tuned to allow to query a DL knowledge

base in different ways. How the DL knowledge base and the logic program are matched is under control of the knowledge designer.

The actual implementation combines a DL engine and an ASP solver, whose interaction is clearly separated. The two sides can transfer knowledge bidirectionally through dl-atoms, which serves as an interface. The basic idea of dl-atoms is to provide a means for posing queries to the DL base $\mathcal{T}$ from the program P, by exploiting the native query facilities of the DL engine. In the course of this, also knowledge can flow from P to $\mathcal{T}$.

More in detail, a query Q can be a concept/role instance $C(X)/R(X,Y)$, or a subsumption $C \sqsubseteq D$. When submitting a query, a dl-atom allows to modify the extensional part (ABox) of $\mathcal{T}$, by adding positive ($\uplus$) or negative ($\cup$) assertions that are computed by the logic program P.[35] The dl-atom evaluates to true iff the modified $\mathcal{T}$ proves Q.

For example, the dl-atom $DL[Wine](\text{“}ChiantiClassico\text{”})$ asks whether it holds that $\mathcal{T} \models Wine(\text{“}ChiantiClassico\text{”})$; a dl-atom with a variable, $DL[Wine](X)$ evaluated to true for all the known individual x such that $\mathcal{T} \models Wine(x)$ holds.

The atom $DL[RedWine \uplus my\_red; Wine](X)$ adds all assertions $RedWine(c)$ to $\mathcal{T}$, such that $my\_red(c)$ holds in the logic program P, while $DL[RedWine \cup my\_white; hasColor](X, \text{“}Red\text{”})$ adds all assertions $\neg RedWine(c)$ to $\mathcal{T}$ such that $my\_white(c)$ holds in P. In both cases, the resulting theory $\mathcal{T}'$ is used for the query entailment test.

More formally, a *dl-program* [40, 35] is a pair $(\mathcal{T}, P)$ where P consists of rules of the form (31) where $l = 1$ and based on a function-free first order language, each a_i is a classical literal and each b_j is either a classical literal or a dl-atom; an extension allowing arbitrary $l \geq 0$ (and thus also disjunctive rules) has been considered in [37].

Answer sets of a dl-program $(\mathcal{T}, P)$ are defined via grounding all the rules in P with a set of constants C, where C contains the constants in P and additional constants from $\mathcal{T}$; by default, these additional constants are all the constants in $\mathcal{T}$, but they may also be designated (see [35]). A *model* is a consistent set of classical ground literals M built from the predicates in P and the constants in C. A ground dl-atom $DL[\langle Add\rangle; Q](\mathbf{c})$ is true in M, iff $\mathcal{T} \cup \langle Add\rangle^M \models Q(\mathbf{c})$. Note that $\langle Add\rangle^M$ is dependent on M; this enables a knowledge flow from P to $\mathcal{T}$.

A model M is called a *strong answer set* of $(\mathcal{T}, P)$, if it is the least model of sP^M, which is akin to the famous Gelfond-Lifschitz reduct P^M of an ordinary logic program with respect to M [43]. It generalizes P^M by handling dl-atoms, which are treated like ordinary atoms. That is, sP^M contains all rules obtained from the grounding of P by

- removing all rule instances r of form (31) such that for some b_j, where $j \in \{k+1, \ldots, m\}$, it holds that b_j is true in M (which for a classical literal b_j means $b_j \in M$), and
- removing all negation-as-failure literals $not\, b_j$ from the remaining rules.

In case of a dl-program with arbitrary rule heads ($l \geq 0$), in the above definition by “the least model” is replaced “a minimal model.”

dl-programs are decidable, provided that evaluating dl-atoms over $\mathcal{T}$ is decidable; in particular, they are NEXP-complete for $\mathcal{T} \in \mathcal{SHIF}(\mathbf{D})$ and $\mathrm{P}^{\mathrm{NEXP}}$-complete for $\mathcal{T} \in \mathcal{SHOIN}(\mathbf{D})$ [40, 35].

[35] Other modifications have been conceived, which we for simplicity disregard here. They lead to non-monotonic dl-atoms, i.e., queries to $\mathcal{T}'$ that can have non-monotonic behavior, which require special treatment.

As an example dl-program, consider a scenario where a computer network is encoded in an OWL DL knowledge base $\mathcal{T}''$, through the concept $Node$ and the role $wiredTo$. Imagine now that some new node x must be added to $\mathcal{T}''$, and it must be decided to which existing node x should be connected to. When choosing new connections, nodes belonging to the concept $HighTrafficNode$ should be avoided. High traffic nodes could be restricted in a way such that, e.g., $HighTrafficNode \sqsubseteq \geqslant k\ wired$, for some threshold value k. Thus connecting new nodes might trigger new high traffic nodes. This kind of interplay between the two sides of knowledge can be modelled with the following program:

$$connect(X,Y) \leftarrow newNode(X), DL[Node](Y), not\ overloaded(Y).$$
$$overloaded(X) \leftarrow DL[wired \uplus connect; HighTrafficNode](X).$$

The usage of dl-programs facilitates several advanced reasoning tasks: appropriate encodings allow to emulate CWA and *Extended CWA (ECWA)* [44] on top of a DL knowledge base. Similarly, dl-programs can incorporate Poole's-style [81] and a restricted fragment of Reiter's Default Logic [84] over DL bases. We show next how to emulate default reasoning and ECWA in dl-programs. The reader may refer to [35] for an extensive description of applications of dl-programs.

Default Reasoning. Reconsider the candidate reviewer selection scenario in Section 3.8, and suppose we have the following small knowledge base:

$$\mathcal{T} = \{\ \neg ex{:}ConflictingReviewer \sqsubseteq ex{:}CandidateReviewer,\ ex{:}Senior(joe),\ ex{:}Senior(bob),\ ex{:}ConflictingReviewer(bob)\ \}.$$

The rule that a senior author is a candidate reviewer by default (unless a conflict is apparent), can be mimicked by the following dl-program:

$$r_1 : cand\_rev(P) \leftarrow DL[ex{:}Senior](P), not\ conflict(P);$$
$$r_2 : \quad conflict(P) \leftarrow DL[ex{:}CandidateReviewer \uplus cand\_rev; ex{:}ConflictingReviewer](P).$$

Roughly speaking, r_1 encodes the fact that a senior author should be considered as a canidate reviewer, unless a conflict can be proven. Under Answer Set Semantics, r_2 effects maximal application of r_1 over $\mathcal{T}$. The single answer set M will thus be as follows:

$$\{\ cand\_rev(joe),\ conflict(bob)\ \}.$$

Minimal Models and ECWA. If one considers a DL base with disjunctive information, such as:

$$\mathcal{T} = \{\ Publication(p1), \quad Publication \equiv Journal\_Pub \sqcup Conference\_Pub\ \}$$

one can consider the goal of maximizing negative information (thus, minimizing positive knowledge) without raising inconsistency. The program shown next singles out "minimal" models, in the setting of Extended CWA (ECWA):

$$\overline{j\_pub}(X) \leftarrow not\ j\_pub(X).$$
$$\overline{c\_pub}(X) \leftarrow not\ c\_pub(X).$$
$$j\_pub(X) \leftarrow DL[Journal\_Pub \uplus \overline{j\_pub}, Conference\_Pub \uplus \overline{c\_pub}; Journal\_Pub](X).$$
$$c\_pub(X) \leftarrow DL[Journal\_Pub \uplus \overline{j\_pub}, Conference\_Pub \uplus \overline{c\_pub}; Conference\_Pub](X).$$

In simple terms, the first two rules effect CWA on the concepts of journal and conference publication. The last two rules maximally propagate inferred negative information to $\mathcal{T}$. The answer sets, corresponding to minimal models, of the above program are:

$$M_1 = \{j\_pub(p1), \overline{c\_pub}(p1)\},$$
$$M_2 = \{c\_pub(p1), \overline{j\_pub}(p1)\}.$$

The same encoding structure can be extended to select those concept to be kept "fixed" as in the general ECWA setting.

In [35], also *weak answer sets* have been introduced, which are defined like strong answer sets with the only difference that in building the reduct, besides the not-literals, also dl-atoms b_j that are not under not are "evaluated" for rule and literal elimination. However, in contrast to strong answer sets, weak answer sets are not guaranteed to be minimal, in the sense that a weak answer set may contain some other weak answer set properly; intuitively, they are less "grounded" than strong answer sets. Furthermore, dl-programs have been recently extended to support also (union of) conjunctive queries over the DL base [33, 34].

6.2 Tight integration: $\mathcal{DL}$+*log*

$\mathcal{DL}$+*log* [89] is the latest in a chain of extensions of the DL $\mathcal{ALC}$ with rules such as $\mathcal{AL}$-*log*, r- and r$^+$-hybrid knowledge bases. The key semantic choices of $\mathcal{DL}$+*log* can be summarized as follows:

(a) A distinction between rule-predicates and classical predicates.
(b) a fixed, countably infinite domain, whose elements e can be accessed in all interpretations with distinguished constant c_e in a one-to-one correspondence; this is called the *Standard Names Assumption* (note that this implies the UNA, and that interpretations are isomorphic to Herbrand interpretations in absence of function symbols).
(c) Models (called *NM-models*) of $\mathcal{KB} = \langle \mathcal{T}, P \rangle$ are of form $\mathcal{I} \cup M$, where $\mathcal{I}$ is a model of the classical predicates, M of the rules-predicates, after deletion of classical atoms satisfied by $\mathcal{I}$ in P.
(d) The language has no strong negation, and weak negation is limited to rules-predicates, but classical predicates can appear in rules heads; function symbols are not considered.
(e) To ensure decidability, *weak (DL-)safety* is used: each variable X in a rule r must occur in some positive body atom of r, and this atom must have a rule predicate if X occurs in an atom with classical predicate in the head of r.

Note that weak safety allows to access unnamed individuals in classical atoms. For instance, take $\mathcal{KB} = \langle \mathcal{T}, P \rangle$, where $\mathcal{T} = \{author \sqsubseteq \exists isAuthorOf\,,\ author(turing)\}$ and P consists of the weakly DL-safe rule:

$$scientist(X) \leftarrow isAuthorOf(X, Y), not\ likes(X, astrology);$$

Here $isAuthorOf$ is a classical predicate and $scientist$ and $likes$ are rule predicates. The variable Y, which does not occur in any atom with a rule predicate, can access also unknown individuals. We have $\mathcal{KB} = \langle \mathcal{T}, P \rangle \models_{NM} scientist(turing)$ as intuitively

expected, although Y can not be instantiated and might vary from interpretation to interpretation. The same rule expressed as a dl-program would look like

$$scientist(X) \leftarrow DL[isAuthorOf](X, Y), not\ likes(X, astrology)$$

which does not entail $scientist(turing)$. However, the remodeled dl-program

$$scientist(X) \leftarrow DL[\exists isAuthorOf](X), not\ likes(X, astrology);$$

yields the expected answer; using the extended syntax proposed in [33, 34] (allowing also conjunctive queries), this dl-atoms can be expressed as $DL[father(X, Y)](X)$.

The stable model (or answer set) semantics of $\mathcal{DL}$+*log* is conceived in a 2-step reduction.

- In the first step, an interpretation $\mathcal{I}$ of the classical predicates is taken. Then P is grounded and "reduced" with respect to $\mathcal{I}$, by "evaluating" and eliminating classical atoms from rules (that is, classical atoms satisfied in $\mathcal{I}$ and appearing in bodies are eliminated, classical atoms not satisfied in $\mathcal{I}$ and appearing in heads are eliminated, rules which have falsified body and/or true head are eliminated). The resulting ground program $P_{\mathcal{I}}$ contains no classical predicates.
- In the second step, we define M as a stable model of $P_{\mathcal{I}}$ as usual.

The $\mathcal{DL}$+*log* formalism is decidable, if containment between union of conjunctive queries is decidable in $\mathcal{T}$.

For an example, consider the following $\mathcal{KB} = \langle \mathcal{T}, P \rangle$:

$$\begin{aligned}
\mathcal{T} = \{\ & Multilingual \sqsubseteq \neg Monolingual; \\
& Multilingual \sqcup Monolingual \sqsubseteq Author; \\
& Author \sqsubseteq \exists isAuthorOf;\ \ Author(joey)\ \} \\
P = \{ & novelist(X) \vee scientist(X) \leftarrow writer(X); \\
& Monolingual(X) \leftarrow novelist(X); \\
& Multilingual(X) \leftarrow scientist(X); \\
& writer(joey); \\
& scientist(X) \leftarrow writer(X), isAuthorOf(X, Y), not\ likes(X, astrology)\}
\end{aligned} \tag{33}$$

Given a consistent interpretation $\mathcal{I}_1$, s.t. the set of classical atoms $\{Author(joey), Multilingual(joey)\}$ holds in $\mathcal{I}_1$, we have

$$\begin{aligned}
P_{\mathcal{I}_1} = \{\ & novelist(joey) \vee scientist(joey) \leftarrow writer(joey); \\
& \leftarrow novelist(joey); \\
& writer(joey); \\
& scientist(joey) \leftarrow writer(joey), not\ likes(joey, astrology)\}
\end{aligned}$$

The interpretation $M_1 = \{writer(joey), scientist(joey)\}$ is a stable model, while $M_2 = \{writer(joey), novelist(joey)\}$ is not a stable model. Indeed, we have

$$\begin{aligned}
{P_{\mathcal{I}}}^{M_1} = \{\ & novelist(joey) \vee scientist(joey) \leftarrow writer(joey); \\
& \leftarrow novelist(joey); \\
& writer(joey); \\
& scientist(joey) \leftarrow writer(joey)\}
\end{aligned}$$

which has as single minimal model M_1. Since ${P_{\mathcal{I}}}^{M_2} = {P_{\mathcal{I}}}^{M_1}$, M_2 is not a stable model.

If we take $\mathcal{I}_2$, where *joey* belongs to *Monolingual* and *Author*, we get

$$\begin{aligned} P_{\mathcal{I}_2} = \{\ & novelist(joey) \vee scientist(joey) \leftarrow writer(joey); \\ & \leftarrow scientist(joey); \\ & writer(joey); \\ & scientist(joey) \leftarrow writer(joey), not\ likes(joey, astrology)\} \end{aligned}$$

We cannot find any stable model, since in any such M, $likes(joey, astrology)$ must be false, otherwise $scientist(joey)$ would be true, in contradiction with the constraint $\leftarrow$ $scientist(joey)$.

6.3 Full Integration: Hybrid MKNF knowledge bases

Building on Lifschitz's bimodal *Logic of Minimal Knowledge and Negation as Failure (MKNF)* [60], hybrid MKNF knowledge bases [69, 68] aim at a seamless integration of classic and nonmonotonic semantics beyond tight integration approaches. The formalism uses two modal operators: $\mathbf{K}\phi$, which intuitively should mean that ϕ is necessarily known, and $\mathbf{not}\phi$, which intuitively means that ϕ is not true, i.e., there is some scenario in which ϕ is false.

In hybrid MKNF KBs, the rules in P have the form

$$\mathbf{K}h_1 \vee \cdots \vee \mathbf{K}h_l \leftarrow \mathbf{K}b_1, \ldots \mathbf{K}b_m, \mathbf{not}\, b_{m+1}, \ldots, \mathbf{not}\, b_n$$

where all h_i and b_j are function-free first-order atoms; they are seen as MKNF formulas

$$(\forall)\mathbf{K}b_1 \wedge \cdots \wedge \mathbf{K}b_m \wedge \mathbf{not}\, b_{m+1}, \wedge \cdots \wedge \mathbf{not}\, b_n \supset \mathbf{K}h_1 \vee \cdots \vee \mathbf{K}h_l.$$

The first-order part $\mathcal{T}$ is converted to the formula $\mathbf{K}\bigwedge_{\phi\in\mathcal{T}}\phi$, assuming that $\mathcal{T}$ is finite. As in other formalisms, no function symbols are allowed.

The semantics of the hybrid MKNF KB $\mathcal{KB} = \langle \mathcal{T}, P\rangle$ is then defined in terms of the semantics of the conjunction of these MKNF formulas, which we denote by MKNF($\mathcal{KB}$). As in $\mathcal{DL}$+*log*, a fixed, countably infinite domain and the Standard Names Assumption is used, but in addition Herbrand interpretations are explicitly assumed.

In the tradition of Kripke-style semantics for modal logics, models are sets of interpretations $\mathcal{M}$ rather than single interpretations $\mathcal{I}$. Intuitively, a model $\mathcal{M}$ represents a group of interpretations or "possible worlds" $\mathcal{I}$ in which a given formula is true. The operator $\mathbf{K}\phi$ can be seen as the logical necessity operator under modal logic $S5$ axiomatization; in Kripke-semantic terms, this means that given a model $\mathcal{M}$, each world $\mathcal{I}$ can access each world $\mathcal{I}'$ in $\mathcal{M}$ (including itself); thus, a formula $\mathbf{K}\phi$ evaluates to true at a world $\mathcal{I}$, if ϕ evaluates to true at each world $\mathcal{I}'$ in $\mathcal{M}$. Similarly, $\mathbf{not}\ \phi$ evaluates to false at $\mathcal{I}$ if ϕ evaluates to false at some $\mathcal{I}'$ in $\mathcal{M}$. Atoms, propositional combinations of formulas, and quantifiers are evaluated at $\mathcal{I}$ as usual in first-order logic.

A model $\mathcal{M}$ is now an *MKNF model* of $\mathcal{KB} = \langle \mathcal{T}, P\rangle$, if the formula MKNF($\mathcal{KB}$) evaluates to true at each world of $\mathcal{M}$, and it is not possible to increase $\mathcal{M}$ to some $\mathcal{M}' \supset \mathcal{M}$ such that MKNF($\mathcal{KB}$) evaluates to true at some world of $\mathcal{M}'$, if $\mathbf{K}$ would be evaluated with respect to $\mathcal{M}'$ but $\mathbf{not}$ with respect to $\mathcal{M}$. Intuitively, $\mathcal{M}$ is "maximal"

and embodies the Minimal Knowledge Principle in the sense that the more interpretations (possible worlds) a model contains, the less certain knowledge is associated with it. Note that for a modal-free formula ϕ, the formula $\mathbf{K}\phi$ is equivalent to ϕ, as the only maximal model $\mathcal{M}$ such that $\mathcal{M} \models \mathbf{K}\phi$ coincides with the set of all the first-order interpretations $\mathcal{I}$ such that $\mathcal{I} \models \phi$. On the other hand, the **not** operator implements negation as failure and can be read as "there is the possibility that ϕ is false."

Although in hybrid MKNF KBs there is no distinction between classical and rules predicates for defining the semantics, this issue becomes relevant for ensuring decidability of reasoning. To this end, DL-safety of the rules in P is adopted, where predicates that appear in $\mathcal{T}$ are considered as DL-predicates, and all other ones (occurring only in P) as non-DL-predicates. Furthermore, on the first-order side $\mathcal{T}$ is restricted to a decidable DL.

Hybrid MKNF KBs can be seen as a generalization of CARIN [59], $\mathcal{AL}$-*log* [28], and DL-safe rules [70]. They extend, like dl-programs, and $\mathcal{DL}$+*log*, logic programs and description logic faithfully in the sense that the consequences of hybrid KBs $(\emptyset, P)$ and $(\mathcal{T}, \emptyset)$ reflect consequences of stable model semantics and first-order semantics, respectively (where for dl-programs, only consequences given by queries make sense).

For an example, consider the following extension of the hybrid KB (33). The ontology part is extended to

$$\begin{aligned}\mathcal{T} = \{\ & \mathit{Multilingual} \sqsubseteq \neg \mathit{Monolingual};\\ & \mathit{Multilingual} \sqcup \mathit{Monolingual} \sqsubseteq \mathit{Author};\\ & \mathit{Author} \sqsubseteq \exists \mathit{isAuthorOf};\ \ \mathit{Author}(\mathit{joey});\ \ \mathit{Lefthanded} \sqsubseteq \mathit{Author}\ \}\end{aligned}$$

where the last axiom introduces a class of authors using their left hand to write. To the rules part, we add that the rule that authors write with their right hand if they are not left-handed, and this is the default. This leads to the following program part P:

$$\begin{aligned}P = \{ & \mathbf{K}\mathit{novelist}(X) \vee \mathbf{K}\mathit{scientist}(X) \leftarrow \mathbf{K}\mathit{writer}(X);\\ & \mathbf{K}\mathit{Monolingual}(X) \leftarrow \mathbf{K}\mathit{novelist}(X);\\ & \mathbf{K}\mathit{Multilingual}(X) \leftarrow \mathbf{K}\mathit{scientist}(X);\\ & \mathbf{K}\mathit{scientist}(X) \leftarrow \mathbf{K}\mathit{writer}(X), \mathbf{K}\mathit{isAuthorOf}(X,Y), \mathbf{not}\,\mathit{likes}(X,\mathit{astrology});\\ & \mathbf{K}\mathit{writer}(\mathit{joey});\\ & \mathbf{K}\mathit{Righthanded}(X) \leftarrow \mathbf{K}\mathit{Author}(X), \mathbf{not}\ \mathit{Lefthanded}(X)\ \}\end{aligned}$$

Note that compared to $\mathcal{DL}$+*log* (in which the new rule cannot be formulated), it is now possible to use negation as failure over first-order predicates such as *Lefthanded*. As the authors of [69] describe, in some sense "closed world glasses" can be put on classical predicates, allowing to state exceptions.

By treating DL concepts and roles as objective knowledge (i.e., without the $\mathbf{K}$ operator), and the rule predicates as modal, it is possible to port a $\mathcal{DL}$+*log* knowledge base into an equi-satisfiable generalized hybrid MKNF KB. For more details, see [68].

6.4 Assessment

Some noticeable features of the semantics of dl-programs, $\mathcal{DL}$+*log*, and hybrid MKNF knowledge bases are summarized Table 5, following a similar assessment in [22]. For

Table 5. Comparison table for some hybrid approaches.

	dl-programs	$\mathcal{DL}$+*log*	**hybrid MKNF**	**SWRL**
Distinguish classical and rule predicates	+	+	–	–
Domain of Discourse for P				
Herbrand Universe of P	–	+$\sim$	+	–
Combined Signature	+	+$\sim$	+	–
Arbitrary domains	–	–	–	+
Uniqueness of Names				
unique names in HU of P	+	+	+	–
Special equality predicate	+$\sim$	+$\sim$	+	+
No uniqueness	–	–	–	+
Knowledge Interaction: from First-Order Theories to Rules				
Per single model	–	+	–	+
Entailment	+	–	+	–
Knowledge Interaction: from Rules to First-Order Theories				
Per single model	–	+	+	+
Entailment	+	–	–	–
Decidability	+$\sim$	+$\sim$	+$\sim$	–

Legenda: + = yes, – = no, +$\sim$ = yes, with some proviso

the sake of comparison, we have added also SWRL there as a prominent non-hybrid approach. The first row identifies which formalisms have different vocabularies for classical and rule predicates names, respectively. Note that this feature is not distinctive of loose coupling approaches, although it can be seen as an indication of the level of coupling between classic and logic programming semantics.

The second group of features identifies which choice is taken regarding the domain of discourse for the logic programming part P of a knowledge base. The choice varies between taking a single arbitrary domain, such as for SWRL, or adopting a combined, yet overlapping, signature (such as for hybrid MKNF and dl-programs). Such a signature usually defines two distinct domains of discourse and their interaction. In this latter setting, it can be chosen whether to take the Herbrand Universe as domain for P.

As it can be seen in the second group of features, dl-programs, $\mathcal{DL}$+*log*, and hybrid MKNF KBs have unique names in the Herbrand universe of the rules part. In fact, $\mathcal{DL}$+*log* and hybrid MKNF KBs fulfill the UNA in the whole knowledge base, which is implied by the Standard Names Assumptions they adopt. Note, however, that $\mathcal{DL}$+*log* is not committed to Herbrand interpretations of constants in the rules part.

Although under UNA, it is possible to identify different names using a special equality predicate such as $\approx$ in hybrid MKNFs. The same is in principle possible for both $\mathcal{DL}$+*log* and dl-programs, which are extensible with axioms defining an appropriate congruence relation. This has actually been theoretically introduced and implemented for dl-programs [40, 35], which feature the possibility of simulating a congruence relation or defining a customized behavior for equality. Note that dl-programs tolerate non-uniqueness of names on the classical logic side of the knowledge base signature. SWRL

has native features for reasoning with non-uniqueness of names, which is the default setting.

Regarding the interaction from the ontology (first-order theory) to the rules, we distinguish whether the truth of literal with "classical" predicate in a rule depends for model construction on a single model of the first-order part of the hybrid KB, or on entailment from multiple models. Here, "model" is understood in the wider sense of first-order logics interpretation/hybrid model; for hybrid MKNF, it is a first-order interpretation in a MKNF model (which is a set of first-order interpretations). $\mathcal{DL}$+*log* and SWRL work on a single model basis, while dl-programs and hybrid MKNF employ inference from multiple models; in dl-programs, information from the first-order theory is imported to rules only if a query is proven from the (possibly constrained) set of models of the first-order part. Similarly, the operators $\mathbf{K}$ and $\mathbf{not}$ in hybrid MKNFs imply a quantification over multiple first-order models before knowledge can be considered true/false within a rule.

For the reverse direction (from the rules to the first-order part), single-model interaction is understood in the sense that each model $\mathcal{I}$ of the rules part P *constrains* the models of the first order part $\mathcal{T}$ such that only models will be considered in which all classical predicates have a larger extent than in $\mathcal{I}$. Entailment based interaction, instead, simply adds positive conclusions about the classical predicates that can be drawn from the model of the logic program to the first-order part. Note that this may make a difference, if we can have elements in interpretations that can not be accessed via some ground term. Here, only dl-programs are conceived according to the second principle, through the special dl-atom device, which adds conclusions about classical predicates to the ontology.

As a last yet important parameter, we consider decidability. Both dl-programs and hybrid MKNF KBs are decidable, provided that satisfiability checking for the underlying description logic base is decidable, and the rules part is DL-safe (for dl-programs, DL-safety is implicitly ensured). Compared to MKNF, the $\mathcal{DL}$+*log* formalism asks only for *weak* DL-safety, but in turn containment between union of conjunctive queries must be decidable in the underlying first-order theory $\mathcal{T}$.

6.5 Further Aspects

There are many interesting aspects that we can not cover here. Probabilistic and fuzzy hybrid systems under stable model semantics for the rules have been investigated under both the loose coupling and tight coupling approach; see [63, 14, 64]. An extension of RDF(S) with stable models has been proposed in [2].

Besides stable models semantics, the research community also paid attention to hybrid knowledge bases with well-founded semantics on the rules side: for example, a well-founded semantics for dl-programs [41] and for hybrid MKNF knowledge bases [57] has been defined, while *hybrid rules* under well-founded semantics [31] follow the approach of the $\mathcal{DL}$+*log* family.

A rich line of research has investigated the possibility of emulating first-order semantics by mapping first-order theories into equivalent logic programs. A noticeable translation from $\mathcal{SHIQ}$ to positive disjunctive Datalog (which has an exponential blow

up in the worst case) was given in [54]. A correspondence between open logic programming and $\mathcal{ALCN}$ has been shown in [99]. Other attempts to map description logics into answer set semantics are [95] and [1]. A decidable fragment of ASP extended with function symbols that is rich enough to capture $\mathcal{ALC}$ has been recently described [90].

7 Conclusion

Advanced reasoning frameworks for future Semantic Web applications need to deal with both rules and ontologies in an integrated manner, which is currently not supported well and an active area of research. In this article, we have considered a number of rule-based formalisms to work on top of or aside ontology bases. They work at different levels of integration, ranging from a low level, at which the integration is ad hoc, to a high level, where a genuine semantics is given to a combination of rules and ontologies.

In the course of this, we have developed a number of criteria and discriminating features, which we then used to profile the various formalisms and systems. As for implemented systems, we have briefly addressed the languages they support, and we related them to foundational approaches to combining rules and ontologies. Furthermore, we have also discussed selected approaches at the high level that are on the forefront of research, whose impact for future developments remains to be seen.

Looking at the tool support that is currently available, we found that many – and quite diverse – systems and languages exist, and that there is no easy way to change from one system to another in general; this means that once the user gets stuck when modeling her application with a specific system, then she has to port the whole rule base to another system; this is however not always feasible for any arbitrary target system. In this regard, the RIF standardization effort of the W3C is not only useful to promote rule languages, but also to give more freedom to the users in choosing the "right" system for their application. The Semantic Web as such is a good application playground for pushing the frontiers in the implementations and for providing solid and scalable implementation of rule/ontology languages.

When we looked at the issue of combining rules and ontologies into a unifying framework, we found that this is not easy given the quite different features underlying logic programs and ontologies, since the latter are mainly based on classical logic while the former are not. Recent proposals are a step forward but the issue is not resolved yet (as it seems), and more research efforts will be necessary. After the successful ontology initiative of the W3C which resulted in the OWL standard, it is to be hoped that the RIF effort will converge to a useful standard as well, even though this is far less clear given the many facets of rules and views what rules are.

Current and future research centers around the following questions.

Semantics for rules plus ontologies. While a number of proposals for a semantics of rules combined with ontologies have been made, it is not clear whether these proposals are already sufficient and will show satisfactory behavior in relevant cases. What is missing at this point are case studies and large(r) scale examples beyond the toy examples which have been considered in the seminal papers that introduced the approaches. This, in turn, may also provide guidance in the development of a "gold standard" for rules plus ontologies.

Semantic and computational properties. Related with these, we need to know more about the semantic and computational properties of the various approaches for rules and ontologies, and also how they relate to each other. Studies on how knowledge bases in the one formalism can be transformed into knowledge bases in the other formalism are useful in this regard, as well as to understand what scenarios can be expressed in a formalism (and which not). Related to this is the issue of computational complexity, which gives us however a somewhat coarser view than the expressiveness of a formalism in terms of (sets of) models that it can represent. Current complexity studies provide us with basic results, but more refined ones and studies of expressiveness issues are missing.

Efficient implementations, algorithms. Of most reasoning engines, especially at the high level of integration, only simple prototypes or even no implementations are available. Implementations, however, are barely needed in order to experiment with a formalism not only to measure performance, but also to understand and analyze its behavior. Doing this on paper is cumbersome (and tedious). Guided by the results of complexity studies, efficient implementations have to be developed, and the great challenge of scalability has to be met. This, however, might require to modify the semantics or to develop suitable approximation methods to facilitate reasoning with manageable resources.

Beyond logic rules. The current integration efforts aim at logic rules, be it in the reading of rules as logical clauses, or in the style of (non-monotonic) rules as in logic programming. In fact, there are many more kinds of rules out there which we need to integrate with ontologies as well; for example, production rules as available in traditional expert systems engines, that are based on an operational semantics; business rules, which are used in the context of business policies and whose semantics is not always clear; etc.

Knowledge combination/integration beyond rules and ontologies. Connected to the previous issue, knowledge integration beyond a simple pair of a rules and an ontology part is an issue. Both the rules and the ontology part need not be homogeneous, but composed of parts itself that have difference semantics; furthermore, knowledge bases of a different kind than rules (e.g., descriptions of temporal processes like work flows or protocols in a temporal logic, or action theories) may need to be integrated. This calls for a logic framework in which knowledge modules, having different native semantics, can be put together under a meaningful semantics, ideally in a plug and play manner – realizing this vision is a challenging goal.

References

1. G. Alsaç and C. Baral. Reasoning in description logics using declarative logic programming. Technical report, CS Dept, Arizona State University, 2001.
2. A. Analyti, G. Antoniou, C. V. Damásio, and G. Wagner. Stable Model Theory for Extended RDF Ontologies. In *Proceedings of the Fourth International Semantic Web Conference* (*ISWC 2005*), pages 21–36, 2005.
3. G. Antoniou, C. V. Damásio, B. Grosof, I. Horrocks, M. Kifer, J. Maluszynski, and P. F. Patel-Schneider. Combining Rules and Ontologies: A survey. Technical Report

IST506779/Linköping/I3-D3/D/PU/a1, Linköping University, February 2005. IST-2004-506779 REWERSE Deliverable I3-D3. http://rewerse.net/publications/.
4. F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications. 2nd Edition.* Cambridge University Press, 2007.
5. F. Baader and B. Hollunder. Embedding defaults into terminological representation systems. *J. Automated Reasoning*, 14:149–180, 1995.
6. C. Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press, 2002.
7. S. Baselice, P. A. Bonatti, and G. Criscuolo. On finitely recursive programs. In *Proceedings 23rd International Conference on Logic Programming (ICLP 2007)*, volume 4670 of *Lecture Notes in Computer Science*, pages 89–103. Springer, 2007.
8. T. 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/.
9. H. Boley, M. Kifer, P.-L. Pătrânjan, and A. Polleres. Rule interchange on the web. In *Reasoning Web 2007*, volume 4636 of *Lecture Notes in Computer Science* (*LNCS*), pages 269–309. Springer, Sept. 2007.
10. H. Boley and M. Kifer (eds.). RIF Basic Logic Dialect, Oct. 2007. W3C Working Draft, available at http://www.w3.org/TR/2007/WD-rif-bld-20071030.
11. P. A. Bonatti. Reasoning with infinite stable models. *Artificial Intelligence*, 156(1):75–111, 2004.
12. D. Brickley and R. Guha (eds.). RDF vocabulary description language 1.0: RDF Schema, Feb. 2004. W3C Recommendation, available at http://www.w3.org/TR/rdf-schema/.
13. F. Bry, N. Eisinger, T. Eiter, T. Furche, G. Gottlob, C. Ley, B. Linse, R. Pichler, and F. Wei. Foundations of rule-based query answering. In *Reasoning Web 2007*, volume 4636 of *Lecture Notes in Computer Science* (*LNCS*), pages 1–153. Springer, Sept. 2007.
14. A. Calì and T. Lukasiewicz. Tightly integrated probabilistic description logic programs for the semantic web. In *Logic Programming, 23rd International Conference, ICLP 2007*, pages 428–429, 2007.
15. F. Calimeri, S. Cozza, G. Ianni, and N. Leone. DLV-Complex homepage, since 2008. http://www.mat.unical.it/dlv-complex.
16. D. Calvanese, G. D. Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Embedding defaults into terminological representation systems. *J. Automated Reasoning*, 39:385–429, 2007.
17. J. Carroll, C. Bizer, P. Hayes, and P. Stickler. Named graphs. *Journal of Web Semantics*, 3(4), 2005.
18. K. G. Clark, L. Feigenbaum, and E. Torres. SPARQL Protocol for RDF, Nov. 2007. W3C Proposed Recommendation, available at http://www.w3.org/TR/2007/PR-rdf-sparql-protocol-20071112/.
19. D. Connolly (ed.). Gleaning Resource Descriptions from Dialects of Languages (GRDDL), Sept. 2007.
20. C. V. Damásio, A. Analyti, G. Antoniou, and G. Wagner. Supporting open and closed world reasoning on the web. In *PPSWR*, volume 4187 of *Lecture Notes in Computer Science*, pages 149–163. Springer, 2006.
21. J. de Bruijn. *Semantic Web Language Layering with Ontologies, Rules, and Meta-Modeling*. PhD thesis, Faculty of Mathematics, Computer Science and Physics of the University of Innsbruck, Innsbruck, Austria, 2008.
22. J. de Bruijn, T. Eiter, A. Polleres, and H. 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)*, volume 4092 of *Lecture Notes in Computer Science* (*LNCS*), pages 1–22, Guilin, China, 2006. Springer.

23. J. de Bruijn, T. Eiter, A. Polleres, and H. Tompits. Embedding non-ground logic programs into autoepistemic logic for knowledge base combination. In M. Veloso, editor, *Proceedings 20th International Joint Conference on Artificial Intelligence* (*IJCAI 2007*), pages 304–309. AAAI Press/IJCAI, 2007.
24. J. de Bruijn, D. Pearce, A. Polleres, and A. Valverde. A logic for hybrid rules. In *Proceedings Second International Conference on Rules and Rule Markup Languages for the Semantic Web* (*RuleML 2006*). IEEE, 2006. Available at `http://2006.ruleml.org/online-proceedings/rule-integ.pdf`.
25. J. de Bruijn, A. Polleres, R. Lara, and D. Fensel. OWL$^-$. Final draft d20.1v0.2, WSML, 2005.
26. J. de Bruijn (ed.). RIF RDF and OWL Compatibility, Oct. 2007. W3C Working Draft, available at `http://www.w3.org/TR/2007/WD-rif-bld-20071030`.
27. M. Dean, G. Schreiber, S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein. OWL Web Ontology Language Reference, Feb. 2004. W3C Recommendation.
28. F. M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. $\mathcal{AL}$-log: Integrating Datalog and Description Logics. *Journal of Intelligent Information Systems*, 10(3):227–252, 1998.
29. F. M. Donini, D. Nardi, and R. Rosati. Description logics of minimal knowledge and negation as failure. *ACM Trans. Comput. Log.*, 3(2):177–225, 2002.
30. W. Drabent, J. Henriksson, and J. Maluszynski. HD-Rules: a hybrid system interfacing prolog with dl-reasoners. In *2nd International Workshop on Applications of Logic Programming to the Web, Semantic Web and Semantic Web Services. ALPSWS2007*, 2007.
31. W. Drabent and J. Maluszynski. Well-founded semantics for hybrid rules. In *Web Reasoning and Rule Systems, First International Conference, RR 2007*, pages 1–15, 2007.
32. M. Duerst and M. Suignard. Internationalized Resource Identifiers (IRIs). RFC 3987 (Proposed Standard), Jan. 2005.
33. T. Eiter, G. Ianni, T. Krennwallner, and R. Schindlauer. Exploiting conjunctive queries in description logic programs. In *Proceedings of the 2007 International Workshop on Description Logics (DL2007)*, pages 259–266, 2007.
34. T. Eiter, G. Ianni, T. Krennwallner, and R. Schindlauer. Exploiting conjunctive queries in description logic programs. Technical Report INFSYS RR-1843-08-02, Institut für Informationssysteme, Technische Universität Wien, A-1040 Vienna, Austria, Mar. 2008. Extended version of the DL'07/ISAIM'08 abstract.
35. T. Eiter, G. Ianni, T. Lukasiewicz, R. Schindlauer, and H. Tompits. Combining Answer Set Programming with Description Logics for the Semantic Web. Technical Report INFSYS RR-1843-07-04, Institut für Informationssysteme, TU Wien, Mar. 2007. To appear in *Artificial Intelligence*.
36. T. Eiter, G. Ianni, A. Polleres, and R. Schindlauer. Answer set programming for the semantic web, June 2006. Slides available at `http://asptut.gibbi.com/`.
37. T. Eiter, G. Ianni, A. Polleres, R. Schindlauer, and H. Tompits. Reasoning with rules and ontologies. In P. Barahona, F. Bry, E. Franconi, U. Sattler, and N. Henze, editors, *Reasoning Web 2006*, volume 4126 of *LNCS*, pages 93–127. Springer, Sept. 2006.
38. T. Eiter, G. Ianni, R. Schindlauer, and H. Tompits. A Uniform Integration of Higher-Order Reasoning and External Evaluations in Answer Set Programming. In L. P. Kaelbling and A. Saffiotti, editors, *Proceedings of the 19th International Joint Conference on Artificial Intelligence* (*IJCAI 2005*), pages 90–96. Professional Book Center, 2005.
39. T. Eiter, G. Ianni, R. Schindlauer, and H. Tompits. Effective Integration of Declarative Rules with External Evaluations for Semantic Web Reasoning. In Y. Sure and J. Domingue, editors, *Proceedings of the Third European Semantic Web Conference* (*ESWC 2006*), number 4011 in LNCS, pages 273–287. Springer, 2006.

40. T. Eiter, T. Lukasiewicz, R. Schindlauer, and H. Tompits. Combining answer set programming with description logics for the Semantic Web. In *Proceedings KR-2004*, pages 141–151, 2004.
41. T. Eiter, T. Lukasiewicz, R. Schindlauer, and H. Tompits. Well-Founded Semantics for Description Logic Programs in the Semantic Web. In *Proceedings of the ISWC 2004 Workshop on Rules and Rule Markup Languages for the Semantic Web* (*RuleML 2004*), volume 3323 of *Lecture Notes in Computer Science* (*LNCS*), pages 81–97. Springer Verlag, 2004.
42. The Friend of a Friend (FOAF) Project. `http://www.foaf-project.org/`.
43. M. Gelfond and V. Lifschitz. Classical Negation in Logic Programs and Disjunctive Databases. *New Generation Computing*, 9:365–385, 1991.
44. M. Gelfond, H. Przymusinska, and T. C. Przymusinski. The Extended Closed World Assumption and its Relationship to Parallel Circumscription. In *Proceedings Fifth ACM Symposium on Principles of Database Systems* (*PODS '86*), pages 133–139, 1986.
45. B. N. Grosof, I. Horrocks, R. Volz, and S. Decker. Description logic programs: Combining logic programs with description logics. In *Proceedings WWW-2003*, pages 48–57, 2003.
46. T. R. Gruber. A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*, 5:199–220, 1993.
47. V. Haarslev and R. Möller. RACER System Description. In *Proceedings of the First International Joint Conference on Automated Reasoning* (*IJCAR 2001*), volume 2083 of *Lecture Notes in Computer Science* (*LNCS*), pages 701–705. Springer Verlag, 2001.
48. P. Hayes. RDF semantics. `http://www.w3.org/TR/rdf-mt/`.
49. S. Heymans, D. V. Nieuwenborgh, and D. Vermeir. Open answer set programming for the semantic web. *J. Applied Logic*, 5(1):144–169, 2007.
50. S. Heymans, L. Predoiu, C. Feier, J. de Bruijn, and D. van Nieuwenborgh. G-hybrid knowledge bases. In *Proceedings of the ICLP'06 Workshop Workshop on Applications of Logic Programming in the Semantic Web and Semantic Web Services (ALPSWS2006)*, pages 39–54, 2006.
51. I. Horrocks, O. Kutz, and U. Sattler. The even more irresistible $\mathcal{SROIQ}$. In *Proceedings of the 10th International Conference of Knowledge Representation and Reasoning (KR-2006)*, pages 57–67, 2006.
52. I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. SWRL: A Semantic Web Rule Language Combining OWL and RuleML, May 2004. W3C Member Submission. `http://www.w3.org/Submission/SWRL/`.
53. I. Horrocks, U. Sattler, and S. Tobies. Practical Reasoning for Very Expressive Description Logics. *Logic Journal of the IGPL*, 8(3):239–264, 2000.
54. U. Hustadt, B. Motik, and U. Sattler. Reasoning in description logics by a reduction to disjunctive datalog. *J. Autom. Reasoning*, 39(3):351–384, 2007.
55. M. Kifer. Nonmonotonic reasoning in FLORA-2. In C. Baral, G. Greco, N. Leone, and G. Terracina, editors, *8th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'05)*, volume 3662 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2005.
56. M. Kifer, G. Lausen, and J. Wu. Logical foundations of object-oriented and frame-based languages. *Journal of the ACM*, 42(4):741–843, 1995.
57. M. Knorr, J. J. Alferes, and P. Hitzler. A well-founded semantics for hybrid mknf knowledge bases. In *Proceedings of the 2007 International Workshop on Description Logics (DL2007)*, pages 347–354, 2007.
58. N. Leone, G. Pfeifer, W. Faber, T. Eiter, G. Gottlob, S. Perri, and F. Scarcello. The dlv system for knowledge representation and reasoning. *ACM Trans. Comput. Log.*, 7(3):499–562, 2006.
59. A. Y. Levy and M.-C. Rousset. Combining Horn Rules and Description Logics in CARIN. *Artificial Intelligence*, 104(1-2):165–209, 1998.

60. V. Lifschitz. Nonmonotonic databases and epistemic queries. In *Proceedings IJCAI-91*, pages 381–386, 1991.
61. J. W. Lloyd. *Foundations of logic programming; (2nd extended ed.)*. Springer, New York, NY, USA, 1987.
62. J. W. Lloyd and R. W. Topor. Making prolog more expressive. *Journal of Logic Programming*, 1(3):225–240, 1984.
63. T. Lukasiewicz. Probabilistic description logic programs. *Int. J. Approx. Reasoning*, 45(2):288–307, 2007.
64. T. Lukasiewicz and U. Straccia. Description logic programs under probabilistic uncertainty and fuzzy vagueness. In *Symbolic and Quantitative Approaches to Reasoning with Uncertainty, 9th European Conference, ECSQARU*, pages 187–198, 2007.
65. A. Malhotra, J. Melton, and N. Walsh (eds.). XQuery 1.0 and XPath 2.0 Functions and Operators , Jan. 2007. W3C Recommendation, available at `http://www.w3.org/TR/xpath-functions/`.
66. D. Marin. A formalization of RDF. Technical Report TR/DCC-2006-8, TR Dept. Computer Science, Universidad de Chile, 2006.
67. B. Motik, I. Horrocks, R. Rosati, and U. Sattler. Can OWL and logic programming live together happily ever after? In *Proceedings ISWC-2006*, volume 4273 of *LNCS*, pages 501–514. Springer, 2006.
68. B. Motik and R. Rosati. Closing semantic web ontologies. Technical report, University of Manchester, March 2007. Available from `http://web.comlab.ox.ac.uk/oucl/work/boris.motik/publications/mr06closing-report.pdf`.
69. B. Motik and R. Rosati. A faithful integration of description logics with logic programming. In *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 477–482, 2007.
70. B. Motik, U. Sattler, and R. Studer. Query Answering for OWL-DL with Rules. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, 3(1):41–60, 2005.
71. M. Nilsson, A. Powell, P. Johnston, and A. Naeve. Expressing dublin core metadata using the resource description framework (rdf), Jan. 2008. DCMI Recommendation.
72. J. Z. Pan, E. Franconi, S. Tessaris, G. Stamou, V. Tzouvaras, L. Serafini, I. R. Horrocks, and B. Glimm. Specification of Coordination of Rule and Ontology Languages. Project Deliverable D2.5.1, KnowledgeWeb NoE, June 2004.
73. P. F. Patel-Schneider, P. Hayes, and I. Horrocks. OWL Web Ontology Language Semantics and Abstract Syntax, Feb. 2004. W3C Recommendation.
74. D. Pearce. Equilibrium logic. *Ann. Math. Artif. Intell.*, 47(1-2):3–41, 2006.
75. D. Pearce and A. Valverde. Quantfied equilibrium logic. Technical report, Universidad Rey Juan Carlos, 2006.
76. J. Pérez, M. Arenas, and C. Gutierrez. Semantics and complexity of sparql. In *International Semantic Web Conference (ISWC 2006)*, pages 30–43, 2006.
77. A. Polleres. From SPARQL to rules (and back). In *Proceedings of the 16th World Wide Web Conference (WWW2007)*, Banff, Canada, May 2007.
78. A. Polleres, C. Feier, and A. Harth. Rules with contextually scoped negation. In *Proceedings of the 3rd European Semantic Web Conference (ESWC2006)*, volume 4011 of *Lecture Notes in Computer Science* (*LNCS*), Budva, Montenegro, June 2006. Springer.
79. A. Polleres, C. Feier, and A. Harth. Rules with contextually scoped negation. In *Proceedings ESWC-2006*, volume 4011 of *LNCS*, pages 332–347. Springer, 2006.
80. A. Polleres, F. Scharffe, and R. Schindlauer. SPARQL++ for mapping between RDF vocabularies. In *OTM 2007, Part I : Proceedings of the 6th International Conference on Ontologies, DataBases, and Applications of Semantics (ODBASE 2007)*, volume 4803 of *Lecture Notes in Computer Science* (*LNCS*), pages 878–896, Vilamoura, Algarve, Portugal, Nov. 2007. Springer.

81. D. Poole. A Logical Framework for Default Reasoning. *Artificial Intelligence*, 36:27–47, 1988.
82. E. Prud'hommeaux and A. Seaborne (eds.). SPARQL Query Language for RDF, Jan. 2007. W3C Recommendation, available at `http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/`.
83. R. Reiter. On Closed-World Databases. In H. Gallaire and J. Minker, editors, *Logic and Data Bases*, pages 55–76. Plenum Press, New York, 1978.
84. R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.
85. R. Rosati. Towards Expressive KR Systems Integrating Datalog and Description Logics: Preliminary Report. In *Proceedings of the 1999 International Workshop on Description Logics* (*DL '99*), pages 160–164, 1999.
86. R. Rosati. On the Decidability and Complexity of Integrating Ontologies and Rules. *Journal of Web Semantics*, 3(1):61–73, 2005.
87. R. Rosati. Semantic and computational advantages of the safe integration of ontologies and rules. In *Principles and Practice of Semantic Web Reasoning, Third International Workshop, PPSWR 2005*, volume 3703 of *Lecture Notes in Computer Science*, pages 50–64. Springer, 2005.
88. R. Rosati. Integrating Ontologies and Rules: Semantic and Computational Issues. In P. Barahona, F. Bry, E. Franconi, U. Sattler, and N. 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, Sept. 2006.
89. R. Rosati. $\mathcal{DL}$+*log*: Tight Integration of Description Logics and Disjunctive Datalog. In *Proceedings of the Tenth International Conference on Principles of Knowledge Representation and Reasoning* (*KR 2006*), pages 68–78. AAAI Press, 2006.
90. M. Simkus and T. Eiter. FDNC: Decidable non-monotonic disjunctive logic programs with function symbols. In *Logic for Programming, Artificial Intelligence, and Reasoning, 14th International Conference, LPAR*, pages 514–530, 2007. Full paper Tech. Rep. INFSYS RR-1843-08-01, TU Vienna. `http://www.kr.tuwien.ac.at/research/reports/rr0801.pdf`.
91. M. Sintek and S. Decker. TRIPLE - A Query, Inference, and Transformation Language for the Semantic Web. In *Proceedings of the First International Semantic Web Conference* (*ISWC 2002*), volume 2342 of *Lecture Notes in Computer Science* (*LNCS*), pages 364–378, 2002.
92. E. Sirin, B. Parsia, B. Cuenca Grau, A. Kalyanpur, and Y. Katz. Pellet: A practical OWL-DL reasoner. Technical Report 68, UMIACS, University of Maryland, 2005.
93. U. Straccia. Reasoning about Uncertainty. In *Reasoning Web, Fourth International Summer School 2008, Tutorial Lectures*, LNCS. Springer, 2008. This volume.
94. Y. Sure, S. Bloehdorn, P. Haase, J. Hartmann, and D. Oberle. The SWRC ontology - semantic web for research communities. In C. Bento, A. Cardoso, and G. Dias, editors, *Proceedings of the 12th Portuguese Conference on Artificial Intelligence - Progress in Artificial Intelligence (EPIA 2005)*, volume 3803 of *LNCS*, pages 218 – 231, Covilha, Portugal, DEC 2005. Springer.
95. T. Swift. Deduction in ontologies via ASP. In *Proceedings LPNMR-2004*, volume 2923 of *LNCS*, pages 275–288, 2004.
96. H. 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.
97. D. Tsarkov and I. Horrocks. Fact++ Description Logic Reasoner: System Description. In *Proceedings of the Third International Joint Conference on Automated Reasoning* (*IJCAR 2006*), 2006.

98. J. D. Ullman. *Principles of Database & Knowledge Base Systems*. Comp. Science Press, 1989.
99. K. Van Belleghem, M. Denecker, and D. De Schreye. A strong correspondence between description logics and open logic programming. In *Proceedings ICLP-1997*, pages 346–360, 1997.
100. A. Van Gelder, K. A. Ross, and J. S. Schlipf. The Well-Founded Semantics for General Logic Programs. *Journal of the ACM*, 38(3):620–650, 1991.
101. W3C. The Resource Description Framework. `http://www.w3.org/RDF/`.
102. K. Wang, D. Billington, J. Blee, and G. Antoniou. Combining Description Logic and Defeasible Logic for the Semantic Web. In *Proceedings of the ISWC 2004 Workshop on Rules and Rule Markup Languages for the Semantic Web* (*RuleML 2004*), volume 3323 of *Lecture Notes in Computer Science* (*LNCS*), pages 170–181. Springer Verlag, 2004.
103. J. Wielemaker, G. Schreiber, and B. Wielinga. Prolog-based infrastructure for RDF: Scalability and performance. In D. Fensel, K. Sycara, and J. Mylopoulos, editors, *Proceedings ISWC-2003*, volume 2870 of *LNCS*, pages 644–658, Berlin, Germany, Oct. 2003. Springer.
104. Z. Wu, G. Eadon, S. Das, E. I. Chong, V. Kolovski, M. Annamalai, and J. Srinivasan. Implementing and Inference Engine for RDFS/OWL Constructs and User-Defined Rules in Oracle. In *Proccedings of ICDE 2008*. IEEE, 2008. To appear.
105. G. Yang and M. Kifer. Reasoning about anonymous resources and meta statements on the semantic web. *Journal on Data Semantics*, 1:69–97, 2003. Printed in LNCS Series, vol. 2800, Springer.