LOGICS FOR THE SEMANTIC WEB

Pascal Hitzler, Jens Lehmann, Axel Polleres

Reader: Patrick Hayes

1 INTRODUCTION

A major international research effort is currently under way to improve the existing World Wide Web (WWW), with the intention to create what is often called the Semantic Web [Berners-Lee et al., 2001; Hitzler et al., 2010]. Driven by the World Wide Web Consortium (W3C) and its director Sir Tim Berners-Lee (inventor of the WWW), and heavily funded by many national and international research funding agencies, Semantic Web has become an established field of research. It integrates methods and expertise from many subfields of Computer Science and Artificial Intelligence [Studer, 2006], and it has now reached sufficient maturity for first industrial scale applications [Hamby, 2012; Hermann, 2010]. Correspondingly, major IT companies are starting to roll out applications involving Semantic Web technologies; these include Apple’s Siri, IBM’s Watson system, Google’s Knowledge Graph, Facebook’s Open Graph Protocol, and schema.org as a collaboration between major search engine providers including Microsoft, Google, and Yahoo!.

The Semantic Web field is driven by the vision to develop powerful methods and technologies for the reuse and integration of information on the Web. While current information on the Web is mainly made for human consumption, it shall in the future be made available for automated processing by intelligent systems. This vision is based on the idea of describing the meaning—or semantics—of data on the Web using metadata—data that describes other data—in the form of so-called ontologies [Hitzler et al., 2010]. Ontologies are essentially knowledge bases represented using logic-based knowledge representation languages. This shall enable access to implicit knowledge through logical deduction [Hitzler and van Harmelen, 2010], and its use for search, integration, browsing, organization, and reuse of information.

Of course, the idea of adopting knowledge representation languages raises the question which of the many approaches discussed in the literature should be adopted and promoted to Web standards (officially called W3C Recommendations). In this chapter, we give an overview of the most important present standards as well as their origins and history.

The idea that the World Wide Web shall have capabilities to convey information for processing by intelligent systems, and not only by humans, has already been part of its original design [Berners-Lee, 1996]. The World Wide Web was initiated
in 1990, and immediately showed exponential growth [Berners-Lee, 1996]. In the meantime, it has become a very significant technological infrastructure of modern society.

In the 1990s, the Semantic Web vision\(^1\) was mainly driven by the W3C Metadata Activity [W3C Metadata, revision of 23 August 2002] which produced the first version of the Resource Description Framework (RDF) which we will discuss in Section 2. The Semantic Web Activity [W3C Semantic Web, revision of 19 June 2013] replaced the Metadata Activity in 2001, and has installed several standards for representing knowledge on the Web, most notably two revisions of RDF, the Web Ontology Language (OWL) discussed in Section 3, and the Rule Interchange Format (RIF) discussed in Section 4. In Section 5, we discuss some of the particular challenges which must be faced when adopting logic-based knowledge representation languages for the Semantic Web, and in Section 6 we discuss some of the more recent research developments and questions. Note that we give a more detailed technical account for RDF than for OWL and RIF, because the latter are closely related to description logics and logic programming, respectively, and the reader is referred to the corresponding chapters in this volume for additional background and introductions.

2 RDF AND RDF SCHEMA

The Resource Description Framework (RDF) [Manola et al., 2004; Hayes, 2004] comprises a simple data format as well as a basic schema language, called RDF Schema [Brickley and Guha, 2004]. While historically often termed a “mediadata” standard, that is, an exchange format for data about documents and resources, in the meantime, RDF has been well established as a universal data exchange format for classical data integration scenarios, and particularly for publishing and exchanging structured data on the Web [Polleres et al., 2011].

Informally, all RDF data can be understood as a set of subject–predicate–object triples, where all subjects and predicates are Uniform Resource Identifiers (URIs)\(^2\) [Berners-Lee et al., 2005], and in the object position both URIs and literal values (such as numbers, strings, etc.) are allowed. Such a simple, triple based format was chosen since on the one hand, it can accommodate for any kind of metadata in the form of predicate-value pairs, and on the other hand, any more complex relational or object-oriented data can be decomposed into such triples in a fairly straightforward manner [Berners-Lee, 2006].\(^3\)

Last, but not least, since URIs are being used as constant symbols in the language of RDF, any RDF triple may likewise be viewed as a generalization of

---

1 The term Semantic Web became popular in the aftermath of the widely cited popular science article [Berners-Lee et al., 2001]. We were able to trace the term Semantic, in relation to Web, back to a 1994 presentation by Tim Berners-Lee [Berners-Lee, 1994].

2 URIs are a generalization of URLs.

3 See also the discussion of semantic networks in the chapter on description logics, in this volume.
a “link” on the Web; as opposed to plain links on the traditional Web, on the Semantic Web, links can be associated with an arbitrary binary relation, which again is represented by a URI. We note that this idea of “typed” links was already part of Tim-Berners Lee’s original design ideas of the Web [Berners-Lee, 1993; Berners-Lee and Fischetti, 1999]. For instance, on the website of the W3C (http://www.w3.org), if one wants to state that the page with the URI http://www.w3.org/DesignIssues/TimBook-old/History.html was created by Tim Berners-Lee, this fact may be viewed as such a typed link, and consequently as an RDF subject-predicate-object triple, as shown in Fig. 1. Here, the URI dc:creator is used to denote the has-creator relation between a resource and its creator. This common view of typed links as labeled edges linking between resources also leads to sets of RDF triples often being called “RDF graphs.”

A distinguished relation within RDF, represented by the URI rdf:type is the is-a relation, that allows to denote membership to a certain class, where classes are again represented by URIs, for instance the class foaf:Person. Another important feature of RDF is that so called “blank nodes” can be used in the subject or object positions of triples to denote unnamed or unknown resources. This allows to model incomplete information in the form of existentials. For instance, the RDF graph in Fig. 2 extends the information in Fig. 1 by the fact that Tim Berners-Lee is a Person, is named “Timothy Berners-Lee” and knows some person named “Dan Brickley”.

In the following, after giving a brief history of the RDF standard (Section 2.1), we will present the RDF Data model along with a short discussion of different syntactic representations and the semantics of RDF (Section 2.2). We continue with a discussion of RDF Schema in Section 2.3 and the query language SPARQL in Section 2.4.
2.1 Brief History of RDF

The standardisation of RDF has been preceded by two earlier proposals for metadata standards in the form of W3C member submissions, namely (i) the Channel Definition Format [Ellerman, 1997] and (ii) the Meta Content Framework (MCF) [Guha and Bray, 1997]. While the former comprised an XML format with a fixed term of metadata properties for describing information channels on the Web, (somewhat similar to RSS nowadays), the latter (MCF) was strictly extensible and evolved into the first version of RDF [Lassila and Swick, 1999], published in 1999.

Another important metadata initiative from the digital libraries community, Dublin Core [Nilsson et al., 2008], which started around the same time but outside of W3C, later on adopted RDF as a representation syntax, becoming one of the most prominent RDF vocabularies, see Section 2.3 below.

The first official standard recommendation of RDF from 1999 was extended in 2004 by a formal definition of the Semantics of RDF [Hayes, 2004], decoupling the syntactical representation in XML from the RDF data model.

Since then RDF has been used in various contexts and experienced wide adoption. In 2009 the W3C held a workshop on future directions of RDF [Herman, 2009], discussing several extensions but also simplifications of the standard. These extensions are currently under discussion in the ongoing W3C RDF 1.1 working group.4

4See http://www.w3.org/2011/rdf-wg/.
2.2 Different Syntactic Representations and Semantics

There are various serialisations for RDF. Fig. 3 shows different syntactic representations of the six triples in the RDF graph from Fig. 2 in some of these serialisation syntaxes: N-Triples [Beckett and McBride, 2004a], cf. Fig. 3(a) is a simple line-based format that serialises one RDF triple per line terminating each line triple with a full-stop ‘.’, enclosing URIs in angle brackets, and literals in quotes; blank nodes are given alphanumeric identifiers (also called blank node label, preceded by the prefix ‘_:’). Turtle [Beckett and Berners-Lee, 2011], shown in Fig. 3(b) extends the simple N-Triples format by shortcuts making the language more legible, such as namespace prefix and base URI declarations, similar to XML, for abbreviating URIs, as well as the possibility to separate predicate-object groups for triples with the same subject by semicolon, etc. The original RDF/XML [Beckett and McBride, 2004b] syntax was an XML format, that encoded predicates as XML elements, with some abbreviations, such as rdf:type triples that refer to class membership of a node can be also directly encoded as XML elements, an example of which is given in Fig. 3(c).

Other serialization syntaxes for RDF, which do not detail herein, include the RDFa [Herman et al., 2013], which provides means to syntactically embed RDF directly as markup into (X)HTML documents. We note that RDFa is particularly similar – and in fact intertranslatable – to other metadata markup formats in HTML such as the increasingly popular microdata format [Hickson, 2012] (which is actively promoted by schema.org).

The fundamental difference between RDF and general XML is that the intuition of the RDF data model is that the syntactic representation, and also the order of triples is irrelevant, which intuitively implies a notion of equivalence between RDF graphs that is independent of the serialization. While this RDF data model was not formally described in the 1999 version of RDF [Lassila and Swick, 1999], formal definitions were introduced in the specification of 2004, where a formal model-theoretic semantics was defined [Hayes, 2004]; and later refined in the recent RDF1.1 specification [Hayes and Patel-Schneider, 2014]. For the exposition of this formal semantics in this chapter, we will stick with a notation similar to the one introduced in [Gutiérrez et al., 2004] rather than quoting the original W3C specification verbatim.

DEFINITION 1. The set of RDF terms $U \cup L \cup B$ consists of elements from three infinite disjoint sets $U$ (RDF URI references), $L$ (RDF Literals), and $B = \{b_j : j \in \mathbb{N}\}$ (RDF blank nodes).

Note that for the exposition herein we restrict literals to plain string literals, in general RDF also offers language tagged literals, as well as so called “typed” literals, that is, pairs $(l,d)$ where $l \in L$ is a string, and $d$ is either a string language tag [Phillips and Davis, 2006], or, respectively, $d \in D$ is a URI representing a datatype (such as e.g. http://www.w3.org/2001/XMLSchema#decimal), see also [Biron and Malhotra, 2004].
<http://www.w3.org/DesignIssues/TimBook-old/History.html> 
  <http://purl.org/dc/elements/1.1/creator> 
  <http://www.w3.org/People/Berners-Lee/card#i> . 
<http://www.w3.org/People/Berners-Lee/card#i> 
  <http://xmlns.com/foaf/0.1/name> 
  "Timothy Berners-Lee" . 
<http://www.w3.org/People/Berners-Lee/card#i> 
  <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> 
  <http://xmlns.com/foaf/0.1/Person> . 
<http://www.w3.org/People/Berners-Lee/card#i> 
  <http://xmlns.com/foaf/0.1/known> 
    _:b1 . 
    _:b1 
      <http://xmlns.com/foaf/0.1/name> 
      "Dan Brickley" . 
    _:b1 
      <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> 
      <http://xmlns.com/foaf/0.1/Person> . 
(a)

@base <http://www.w3.org/> . 
@prefix dc: <http://purl.org/dc/elements/1.1/> . 
@prefix foaf: <http://xmlns.com/foaf/0.1/> . 
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> . 
@prefix rdfs: <http://http://www.w3.org/2000/01/rdf-schema#> . 
<DesignIssues/TimBook-old/History.html> dc:creator <People/Berners-Lee/card#i> . 
<People/Berners-Lee/card#i> foaf:name "Timothy Berners-Lee" ; 
  rdf:type foaf:Person ; 
  foaf:knows [ foaf:name "Dan Brickley" ; 
    rdf:type foaf:Person ] . 
(b)

<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:foaf="http://xmlns.com/foaf/0.1/"
    xmlns:rdfs="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <rdf:Description rdf:about="http://www.w3.org/DesignIssues/TimBook-old/History.html"> 
      <dc:creator rdf:resource="http://www.w3.org/People/Berners-Lee/card#i"/> 
    </rdf:Description>
    <foaf:Person rdf:about="http://www.w3.org/People/Berners-Lee/card#i"> 
      <foaf:name>Timothy Berners-Lee</foaf:name> 
      <foaf:knows> 
        <foaf:Person> 
          <foaf:name>Dan Brickley</foaf:name> 
        </foaf:Person> 
      </foaf:knows> 
    </foaf:Person>
</rdf:RDF>

(c)

Figure 3. An RDF Graph in N-Triples, Turtle, and RDF/XML syntax
Figure 4. The RDF Graph from Fig. 2 in first-order logic using ternary encoding with an auxiliary predicate \textit{triple}, unary/binary encoding, and F-Logic-style frames.
DEFINITION 2. A triple \((s, p, o) \in (U \cup B) \times U \times (U \cup B \cup L)\) is called an RDF triple, where\(s\) is called the subject, \(p\) the predicate and \(o\) the object.

An RDF graph (or, just graph) is a set of RDF triples. A (proper) subgraph is a (proper) subset of a graph. The universe of a graph \(G\), \(\text{universe}(G)\), is the set of elements of \(U \cup B \cup L\) that occur in the triples of \(G\). The vocabulary \(V_G\) of a graph \(G\) is the set \(\text{universe}(G) \cap (U \cup L)\).

Finally a triple, or graph, respectively, is called ground, if it does not contain any blank nodes.

The intention of blank nodes in RDF suggests that graphs that only differ in the identifiers used for blank nodes in a concrete syntactical representation should be considered equivalent. Likewise, a graph \(G_1\) that can be turned into a subgraph \(G_2\) of by just renaming blank nodes in \(G_1\) to RDF terms from the universe of \(G_2\), does not carry more information than \(G_2\) and should thus be considered “entailed” by \(G_2\). This intention is reflected in the formal model-theoretic semantics of RDF that is – in principle – based on the usual idea of first-order interpretations, with the caveat that elements of \(U\) both reflect binary relations and constants at the same time. This leads to a somewhat non-standard definition of interpretations in RDF.

DEFINITION 3. (from [Hayes, 2004, Section 1], slightly simplified.) A simple RDF interpretation \(I = \langle \Delta, \Delta_p, I^{\text{EXT}}, I^{\text{Terms}}, L_V \rangle\) over an RDF vocabulary \(V\) consists of

- a non-empty domain \(\Delta\), i.e. the set of resources, which contains the set \(L_V = L \cap V\).
- a non-empty set of properties \(\Delta_P\), not necessarily disjoint with \(\Delta\)
- a function \(I^{\text{EXT}} : \Delta_P \rightarrow \Delta \times \Delta\), which maps properties to binary relations
- a function \(I^{UL} : U_V \cup L_V \rightarrow \Delta\), where \(I^{UL}(l) = l\) for \(l \in L_V\) (that is, literals are interpreted as themselves)

Finally, satisfaction of an RDF triple, or graph, respectively, under an interpretation \(I\) is defined as follows.

DEFINITION 4. An interpretation \(I\) satisfies a ground triple triple \(t = (s, p, o)\), written \(I \models t\) if \(s, p, o \in V\), \(I^{\text{Terms}}(p) \in \Delta_P\) and \((I^{\text{Terms}}(s), I^{\text{Terms}}(o)) \in I^{\text{EXT}}(I^{\text{Terms}}(p))\). Accordingly, a ground RDF graph \(G\) is satisfied under \(I\), written \(I \models G\) if and only if \(I \models t\) for all \(t \in G\). Finally, a non ground graph \(G'\) is satisfied under \(I\) if there exists an extension \([I^{UL} + A]\) of \(I^{UL}\) by an assignment \(A : B \rightarrow \Delta\), such that

\[
([I^{\text{Terms}} + A](s), [I^{\text{Terms}} + A](o)) \in I^{\text{EXT}}(I^{\text{Terms}}(p))
\]

\[5\]As mentioned above, as opposed to [Hayes, 2004] we do not consider typed literals nor language tagged literals here, but only plain string literals.
for all $t \in G$.

We note that when looking at Definition 4 that the semantics of blank nodes corresponds exactly to that of existential variables in first-order logic.

Simple entailment between (sets of) RDF graphs is then defined following usual terminology.

**DEFINITION 5.** Given a set $S$ of RDF graphs (simply) entails a graph $G$, written $S \models G$, if every interpretation which satisfies every member of $S$ also satisfies $G$.

Given the intention outlined above, entailment should also be expressible in terms of blank node mappings.

**DEFINITION 6.** Here, A map is a function $\mu : U \cup B \cup L \rightarrow UBL$ preserving URIs and literals, i.e., $\mu(u) = u$ and $\mu(l) = l$ for all $u \in U$ and $l \in L$.

Using such maps, indeed the notion of entailment between two RDF graphs can be defined via the so-called interpolation lemma from [Hayes, 2004, Section 2], rather than in a model-theoretic way.

**LEMMA 7 Interpolation Lemma.** Let $G_1$, $G_2$ be RDF graphs, then $G_1 \models G_2$ if a subgraph of $G_1$ is an instance of $G_2$, that is, if there exists a map $\mu$, such that $\mu(G_2)$ is a subgraph of $G_1$.

Given $G_1$, $G_2$, deciding whether there exists such a map, boils down to graph homomorphism, which is well known to be an NP-complete problem [Garey and Johnson, 1979], and therefore also NP-completeness of simple RDF entailment follows. Fragments of RDF where entailment is tractable include obviously ground graphs, but also graphs where blank nodes are not used in a cyclic fashion across triples [Pichler et al., 2008].

Obviously, due to this existential semantics of blank nodes there could be inner redundancy in an RDF graph, that is, if there is a homomorphism of $G$ to itself. This redundancy is called non-leaness in RDF terminology.

**DEFINITION 8.** A graph $G$ is lean if there is no map $\mu$ such that $\mu(G)$ is a proper subgraph of $G$.

Unfortunately, as a consequence of the NP-completeness of simple entailment, deciding leanness is also intractable, namely coNP-complete [Gutiérrez et al., 2004].

As a side note, let us note that it has often been critizized by practitioners that the existential treatment of blank nodes, which leads to this high complexity, puts an unnecessary burden on RDF users and implementers, and moreover is not consistently followed in neighbouring standards that build on top of RDF [Mallea et al., 2011].

**Relation of RDF to other Logical Formalisms**

Another way to show NP-completeness of RDF simple entailment is that RDF entailment can straightforwardly be encoded into entailment of first-order-logic
formulae with existentials and conjunction only, which is well known to be just another formulation of conjunctive query containment [Chandra and Merlin, 1977], as shown in the following theorem, which is implicit in [de Bruijn and Heymans, 2007].

**THEOREM 9.** Given RDF graphs $G_1$ and $G_2$, we have that $G_1 \models G_2$ if and only if $\mathcal{T}(G_1) \models_{\text{FOL}} \mathcal{T}(G_2)$ where a first order theory $\mathcal{T}(G)$ is obtained from a graph $G$ as follows

$$\mathcal{T}(G) = \exists x \in V_G \cap B x \land \bigwedge_{(s,p,o) \in G} \text{triple}(s,p,o)$$

An example for this encoding into first-order logic is shown in Fig.4(a); Another common way to encode RDF into first-order logic is using unary predicates for triples modeling an is-a relationship, i.e. rdf:type triples, and binary predicates for all other properties, cf. Fig. 4(b). We note though that this representation is of somewhat limited use to encode arbitrary RDF graphs, since for instance blank nodes in the object positions of rdf:type triples, which is perfectly fine in the general setting of RDF, would result in a second-order formula.

Note that, translation to a first-order setting in [de Bruijn and Heymans, 2007] uses F-Logic [Kifer et al., 1995] instead of classical first-order logic, which may be considered as syntactic sugar; a respective encoding of RDF triples in F-Logic frame syntax is shown in 4(c).

### 2.3 RDF Schema (RDFS)

The generic semantics defined by simple RDF interpretations is restricted to interpretations that give a special meaning to the RDF and RDFS vocabulary, that is, for URIs in the rdf: (http://www.w3.org/1999/02/22-rdf-syntax-ns#) and rdfs: (http://www.w3.org/2000/01/rdf-schema#) namespaces, cf. the respective prefix declarations in Fig 3(b) for the full URIs. This special semantics allows to express simple ontologies, in the form of (i) defining subclass and subproperty hierarchies, and (ii) defining domain and range restrictions of properties.

The RDFS semantics restricts interpretations as per Def. 3 above such that (i) a set of axiomatic triples, cf. [Hayes, 2004, Sections 3.1 and 4.1] are true in any RDFS interpretation, and (ii) a set of entailment rules holds, that affect how rdfs:subClassOf, rdfs:subPropertyOf, rdfs:domain, rdfs:range, etc. triples are interpreted. Figure 5 shows some of the RDFS axiomatic triples. Further, with the encoding of RDF Graphs into first-order logic from Fig. 4(a) in mind, the RDFS entailment rules can to a large extent be approximated by the first-order rules shown in Table 1 (from [Eiter et al., 2008b]). The fact that these rules are simple Horn rules and that RDF is encodable as a set of facts makes reasoning in RDFS thus amenable to simple (Datalog) rule engines. Muñoz et al. [Muñoz et al., 2007] have argued for a simpler set of entailment rules that leaves out inferences that might be considered redundant for many applications of RDF and RDFS, for instance leaving out axiomatic triples or rules like the first seven rules in Table 1.
The RDFS entailment rules, written as first-order Horn rules:

\[
\forall S, P, O \ (\text{triple}(S, P, O) \supset \text{triple}(S, \text{rdf:type}, \text{rdfs:Resource}))
\]
\[
\forall S, P, O \ (\text{triple}(S, P, O) \supset \text{triple}(P, \text{rdf:type}, \text{rdf:Property}))
\]
\[
\forall S, P, O \ (\text{triple}(S, P, O) \supset \text{triple}(O, \text{rdf:type}, \text{rdfs:Resource}))
\]
\[
\forall C \ (\text{triple}(C, \text{rdf:type}, \text{rdfs:Class}) \supset \text{triple}(C, \text{rdfs:subClassOf}, \text{rdfs:Resource}))
\]
\[
\forall S, C \ (\text{triple}(S, \text{rdf:type}, C) \supset \text{triple}(C, \text{rdf:type}, \text{rdfs:Class}))
\]
\[
\forall C \ (\text{triple}(C, \text{rdf:type}, \text{rdfs:Class}) \supset \text{triple}(C, \text{rdfs:subClassOf}, C))
\]
\[
\forall P \ (\text{triple}(P, \text{rdf:type}, \text{rdf:Property}) \supset \text{triple}(P, \text{rdfs:subPropertyOf}, P))
\]
\[
\forall S, P, O \ (\text{triple}(S, P, O) \land \text{triple}(P, \text{rdfs:range}, C) \supset \text{triple}(C, \text{rdf:type}, C))
\]
\[
\forall C_1, C_2, C_3 \ (\text{triple}(C_1, \text{rdfs:subClassOf}, C_2) \land \text{triple}(C_2, \text{rdfs:subClassOf}, C_3) \supset \text{triple}(C_1, \text{rdfs:subClassOf}, C_3))
\]
\[
\forall S, C_1, C_2 \ (\text{triple}(S, \text{rdf:type}, C_1) \land \text{triple}(C_1, \text{rdfs:subClassOf}, C_2) \supset \text{triple}(S, \text{rdf:type}, C_2))
\]
\[
\forall P_1, P_2, P_3 \ (\text{triple}(P_1, \text{rdfs:subPropertyOf}, P_2) \land \text{triple}(P_2, \text{rdfs:subPropertyOf}, P_3) \supset \text{triple}(P_1, \text{rdfs:subPropertyOf}, P_3))
\]
\[
\forall S, P_1, P_2, O \ (\text{triple}(S, P_1, O) \land \text{triple}(P_1, \text{rdfs:subPropertyOf}, P_2) \supset \text{triple}(S, P_2, O))
\]

Figure 5. Some of the axiomatic triples that are true under the RDFS semantics.

Table 1. RDFS entailment rules, written as first-order Horn rules

By giving a special semantics to the rdfs: vocabulary, RDF Schema enables the meta-description of other RDF vocabularies with the goal that additional triples will be entailed under the RDFS semantics. As such RDFS can be considered a simple ontology language, that is used within popular Web ontologies such as the Friend-of-a-friend (foaf:) or the Dublin Core (dc:) vocabulary descriptions [Brickley and Miller, 2007; Nilsson et al., 2008]. Fig. 6 shows a snippet of the FOAF ontology, along with some additional triples that can be derived from this RDFS ontology together with the RDF graph from Fig. 3.

Relation of RDFS to other Logical Formalisms

A mapping from RDF interpretations to first-order logics is given in [Franconi et al., 2005]. This picture is completed in [de Bruijn and Heymans, 2007], embedding RDF(S) within the framework of F-Logic [Kifer et al., 1995], and also covering the extensional semantics of RDFS [Hayes, 2004, Section 4.2]; additional considerations regarding special semantics of datatype literals (cf. [Hayes,
foaf:Person rdfs:subClassOf foaf:Agent .
foaf:knows rdfs:domain foaf:Person .
foaf:knows rdfs:range foaf:Person .
foaf:name rdfs:subPropertyOf rdfs:label .
...

(a)

<Person/Berners-Lee/card#i> rdf:type foaf:Agent .
<Person/Berners-Lee/card#i> rdfs:label "Timothy Berners-Lee" .
_:b1 rdf:type foaf:Agent .
_:b1 rdfs:label "Dan Brickley" .

(b)

Figure 6. (a) Some RDFS statements from the FOAF vocabulary description, plus (b) some additional triples inferrable from these statements together with the RDF graph from Fig. 3

2004, Section 5]) are covered in [de Bruijn and Heymans, 2010]. Another recent paper by Franconi et al. [Franconi et al., 2013] discusses the logic of the extensional RDF semantics, which is only a non-normative part of the RDF specification, in more detail; remarkably, the authors come to the conclusion that the extensional RDFS semantics can likewise be implemented by a set of inference rules, where the closure is computable in polynomial time in a forward-chaining manner, thus contradicting the conjecture in the original RDF specification that the extensional semantics would “require more complex inference rules” [Hayes, 2004, Section 4.2] The core semantics of RDFS, that is reasoning about rdfs:subClassOf, rdfs:subPropertyOf, rdfs:domain, and rdfs:range has also been associated in the literature with a minimalistic fragment of the Description Logics family DL-Lite [Calvanese et al., 2007; Poggi et al., 2008; Franconi et al., 2013]; a respective mapping from RDFS statements to DL-Lite is shown in Table 2. We note though that only restricted RDF graphs, that do not use the RDF and RDFS vocabulary in a “non-standard” [de Bruijn and Heymans, 2007] fashion (e.g., using rdfs:subclass in an object position, or – as already mentioned above – blank nodes in the object position of rdf:type triples) are amenable to such an embedding into DL-Lite.

2.4 SPARQL

In order to facilitate queries over RDF and RDFS, the W3C has defined a standard query language, SPARQL [Harris and Seaborne, 2013], which at its core facilitates conjunctive queries over RDF graphs. Such conjunctive queries are called Basic Graph Patterns (BGP) in SPARQL and syntactically expressed as RDF graphs with (‘?’-prefixed) variables allowed in subject, predicate or object positions of
Table 2. DL-Lite axioms vs. RDF(S)

<table>
<thead>
<tr>
<th>DL-Lite</th>
<th>RDFS</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_1 \sqsubseteq A_2$</td>
<td>$A_1 \text{rdfs:subClassOf } A_2$.</td>
</tr>
<tr>
<td>$\exists P \subseteq A$</td>
<td>$P \text{rdfs:domain } A$.</td>
</tr>
<tr>
<td>$\exists P^- \subseteq A$</td>
<td>$P \text{rdfs:range } A$.</td>
</tr>
<tr>
<td>$P_1 \sqsubseteq P_2$</td>
<td>$P_1 \text{rdfs:subPropertyOf } P_2$.</td>
</tr>
</tbody>
</table>

$A(x)$ \quad $x \text{rdf:type } A$.  
$P(x,y)$ \quad $x \; P \; y$.  

SELECT ?X WHERE {  
  ?Y foaf:name "Dan Brickley" }  

answer(x) \rightarrow  
triple(x, type, Person) \land triple(x, knows, y) \land triple(y, name, "Dan Brickley")

Figure 7. A simple SPARQL query asking for persons who know someone named “Dan Brickley” and its corresponding conjunctive query in first-order syntax

SPARQL allows more complex patterns on top of BGPs, such as unions of patterns, optional query patterns and filters, where the expressivity of the SPARQL in its version 1.0 [Prud’hommeaux, 2008] language was shown to capture Relational Algebra, or non-recursive Datalog with negation, respectively, by Angles and Gutierrez [Angles and Gutierrez, 2008].

We note that the recent SPARQL 1.1 [Harris and Seaborne, 2013] specification has additional expressivity e.g. by allowing aggregates plus a basic form of regular path queries, which can no longer be captured in non-recursive Datalog with negation alone [Polleres and Wallner, 2013]. Moreover, SPARQL in its version 1.0 was solely defined in terms of RDF simple entailment, SPARQL 1.1 defines which additional answers a SPARQL query should return under RDFS and OWL semantics (see Section 3 below).

3 DAML/OIL AND OWL

3.1 A Brief History

While the previously described RDF and RDFS languages already allow to model domain knowledge, they are not very expressive and often insufficient for capturing the necessary relationships and constraints. Therefore, the development of richer representations was an early goal in the Semantic Web initiative, which eventually
led to the OWL ontology language.

One of the main predecessors of OWL are frame based systems. While the notion of frames was previously introduced in different contexts, e.g. [Minsky, 1975], a major development were the structured inheritance networks developed at the end of the 70s in [Brachman, 1978]. In those systems, the core modelling structures are frames – now more commonly referred to as classes in object oriented programming languages and ontology languages. Frames usually had specific attributes (also called properties) describing them. This differs from the previously described RDFS language and OWL itself in which properties are autonomous entities. Using domain and range axioms, RDFS properties can be used to model frame-like structures. Another consequence of properties being autonomous entities, is that their usage in RDFS and OWL is not restricted to instances of a single class/frame.

While early frame based systems lacked formal semantics, this deficit was overcome by the introduction of description logics (DLs). The first DL-based KR system is KL-ONE [Brachman and Schmolze, 1985]. We refrain from describing those in detail as they are already covered in the chapter on description logics in this volume. In contrast to some frame-based systems, DLs have a clear focus on logic based semantics and reasoning, which are now considered essential for an ontology language [Baader, 2003]. DLs later became the formal foundation of the OWL ontology language and enjoy an increase in popularity and usage. OWL and the underlying DLs go far beyond early frame based systems and RDF, e.g. it supports specific characteristics like functionality and transitivity for properties as well as complex class expressions.

A major turning point after the gradual introduction of some key technologies like frames and description logics more than 30 years ago was the rise of the World Wide Web in the 90s. Web technologies, e.g. XML, had a major influence on OWL. After the introduction of a first RDF recommendation in 1999, standardisation efforts on the introduction of an ontological layer in the Semantic Web intensified while a new version of RDF was developed in parallel. Ultimately, this resulted in the Web Ontology Language OWL becoming an official W3C recommendation in 2004, which was published together with the revised RDF W3C recommendation. The predecessors of OWL are DAML, OIL and to a lesser extent SHOE. SHOE is an extension of HTML, which was developed around 1996 and makes it possible to annotate web documents with machine-readable information. While the project is no longer active, it influenced the development of the DAML+OIL language. DAML (DARPA Agent Markup Language) was a funding program in the US, which started in 1999 and involved James Hendler and Tim Bernes-Lee. The program pursued the development of machine readable knowledge representation languages for the web. A main result of the DAML program was the DAML language, which was already based on RDF. In parallel to the development of DAML, the aim of OIL (Ontology Inference Layer) was to provide an infrastructure for the Semantic Web [Fensel et al., 2001]. The authors of [Fensel et al., 2001]

6http://www.cs.umd.edu/projects/plus/SHOE/
Logics for the Semantic Web

state that RDFS served as starting point for OIL and they developed it into a "full-fledged Web-based ontology language" including formal semantics. OIL development started at the end of the 90s. Finally, in December 2000 the first version of the language DAML+OIL [McGuinness et al., 2002] was released. While using DAML as a foundation, this language focused on the inclusion of the clear semantics underlying OIL. It also used the expressive power of OIL, specifically the S$HT\mathcal{Q}$ description logic [Horrocks et al., 2003]. DAML+OIL was subsequently used as starting point for the W3C Web Ontology Working Group. In 2004, this working group produced the W3C recommendation OWL – Web Ontology Language [Web Ontology Working Group, 10 February 2004]. Since then OWL served as a backbone for knowledge representation on the web and several inference algorithms for it, in particular for OWL DL, were developed and implemented. As the people involved in the development of DAML, OIL and OWL often had different technological backgrounds, ideas and expertise than those working on the RDF specifications, joining those two strands was a tense and difficult process. This is one of the reasons why different OWL dialects were created with varying compatibility with RDF.

In 2009, after several years of refinements, OWL 2 became a W3C recommendation [OWL Working Group, 27 October 2009] (note that the 2012 version of the recommendation document [Hitzler et al., 2012] contains only very minor changes, most of them editorial). OWL 2 was started as an incremental improvement of OWL, but during the development of the language, it turned out that in sum the required changes and addressed deficiencies are substantial: "None of these problems are severe, but, taken together, they indicate a need for a revision of OWL 1" [Grau et al., 2008]. From a knowledge representation perspective, OWL 2 mainly builds on \textit{SROIQ}(D), whereas OWL 1 mainly used \textit{SHOIN}(D) – see the description logic chapter in this volume for details.

3.2 Quick Introduction to OWL

Based on the introduction of description logics in the corresponding chapter in this volume, we will now describe the Web Ontology Language OWL. For brevity, we focus on the OWL DL dialect. In essence, OWL DL is based on description logics extended by several features to make it suitable as a web ontology language, e.g. using URIs/IRIs as identifiers, imports of other ontologies and annotations of URIs and axioms. By basing OWL DL on description logics, it can make use of the theory developed for DLs, in particular sophisticated reasoning algorithms. In OWL, different naming conventions are usually used compared to description logics. OWL classes correspond to concepts in description logics and properties correspond to roles.

While being based on description logics, OWL is also seen as a language extending RDF in the Semantic Web layer cake. However, the semantics of RDF differs from that of description logics and does in general not necessarily lead to the same logical consequences. Due to being based on RDF and DLs, there are
two different definitions of formal semantics: Direct Semantics, which are based on DLs, and RDF-based Semantics.

In general, OWL offers more convenience constructs than the corresponding description logics, but does not extend its expressivity. For instance, the domain and range constructs inherited from RDF are logically redundant, i.e. can be expressed using other constructs, but are part of the language, since they simplify modelling for knowledge engineers.

It should be noted that OWL does not make the unique name assumption, so different individuals can be mapped to the same domain element. It allows us to express equality and inequality between individuals \((a = b, a \neq b)\) using \texttt{owl:sameAs} and \texttt{owl:differentFrom}. Most algorithms for description logics already supported this distinction before the OWL specification was created. Not

<table>
<thead>
<tr>
<th>OWL expression / axiom</th>
<th>OWL 2</th>
<th>DL syntax</th>
<th>Manchester syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thing</td>
<td>⊤</td>
<td>Thing</td>
<td></td>
</tr>
<tr>
<td>Nothing</td>
<td>⊥</td>
<td>Nothing</td>
<td></td>
</tr>
<tr>
<td>intersectionOf</td>
<td>(C_1 \sqcap \ldots \sqcap C_n)</td>
<td>(C_1) and \ldots and (C_n)</td>
<td></td>
</tr>
<tr>
<td>unionOf</td>
<td>(C_1 \sqcup \ldots \sqcup C_n)</td>
<td>(C_1) or \ldots or (C_n)</td>
<td></td>
</tr>
<tr>
<td>complementOf</td>
<td>(\lnot C)</td>
<td>not (C)</td>
<td></td>
</tr>
<tr>
<td>oneOf</td>
<td>({x_1} \sqcup \ldots \sqcup {x_n})</td>
<td>({x_1, \ldots, x_n})</td>
<td></td>
</tr>
<tr>
<td>allValuesFrom</td>
<td>(\forall r.C)</td>
<td>(r) only (C)</td>
<td></td>
</tr>
<tr>
<td>someValuesFrom</td>
<td>(\exists r.C)</td>
<td>(r) some (C)</td>
<td></td>
</tr>
<tr>
<td>maxCardinality</td>
<td>((✓))</td>
<td>(\leq n r C)</td>
<td>(r) max (n)</td>
</tr>
<tr>
<td>minCardinality</td>
<td>((✓))</td>
<td>(\geq n r C)</td>
<td>(r) min (n)</td>
</tr>
<tr>
<td>cardinality</td>
<td>((✓))</td>
<td>(\leq n r C \sqcap \geq n r C)</td>
<td>(r) exact (n) (C)</td>
</tr>
<tr>
<td>hasSelf</td>
<td>(✓)</td>
<td>(\exists s.\text{Self})</td>
<td>(r) Self</td>
</tr>
<tr>
<td>subClassOf</td>
<td>(C_1 \sqsubseteq C_2)</td>
<td>(C_1) SubClassOf: (C_2)</td>
<td></td>
</tr>
<tr>
<td>equivalentClass</td>
<td>(C_1 \equiv C_2)</td>
<td>(C_1) EquivalentTo: (C_2)</td>
<td></td>
</tr>
<tr>
<td>disjointWith</td>
<td>(C_1 \equiv \lnot C_2)</td>
<td>(C_1) DisjointWith: (C_2)</td>
<td></td>
</tr>
<tr>
<td>sameAs</td>
<td>({x_1} \equiv {x_2})</td>
<td>(x_1) SameAs: (x_2)</td>
<td></td>
</tr>
<tr>
<td>differentFrom</td>
<td>({x_1} \sqsubseteq \lnot{x_2})</td>
<td>(x_1) DifferentFrom: (x_2)</td>
<td></td>
</tr>
<tr>
<td>domain</td>
<td>(\forall r.\top \subseteq C)</td>
<td>(r) Domain: (C)</td>
<td></td>
</tr>
<tr>
<td>range</td>
<td>(\top \subseteq \forall r.C)</td>
<td>(r) Range: (C)</td>
<td></td>
</tr>
<tr>
<td>subPropertyOf</td>
<td>(r_1 \sqsubseteq r_2)</td>
<td>(r_1) SubPropertyOf: (r_2)</td>
<td></td>
</tr>
<tr>
<td>equivalentProperty</td>
<td>(r_1 \equiv r_2)</td>
<td>(r_1) EquivalentTo: (r_2)</td>
<td></td>
</tr>
<tr>
<td>inverseOf</td>
<td>(r_1 \equiv r_2^\circ)</td>
<td>(r_1) InverseOf: (r_2)</td>
<td></td>
</tr>
<tr>
<td>TransitiveProperty</td>
<td>(r^+ \subseteq r)</td>
<td>(r) Characteristics: Transitive</td>
<td></td>
</tr>
<tr>
<td>FunctionalProperty</td>
<td>(\top \subseteq \leq 1 r)</td>
<td>(r) Characteristics: Functional</td>
<td></td>
</tr>
<tr>
<td>ReflexiveProperty</td>
<td>(✓)</td>
<td>(\text{Ref}(r))</td>
<td>(r) Characteristics: Reflexive</td>
</tr>
<tr>
<td>propertyChainAxiom</td>
<td>(✓)</td>
<td>(r_1 \circ \ldots \circ r_n \subseteq s)</td>
<td>(s) SubPropertyChain: (r_1 \circ \ldots \circ r_n)</td>
</tr>
</tbody>
</table>

Table 3. OWL constructs in DL and Manchester OWL syntax (excerpt). \(✓\) indicates that the construct is only available in OWL 2 and \((✓)\) indicates that it was extended in OWL 2.
making the unique names assumption is crucial in the Semantic Web, where it is often the case that many knowledge bases contain information about the same entity. In this case, a common approach is that each knowledge base uses their own URI namespace and \texttt{owl:sameAs} is used to connect individuals.

Table 3 shows, for some examples, how constructs in OWL can be mapped to description logics. We can see that some features can be mapped directly to description logics, e.g. union, and others are syntactic sugar, e.g. functional properties. OWL has different syntactic formats, in which a knowledge base can be stored. Since it can be converted to RDF, formats like RDF/XML or Turtle can be used. There is also a special XML syntax called OWL/XML as well as the Manchester OWL Syntax. For details on Manchester OWL syntax (e.g. used in the Protégé editor) see [Horridge and Patel-Schneider, 2008] and the OWL 2 Manchester Syntax Specification [OWL Working Group, 11 December 2012]. The latter is popular in ontology editors. However, the RDF-based syntax plays a special role since its support is required for tools to be compliant with the OWL standard. Examples for Manchester OWL Syntax are shown on the right column in Table 3. Note that OWL 2 also supports the creation of datatypes as discussed in detail in [Motik and Horrocks, 2008], but a discussion of them is omitted for brevity in this section.

### 3.3 Relations to other Formalisms

OWL 1 comes in three flavors: OWL Lite, OWL DL, and OWL Full. For OWL 1, OWL Lite corresponds to the description logic $\mathcal{SHIF}(D)$ and OWL DL to the description logic $\mathcal{SHOIN}(D)$. OWL Full contains features not expressible in description logics, but needed to be compatible with RDFS. In this sense, OWL Full can be seen as the union of RDFS and OWL DL in terms of language features or, alternatively, as OWL without syntactic restrictions.

The latest version OWL 2 is again split in two flavors OWL 2 DL and OWL 2 Full. OWL 2 DL corresponds roughly to the logic $\mathcal{SROIQ}(D)$. An exception for this are the so called \textit{keys}, which essentially state that certain property values and class memberships are sufficient to uniquely identify an individual. This language feature is derived from relational database technology and cannot be expressed in DLs. A further interesting feature of OWL 2 DL is meta-modelling via \textit{punning}, which allows to use the same URI for an entity denoting a class and an individual. Internally, those are then semantically treated as separate entities.

As in OWL, there is also an OWL 2 Full variant introduced for RDFS compatibility. In addition, three profiles were introduced: EL, QL, and RL. Each profile imposes, usually syntactical, restrictions on OWL in order to allow for more efficient reasoning. OWL 2 EL is aimed at applications which require expressive property modelling and is based on the logic $\mathcal{EL}^{++}$, which guarantees polynomial reasoning time wrt. ontology size for all standard inference problems. QL is targeted at applications with massive volumes of instance data. In QL, query answering can be implemented on top of conventional relational database
systems and sound and complete conjunctive query answering methods can be implemented in LOGSPACE. As in the EL profile, the standard inference problems run in polynomial time. RL is aimed at scalable applications, which however, do not want to sacrifice too much expressive power. Reasoning algorithms for it can be implemented in rule-based engines and run in polynomial time. The EL and QL languages are subsets of OWL 2 DL, whereas RL provides two variants where one is subset of OWL 2 Full and the other one is a subset of OWL 2 DL.

Compared to RDFS, OWL Full is much more expressive by allowing the construction of complex concepts via boolean connectors (conjunctions, disjunction, negation) as well as cardinality restrictions (minimum, maximum and exact cardinality). Furthermore, it includes several other features such as class disjointness and more fine-granular property modelling. Properties can be declared to be reflexive, functional, symmetric or equivalent to other properties. Due to those characteristics, OWL is usually seen as a full-fledged ontology language, whereas RDFS is more suitable for lightweight vocabularies. However, while RDFS allows reification as a method for adding contextual information (see discussion below in Section 6), this is not allowed in OWL DL and generally not supported in standard description logics.

More details on the OWL language and its formal foundations can be found in the [Hitzler et al., 2010]. For technical details, we refer to the W3C recommendations, in particular those for OWL 2 [OWL Working Group, 27 October 2009].

4 RULES

Rules come in many guises. In one of their most basic forms, they consist of statements of the form

$$\bigwedge_{i} A_i \rightarrow B,$$

where $B$ (the head of the rule) and all $A_i$ (which form the body of the rule) are atomic formulas from first-order predicate logic, and all variables in the rule are considered universally quantified. Function symbols may or may not be allowed. Additional logical connectives may be allowed, e.g. disjunctions or existential quantifiers in the rule head. Constructs from other, e.g. modal or non-monotonic, logics may be allowed. Atomic formulas in head or body may be replaced by procedural built-ins or other executable commands. If a formal semantics is defined for a rules language, it may range from a full first-order predicate logic semantics to an entirely procedural specification. Logic programming, as discussed in the chapter by Robert Kowalski in this volume, is one rather prominent example of such a rules language.

Many rules paradigms had already been well established in research and industry, when the World Wide Web Consortium set out to define a recommended
standard for modeling ontologies for the Semantic Web. Rules, in particular in the broad sense of logic programming, were a very strong contender for the base paradigms on which to base this recommended standard. As discussed in Section 3, description logics were eventually chosen, but a significant research and industrial interest remained in giving rules a more prominent role, and in the aftermath of the 2004 W3C OWL specification [Web Ontology Working Group, 10 February 2004], the ensuing discussions on the role of rules for the Semantic Web were sometimes rather fierce [Horrocks et al., 2005a; Kifer et al., 2005].

One of the prominently discussed paradigms was F-Logic [Kifer et al., 1995], in its variant as a primarily syntactic extension of logic programming [Angele and Lausen, 2004]. This included industrial strength systems [Angele, 2014], W3C member submissions [de Bruijn et al., 2005], research investigations (e.g., [Friedland et al., 2004; Roman et al., 2005]), and industrial applications (e.g., [Angele and Gesmann, 2007; Angele et al., 2008]). Central to F-Logic is its use of a frame-based syntax.

RuleML, the Rule Markup Initiative, is another long-standing effort which aims at developing the Rule Markup Language RuleML “as the canonical Web language for rules using XML markup, formal semantics, and efficient implementations” (cited from http://ruleml.org). RuleML is set to encompass the entire rule spectrum.

The Semantic Web Rules Language SWRL [Horrocks et al., 2004; Horrocks et al., 2005b] has been presented in the aftermath of the W3C OWL specification, as an early effort to accommodate rules modeling in a way compatible with the description logic paradigm. In its original formulation, SWRL simply added rules with a first-order predicate logic semantics to OWL, but reasoning systems and research discussions soon converged towards reading SWRL rules in a more restricted way, known as DL-safety, which was more akin to the Herbrand semantics usually considered in logic programming, was more readily implementable, and retained a key design rationale of description logics, namely decidability [Motik et al., 2005; Krisnadhi et al., 2011]. SWRL became rather prominent in the wake of OWL, and the notion of DL-safety provided a key notion towards subsequent research into the integration of description logics and rules – see Section 6 for pointers to more recent developments on this issue.

The Rule Interchange Format RIF [Kifer and Boley, 2013] was finally established as a W3C recommended standard for exchanging rules between rule systems. It draws, in part, on both F-Logic and RuleML. In particular, it sports a frame-based syntax inspired by F-Logic and draws from RuleML for its normative XML-based syntax. Set up as an exchange language, rather than as a full-blown knowledge representation language, RIF has several dialects as well as an extensible framework. Of the dialects, RIF Core [Boley et al., 2013] corresponds to Datalog, RIF BLD (the Basic Logic Dialect) [Boley and Kifer, 2013a] corresponds to Horn logic, and RIF PRD (the Production Rule Dialect) [de Sainte Marie et
al., 2013] captures main aspects of production rule systems [Klahr et al., 1987] which incorporate ad-hoc computational mechanisms (such as side-actions triggered by rule executions, e.g. printing a document). Each of these dialects, even the strongly logic-based ones RIF Core and RIF BLD, sport some use of datatypes and built-ins. RIF FLD (the Framework for Logic Dialects) [Boley and Kifer, 2013b; Kifer, 2008] provides a means to define further RIF dialects.

Finally, as discussed above, we note that the RDFS semantics is expressible in terms of Horn rules. On top of that, the W3C has defined a combined semantics for combining arbitrary Horn rules encoded in RIF [Boley and Kifer, 2013a] with RDF(S) and OWL [de Bruijn, 2013]. In the academic literature there have been several other proposals to extend RDF by rules beyond Horn rules, such as ERDF [Analyti et al., 2008] which provides a syntax for normal logic programs (that is, rules with default negation interpreted under the stable model semantics [Gelfond and Lifschitz, 1988]), or N3 [Berners-Lee et al., 2008] which also allows default negation in rule bodies; although N3's semantics is only defined informally, its reference implementation CWM\(^{10}\) implements the perfect model semantics [Przymusinski, 1988]).\(^{11}\)

5 PARTICULAR CHALLENGES TO USING LOGIC-BASED KNOWLEDGE REPRESENTATION ON THE WEB

The use of logic-based knowledge representation and reasoning at the scale of the World Wide Web poses a number of particular challenges which have so far not received primary attention in logic research. We list some of them in the following.

The added value of a good machine-processable syntax for knowledge representation formalisms is easily underestimated. However, it is a fundamental basis for knowledge exchange and integration which needs to be approached carefully in order to obtain a widest possible agreement between stakeholders. The World Wide Web Consortium has gone a long way in establishing recommended standards for knowledge representation for the Semantic Web, in particular through their work on RDF [Lassila and Swick, 2004; Cyganiak and Wood, 2013], OWL [Smith et al., 2004; Hitzler et al., 2012], and RIF [Boley et al., 2013; Boley and Kifer, 2013a], but also by establishing special-purpose shared vocabularies based on these, e.g. SKOS Simple Knowledge Organization System [Miles and Bechhofer, 2009], SSN Semantic Sensor Networks [Compton et al., 2012], provenance [Groth and Moreau, 2010].

Investigating the scalability of automated reasoning approaches is, of course, an established line of research in computer science. However, dealing with Web scale data lifts this issue to yet another level. Shared memory parallelization of reasoning is highly effective [Kazakov et al., 2011], however it breaks down if data size exceeds capacities. Massive distributed memory parallelization has started to

\(^{10}\)http://www.w3.org/2000/10/svav/doc/cwm.html

\(^{11}\)according to personal communication with Dan Connolly.
be investigated [Mutharaju et al., 2013; Schlicht and Stuckenschmidt, 2010; Urbani et al., 2011; Urbani et al., 2012; Weaver and Hendler, 2009; Zhou et al., 2012], but there is as yet insufficient data for casting a verdict if distributed memory reasoning will be able to meet this challenge. Some authors even call for the investigation of non-deductive methods, e.g. borrowed from machine learning or data mining, as a partial replacement for deductive approaches [Hitzler and van Harmelen, 2010].

Automated reasoning applications usually rely on clean, single-purpose, and usually manually created or curated knowledge bases. In a Web setting, however, it would often be an unrealistic assumption that such input would be available, or would be of sufficiently small volume to make manual curation a feasible approach. In some cases, this problem may be reduced by crowdsourcing data curation [Acosta et al., 2013]. Nevertheless, on the Web we should expect high-volume or high-throughput input which at the same time is multi-authored, multi-purposed, context-dependent, contains errors and omissions, and so forth [Hitzler and van Harmelen, 2010; Janowicz and Hitzler, 2012]. The aspects just mentioned are often referred to as the volume (size of input data), velocity (speed of data generation) and variety aspects of data, in fact these three V’s are usually discussed within the much larger Big Data context, within which many Semantic Web challenges can be located [Hitzler and Janowicz, 2013].

To give just one example of variety which is particularly challenging in a Semantic Web context, consider basic geographical notions such as forest, river, or village, which depend heavily on social agreement and tradition, and are furthermore often influenced by economic or political incentives [Janowicz and Hitzler, 2012]. This type of variety is often referred to as semantic heterogeneity, and it cannot be overcome by simply enforcing a single definition: In fact, the different perspectives are often incompatible and would result in logical inconsistencies if combined. Research on the question how to deal with semantic heterogeneity may, of course, be more a question of pragmatics than of formal logic, yet the body of literature dealing with this issue is still too small to confidently locate major promising approaches. Formal logical approaches which have been proposed as partial solutions include fuzzy or probabilistic logics [Klinov and Parsia, 2008; Łukasiewicz and Straccia, 2009; Straccia, 2001], paraconsistent reasoning [Maier et al., 2013], and the use of defaults or other non-monotonic logics related to reasoning with knowledge and belief [Baader and Hollunder, 1995; Bonatti et al., 2009; Donini et al., 2002; Eiter et al., 2008a; Knorr et al., 2011; Motik and Rosati, 2010; Sengupta et al., 2011], but the larger issue remains unresolved. Others have advocated the use of ontology design patterns for meeting the challenge of semantic heterogeneity [Gangemi, 2005; Janowicz and Hitzler, 2012], but it is currently not clear how far this will carry.

There exist a multitude of different knowledge representation paradigms based on different and often incompatible design principles. Logical features which appear useful for modeling such as uncertainty handling or autoepistemic introspection are often studied in isolation, while the high-variety setting of the Semantic Web would suggest that combinations of features need to be taken into account in
realistic settings. However, merging different knowledge representation paradigms often results in unwieldy, highly complex logics for which strong automated reasoning support may be difficult to obtain [de Bruijn et al., 2011; de Bruijn et al., 2010; Knorr et al., 2012]. Even W3C recommended standards, which on purpose are designed to be of limited variety, expose this issue. The OWL 2 DL profile is essentially a traditional description logic, but if serialised in RDF (as required by the standard), the RDF formal semantics is not equivalent to the OWL formal semantics, and the sets of logical consequences defined by these two formal semantics for an OWL file (serialised in RDF) are not contained within each other. The OWL 2 Full profile was established to encompass all of both OWL 2 DL and RDF Schema, but we are not aware of any practical use of its formal semantics. Concerning the relationship between OWL and RIF, in contrast, the gap seems to be closing now, as discussed in Section 6 below.

Another practical obstacle to the use of formal logic and reasoning on the Semantic Web is the availability of strong and intuitive tools and interfaces, of industrial strength, which would relieve application developers from the burden of becoming an expert in formal logic and Semantic Web technologies. Of course, many useful tools are available, e.g. [Lehmann, 2009; Calvanese et al., 2011; David et al., 2011; Horridge and Bechhofer, 2011; Tudorache et al., 2013], and some of them are of excellent quality, but a significant gap remains to meet practical requirements.

6 RECENT DEVELOPMENTS

Concerning more recent developments and investigations concerning the use of logic-based knowledge representation for the Semantic Web, it appears to make sense to distinguish between theoretical advances and advances concerning dissemination into practice and applications.

On the theory side, a convergence of different paradigms is currently happening, in particular with respect to the description-logic-based and the rule-based paradigm. The following are some of the most prominent recent developments.

- The introduction of role chains and some other constructs in OWL 2 [Hitzler et al., 2012] has made a significant step towards closing the gap between Horn logic and description logics, by making many more rules expressible in major description logics [Krötzsch et al., 2008; Krötzsch, 2010]. Another recently introduced syntax construct, called nominal schemas [Carral Martínez et al., 2012; Carral et al., 2013], further enables the expression of many monotonic rules [Krisnadhi et al., 2011], up to a complete coverage of $n$-ary Datalog under the Herbrand semantics [Knorr et al., 2012]. Research concerning algorithmization and reasoning tool support are under way [Krötzsch et al., 2011; Steigmiller et al., 2013].

- Datalog has recently seen a revival due to results investigating the theoretical and practical implications of adding existentially quantified variables to rule
heads. The general approach is known as existential rules and was most prominently introduced under the name Datalog$^{\dagger-}$ [Cal`ı et al., 2012]. The paradigm on the one hand generalizes Datalog, and on the other hand is very akin in spirit to the so-called $\mathcal{EL}$ family of description logics which entered the mainstream with the introduction of the tractable$^{12}$ description logic $\mathcal{EL}^{++}$ [Baader et al., 2005] and its applications [Baader et al., 2006] – a line of work which eventually led to the OWL 2 profile known as OWL EL [Motik et al., 27 October 2009]. Currently, existential rules are under investigation from many different angles and by researchers of different backgrounds.

- Rule paradigms, and in particular logic programming in its many variants, have long been investigated from the perspective of non-monotonic logics, in particular in order to deal with types of commonsense reasoning related to defaults. Similar investigations have been pursued in recent years in order to extend description logics with such non-monotonic features, resulting in a significant body of work (e.g., [Baader and Hollunder, 1995; Bonatti et al., 2009; Donini et al., 2002; Eiter et al., 2008a; Giordano et al., 2013; Grimm and Hitzler, 2008; Huang et al., 2013; Knorr et al., 2011; Lukasiewicz and Straccia, 2009; Motik and Rosati, 2010; Sengupta et al., 2011]) which seems to converge towards a unifying perspective [de Bruijn et al., 2010; Knorr et al., 2012].

The situation concerning the dissemination of logic-based methods to Web practice is much less clear, partly because these types of investigations are mainly industry-driven and thus often not well documented in the research literature. RDF-based reasoning is often being used, as existing RDF triple stores often provide the required support. The use of ontologies in expert-system-like applications is also rather common, in particular in industrial settings where input data is more controlled or curateable. So-called ontology-based data access (OBDA) [Calvanese et al., 2011], which rests on the idea of using shared ontologies as access layers for databases, is currently under heavy research investigation and will likely stay so for some time. For use of deep KR on the open Web major challenges remain [Noy and McGuinness, 2013; Hitzler and van Harmelen, 2010; Jain et al., 2010] which range from logical issues, e.g. how to deal with noisy data, to pragmatic issues, e.g. the development of modeling best practices and strong tool support.

Adding contextual information to RDF(S), OWL and SPARQL

A particularly relevant topic that has appeared in various disguises over the years is the lack of means to express contextual information alongside RDF(S) & OWL. Various approaches have been proposed in the literature to extend RDF(S), OWL

\footnote{To be precise, the worst-case computational complexity of computing the class hierarchy of all named classes is polynomial with respect to time.}
and SPARQL by contextual information such as temporal information associated to RDF triples [Gutierrez et al., 2007] and OWL statements [Motik, 2012], fuzzy annotations [Vaneková et al., 2005; Mazzieri and Dragoni, 2005; Straccia, 2009], or defining more general forms of “Annotated RDF” [Udrea et al., 2010; Zimmermann et al., 2012]. Note that the importance of adding such contextual meta-information to RDF data at the level of triples is also a topic within the ongoing developments within RDF1.1 [Herman, 2009], where not only the semantics of such extensions, but also their syntactic representation within RDF is under discussion; while the basic RDF vocabulary offers the possibility to add meta-descriptions through so-called reification, supported by special vocabulary terms \( \text{rdfs:Statement}, \text{rdfs:subject}, \text{rdfs:predicate}, \text{rdfs:object} \), these vocabulary terms are not given any special semantics and are often perceived as cumbersome in practice. Alternative proposals for expressing contextual information include so-called named graphs [Carroll et al., 2005], i.e., using again URIs to identify RDF graphs themselves, which then allows to use those identifiers in RDF triples to add contextual meta-information to the triples in these graphs; different syntax proposals for named graphs include N3 [Berners-Lee et al., 2008], TRIG/TRIX [Carothers et al., 2013; Carroll et al., 2005], and N-Quads [Carothers, 2013].

ACKNOWLEDGEMENTS

Pascal Hitzler acknowledges support by the National Science Foundation under award 1017225 "III: Small: TROn – Tractable Reasoning with Ontologies." Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. Jens Lehmann acknowledges support by grants from the European Union’s 7th Framework Programme provided for the projects GeoKnow (GA no. 318159) and LOD2 (GA no. 257943).

BIBLIOGRAPHY

[Angele et al., 2008] Jürgen Angele, Michael Erdmann, and Dirk Wenke. Ontology-based knowledge management in automotive engineering scenarios. In Martin Hepp, Pieter De Leenheer,


[Franconi et al., 2005] Enrico Franconi, Jos de Bruijn, and Sergio Tessaris. Logical reconstruction of normative RDF. In Workshop on OWL: Experiences and Directions (OWLED’05), 2005.


