Description Logics and the Semantic Web

Umberto Straccia

ISTI - CNR,
http://gaia.isti.cnr.it/~straccia
straccia@isti.cnr.it

2007
Outline

1. The Semantic Web and Ontologies
   - Today’s Web
   - The Semantic Web Impact
   - Semantic Web Technologies
     - Explicit Metadata
     - Ontologies
     - Web Ontology Languages
     - RDF and RDFS
     - OWL

2. From OWL to Description Logics
   - What Are Description Logics?
   - Short History of Description Logics
   - DLs Basics
   - OWL DL and OWL Lite as Description Logic

3. Description Logic Reasoning
   - Basic Inference Problems (Formally)
   - Reasoning in DLs: Basics
     - \(\mathcal{ALC}\) Tableau rules
Foundations of the Semantic Web
Ontologies and OWL

http://www.semanticweb.org/
Today’s Web

- Most of today’s Web content is suitable for human consumption
  - even Web content that is generated automatically from databases is usually presented without the original structural information found in databases
- Typical Web uses today people’s
  - seeking and making use of information, searching for and getting in touch with other people, reviewing catalogs of online stores and ordering products by filling out forms
Keyword-Based Search Engines

- Current Web activities are not particularly well supported by software tools
  - Except for keyword-based search engines (e.g. Google, AltaVista, Yahoo)
- The Web would not have been the huge success it was, were it not for search engines
Problems with Keyword-based Search Engines

- High recall, low precision
- Low or no recall
- Results are highly sensitive to vocabulary
- Results are single Web pages
- Human involvement is necessary to interpret and combine results
- Results of Web searches are not readily accessible by other software tools
Impossible (?) Using Today’s Web

- Complex queries involving background knowledge
  - Find information about “animals that use sonar but are not either bats or dolphins”, e.g. Barn Owl

- Book me a holiday next weekend somewhere warm, not too far away, and where they speak French or English
What is the Problem?

- Consider a typical web page. Markup consists of:
  - rendering information (e.g., font size and colour)
  - Hyper-links to related content

Semantic content is accessible to humans but not (easily) to computer ...
The Key Problem of Today’s Web

- The meaning of Web content is not machine-accessible
  - lack of semantics
- It is simply difficult to distinguish the meaning between these two sentences:
  - “I am a soccer player”
  - “You may think I’m a soccer player”
The Semantic Web Approach

- Represent Web content in a form that is more easily machine-processable
- Use intelligent techniques to take advantage of these representations
- The Semantic Web will gradually evolve out of the existing Web, it is not a competition to the current WWW
Knowledge management concerns itself with acquiring, accessing, and maintaining knowledge within an organization.

Key activity of large businesses: internal knowledge as an intellectual asset.

It is particularly important for international, geographically dispersed organizations.

Most information is currently available in a weakly structured form (e.g. text, audio, video).
Limitations of Current Knowledge Management Technologies

- **Searching information**
  - Keyword-based search engines

- **Extracting information**
  - Human involvement necessary for browsing, retrieving, interpreting, combining

- **Maintaining information**
  - Inconsistencies in terminology, outdated information

- **Viewing information**
  - Impossible to define views on Web knowledge
Semantic Web Enabled Knowledge Management

- Knowledge will be organized in conceptual spaces according to its meaning
- Automated tools for maintenance and knowledge discovery
- Semantic query answering
- Query answering over several documents
- Defining who may view certain parts of information (even parts of documents) will be possible
The Semantic Web Impact - B2C Electronic Commerce

- A typical scenario: user visits one or several online shops, browses their offers, selects and orders products
- Ideally humans would visit all, or all major online stores; but too time consuming
- Shopbots are a useful tool
Limitations of Shopbots

- They rely on wrappers: extensive programming required
- Wrappers need to be reprogrammed when an online store changes its outfit
- Wrappers extract information based on textual analysis
  - Error-prone
  - Limited information extracted
Software agents that can interpret the product information and the terms of service

- Pricing and product information, delivery and privacy policies will be interpreted and compared to the user requirements

Information about the reputation of shops

Sophisticated shopping agents will be able to conduct automated negotiations
Summary of knowledge flow process

- **Data** = Documents, Pages, Blobs...
- **Structured Data** = Data + Syntax
- **Semantics** = Meaning of Structured Data elements
- **Information** = Structured Data + Semantics
- **Knowledge** = Purposeful Combination of Information
- **The Web**: A Data repository
- **The Semantic Web** = Putting Semantics on the Web
Semantic Web Technologies

- Explicit Metadata
- Ontologies
- Logic and Inference
Web content is currently formatted for human readers rather than programs

HTML is the predominant language in which Web pages are written (directly or using tools)

Vocabulary describes presentation
Problems with HTML

- Humans have no problem with this
- Machines (software agents) do:
  - How distinguish my research topics from my hobbies
  - How determine my location
Better Representation

<topicList>
  <category>
    <name> Knowledge Representation and Reasoning </name>
    <topic> Description Logic </topic>
    <topic> Logic Programming </topic>
    <topic> Uncertainty and Logics </topic>
  </category>
  <category>
    <name> Information Retrieval </name>
    <topic> Distributed Information Retrieval </topic>
    <topic> Schema Matching </topic>
    <topic> Rank Aggregation </topic>
  </category>
</topicList>
Explicit Metadata

- This representation is far more easily processable by machines
- Metadata: data about data
  - Metadata capture part of the meaning of data
- Semantic Web does not rely on text-based manipulation, but rather on machine-processable metadata
Ontologies

- The term ontology originates from philosophy
  - The study of the nature of existence
- Different meaning from computer science
  - An ontology is an explicit and formal specification of a conceptualization
  - Formal and machine manipulable model of a domain of interest
Typical Components of Ontologies

- **Terms** denote important concepts (classes of objects) of the domain
  - e.g. professors, staff, students, courses, departments
  - New concepts can be formed by combining existing ones (e.g. a male professor)
  - Meaning (semantics) of such terms is formally specified

- **Relationships** between these terms: typically class hierarchies
  - a class $C$ to be a subclass of another class $C'$ if every object in $C$ is also included in $C'$
  - e.g. all professors are staff members
Further Components of Ontologies

- Properties:
  - e.g. X teaches Y

- Value restrictions
  - e.g. only faculty members can teach courses

- Disjointness statements
  - e.g. faculty and general staff are disjoint

- Logical relationships between objects
  - e.g. every department must include at least 10 faculty
Example of a Class Hierarchy

- University People
  - Staff
    - Academic Staff
      - Regular Faculty Staff
      - Research Staff
      - Visiting Staff
    - Administration Staff
    - Technical Support Staff
  - Students
    - Undergraduate
    - Postgraduate
Another Example of Structure of an Ontology

Ontologies typically have two distinct components:

- **Names for important concepts in the domain**
  - **Elephant** is a concept whose members are a kind of animal
  - **Herbivore** is a concept whose members are exactly those animals who eat only plants or parts of plants
  - **Adult_Elephant** is a concept whose members are exactly those elephants whose age is greater than 20 years

- **Background knowledge/constraints on the domain**
  - **Adult_Elephants** weigh at least 2,000 kg
  - All **Elephants** are either **African_Elephants** or **Indian_Elephants**
  - No individual can be both a **Herbivore** and a **Carnivore**
http://protege.stanford.edu/
A Semantic Web – First Steps

Make web resources more accessible to automated processes

- Extend existing rendering markup with **semantic markup**
  - Metadata annotations that describe content/function of web accessible resources
- Use Ontologies to provide **vocabulary** for annotations
  - “Formal specification” is accessible to machines
- A prerequisite is a standard web ontology language
  - Need to agree common **syntax** before we can share semantics
  - Syntactic web based on **standards** such as HTTP and HTML
The Role of Ontologies on the Web

- Ontologies provide a shared understanding of a domain: semantic interoperability
  - overcome differences in terminology
  - mappings between ontologies
- Ontologies are useful for the organization and navigation of Web sites
Ontologies are useful for improving the accuracy of Web searches
- search engines can look for pages that refer to a precise concept in an ontology

Web searches can exploit generalization/specialization information
- If a query fails to find any relevant documents, the search engine may suggest to the user a more general query
- If too many answers are retrieved, the search engine may suggest to the user some specializations
Ontology Design and Deployment

Given key role of ontologies in the Semantic Web, it will be essential to provide tools and services to help users:

- Design and maintain high quality ontologies, e.g.:
  - Meaningful – all named classes can have instances
  - Correct – captured intuitions of domain experts
  - Minimally redundant – no unintended synonyms
  - Richly axiomatised – (sufficiently) detailed descriptions

- Store (large numbers) of instances of ontology classes, e.g.:
  - Annotations from web pages

- Answer queries over ontology classes and instances, e.g.:
  - Find more general/specific classes
  - Retrieve annotations/pages matching a given description

- Integrate and align multiple ontologies
Web Ontology Languages

- Wide variety of languages for “Explicit Specification”
  - Graphical notations
    - Semantic networks

![Semantic Network Diagram](image)
Web Ontology Languages

- Wide variety of languages for “Explicit Specification”
  - Graphical notations
    - Topic Maps (see http://www.topicmaps.org/)

![Diagram of Topic Maps](http://www.topicmaps.org/)
Web Ontology Languages

- Wide variety of languages for “Explicit Specification”
  - Graphical notations
    - UML
Web Ontology Languages

- Wide variety of languages for “Explicit Specification”
  - Graphical notations
    - RDF
Web Ontology Languages

- Wide variety of languages for “Explicit Specification”
  - Logic based
    - Description Logics (e.g., OIL, DAML+OIL, OWL)
      “A Happy father is a Man, which has a daughter”

\[ \text{Happy\_Father} \sqsubseteq \text{Man} \sqcap \exists \text{has\_child}.\text{Female} \]
Web Ontology Languages

- Wide variety of languages for “Explicit Specification”
  - Logic based
    - Rules (e.g., RuleML, LP/Prolog)
      - “A Man, which has a daughter is a Happy father”
      \[
      \text{Happy} \_\text{Father}(x) \leftarrow \text{Man}(x), \text{Has} \_\text{child}(x, y), \text{Female}(y)
      \]
Web Ontology Languages

- Wide variety of languages for “Explicit Specification”
  - Logic based
    - First Order Logic (e.g., KIF)
      
      “A Man, which has a daughter buys a lot of clothes”
      \[
      \forall x \forall y \forall z. \text{Man}(x) \land \text{Has\_child}(x, y) \land \text{Female}(y) \rightarrow \\
      \text{BuysALot}(x, z) \land \text{Clothes}(z)
      \]
Web Ontology Languages

- Wide variety of languages for “Explicit Specification”
  - Logic based
    - Conceptual graphs
Web Ontology Languages

- Wide variety of languages for “Explicit Specification”
  - Logic based
    - Non-classical logics (e.g., Flogic, Non-Monotonic, Modal Logics)
Web Ontology Languages

- Wide variety of languages for “Explicit Specification”
  - Probabilistic/Fuzzy

80% of Birds fly

\[ \langle Bird(x) \rightarrow Fly(x), 0.8 \rangle \]

Usually, a sports car is very fast

\[ \langle \forall x \forall y. (SportsCar(x) \land HasSpeed(x, y)) \rightarrow \text{very}(High(y)), 0.9 \rangle \]
Web Ontology Languages: Summary

- Wide variety of languages for “Explicit Specification”
  - Graphical notations
    - Semantic networks
    - Topic Maps (see http://www.topicmaps.org/)
    - UML
    - RDF
  - Logic based
    - Description Logics (e.g., OIL, DAML+OIL, OWL)
    - Rules (e.g., RuleML, LP/Prolog)
    - First Order Logic (e.g., KIF)
    - Conceptual graphs
    - Non-classical logics (e.g., Flogic, Non-Mon, modalities)
  - Probabilistic/fuzzy
- Degree of formality varies widely
  - Increased formality makes languages more amenable to machine processing (e.g., automated reasoning)
Many languages use “object oriented” model based on:

- **Objects/Instances/Individuals**
  - Elements of the domain of discourse
  - Equivalent to constants in FOL

- **Types/Classes/Concepts**
  - Sets of objects sharing certain characteristics
  - Equivalent to unary predicates in FOL

- **Relations/Properties/Roles**
  - Sets of pairs (tuples) of objects
  - Equivalent to binary predicates in FOL

Such languages are/can be:
- Well understood
- Formally specified
- (Relatively) easy to use
- Amenable to machine processing
Web “Schema” Languages

- Existing Web languages extended to facilitate content description
  - XML → XML Schema (XMLS)
  - RDF → RDF Schema (RDFS)

- XMLS not an ontology language
  - Changes format of DTDs (document schemas) to be XML
  - Adds an extensible type hierarchy
    - Integers, Strings, etc.
    - Can define sub-types, e.g., positive integers

- RDFS is recognisable as an ontology language
  - Classes and properties
  - Sub/super-classes (and properties)
  - Range and domain (of properties)
RDF and RDFS

- RDF stands for Resource Description Framework
- RDF is a data model for objects and relations between them
- RDF Schema is a vocabulary description language
- Describes properties and classes of RDF resources
  - Class, Property
  - type, subClassOf, subPropertyOf
  - range, domain
- Provides semantics for generalization hierarchies of properties and classes
- RDF is graphical formalism (+ XML syntax + semantics)
  - for representing metadata
  - for describing the semantics of information in a machine-accessible way

(http://www.w3.org/RDF)
The RDF Data Model

- Statements are of the form
  \[
  \langle \text{subject}, \text{predicate}, \text{object} \rangle
  \]
called triples: e.g.

\[
\langle \text{umberto}, \text{plays}, \text{soccer} \rangle
\]

- can be represented graphically as:

  \[
  \text{umberto} \xrightarrow{\text{plays}} \text{soccer}
  \]

- Statements describe properties of resources
- A resource is any object that can be pointed to by a URI:
  - a document, a picture, a paragraph on the Web;
  - a book in the library, a real person (?)
  - isbn://5031-4444-3333
  - ...
- Properties themselves are also resources (URIs)
URIs

- URI = Uniform Resource Identifier
- The generic set of all names/addresses that are short strings that refer to resources
- URLs (Uniform Resource Locators) are a particular type of URI, used for resources that can be accessed on the WWW (e.g., web pages)
- In RDF, URIs typically look like “normal” URLs, often with fragment identifiers to point at specific parts of a document:

  http://www.somedomain.com/some/path/to/file#fragmentID
Linking Statements

- The subject of one statement can be the object of another.
- Such collections of statements form a directed, labeled graph.

Note that the object of a triple can also be a “literal” (a string).
RDF has an XML syntax that has a specific meaning:

- Every Description element describes a resource
- Every attribute or nested element inside a Description is a property of that Resource
- We can refer to resources by using URIs

```xml
<Description about="some.uri/person/giuseppe_amato">
   <hasColleague resource="some.uri/person/giuseppe_amato"/>
</Description>
<Description about="some.uri/person/umberto_straccia">
   <hasHomePage> http://gaia.isti.cnr.it/~straccia</hasHomePage>
</Description>
<Description about="some.uri/person/fabrizio_sebastiani">
   <hasColleague resource="some.uri/umberto_straccia"/>
</Description>
```
RDF gives a formalism for meta data annotation, and a way to write it down in XML, but it does not give any special meaning to vocabulary such as `subClassOf` or `type`.

- Interpretation is an arbitrary binary relation.

RDF Schema allows you to define vocabulary terms and the relations between those terms.

- It gives “extra meaning” to particular RDF predicates and resources.
- This “extra meaning”, or semantics, specifies how a term should be interpreted.
RDFS Example

- RDF Schema terms (just a few examples):
  - Class
  - Property
  - type
  - subClassOf
  - range
  - domain

- These terms are the RDF Schema building blocks (constructors) used to create vocabularies:

  \(<\text{Person}, \text{type, Class}>\)
  \(<\text{hasColleague}, \text{type, Property}>\)
  \(<\text{Professor}, \text{subClassOf,Person}>\)
  \(<\text{Carole}, \text{type,Professor}>\)
  \(<\text{hasColleague}, \text{range,Person}>\)
  \(<\text{hasColleague}, \text{domain,Person}>\)
RDF/RDFS “Liberality”

- No distinction between classes and instances (individuals)
  
  \(<\text{Species}, \text{type}, \text{Class}>\>
  \(<\text{Lion}, \text{type}, \text{Species}>\>
  \(<\text{Leo}, \text{type}, \text{Lion}>\>

- Properties can themselves have properties
  
  \(<\text{hasDaughter}, \text{subPropertyOf}, \text{hasChild}>\>
  \(<\text{hasDaughter}, \text{type}, \text{familyProperty}>\>

- No distinction between language constructors and ontology vocabulary, so constructors can be applied to themselves/each other
  
  \(<\text{type}, \text{range}, \text{Class}>\>
  \(<\text{Property}, \text{type}, \text{Class}>\>
  \(<\text{type}, \text{subPropertyOf}, \text{subClassOf}>\>
RDF/RDFS Semantics

- RDF has “Non-standard” semantics in order to deal with this
- Semantics given by RDF Model Theory (MT)

In RDF MT, an interpretation \( I \) of a vocabulary \( V \) consists of:

- \( IR \), a non-empty set of resources, called the domain of \( I \).
- \( IS \), a mapping from URI references in \( V \) into \( IR \).
- \( IP \), a distinguished subset of \( IR \) (the set of properties of \( I \)).
  - A vocabulary element \( v \in V \) is a property iff \( IS(v) \in IP \).
- \( IEXT \), a mapping from \( IP \) into the powerset of \( IR \times IR \), \( IEXT(x) \) is called the extension of \( x \).
  - I.e., a set of elements \( \langle x, y \rangle \), with \( x, y \) elements of \( IR \).
  - I.e., is a set of pairs which identify the arguments for which the property is true.
  - This trick of distinguishing a relation as an object from its relational extension allows a property to occur in its own extension.
- \( IL \), a mapping from typed literals in \( V \) into \( IR \).
- A distinguished subset \( LV \) of \( IR \), called the set of literal values, which contains all the plain literals in \( V \).

Class interpretation \( ICEXT \) simply induced by \( IEXT(IS(type)) \):

\[
ICEXT(C) = \{ x | \langle x, C \rangle \in IEXT(IS(type)) \}
\]

(http://www.w3.org/TR/rdf-mt/)
Example RDF/RDFS Interpretation
RDFS Interpretations

- RDFS adds extra constraints on interpretations
  - E.g., interpretations of $\langle C, \text{subClassOf}, D \rangle$ constrained to those where $\text{ICEXT}(\text{IS}(C)) \subseteq \text{ICEXT}(\text{IS}(D))$

- Can deal with triples such as
  - $\langle \text{Species}, \text{type}, \text{Class} \rangle$
  - $\langle \text{Lion}, \text{type}, \text{Species} \rangle$
  - $\langle \text{Leo}, \text{type}, \text{Lion} \rangle$
  - $\langle \text{SelfInst}, \text{type}, \text{SelfInst} \rangle$

- And even with triples such as
  - $\langle \text{type}, \text{subPropertyOf}, \text{subClassOf} \rangle$

- But not clear if meaning matches intuition (if there is one)
Example: Multimedia Information retrieval

- Media dependent properties:
  - Object features:
    - color, shape, texture
    - structure
  - Examples:
    - o1: IsAbout (o1, Snoopy) = 0.8
    - o2: Object Form Layer

- Media independent properties:
  - Object Semantics Layer
  - Examples:
    - Snoopy is a dog
    - Birds and Dogs are animals
    - Woodstock is a bird

RDF and RDFS graph:

- Image
- hasImageRegion
- isAbout
- ImageRegion
- Animal
- Bird
- Dog
- Woodstock
- Snoopy
Example: Multimedia Information retrieval

```
<LivingThing,type,Class>
<Image,type,Class>
<ImageRegion,type,Class>
<Animal,type,Class>
<Dog,type,Class>
<Bird,type,Class>
<Animal,subClassOf,LivingThing>
<Dog,subClassOf,Animal>
<Bird,subClassOf,Animal>
<hasImageRegion,type,Property>
<hasImageRegion,domain,Image>
<hasImageRegion,range,ImageRegion>
<i,type,Image>
<o1,type,ImageRegion>
<o2,type,ImageRegion>
<i,hasImageRegion,o1>
<i,hasImageRegion,o2>
<woodstock,type,Bird>
<snoopy,type,Dog>
<o1,isAbout,snoopy>
<o2,isAbout,woodstock>
```

```
<hasColor,type,Property>
<hasColor,domain,ImageRegion>
<hasColor,range,Color>
<hasShape,type,Property>
<hasShape,domain,ImageRegion>
<hasShape,range,Shape>
<hasTexture,type,Property>
<hasTexture,domain,ImageRegion>
<hasTexture,range,Texture>
```
Example: Multimedia Information retrieval

media dependent properties

Object features:
- color, shape, texture, structure

media independent properties

Object Semantics Layer

Snoopy is a dog
Birds and Dogs are animals
Woodstock is a bird

u1
IsAbout (o1, Snoopy) = .8

o2

Object Form Layer

Class

Image

ImageRegion

hasImageRegion

i

o1

o2

type

domain

range

type

subClassOf

isAbout

Property

LivingThing

Animal

type

subClassOf

Dog

Bird

Snoopy

Woodstock

U. Straccia (ISTI - CNR)

DLs & SW

2007

51 / 167
Example: Multimedia Information retrieval

Query: “Find image regions about animals”
In SPARQL:  SELECT ?r 
    WHERE { ?i isABout ?o . 
            ?o type Animal } 
http://www.w3.org/TR/rdf-sparql-query/
Problems with RDFS

- RDFS too weak to describe resources in sufficient detail
  - No localised range and domain constraints
    - Can’t say that the range of hasChild is person when applied to persons and elephant when applied to elephants
  - No existence/cardinality constraints
    - Can’t say that all instances of person have a mother that is also a person, or that persons have exactly 2 parents
  - No transitive, inverse or symmetrical properties
    - Can’t say that isPartOf is a transitive property, that hasPart is the inverse of isPartOf or that touches is symmetrical
  - ...

- Difficult to provide reasoning support
  - No “native” reasoners for non-standard semantics
  - May be possible to reason via FO axiomatisation
  - Some reasoners exists: e.g., Jena

http://jena.sourceforge.net/
Web Ontology Language Requirements

Desirable features identified for Web Ontology Language:

- Extends existing Web standards
  - Such as XML, RDF, RDFS
- Easy to understand and use
  - Should be based on familiar KR idioms
- Formally specified
- Of “adequate” expressive power
- Possible to provide automated reasoning support
From RDF to OWL

Desirable features identified for Web Ontology Language:

- Two languages developed to satisfy above requirements
  - **OIL**: developed by group of (largely) European researchers (several from EU OntoKnowledge project)
  - **DAML-ONT**: developed by group of (largely) US researchers (in DARPA DAML programme)
- Efforts merged to produce **DAML+OIL**
  - Development was carried out by “Joint EU/US Committee on Agent Markup Languages”
  - Extends (“DL subset” of’) RDF
- DAML+OIL submitted to W3C as basis for standardisation
  - Web-Ontology (**WebOnt**) Working Group formed
  - WebOnt group developed **OWL** language based on DAML+OIL

http://www.w3.org/2004/OWL/
http://www.schemaweb.info/
OWL Language

- A richer ontology language
- Relations between classes,
  - e.g., disjointness
- cardinality
  - e.g. “exactly one”
- Richer typing of properties
- Characteristics of properties (e.g., symmetry)
Tradeoff between Expressive Power and Efficient Reasoning Support

- The richer the language is, the more inefficient the reasoning support becomes
- Sometimes it crosses the border of noncomputability
- We need a compromise:
  - A language supported by reasonably efficient reasoners
  - A language that can express large classes of ontologies and knowledge
Examples of Reasoning Support

- **Class membership**
  - If \( x \) is an instance of a class \( C \), and \( C \) is a subclass of \( D \), then we can infer that \( x \) is an instance of \( D \)

- **Equivalence of classes**
  - If class \( A \) is equivalent to class \( B \), and class \( B \) is equivalent to class \( C \), then \( A \) is equivalent to \( C \), too
Examples of Reasoning Support

- **Consistency**
  - $X$ instance of classes $A$ and $B$, but $A$ and $B$ are disjoint
  - This is an indication of an error in the ontology

- **Classification**
  - Certain property-value pairs are a sufficient condition for membership in a class $A$; if an individual $x$ satisfies such conditions, we can conclude that $x$ must be an instance of $A$
Uses for Reasoning

- Reasoning support is important for
  - checking the consistency of the ontology and the knowledge
  - checking for unintended relationships between classes
  - automatically classifying instances in classes

- Checks like the preceding ones are valuable for
  - designing large ontologies, where multiple authors are involved
  - integrating and sharing ontologies from various sources
Reasoning Support for OWL

- Semantics is a prerequisite for reasoning support
- Formal semantics and reasoning support are usually provided by
  - mapping an ontology language to a known logical formalism
  - using automated reasoners that already exist for those formalisms
- OWL is (partially) mapped on a description logic, and makes use of reasoners such as FaCT, RACER, Pellet
- Description logics are a subset of predicate logic for which efficient reasoning support is possible

http://www.cs.man.ac.uk/~horrocks/FaCT/
http://gaia.isti.cnr.it/~straccia
OWL Language

- **Three species of OWL**
  - **OWL full** is union of OWL syntax and RDF (Undecidable)
  - **OWL DL** restricted to FOL fragment (decidable in NEXPTIME)
  - **OWL Lite** is “easier to implement” subset of OWL DL (decidable in EXPTIME)

- **Semantic layering**
  - OWL DL within Description Logic (DL) fragment
  - DL semantics officially definitive

- **OWL DL based on** $SHIQ$ Description Logic
  - In fact it is equivalent to $SHOIN(D_n)$ DL

- **OWL DL Benefits from many years of DL research**
  - Well defined semantics
  - Formal properties well understood (complexity, decidability)
  - Known reasoning algorithms
  - Implemented systems (highly optimised)
OWL Full

- It uses all the OWL languages primitives
- It allows the combination of these primitives in arbitrary ways with RDF and RDF Schema
- OWL Full is fully upward-compatible with RDF, both syntactically and semantically
- OWL Full is so powerful that it is undecidable
- No efficient reasoning support
OWL DL (Description Logic) is a sublanguage of OWL Full that restricts application of the constructors from OWL and RDF:
- Application of OWL’s constructors’ to each other is disallowed
- Therefore it corresponds to a well studied description logic

OWL DL permits efficient reasoning support

But we lose full compatibility with RDF:
- Not every RDF document is a legal OWL DL document
- Every legal OWL DL document is a legal RDF document.
OWL Lite

- An even further restriction limits OWL DL to a subset of the language constructors
  - E.g., OWL Lite excludes enumerated classes, disjointness statements, and arbitrary cardinality

- The advantage of this is a language that is easier to grasp, for users
  - grasp, for users
  - implement, for tool builders

- The disadvantage is restricted expressivity
Upward Compatibility between OWL Species

- Every legal OWL Lite ontology is a legal OWL DL ontology
- Every legal OWL DL ontology is a legal OWL Full ontology
- Every valid OWL Lite conclusion is a valid OWL DL conclusion
- Every valid OWL DL conclusion is a valid OWL Full conclusion