Semantic web*

Jérôme Euzenat and Marie-Christine Rousset

Abstract The semantic web aims at making web content interpretable. It is no less than offering knowledge representation at web scale. The main ingredients used in this context are the representation of assertional knowledge through graphs, the definition of the vocabularies used in graphs through ontologies, and the connection of these representations through the web. Artificial intelligence techniques and, more specifically, knowledge representation techniques, are put to use and to the test by the semantic web. Indeed, they have to face typical problems of the web: scale, heterogeneity, incompleteness, and dynamics. This chapter provides a short presentation of the state of the semantic web and refers to those techniques at work in the semantic web.

1 Introduction

The idea of using knowledge representation on the worldwide web appeared rapidly after the development of the web. Systems like SHOE [Luke et al, 1997] and Ontobroker [Fensel et al, 1998] integrated formal knowledge representation in web pages while Ontoserver [Farquhar et al, 1995] and HyTroeps [Euzenat, 1996] used the web to browse and edit knowledge.

Jérôme Euzenat
Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LIG, F-38000 Grenoble France
e-mail: Jerome.Euzenat@inria.fr

Marie-Christine Rousset
Univ. Grenoble Alpes & Institut Universitaire de France, CNRS, Inria, Grenoble INP, LIG, F-38000 Grenoble, France
e-mail: Marie-Christine.Rousset@imag.fr

* in: Pierre Marquis, Odile Papini, Henri Prade (eds), A Guided tour of artificial intelligence research, Volume 3, Chapter 6, pp181-207, Springer, 2020 DOI: 10.1007/978-3-030-06170-8_6

In 1998, Tim Berners-Lee described the principles of what he called the 'semantic web'. He already had launched, within the W3C[2], the development of the RDF language whose initial goal was to annotate web pages. At that time, he wrote '*A Semantic Web is not Artificial Intelligence*', but later he added that it was '*Knowledge Representation goes Global*' [Berners-Lee, 1998].

In 2000, this idea had not taken off nor been subject to ambitious developments. The US DARPA launched the DAML (*DARPA Agent Markup Language*) program directed by James Hendler, the initiator of SHOE. Right afterwards, was held a seminar on '*Semantics for the Web*' [Fensel et al, 2003] which steered the EU OntoWeb thematic network, itself leading to the Knowledge Web network of excellence. Such efforts have involved many artificial intelligence researchers in the development of semantic web technologies [Euzenat, 2002].

The semantic web is often presented as a 'web for machines': it aims at representing knowledge on the web with formal languages that can be processed by machines. For that purpose, it is natural to use techniques developed in artificial intelligence.

Indeed, the main target application for the semantic web is information retrieval (Chapter "Artificial intelligence and information retrieval" of this volume): being able to retrieve precise information on the web because documents are annotated with 'semantic metadata'. Although this goal seemed remote in 2000, it is nowadays far more tangible. In the interval, we witnessed the use of of Open Graph by Facebook, the definition of the lightweight ontology schema.org by four of the main search engines (Bing, Google, Yahoo, Yandex), and the use by Google of the *knowledge graph* providing structured answers to queries instead of lists of documents.

Since then, many other types of applications have emerged, each based on the annotation of resources using the semantic web knowledge representation languages. Such applications are referred to as applications of semantic technologies. In particular, this covers:

- Semantic web services in which web service interfaces (input/output) are semantically annotated,
- Semantic peer-to-peer systems in which shared resources are semantically annotated,
- Semantic social networks in which social relations and people profiles are semantically annotated,
- Semantic desktop in which personal information (agenda, address book, etc.) is semantically expressed,
- Semantic web of things in which sensors, effectors and the information they exchange are semantically annotated,
- The web of data in which data sources are expressed and linked in semantic web languages (see Section 2.2).

The semantic web has thus spread out of its initial playground. It is now a broad experimentation field in which various problems are investigated such as trust in peer-to-peer systems, statistical data integration, or smart city monitoring. As web

[2] *WorldWide Web Consortium*: the organisation in charge of recommending web technologies.

technologies, semantic technologies have the potential to progressively affect all computer developments [Janowicz et al, 2015].

We focus on the relationships between artificial intelligence techniques and semantic web technologies. For that purpose, we shall consider three problems at the heart of the semantic web: knowledge representation (Section 2), reasoning with knowledge and data (Section 3), and relating data and knowledge sources (Section 4). This organisation may seem arbitrary. Indeed, it is not organised chronologically, e.g. the definition of an ontology may occur before or after that of data, and reasoning may occur within a simple data graph or across a network of ontologies. However, each section corresponds to one type of activity with a well-identified set of techniques. Each relies on artificial intelligence techniques, mostly from knowledge representation and reasoning, and could take advantage of other techniques described in this book. We can only survey them: there exist more extensive treatments of these [Antoniou and van Harmelen, 2008; Hitzler et al, 2009; Abiteboul et al, 2011].

2 Publishing data on the web

The first question solved by the semantic web is the expression of factual data on the web. To address it, a data expression language, adapted to the characteristics of the web, is required. The RDF (*Resource Description Framework*) language has been developed and recommended by the W3C and adopted in practice.

2.1 RDF: simple conceptual graphs

RDF [Lassila and Swick, 1999; Cyganiak et al, 2014] is a formal language for expressing assertion of relations between 'resources'. It is first used to annotate documents written in non structured languages, or as interface between data sources expressed in languages with a comparable semantics, e.g. data bases.

An RDF graph is a set of triples $\langle$subject, predicate, object$\rangle$. The three elements may be IRIs (*Internationalized Resource Identifiers* generalising URIs, i.e. *Uniform Resource Identifiers* [Berners-Lee et al, 1998] themselves comprising URLs), literals, e.g. character strings or dates, or variables, also called 'blanks'.

The use of IRIs is specific to semantic web technologies, with respect to artificial intelligence practice. IRIs behave as simple identifiers. However, they are generally associated with a name space identifying their provenance (vocabulary or data source). For instance, `foaf:Person` identifies the `Person` concept in the friend-of-a-friend name space `http://xmlns.com/foaf/0.1/` shortened as `foaf:`. Thus an RDF graph may use different vocabularies with very little risks of conflict or ambiguity.

This set of triples can be represented in a natural way as a graph, or more precisely as a directed labelled multigraph, in which the elements appearing as subject or object are vertices and each triple is represented by an edge between the subject

and object labelled by the predicate (see Figure 1). Such a graph may be encoded as an RDF/XML [Gandon and Schreiber, 2014] or N-triple [Carothers and Seaborne, 2014] document.

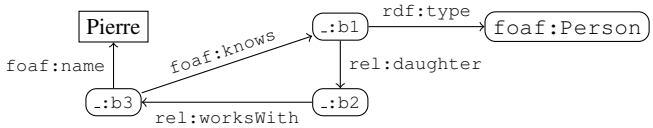

Fig. 1 A RDF graph: Pierre knows a parent of one of his colleagues. Blank nodes are identified by ids in the _ name space.

Figure 1 displays an RDF graph fragment. The IRIs of some resources belong to specific vocabularies: FOAF, RDF, rel. The literals are represented by a rectangle, such as "Pierre". Non-labelled vertices represent variables, also called blanks. Intuitively, this graph may be understood as: 'Someone called Pierre knows a parent of one of his colleagues'. This intuitive semantics is not sufficient for an automated treatment; hence, RDF has been given a formal semantics.

The semantics of an RDF graph is defined in model theory [Hayes and Patel-Schneider, 2014]. RDF is thus a proper logics. This semantics has this peculiarity that predicates, which naturally correspond to classical dyadic predicates, may also be considered as resources. Hence, the triple $\langle$`rdf:type`, `rdf:type`, `rdf:Property`$\rangle$ is legitimate in RDF and can be interpreted (it indeed means that `rdf:type` denotes a predicate). This is achieved by interpreting triples in two steps: a first step associates a denotation to each IRI used and a second step interprets those used in predicate position as binary relations. This is the main specificity of the RDF semantics with respect to that of first-order logics.

Some RDF graphs may be translated as formulas in a positive (without negation), conjunctive, existential and function-free first-order logic. To each triple $\langle s,p,o\rangle$ corresponds the atomic formula $p(s,o)$, such that p is a predicate name, and o and s are constants if these elements are IRIs or literals, and variables otherwise. A graph is translated as the existential closure of the conjunction of atomic formulas associated to its triples. Hence, the graph of Figure 1 is translated by:

$$\exists ?b1,?b2,?b3;\texttt{rdf:type}(?b1,\texttt{foaf:Person})\wedge \texttt{rel:daughter}(?b1,?b2)\wedge\texttt{rel:worksWith}(?b2,?b3)\wedge \texttt{foaf:name}(?b3,\text{"}\textit{Pierre}\text{"})\wedge\texttt{foaf:knows}(?b3,?b1)$$

The models of such a formula are isomorphic to those of the direct semantics of graphs. This logical translation of RDF illustrates the proximity with other representations: logics, of course, as well as positive Datalog (see Chapter "Databases and artificial intelligence" of this volume) or conceptual graphs (see Chapter "Reasoning with ontologies" of Volume 1).

RDF is a simple and stable language. It fulfils adequately its function on the semantic web. It benefits from a good software support in many programming languages.

RDF 1.1 has acknowledged the notion of *RDF Dataset* which is a collection of RDF graphs and that of *named graphs* which are graphs identified by IRIs. This identification may be exploited to assert statements (triples) about graphs and hence triples. Such statements may be about the origin of the graph or the confidence in what it asserts. This function was previously fulfilled by '*quads*', i.e. triples which were usually tagged by a IRI.

It has also embraced various formats used for expressing RDF graphs: beyond RDF/XML and n-triples. The new well-defined formats are RDFa —enabling the embedding of RDF within HTML—, Turtle, N-Quads —extending ntriples by quads—, and JSON-LD. These formats are usually supported by RDF data management systems, so-called *triple stores*.

2.2 The web of data

In 2006, from the observation that the semantic web development was lagging due to the lack of resources, Tim Berners-Lee put forth a methodology for publishing large quantities of data on the web so that they can help the semantic web [2006].

This defined the '*web of data*' through four principles for publishing data on the web, summarised as [Berners-Lee, 2006; Heath and Bizer, 2011]:

1. Resources are identified by IRIs.
2. IRIs are dereferenceable, i.e. can be looked up on the web.
3. When a IRI is dereferenced, a description of the identified resource should be returned, ideally adapted through content negotiation.
4. Published web data sets must contain links to other web data sets.

Although not explicitly specified, linked data sources are more usable if they are published with semantic web technologies: IRIs for identifying resources, RDF for describing them, OWL for defining the used vocabularies, SPARQL for accessing data, and asserting links between sources through the `owl:sameAs` predicate.

The 'five stars rating', to be interpreted as incremental steps, has been introduced for measuring how much of this usability is achieved, namely:

⋆ Publish data on the web in any format, e.g. a scan of a table in PDF;
⋆⋆ Use structured data formats, e.g. a table in Excel instead of its scan in PDF;
⋆⋆⋆ Use non-proprietary formats, e.g. CSV instead of Excel, such that users have direct access to the raw data;
⋆⋆⋆⋆ Use universal formats to represent data, such as RDF, which encapsulates both syntax and semantics;
⋆⋆⋆⋆⋆ Link data to other data sets on the web, thereby providing context.

The first three stars are easy to reach and these already enable some data reuse. However, humans still have to handle all the semantic issues related to integration. In order to have data that is more easily discoverable and interoperable, it is necessary to reach the fourth and the fifth stars.

The web of data is thus a huge RDF graph reachable through the HTTP protocol like the web. Very quickly it developed around DBpedia [Bizer et al, 2009], a massive extraction of Wikipedia in RDF. Many efforts linked their resources to DBpedia. Soon after, several governments encouraged the publication of their (open) data in RDF. Since then, libraries, museums, research institutions of various kinds joined the linked open data cloud. In 2016, it contained 9960 data sources providing more than 150 billion triples using 576 vocabularies or ontologies[3].

2.3 Querying RDF with SPARQL

The RDF recommendation [Hayes and Patel-Schneider, 2014] defines the RDF semantics so that developers can assess the soundness and completeness of their inference mechanisms. Such a mechanism is however discussed in W3C documents in direct relation with simple conceptual graphs (see Chapter "Reasoning with ontologies" of Volume 1). RDF consequence can be tested by looking for a labelled graph homomorphism. Although finding a homomorphism is an NP-complete problem, efficient algorithms are available for implementing it.

A major way to deal with RDF graphs consists of answering structured queries, as opposed to testing if a formula is a consequence. SPARQL [Prud'hommeaux and Seaborne, 2008; Harris and Seaborne, 2013] is the recommended query language for RDF. It is inspired by the SQL relational data base query language. It returns the values of variables that make a graph pattern consequence of a given graph. For instance, the query of Figure 2 returns the answer: $\{\langle$"*Pierre*", `_:b3`$\rangle\}$ from the graph of Figure 1.

```
SELECT ?n, ?z
FROM Figure 1
WHERE {
    ?x rel:daughter ?y .
    ?y rel:worksWith ?z .
    ?z foaf:knows ?x .
    OPTIONAL { ?z foaf:name ?n }
}
```

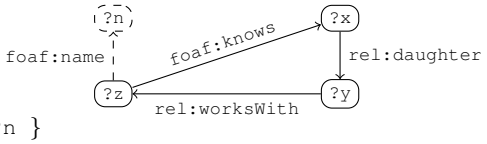

Fig. 2 A SPARQL query and the corresponding graph pattern (the optional part is displayed in dashed lines).

SPARQL is based on the graph pattern concept (RDF graphs containing variables) composed through classical operators (conjunction, disjunction, optional conjunction, filtering). The answers are provided by graph homomorphisms between the queried graph and the graph patterns. These homomorphisms are then combined by operators reflecting the semantics of the composition, like in SQL [Pérez et al, 2009]. SPARQL is thus a hybrid language between conceptual graphs (see Chapter "Reasoning with ontologies" of Volume 1) and data bases.

[3] Sources: `http://stats.lod2.eu/` and `http://lov.okfn.org`.

Beside the SELECT query form, it is possible to use a CONSTRUCT query form which returns a graph. As relational SELECT queries extract a relation from relations, a SPARQL CONSTRUCT query extracts a graph from graphs.

The following query is a rewriting of that of Figure 2 as a CONSTRUCT query: instead of returning a set of homomorphisms (or assignments to the SELECT variables which satisfy the pattern), it will return an RDF graph made of the CONSTRUCT graph pattern whose variables are instantiated with the resulting homomorphisms.

```
CONSTRUCT {
    ?z rdf:knowsTheFatherOf ?y .
    ?z foaf:name ?n }
FROM Figure 1
WHERE {
    ?z foaf:knows/rel:daughter ?y .
    ?y rel:workWith+ ?z .
    OPTIONAL { ?z foaf:name ?n }
}
```

The query also uses paths which have been introduced in SPARQL 1.1. Triple patterns may use regular path, such as the sequence of `foaf:knows` and `rel:daughter` or the unbounded sequence of `rel:workWith` predicate instead of atomic predicates. This leads to suppress any unnecessary variables from the initial graph pattern.

In addition, SPARQL 1.1 [Harris and Seaborne, 2013] introduced the opportunity to combine queries and sources. In particular, queries can integrate subqueries, evaluate a query part againts a specific named graph (with the FROM NAMED and GRAPH clauses), and evaluate federated queries, i.e. queries against several data sources whose results are combined (with the SERVICE construct).

2.4 Beyond SPARQL: streams and navigation

Many applications in sensor networks, smart cities or the web of things are generating streams of RDF triples or quads, i.e. in this context, RDF triples labelled by their occurrence time. One may want to query such streams for detecting particular events occurring and alerting some monitor about it. Because the expressiveness of triples is very poor, these queries rely on graph patterns. The difficulty is that applications should be able to deal quickly with large amounts of real time data. For that purpose, they can take advantage of techniques defined in artificial intelligence for rule compilation (RETE, TREAT), as well as more sophisticated techniques for recognising chronicles [Dousson et al, 1993]. This has to be combined with time windowing techniques matching events only related to closer events.

The C-SPARQL language has been designed for that purpose [Barbieri et al, 2010] and theoretical studies for stream reasoning are under development [Beck et al, 2015].

The path facilities of SPARQL 1.1 are a good way to specify navigational queries and to evaluate queries against the semantic web in a '*follow your nose*' fashion, i.e.

by dereferencing reachable IRIs and querying their own data sources. This querying mode is well adapted to the web of data (Section 2.2).

However, SPARQL queries are usually evaluated against one particular graph specified in the FROM or GRAPH clauses. The new federated query feature also requires that graph patterns be evaluated against one specific server. Thus the SPARQL recommendation does not specify the semantics of evaluating queries against the web. Moreover, the availability of the web is unpredictable and the evaluation would require, in principle, to look-up a potentially infinite number of data sources. To better specify the semantics of such queries evaluated over the web, *reachability-based* and *context-based query semantics* have been defined [Hartig and Pirró, 2016]. They characterise the set of answers to a query by answers with respect to reachable data sources or those selected from the IRIs instantiating the paths. They also identify *web-safe queries* which can be evaluated according to the new semantics and return complete answers.

3 A little knowledge representation goes a long way

The title of this section is an adaptation of James Hendler's catch phrase 'A little semantics goes a long way'.

Once able to express data on the web, the next problem is the definition of the vocabulary used in the RDF graphs. This requires listing the terms used to describe data as well as expressing the axioms constraining the use of such terms allowing a machine to interpret them properly. Such vocabularies are defined as *ontologies* (see Chapters "Reasoning with ontologies" and "Knowledge engineering" of Volume 1). We will consider below ontologies as sets of axioms in a specific logic. Several such logics have been defined.

3.1 RDFS

RDFS (for RDF Schema [Brickley and Guha, 2004]) extends the RDF language by more precisely describing the resources used for labelling the graphs. In RDFS, this extension is seen as the introduction of a specific vocabulary (a set of IRIs) carrying a precise semantics. RDF already introduced a few keywords in the `rdf:` name space for structuring knowledge:

- ⟨`ex:Sonia rdf:type ex:Employee`⟩ asserts that the resource `ex:Sonia` is an instance of the class `ex:Employee`;
- ⟨`rel:worksWith rdf:type rdf:Property`⟩ tells that `rel:worksWith` is a predicate, i.e. a resource used for labelling edges.

RDFS is expressed as RDF triples using a few more keywords in the `rdfs:` name space, such as:

- ⟨`ex:Employee rdf:type rdfs:Class`⟩ indicating that the resource `ex:Employee` as for type `rdfs:Class`, it is thus a class.

- ⟨ex:Employee rdfs:subClassOf foaf:Person⟩ meaning that ex:Employee is a subclass of foaf:Person, thus each instance of ex:Employee is also an instance of foaf:Person, like ex:Sonia;
- ⟨ex:worksWith rdfs:range ex:Employee⟩ asserts that any resource used as the extremity of an edge labelled by rel:worksWith is an instance of the ex:Employee class.

The RDFS semantics extends that of RDF by considering some IRIs as identifying classes and interpreting them in two steps, as properties [Hayes and Patel-Schneider, 2014]. ρDF [Pérez et al, 2009] is the subset of RDFS that concentrate on the purely ontology description features.

However, RDFS only provides limited mechanisms to specify classes further. Its expressiveness is limited to that of frame languages, hence other languages have been introduced to address more demanding needs.

3.2 OWL: description logics on the web

The OWL language [Horrocks et al, 2003; Dean and Schreiber (eds.), 2004; Beckett, 2009] is dedicated to class and property definitions. Inspired from description logics (see Chapter "Reasoning with ontologies" of Volume 1), it provides constructors to constrain them precisely. The OWL syntax, based on RDF, introduces a specific vocabulary in the owl: name space, but contrary to RDFS, this is not sufficient to define the full OWL syntax, such as the owl:Restriction imbrication presented below. Hence, not all RDF graphs using this vocabulary are necessarily valid OWL ontologies: further constraints have to be satisfied.

The semantic of OWL constructors is defined through model theory [Patel-Schneider et al, 2004; Motik et al, 2009c] and directly follows description logics. It is more precise than the semantics assigned to the RDF graphs representing OWL ontologies: more can be deduced from the ontology.

OWL allows one to finely describe relations, e.g. the relation daughter, is here defined as a subrelation of the inverse of hasParent whose codomain is of female gender:

```
<owl:ObjectProperty rdf:about="rel:daughter">
  <owl:subPropertyOf>
    <owl:ObjectProperty>
      <owl:inverseOf rdf:resource="rel:hasParent"/>
    </owl:ObjectProperty>
  </owl:subPropertyOf>
  <rdfs:range>
    <owl:Restriction>
      <owl:onProperty rdf:resource="foaf:gender"/>
      <owl:hasValue>female</owl:hasValue>
    </owl:Restriction>
  </rdfs:range>
</owl:ObjectProperty>
```

We provide the main OWL constructors and their intuitive semantics (see [Motik et al, 2009b] for a formal definition):

- RDF keywords (`rdf:type`, `rdf:Property`) and RDFS' (`rdfs:subClassOf`, `rdfs:subPropertyOf`, `rdfs:range`, `rdfs:domain`) are used with the same semantics.
- `owl:Class` is a new (meta)class.
- `owl:sameAs` and `owl:differentFrom` are used to assert that two resources are equal or different.
- `owl:inverseOf` asserts that a property p is the converse of a property p' (in this case, the triple $\langle s\ p\ o\rangle$ entails $\langle o\ p'\ s\rangle$); other characteristics may be assigned to properties such as reflexivity (`owl:ReflexiveProperty`), transitivity (`owl:TransitiveProperty`), symmetry (`owl:SymmetricProperty`) or functionality (`owl:FunctionalProperty`).
- `owl:allValuesFrom` associates a class c to a relation p. This defines the class of objects x such that if $\langle x\ p\ y\rangle$ holds, then y belong to c (this is a universally quantified role in description logics). `owl:someValuesFrom` encodes existentially quantified roles.
- `owl:minCardinality` (resp. `owl:maxCardinality`) defines the class of objects related to at least (resp. at most) a specific number of objects through a given property. A qualified version of these constructors constrains, in addition, that these objects belong to a particular class.
- `owl:oneOf` defines a class in comprehension by enumerating the set of its instances.
- `owl:hasValue` constrains a property to have a particular individual as value.
- `owl:disjointWith` asserts that two classes cannot have a common instance.
- `owl:unionOf`, `owl:intersectionOf` and `owl:complementOf` define a class as the disjunction, the conjunction or the negation of other classes.
- `owl:hasSelf` defines the class of objects related to themselves through a specific relation.
- `owl:hasKey` asserts that a set of properties is a key for a class, i.e. that two distinct instances cannot share the same values for these properties.
- `owl:propertyChainAxiom` composes several relations and properties to obtain a new relation or property.

We have not mentioned all constructors. Many of them can be trivially implemented by using the cited ones, e.g. `owl:equivalentClass` asserting that two classes are equivalent can be expressed with two `rdfs:subClassOf` assertions. OWL also uses data types that are not considered here. They are however important as they can lead to inconsistency.

OWL 2 interprets differently and independently a IRI if it has to be considered as a class or a property. This constraint does not apply to OWL full.

3.3 Expressiveness/efficiency trade-off

The expressiveness/efficiency trade-off has always been present in the work on description logics (see Chapter "Reasoning with ontologies" of Volume 1). It is possible to adopt a unified encoding of description logics preserving the opportunity to

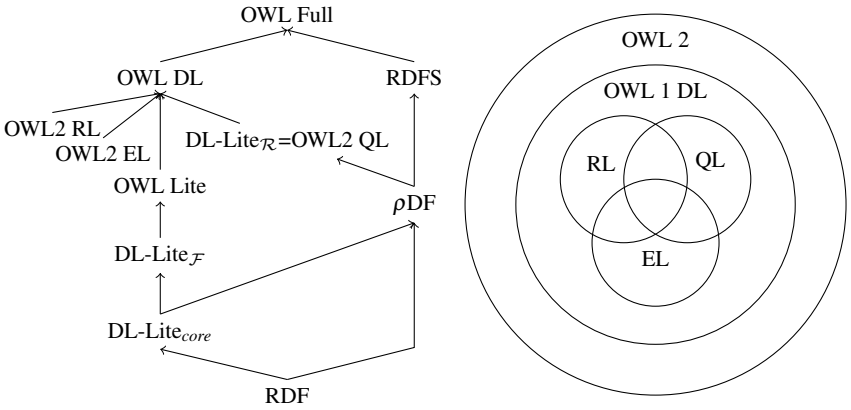

Fig. 3 A hierarchy of ontology languages for the semantic web. Arrows and set inclusion represent the expressibility order between these different languages.

choose the one corresponding to a particular trade-off [Euzenat and Stuckenschmidt, 2003]. This attitude has been adopted in OWL 1 [Dean and Schreiber (eds.), 2004] through only three languages:

- OWL Lite covers a subset of the available constructors and their use may be restricted, e.g. cardinality constraints can only use 0 or 1. It corresponds to the $\mathcal{SHIF}$ description logics for which optimized reasoners based on tableau methods have been implemented and used in practice with a reasonable performances, despite that the reasoning problems such as subsumption checking or satistiability checking are known to be EXPTIME-complete [Baader et al, 2003].
- OWL DL extends OWL Lite by covering all constructors including number restrictions, but enforcing specific constraints, e.g. a resources cannot be both a class and a property. It corresponds to the $\mathcal{SHOIN}$ description logics for which the reasoning problems are decidable but with high complexity (NEXPTIME-complete). Nevertheless, like for OWL-Lite, practical reasoners such as FaCT++ [Tsarkov and Horrocks, 2006], Pellet [Sirin et al, 2007] or HermiT [Glimm et al, 2014] have acceptable performances on most OWL DL ontologies built in practice.
- OWL Full is a semantic extension of both OWL DL and RDF(S) and cannot be translated into a description logic language. In addition, the fact that arbitrary roles can be used in number restrictions would make entailment (and thus subsumption) in the underlying logic undecidable [Horrocks et al, 2000].

However, the various usages of these languages may have led to a different organisation. For instance, some applications may be more easily dealt with a version of OWL DL without disjunction, with intuitionistic negation and such that classes may be considered as instances. Such a language does not exist because OWL has not been defined modularly.

Hence, some languages have been studied independently from the W3C in order to offer efficient reasoning services. First, description logic research has proposed

languages for answering efficiently to queries modulo ontologies. DL-Lite [Calvanese et al, 2007; Artale et al, 2009] is a subfamilly of OWL which contains no disjunction, limited constraints on properties and a restricted negation (see Chapter "Reasoning with ontologies" of Volume 1). In consequence, conjunctive queries in DL-Lite may be evaluated efficiently by rewriting them into SQL queries and using existing database management systems. This family features a language, DL-Lite$_\mathcal{R}$, more expressive than ρDF, for which reasoning is tractable.

Another approach [Voltz, 2004; Hustadt et al, 2007] consists of determining a subset of OWL whose expressiveness is lower than that of OWL Lite and to complement it with a rule language in such a way that its complexity remains polynomial.

OWL 2 integrated these approaches by introducing the notion of language profiles [Motik et al, 2009a]. In addition to the languages of OWL 1, it provides profiles that correspond to tractable description logics in which subsumption checking can be performed in polynomial time:

OWL 2 EL suppresses all universal quantification from OWL DL, i.e. it forbids universal quantification, cardinality restrictions, disjunction, negation and relation properties as well as inverse role, but it preserves role composition. If the ontology is made only of concept definitions, i.e. if general concept inclusions are forbidden, EL knowledge bases correspond to simple conceptual graphs (see Chapter "Reasoning with ontologies" of Volume 1). General concept inclusions in the ontology correspond to existential rules that have been studied both in Datalog$^+_-$ [Calì et al, 2009a] and as an extension of conceptual graphs with rules (see Chapter "Reasoning with ontologies" of Volume 1);

OWL 2 QL corresponds to DL-Lite$_\mathcal{R}$;

OWL 2 RL suppresses any construction which may generate unknown objects and direct the axioms so that, only some expressions (existential) are authorised in the superclass and others (universals) are expressed in subclass of a `rdfs:subClassOf` axiom.

Figure 3 offers two views of the relations between the main languages defined around OWL and RDFS.

3.4 Reasoning

Reasoning is at the core of many tasks performed by semantic web applications, such as data interlinking, answering queries through ontologies, or data consistency checking. The availability of efficient reasoning tools is a decisive criterion for the adoption of a particular language for expressing ontological constraints. Such tools may be embedded in query systems in order to process sophisticated queries and link and enrich the returned answers (see Section 3.5).

However, the problem of determining if an RDF graph is the consequence of another is already an NP-complete problem. Efficient algorithms have been developed to compute the graph homomorphisms which correspond to this problem, based on backtracking methods pioneered in constraint networks (see Chapter "Constraint reasoning" of Volume 2).

In the semantic web, several factors make the problem more difficult:

- One may want to achieve reasoning on the whole semantic web, i.e. all the RDF and OWL information available on the web (see Section 2.4).
- These graphs and the reasoning to be performed depend on ontologies for which reasoning problems may be of high complexity or even undecidable.
- Last, but not least, this information is distributed and heterogeneous.

Research on reasoning with the semantic web languages are being pursued in several directions:

- developing efficient decision procedures for the underlying description logics [Horrocks and Sattler, 2007; Motik et al, 2009d], and
- isolating OWL fragments for which efficient procedures are known.

In the semantic web setting, the size of the ontologies is usually much smaller than the size of the data described using these ontologies. Therefore, it is of utter importance to distinguish reasoning tasks concerning the ontological part only (such as subsomption test or query rewriting) from reasoning tasks involving data (such as instance checking or answering queries). In the latter case, measuring the data complexity is more meaningful for evaluating the practical performances of the reasoning algorithms than measuring the combined complexity that takes into account the size of the ontology and possibly the size of the query in addition to the size of the data. For instance, answering conjunctive queries by query rewriting in DL-Lite has an exponential combined complexity in the worst-case but its data complexity is in AC_0, a highly parallelizable subclass of LOGSPACE, just as in standard relational data bases for which optimisations are available.

3.5 Querying modulo ontologies

The definition of SPARQL through graph homomorphisms does not make it easy to apply it with other semantics than that of simple RDF graphs. Indeed, without correspondence results between consequence and homomorphisms for such languages, query evaluation is disconnected from their semantics. This happens if one queries RDF graphs without taking into account their RDFS Schema or OWL ontologies. SPARQL could have been grounded on the notion of logical consequences instead of graph homomorphism [Alkhateeb et al, 2009]. With such a foundation, any ontology language using model theory for expressing its semantics could be used for interpreting queries.

Various attempts have been made to deal with this situation. For instance, it is possible to answer a SPARQL query with respect to an ontology written in ρDF or DL-Lite through simple rewriting of the query as a PSPARQL [Alkhateeb et al, 2008], SPARQL 1.1 or SQL query. Similar results have been obtained by restricting SPARQL itself [Pérez et al, 2010]. Similarly, the SPARQL-DL language [Sirin and Parsia, 2007] evaluates a fragment of SPARQL (conjunctive queries with limited use of variables) with respect to an OWL DL ontology.

SPARQL 1.1 introduced the notion of SPARQL entailment regime [Glimm and Ogbuji, 2013] which replaces graph homomorphism by the notion of logical entailment. Hence, the queried data may be interpreted under OWL or RDFS semantics among others. It also introduces further constraints on the types of authorised query pattern, expected answers, or what to do in case of inconsistency.

Independently from SPARQL, other works have extended Datalog to deal with queries written in DL-Lite$_\mathcal{R}$ (hence in ρDF) [Calì et al, 2009b] or studied the complexity of such queries when introducing the SPARQL disjunction operator [Artale et al, 2009]. This work has sparked the new domain of ontology-based data access (see Chapter "Reasoning with ontologies" of Volume 1) in which such an ontology is used for extracting more conveniently data from relational data bases. This, in particular, contributes exposing traditional data bases as SPARQL endpoints.

3.6 Rules

Concerning rules, the situation is less clear than for other semantic web languages. The W3C has recommended a '*rule interchange format*' (RIF [Boley et al, 2013]) whose goal is to offer both logic programming rules (Chapter "Logic programming" of Volume 2) and '*business rules*', a kind of rule triggering an action when a condition is satisfied. Two dialects, sharing part of their syntax (RIF-Core), have been defined: BLD for '*Basic Logic Dialect*' and PRD for '*Production Rule Dialect*'. RIF has a syntax different from those provided by other languages. In particular, the RDF syntax of RuleML and SWRL [Horrocks et al, 2004] has not been retained for RIF.

From a theoretical standpoint, all work developed in logic programming applies naturally to the semantic web and can be transposed in the RIF-BLD dialect. Moreover, some connection between rules and ontologies have been specified [de Bruijn and Welty, 2013].

Combining description logics and Datalog has been studied [Levy and Rousset, 1998]. In the semantic web context, work has started from less expressive fragments: DLP ('*description logic programs*' [Grosof et al, 2003]) is a minimal language combining Horn clauses and description logics. It cannot express all OWL Lite, nor all logic programming. However, it could be adopted to reason at large scale. DLP has been the inspiration for OWL 2 RL.

A closer integration of description logics (Chapter "Reasoning with ontologies" of Volume 1) and of '*answer set programming*' (Chapter "Logic programming" of Volume 2), which reintroduces aspects such as negation-as-failure and the closed world assumption, has also been considered [Eiter et al, 2008; Motik and Rosati, 2010]. Other works have attempted the definition of description logics on a logic programming language basis [Hustadt et al, 2007, 2008].

From the conceptual graph side, logical rules have been studied as an extension of simple conceptual graphs (see Chapter "Reasoning with ontologies" of Volume 1, [Baget et al, 2011]) which can be directly applied to RDF given the proximity of these formalisms.

3.7 Robust inference

A typical property of the web is the huge quantity of available information. The semantic web is no exception. Unfortunately, it also contains erroneous, outdated, redundant or incomplete pieces of information. A human user is, in general, able to detect these problems and to overcome them. It is thus necessary to develop reasoning modes taking the semantic web into account, i.e. which remain as faithful as possible to the recommended languages, without being disrupted by errors found in the data sources. In summary, robust reasoners are needed.

Various techniques may be adapted to the semantic web context to achieve robustness: paraconsistent logics or statistical models, intuitionistic logics, etc. Some proposals use non monotonic logics [Pührer et al, 2010] to counter inconsistency. Anytime reasoning or reasoning under resource constraints might be used to overcome the huge size of the web. However, this aspect is not very developed.

Other works aim to reduce the impact of errors in ontologies. They only take into account axioms that extend an ontology by defining new concepts and not those that constrain the meaning of existing ones [Hogan et al, 2009]. This is a radical option to obtain conservative extensions (see Section 4.3).

4 Dealing with heterogeneity

The semantic web being grounded in the same principles of autonomy and independence as the web, ontologies used by various sources of knowledge may be different. It is thus necessary to reconcile them and to express relations between these ontologies.

4.1 From alignments to networks of ontologies

The first problem is the description of relations between data or ontologies, i.e. that an individual in a data source is the same as another in another source or that a class in one ontology is more general than another class in another ontology. In general, such a relation, called a correspondence expresses the relation between two entities of two different ontologies. For instance, one may want to express that the property `daughter` is more specific than the property `child`:

$$\langle \texttt{rel:daughter}, \sqsubseteq, \texttt{my:child} \rangle$$

Such correspondences, when they are generated by ontology matching systems are often assigned a confidence measure.

However, correspondences may go beyond the ontology language. This is especially useful when the ontology languages are not very expressive, e.g. SKOS. For instance, the former correspondence could have been expressed as a daughter to be equivalent to the conjunction of the converse of the parent relation whose domain is

Female:

$$\langle \texttt{rel:daughter}, \equiv, \texttt{my:parent}^{-1} \sqcap \texttt{domain}(\texttt{my:Female}) \rangle$$

The EDOAL language [David et al, 2011] provides description logics constructors within the alignment language and can thus express such correspondences. It offers four types of features:

- Constructions of entities from other entities can be expressed through algebraic operators (here $\sqcap$).
- Restrictions can be expressed on entities in order to narrow their scope (here $\texttt{domain}(\texttt{my:Female})$).
- Transformations of property values can be specified, e.g. concatenation of first and last name to give the name or conversion from m/s to miles-per-hour.
- Link keys can be defined for expressing conditions under which, instances of the aligned entities should be considered equivalent (see Section 4.5).

A set of such correspondences between the two same ontologies is called an alignment [Euzenat and Shvaiko, 2013]. More generally, it may be necessary to express networks of ontologies involving several ontologies and alignments between these.

The Distributed Ontology Language (DOL) [Mossakowski et al, 2015] has been designed for the purpose of dealing in a unified manner with ontologies defined in different languages. Such languages are considered in the categorical formalisms of institutions and operations are defined to use them together. In the DOL framework, alignments and networks of ontologies are syntactic objects. Alignments are interpreted as ontologies which relate two other ontologies. DOL is becoming an Object Management Group standard and is supported by the Hets tool.

4.2 Semantics of alignments

There exists a consensus on the necessity to express alignments between ontologies. However, contrary to ontology languages, there is no recommendation for expressing alignments, networks of ontologies and their semantics.

A first solution is simply to use OWL to express ontologies and alignments. This is called the reduced semantics [Euzenat and Shvaiko, 2013]. OWL offers the `owl:sameAs` relation to express that two individuals are the same. This is particularly used in the web of data (Section 2.2) for expressing link sets, very often kept separated from the data sets. It also offers `owl:equivalentClass` and `rdfs:subClassOf` to express the equivalence or subsumption between two classes (the same exists for properties). However, using OWL in this context constrains to merge data sources in order to exploit them: the alignments between ontologies do not exist on their own.

The same remarks apply to the SKOS language [Miles and Bechhofer, 2009] designed for expressing thesauri and which offers relations between terms of different thesauri, such as `skos:broadMatch` and `skos:exactMatch`.

Several proposals have been made to preserve this ontology/alignment duality. C-OWL [Bouquet et al, 2004] is a language for expressing oriented alignments, called mappings, between ontology expressed in OWL. A correspondence expresses how the information from one ontology may be imported to another ontology, but not the other way. Its originality is that networks of ontologies are considered as such and their semantics is provided on top of OWL semantics preserving the locality of each ontology (contrary to the previous solution). The semantics of C-OWL is grounded on that of distributed description logics [Borgida and Serafini, 2003].

The equalising semantics [Zimmermann and Euzenat, 2006] is an alternative to the distributed semantics which does not require that the ontologies be expressed in OWL. It also preserves the interpretations of the various ontologies and projects these interpretations in a common domain of interpretation in which the alignment relations can be interpreted. Global reasoning on networks of ontologies, e.g. composing alignments, can thus be performed.

The DOL framework [Mossakowski et al, 2015] supports these three semantics, and the heterogeneity of ontological formalisms, at least on simple alignments.

4.3 Reasoning in networks of ontologies

Once ontologies are connected through alignments, methods for reasoning with them have to be developed. This can concern several types of problems:

Consistency test: determining if a network of ontologies, or a sub-part of it, has a model;

Entailment test: determining if a statement or a correspondence is entailed by a network of ontologies;

Query answering: determining variable assignments that satisfy a formula.

Solving such problems often requires specific techniques. These are widely open problems which have found only few concrete developments. They raise, in particular, strong scalability problems.

First, most OWL reasoners are able to reason with imported ontologies. However, gathering the ontologies becomes less possible when data has to be considered in addition to axioms. Furthermore, this approach cannot be adopted when data is only accessible through SPARQL endpoints.

In the domain of semantic peer-to-peer systems, peers offer data expressed in their own ontologies related by alignments (usually sets of subsumption statements). SomeWhere [Adjiman et al, 2006] has developed an original approach to consistency testing and query answering. It uses propositional reasoning about the ontology structure in order to determine query evaluation plans, then it evaluates queries from peer to peer.

DRAGO [Serafini and Tamilin, 2005] is a distributed reasoner for C-OWL. Ontologies are described in OWL DL. It is implemented as a distributed tableau reasoner in which the construction of a model for one peer may recursively require the satisfaction tests of formulas obtained through correspondences by other peers

DRAOn [Zimmermann and Le Duc, 2008] is a distributed reasoner determining the consistency of a network of ontologies in the equalising semantics [Zimmermann and Euzenat, 2006]. It works independently of the logics used in each peer: their only requirement is that they support the $\mathcal{ALC}$ logic. However, the alignment language is reduced to the subsumption and the disjunction of concepts and properties.

One of the important aspects of these three reasoners is that they operate in a decentralised manner: each peer is autonomous and has to communicate with others.

More localised attempts have concentrated on reasoning with only two aligned ontologies in order to repair them [Meilicke et al, 2009; Jimenez Ruiz et al, 2009]. These approaches use the reduced semantics (Section 4.1) to identify correspondences leading to inconsistency or incoherence and return a coherent alignment. They typically choose a consistent sub-alignment satisfying some criterion, i.e. they suppress correspondences causing incoherence. Alcomo [Meilicke et al, 2009] can compute optimal repairs which minimally modify the alignment when several inconsistency sources are detected. LogMap [Jimenez Ruiz et al, 2009] decomposes equivalence correspondences into two subclass correspondences. Hence, it is possible to suppress only one of these when repairing. Alignment repair or debugging is directly related to belief revision (see Chapter "Belief revision, belief merging and information fusion" of Volume 1). Revision techniques have been developed for networks of ontologies as well [Euzenat, 2015].

The notion of conservative extension has been defined for description logics [Ghilardi et al, 2006; Lutz et al, 2007]. It means that, when an ontology is extended, e.g. by connecting it to another ontology through an alignment, no additional statement using the signature of the initial ontology can be deduced. This somewhat defeats the purpose of networks of ontologies since their goal is rather to benefit from the knowledge exposed in other locations. However, it may be useful to ensure that ontologies are loosely connected self-sufficient modules. Moreover, reasoning can be implemented efficiently with conservative extensions since all formulas expressible with the vocabulary of an ontology can be derived from this ontology alone. Unfortunately, deciding if an ontology or a network, is a conservative extension of another ontology is hard and even non decidable for $\mathcal{ALCQIO}$ which is the heart of OWL DL [Ghilardi et al, 2006].

4.4 Ontology matching

Ontology matching is the task of finding an alignment between two ontologies [Euzenat and Shvaiko, 2013]. Numerous algorithms have been designed to fulfil this task exploiting techniques developed in various domains, including artificial intelligence. Ontology matching approaches may be organised in:

- content-based approaches which rely on ontology content. These may be based on:
  - Labels used to name or comment the entities found in the ontologies. Techniques may be inspired from computational linguistics (Chapter "Artificial

intelligence and natural language" of this volume) or information retrieval (Chapter "Artificial intelligence and information retrieval" of this volume).
  - Relationships between entities, and prominently the subsumption relation. Techniques from graph theory may be used.
  - Concept extensions, i.e. their instances when available. In this case, machine learning (Chapters "Statistical computational learning" and "Reinforcement learning" of Volume 1), data analysis or deduplication techniques are used.
  - Semantics of ontologies which may be used for expanding alignments or testing their consistency (Section 4.3). Then, techniques from automated reasoning (Chapter "Automated deduction" of Volume 2) and constraint programming (see Chapter "Constraint reasoning" of Volume 2) may be used.

- context-based approaches which rely on relationships that the ontologies have with other resources. These may be:
  - a corpus of documents annotated by one or both of the ontologies on which statistical machine learning (Chapters "Statistical computational learning" and "Reinforcement learning" of Volume 1) may be used;
  - one or several ontologies used as background knowledge for deducing relations between entities through automated reasoning (Chapter "Automated deduction" of Volume 2);
  - user feedback when providing positive or negative assessment on correspondences;
  - the relation with specific resources such as DBPedia, WordNet or UMLS.

However, none of these approaches works satisfactorily in all situations. Hence, systems use several such techniques together. It is not possible to present these different types of systems (see [Euzenat and Shvaiko, 2013]). We briefly consider below those using machine learning, a typical artificial intelligence technique.

Ontology matching is an inductive rather than a deductive task. There are no rules for deducing alignments, only heuristics to finding them. It is thus natural to reuse machine learning techniques to match ontologies. Two types of techniques may contribute: data mining (Chapter "Designing algorithms for machine learning and data mining" of Volume 2), which isolates regularities in large quantity of data, and learning from example (Chapters "Statistical computational learning" and "Reinforcement learning" of Volume 1), which induces correspondence patterns from examples of correspondences.

Learning from examples is less frequent because, as mentioned, matching cannot easily be generalised. It can, however, be used in two cases: first, when a user is able to show some simple alignments from which the system can learn how to match, and, second, when there are references alignments from which it is possible to learn those algorithm parameters that provide the best results [Ehrig et al, 2005].

Data mining algorithms infer the similarity between two concepts from the similarity between their instances, or the probability that the instances of one class be instances of another one. This may be based on symbolic techniques such as formal concept analysis [Stumme and Mädche, 2001; Kalfoglou and Schorlemmer, 2003] (Chapter "Formal concept analysis" of Volume 3), decision trees [Xu and Embley, 2003] or numeric techniques such as neural networks [Mao et al, 2010], Bayes net-

works [Pan et al, 2005], association rule extraction [David et al, 2007] or a mix of such approaches [Doan et al, 2004].

4.5 Data interlinking

Data interlinking consists of finding representations of the same entity in two (or more) data sets and linking these representation through the `owl:sameAs` predicate. The resulting triple is called a link and a set of links across the same two data sets a link set. It thus plays a very important role for the web of data (Section 2.2).

This task may be considered as very close to ontology matching: ontologies are replaced by data sets and alignments by link sets. Like ontology matching this is an inductive task. However, ontology matching is confronted with two ontologies from which it is difficult to extract regularities, though data interlinking faces huge data sets in which regularities can be found. Hence they use different types of techniques. These two activities are rather complementary: link sets may be exploited by extension-based matching techniques and alignments may guide the objects and features to compare within two data sets.

Data interlinking may be divided in two broad types of approaches:

- Similarity-based approaches use a similarity measure to compare entities and, if they are similar enough, they are considered as the same. The similarity either compares different properties of RDF descriptions or project them in a vector space in which similarity is computed.
- Key-based approaches isolate sufficient conditions, called keys or link keys, for two resources to be the same and use them to find them in data sets. Keys are an extension of relational data base keys which must be associated with an alignment if the data sources do not use the same ontologies [Symeonidou et al, 2014]. Link keys express directly in the same structure identification constraints across ontologies [Atencia et al, 2014].

Both types of approaches may use machine learning or statistical [Capadisli et al, 2015] techniques in order to induce "linkage rules" exploiting similarity or link keys. If a sample of links is provided, supervised methods may be used to learn them; otherwise algorithms have to resort on a measure of the quality of the generated links [Atencia et al, 2014].

Linkage rules are used to generate the actual links. Link keys may be translated into SPARQL queries. Silk can express and process similarities to generate links against large data sets [Volz et al, 2009]. LIMES also uses similarities in metric spaces and focusses on exploiting them efficiently [Ngonga Ngomo and Auer, 2011]. Linkage rules have also been expressed in probabilistic Datalog [Al-Bakri et al, 2016] or description logics [Gmati et al, 2016]. Hence, reasoning can be interleaved with data interlinking.

5 Perspectives

Research on efficient inference with semantic web languages (OWL, SPARQL), is very active. It is at the crossroad between data bases, in which the huge amount of data rules, and knowledge representation and reasoning (see Chapter "Automated deduction" of Volume 2), in which expressiveness prevails. These traditional problems, never dealt with at such a scale, are combined with distributed system problems such as communication, latency and caches. It is thus an important source of problems for artificial intelligence, and specifically for automated reasoning. From the representation standpoint, semantic web languages are not very expressive and oriented towards data expression. There exist work for providing non monotonic [Eiter et al, 2008] or fuzzy [Stoilos et al, 2010] versions of semantic web representations.

The semantic web has reached some maturity. Its technologies are used in various domains and could be in more. Thus users are requesting for making semantic web technologies more reliable. Methods coming from programming languages are used for providing guarantees. This is the case, for instance, of work on SPARQL query containment which aims at ensuring that a query can have an answer or that a data source can indeed provide an answer to a query before actually evaluating the query [Chekol et al, 2012]. There are also work on supporting debugging of knowledge expressed on the web, such as pinpointing used to narrow down the search for inconsistency causes in ontological reasoning or alignment repair [Schlobach, 2005].

Another evolution of the semantic web goes out of the strict realm of artificial intelligence: it is the involvement of users and society as a whole. This trend has now pervaded all computing activities, in particular, through crowdsourcing various kinds of tasks. This applies to query systems in which users can provide feedback on the quality of answers and interactive ontology matching in which matchers ask users for specific help.

Finally, the success of the semantic web should trigger work on its evolution. Indeed, a large, uncontrollable mass of knowledge used directly by machines cannot be stopped for maintenance. It has to continuously evolve without hiccup. Hence specific mechanisms should be designed in order to support coherently evolving ontology, data, alignments and links.

6 Conclusion

As the web, the semantic web is not a well-defined object. Both may be reduced to the technologies of which they are made: URL, HTTP, HTML on the one hand, IRI, RDF, OWL, SPARQL and alignments on the other hand. In both cases, such a description would miss a lot.

Although the semantic web has initially been designed around these technologies, as time passes, they are used to encompass more diverse objects: from minimal RDF snippets to encyclopaedic data bases, from simple schemas to foundational on-

tologies, from path queries against the web to elaborate queries against triple stores, from tiny devices to large back-ends. Embracing all these situations threatens the unity of semantic technologies. It is thus remarkable that the semantic web ecosystem remains grounded on these same technologies.

For artificial intelligence research, the semantic web is a wonderful application field. As we tried to show it, many techniques originating from knowledge representation (Chapter "Reasoning with ontologies" of Volume 1), automated reasoning (Chapters "Automated deduction" and "Reasoning with propositional logic" of Volume 2), constraint programming (see Chapter "Constraint reasoning" of Volume 2), and machine learning (Chapters "Statistical computational learning" and "Reinforcement learning" of Volume 1) are contributing to the semantic web. Another topic not discussed in this chapter is the filling of the semantic web through data and knowledge acquisition using techniques from knowledge engineering (Chapter "Knowledge engineering" of Volume 1), computational linguistic (Chapter "Artificial intelligence and natural language" of this volume) and machine learning (Chapters "Statistical computational learning" and "Reinforcement learning" of Volume 1).

This is a demanding application field: the scale is gigantic, users are difficult. Moreover, artificial intelligence techniques are confronted with specific constraints. For instance, the distribution of resources over the network requires specific reasoning and query evaluation techniques and soon social constraints may force new developments.

This is also a gratifying application field: each technique which will find a place in semantic web technologies will have millions of users.

References

Abiteboul S, Manolescu I, Rigaux P, Rousset MC, Senellart P (2011) Web data management. Cambridge university press, Cambridge (UK)

Adjiman P, Chatalic P, Goasdoué F, Rousset MC, Simon L (2006) Distributed reasoning in a peer-to-peer setting: Application to the semantic web. Journal of artificial intelligence research 25:269–314

Al-Bakri M, Atencia M, David J, Lalande S, Rousset M (2016) Uncertainty-sensitive reasoning for inferring sameAs facts in linked data. In: Proc. 22nd European Conference on Artificial Intelligence (ECAI), The Hague (NL), pp 698–706

Alkhateeb F, Baget JF, Euzenat J (2008) Constrained regular expressions in SPARQL. In: Arabnia H, Solo A (eds) Proceedings of the international conference on semantic web and web services (SWWS), Las Vegas (NV US), pp 91–99

Alkhateeb F, Baget JF, Euzenat J (2009) Extending SPARQL with regular expression patterns (for querying RDF). Journal of web semantics 7(2):57–73

Antoniou G, van Harmelen F (2008) A semantic web primer, 2nd edn. The MIT press

Artale A, Calvanese D, Kontchakov R, Zakharyaschev M (2009) The DL-Lite family and relations. Journal of artificial intelligence research 36:1–69

Atencia M, David J, Euzenat J (2014) Data interlinking through robust linkkey extraction. In: Proc. 21st European Conference on Artificial Intelligence (ECAI), Prague (CZ), pp 15–20

Baader F, Calvanese D, McGuinness D, Nardi D, Patel-Schneider P (eds) (2003) The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press

Baget J, Leclère M, Mugnier M, Salvat E (2011) On rules with existential variables: Walking the decidability line. Artificial Intelligence 175(9–10):1620–1654

Barbieri DF, Braga D, Ceri S, Della Valle E, Grossniklaus M (2010) C-SPARQL: a continuous query language for RDF data streams. International Journal on Semantic Computing 4(1):3–25

Beck H, Dao-Tran M, Eiter T, Fink M (2015) LARS: A logic-based framework for analyzing reasoning over streams. In: Proc. 29th Conference on Artificial Intelligence (AAAI), Austin (TX US), pp 1431–1438

Beckett D (2009) OWL 2 web ontology language document overview. Recommendation, W3C, http://www.w3.org/TR/owl2-overview/

Berners-Lee T (1998) Design issue: What the semantic web can represent. http://www.w3.org/DesignIssues/RDFnot.html

Berners-Lee T (2006) Design issue: Linked data. http://www.w3.org/DesignIssues/LinkedData

Berners-Lee T, Fielding R, Masinter L (1998) Uniform resource identifiers (URI): Generic syntax. RFC 2396, IETF, http://www.ietf.org/rfc/rfc2396.txt

Bizer C, Lehmann J, Kobilarov G, Auer S, Becker C, Cyganiak R, Hellmann S (2009) DBpedia: a crystallization point for the web of data. Journal of web semantics 7(3):154–165

Boley H, Hallmark G, Kifer M, Paschke A, Polleres A, Reynolds D (2013) RIF core dialect. Recommendation, W3C, http://www.w3.org/TR/rif-core/

Borgida A, Serafini L (2003) Distributed description logics: Assimilating information from peer sources. Journal on data semantics I:153–184

Bouquet P, Giunchiglia F, van Harmelen F, Serafini L, Stuckenschmidt H (2004) Contextualizing ontologies. Journal of web semantics 1(1):325–343

Brickley D, Guha R (2004) RDF vocabulary description language 1.0: RDF schema. Recommendation, W3C, http://www.w3.org/TR/rdf-schema/

Calì A, Gottlob G, Lukasiewicz T (2009a) A general datalog-based framework for tractable query answering over ontologies. In: International Conference on Principles of Database Systems (PODS), pp 77–86

Calì A, Gottlob G, Lukasiewicz T (2009b) A general datalog-based framework for tractable query answering over ontologies. In: Proc. 28th ACM Principle of Database Systems conference (PODS), Providence (RI US), pp 77–86

Calvanese D, De Giacomo G, Lembo D, Lenzerini M, Rosati R (2007) Tractable reasoning and efficient query answering in description logics: the DL-Lite family. Journal of automated reasoning 39(3):385–429

Capadisli S, Auer S, Ngonga Ngomo A (2015) Linked SDMX data: Path to high fidelity statistical linked data. Semantic Web 6(2):105–112

Carothers G, Seaborne A (2014) RDF 1.1 N-triples – a line-based syntax for an RDF graph. Recommendation, W3C, http://www.w3.org/TR/n-triples/

Chekol M, Euzenat J, Genevès P, Layaïda N (2012) SPARQL query containment under SHI axioms. In: Proc. 26th American national conference on artificial intelligence (AAAI), Toronto (ONT CA), pp 10–16

Cyganiak R, Wood D, Lanthaler M (2014) RDF 1.1 concepts and abstract syntax. Recommendation, W3C, http://www.w3.org/TR/rdf11-concepts/

David J, Guillet F, Briand H (2007) Association rule ontology matching approach. International journal of semantic web information systems 3(2):27–49

David J, Euzenat J, Scharffe F, Trojahn C (2011) The Alignment API 4.0. Semantic Web Journal 2(1):3–10

de Bruijn J, Welty C (2013) RIF RDF and OWL compatibility. Recommendation, W3C, http://www.w3.org/TR/rif-rdf-owl/

Dean M, Schreiber (eds) G (2004) OWL web ontology language reference. Recommendation, W3C, http://www.w3.org/TR/owl-ref/

Doan AH, Madhavan J, Domingos P, Halevy A (2004) Ontology matching: a machine learning approach. In: Staab S, Studer R (eds) Handbook on ontologies, Springer Verlag, Berlin (DE), chap 18, pp 385–404

Dousson C, Gaborit P, Ghallab M (1993) Situation recognition: Representation and algorithms. In: Proc. 13th International Joint Conference on Artificial Intelligence (IJCAI), Chambéry (FR), pp 166–174

Ehrig M, Staab S, Sure Y (2005) Bootstrapping ontology alignment methods with APFEL. In: Proc. 4th International Semantic Web Conference (ISWC), Galway (IE), Lecture notes in computer science, vol 3729, pp 186–200

Eiter T, Ianni G, Lukasiewicz T, Schindlauer R, Tompits H (2008) Combining answer set programming with description logics for the semantic web. Artificial intelligence 172(12-13):1495–1539

Euzenat J (1996) Hytropes: a www front-end to an object knowledge management system. In: Proc. 10th demonstration track on knowledge acquisition workshop (KAW), Banff (CA), pp (62)1–12

Euzenat J (ed) (2002) Research challenges and perspectives of the Semantic web. EU-NSF Strategic report, ERCIM, Sophia Antipolis (FR), http://www.ercim.org/EU-NSF/semweb.html

Euzenat J (2015) Revision in networks of ontologies. Artificial intelligence 228:195–216

Euzenat J, Shvaiko P (2013) Ontology matching, 2nd edn. Springer, Heidelberg (DE)

Euzenat J, Stuckenschmidt H (2003) The 'family of languages' approach to semantic interoperability. In: Omelayenko B, Klein M (eds) Knowledge transformation for the semantic web, IOS press, Amsterdam (NL), pp 49–63

Farquhar A, Fikes R, Pratt W, Rice J (1995) Collaborative ontology construction for information integration. Tech. Rep. 63, Knowledge system laboratory, Stanford university, Stanford (CA US), URL ftp://ksl.stanford.edu/pub/KSL_Reports/KSL-95-63.ps

Fensel D, Decker S, Erdmann M, Studer R (1998) Ontobroker: The very high idea. In: Proc. 11th International Florida Artificial Intelligence Research Society Conference (FLAIRS), Sanibel Island (FL US), pp 131–135

Fensel D, Hendler J, Lieberman H, Wahlster W (eds) (2003) Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential. The MIT Press

Gandon F, Schreiber G (2014) RDF 1.1 XML syntax. Recommendation, W3C, http://www.w3.org/TR/rdf-syntax-grammar/

Ghilardi S, Lutz C, Wolter F (2006) Did I damage my ontology? a case for conservative extensions in description logics. In: Proc. 10th International conference on principles of knowledge representation and reasoning, Lake District (UK), pp 187–197

Glimm B, Ogbuji C (2013) SPARQL 1.1 entailment regimes. Recommendation, W3C, http://www.w3.org/TR/sparql11-entailment

Glimm B, Horrocks I, Motik B, Stoilos G, Wang Z (2014) HermiT: An OWL 2 reasoner. Journal of automated reasoning 53(3):245–269

Gmati M, Atencia M, Euzenat J (2016) Tableau extensions for reasoning with link keys. In: Proc. 11th International Workshop on Ontology Matching, Kobe (JP), pp 37–48

Grosof B, Horrocks I, Volz R, Decker S (2003) Description logic programs: combining logic programs with description logic. In: Proc. 12th WorldWide Web conference, Budapest (HU), pp 48–57

Harris S, Seaborne A (2013) SPARQL 1.1 query language. Recommendation, W3C, http://www.w3.org/TR/sparql11-query

Hartig O, Pirró G (2016) SPARQL with property paths on the web. Semantic Web Journal

Hayes P, Patel-Schneider P (2014) RDF semantics. Recommendation, W3C, http://www.w3.org/TR/rdf11-mt/

Heath T, Bizer C (2011) Linked Data: Evolving the Web into a Global Data Space. Morgan & Claypool

Hitzler P, Krötzsch M, Rudolph S (2009) Foundations of semantic web technologies. Chapman & Hall/CRC,

Hogan A, Harth A, Polleres A (2009) Scalable authoritative OWL reasoning for the web. International journal of semantic web information Systems 5(2):49–90

Horrocks I, Sattler U (2007) A tableau decision procedure for SHOIQ. Journal of automated reasoning 39(3):249–276

Horrocks I, Sattler U, Tobies S (2000) Practical reasoning for very expressive description logics. Logic Journal of the IGPL 8(3):239–264

Horrocks I, Patel-Schneider P, van Harmelen F (2003) From SHIQ and RDF to OWL: The making of a web ontology language. Journal of web semantics 1(1):7–26

Horrocks I, Patel-Schneider P, Boley H, Tabet S, Grosof B, Dean M (2004) SWRL: a semantic web rule language combining OWL and RuleML. http://www.w3.org/Submission/SWRL/

Hustadt U, Motik B, Sattler U (2007) Reasoning in description logics by a reduction to disjunctive datalog. Journal of automated reasoning 39(3):351–384

Hustadt U, Motik B, Sattler U (2008) Deciding expressive description logics in the framework of resolution. Information and computation 206(5):579–601

Janowicz K, van Harmelen F, Hendler J, Hitzler P (2015) Why the data train needs semantic rails. AI Magazine 36(1):5–14

Jimenez Ruiz E, Cuenca Grau B, Horrocks I, Berlanga R (2009) Ontology integration using mappings: Towards getting the right logical consequences. In: Proc.

6th European Semantic Web Conference (ESWC), Hersounisous (GR), Lecture notes in computer science, vol 5554, pp 173–188

Kalfoglou Y, Schorlemmer M (2003) Ontology mapping: the state of the art. The knowledge engineering review 18(1):1–31

Lassila O, Swick R (1999) Resource description framework (RDF) model and syntax specification. Recommendation, W3C, http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/

Levy A, Rousset MC (1998) Combining Horn rules and description logics in CARIN. Artificial intelligence 104(1-2):165–209

Luke S, Spector L, Rager D, Hendler J (1997) Ontology-based web agents. In: Proc. 1st International Conference on Autonomous Agents, pp 59–66

Lutz C, Walther D, Wolter F (2007) Conservative extensions in expressive description logics. In: Proc. 20th International joint conference on artificial intelligence, Hyderabad (IN), pp 453–458

Mao M, Peng Y, Spring M (2010) An adaptive ontology mapping approach with neural network based constraint satisfaction. Journal of web semantics 8(1):14–25

Meilicke C, Stuckenschmidt H, Tamilin A (2009) Reasoning support for mapping revision. Journal of logic and computation 19(5):807–829

Miles A, Bechhofer S (2009) SKOS simple knowledge organization system: Reference. Recommendation, W3C, http://www.w3.org/TR/skos-reference

Mossakowski T, Codescu M, Neuhaus F, Kutz O (2015) The distributed ontology, modeling and specification language – DOL. In: Koslow A, Buchsbaum A (eds) The Road to Universal Logic: Festschrift for the 50th Birthday of Jean-Yves Béziau Volume II, Springer, pp 489–520

Motik B, Rosati R (2010) Reconciling description logics and rules. Journal of the ACM 57(5)

Motik B, Cuenca Grau B, Horrocks I, Wu Z, Fokoue A, Lutz C (2009a) OWL 2 web ontology language profiles. Recommendation, W3C, http://www.w3.org/TR/owl2-profiles/

Motik B, Patel-Schneider P, Cuenca Grau B (2009b) OWL 2 web ontology language direct semantics. Recommendation, W3C, http://www.w3.org/TR/owl2-direct-semantics/

Motik B, Patel-Schneider P, Parsia B (2009c) OWL 2 web ontology language structural specification and functional-style syntax. Recommendation, W3C, http://www.w3.org/TR/owl2-syntax/

Motik B, Shearer R, Horrocks I (2009d) Hypertableau reasoning for description logics. Journal of artificial intelligence research 36:165–228

Ngonga Ngomo A, Auer S (2011) LIMES - A time-efficient approach for large-scale link discovery on the web of data. In: Proc. 22nd International Joint Conference on Artificial Intelligence (IJCAI), Barcelona (ES), pp 2312–2317

Pan R, Ding Z, Yu Y, Peng Y (2005) A Bayesian network approach to ontology mapping. In: Proc. 4th International Semantic Web Conference (ISWC), Galway (IE), Lecture notes in computer science, vol 3729, pp 563–577

Patel-Schneider P, Hayes P, Horrocks I (2004) OWL web ontology language semantics and abstract syntax. Recommendation, W3C, http://www.w3.org/TR/owl-absyn/

Pérez J, Arenas M, Gutierrez C (2009) Semantics and complexity of SPARQL. ACM transactions on database systems 34(3):16

Pérez J, Arenas M, Gutierrez C (2010) nSPARQL: A navigational language for RDF. Journal of web semantics 8(4):255–270

Prud'hommeaux E, Seaborne A (2008) SPARQL query language for RDF. Recommendation, W3C, http://www.w3.org/TR/rdf-sparql-query

Pührer J, Heymans S, Eiter T (2010) Dealing with inconsistency when combining ontologies and rules using DL-programs. In: Proc. 7th Extended Semantic Web Conference (ESWC), Hersounissos (GR), pp 193–197

Schlobach S (2005) Debugging and semantic clarification by pinpointing. In: Proc. 2nd European Semantic Web Conference (ESWC), Heraklion (GR), pp 226–240

Serafini L, Tamilin A (2005) DRAGO: Distributed reasoning architecture for the semantic web. In: Proc. 2nd European Semantic Web Conference (ESWC), Hersounisous (GR), Lecture notes in computer science, vol 3532, pp 361–376

Sirin E, Parsia B (2007) SPARQL-DL: SPARQL query for OWL-DL. In: Proc. 3rd OWL Experiences and Directions Workshop (OWLED), Innsbruck (AT)

Sirin E, Parsia B, Grau BC, Kalyanpur A, Katz Y (2007) Pellet: A practical OWL-DL reasoner. Journal of Web Semantics 5(2):51–53

Stoilos G, Stamou G, Pan J (2010) Fuzzy extensions of OWL: Logical properties and reduction to fuzzy description logics. International Journal of Approximate Reasoning 51(6):656–679

Stumme G, Mädche A (2001) FCA-Merge: Bottom-up merging of ontologies. In: Proc. 17th International Joint Conference on Artificial Intelligence (IJCAI), Seattle (WA US), pp 225–234

Symeonidou D, Armant V, Pernelle N, Saïs F (2014) Sakey: Scalable almost key discovery in RDF data. In: Proc. 13th International Semantic Web Conference (ISWC), Riva del Garda (IT), pp 33–49

Tsarkov D, Horrocks I (2006) FaCT++ description logic reasoner: System description. In: Proc. 3rd International Joint Conference (IJCAR), pp 292–297

Voltz R (2004) Web ontology reasoning with logic databases. PhD thesis, Universität Fridericiana zu Karlsruhe

Volz J, Bizer C, Gaedke M, Kobilarov G (2009) Discovering and maintaining links on the web of data. In: Proc. 8th International Semantic Web Conference (ISWC), Chantilly (VA US), Lecture notes in computer science, vol 5823, pp 650–665

Xu L, Embley D (2003) Discovering direct and indirect matches for schema elements. In: Proc. 8th International Conference on Database Systems for Advanced Applications (DASFAA), Kyoto (JP), pp 39–46

Zimmermann A, Euzenat J (2006) Three semantics for distributed systems and their relations with alignment composition. In: Proc. 5th conference on International semantic web conference (ISWC), Athens (GA US), pp 16–29

Zimmermann A, Le Duc C (2008) Reasoning on a network of aligned ontologies. In: Proc. 2nd International conference on web reasoning and rule systems (RR), Karlsruhe (DE), pp 43–57