TECHNISCHE UNIVERSITÄT DRESDEN

FOUNDATIONS OF SEMANTIC WEB TECHNOLOGIES

Ontology Editing

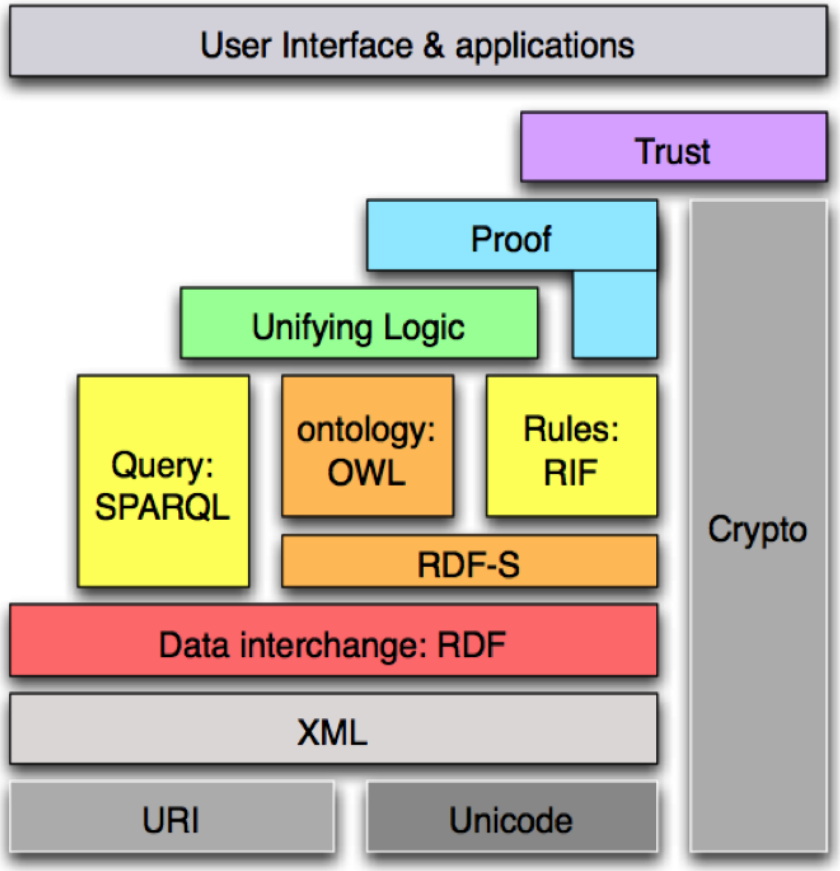
Sebastian Rudolph

Dresden, July 2

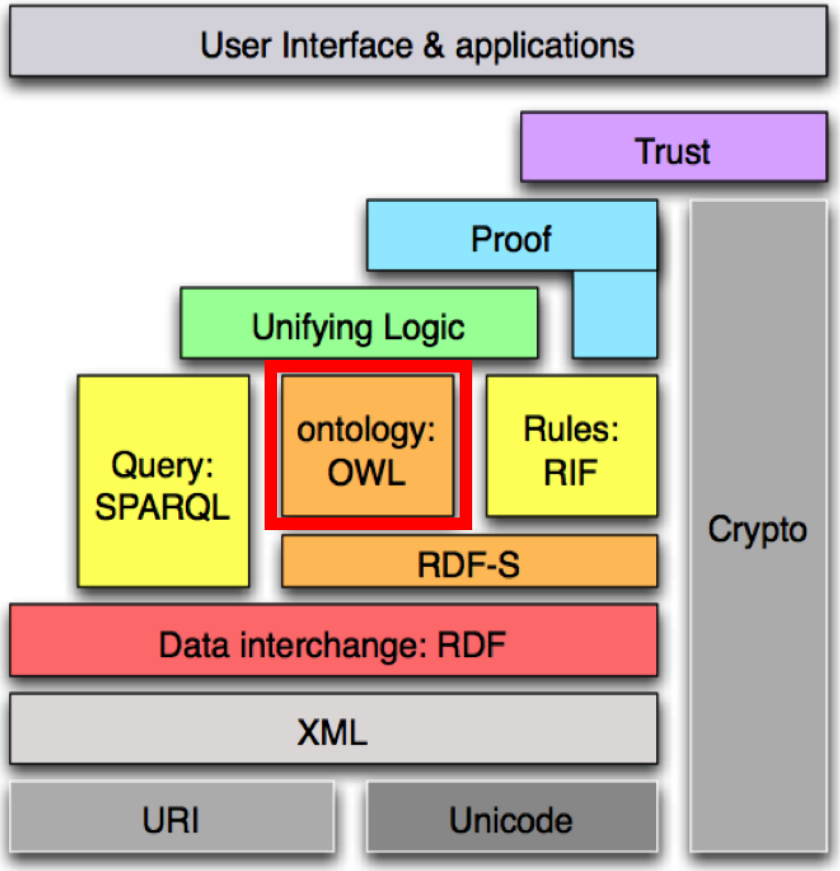
DRESDEN concept
Exzellenz aus Wissenschaft und Kultur

Content

Ontology Engineering



Ontology Engineering



Agenda

- How can we work with an ontology in a programming environment?
- How can we work with an ontology via a graphical interface?

Agenda

- How can we work with an ontology in a programming environment?
- How can we work with an ontology via a graphical interface?

Implementing OWL

What does “implementing OWL” mean?

- modeling – provision of data structures representing an ontology
- parsing – conversion of the textual representation of an ontology (e.g. in RDF/XML) into corresponding data structures
- serialization – conversion of the internal data structure into the textual representation
- manipulation – provision of methods for manipulating the data structures or creating new ontological elements
- inferencing – taking into account the OWL formal semantics

Aspects of an Implementation

- identify functionalities and responsibilities
- representation
  - syntax vs. data model
  - interface vs. implementation
  - locality of information
- parsing/serialization
  - abstraction from the concrete representation (e.g. as triples)
- manipulation
  - granularity
  - dependencies
  - intention of the user
  - strategies
- inferencing
  - distinguishing explicit and implicit consequences
  - external imnplementations

What is an Ontology (from the Implementer's Perspective)

- a certain syntactic representation?
- facts that are represented via the syntactic representation?
- information that is entailed by the represented facts?

What is an Ontology (from the Implementer's Perspective)

- a certain syntactic representation?
- facts that are represented via the syntactic representation?
- information that is entailed by the represented facts?

- these questions are important when working with OWL
- e.g.: which answers to expect when querying these data structures

The OWL API

- the OWL API is such an implementation of OWL
- meant for OWL DL
- an ontology is represented by a set of axioms, which model information about classes, roles and individuals
- when is a class or role “in” an ontology?
  - not precisely defined in OWL spec
  - somewhat depending on the implementation

Data Structures of the OWL API

- provides data structures forrepresenting ontologies
- auxiliary classes for
  - creating,
  - manipulating,
  - parsing,
  - rendering, and
  - inferencing over these structures
- the basic data structures represent objects of an ontology and are organized according to the functional syntax of OWL of OWL

The Ontology Object

- an `OWLOntology` object represents an ontology
- contains a set of axioms
- “knows” what classes and roles are used in the ontology
- may have an IRI as Name/ID and may be loaded from IRIs
  - the ontology’s ID does not have to coincide with the IRI the ontology is loaded from
  - same holds for imported ontologies

Loading an Ontology

```
OWLOntologyManager manager
    =OWLManager.createOWLOntologyManager();
OWLOntologyIRIMapper aMapper
    =new AutoIRIMapper(new File("/ont/"), false);
manager.addIRIMapper(aMapper);
IRI rem=IRI.create("http://www.ex.org/ex.owl");
IRI loc=IRI.create("file:/ont/ex.owl");
OWLOntologyIRIMapper sMapper
    =new SimpleIRIMapper(rem, loc);
manager.addIRIMapper(sMapper);
OWLOntology ont
    =m.loadOntologyFromOntologyDocument(
        IRI.create("file:/ont/ont.owl"));
```

Objects for Entities

- the class `OWLClass` represents an (OWL) class
- does not contain information about which axioms use it
- axioms referring classes belong to an `OWLOntology` object
- methods in `OWLClass` allow acessing information about the class in the context of the ontology (convenience methods)
- non-atomic classes use the interface `OWLClassExpression`
- analogous for properties: `OWLObjectProperty` and `OWLDataProperty`

Adding Axioms

```
OWLDataFactory df=m.getOWLDataFactory();
OWLClass student=df.getOWLClass(IRI.create("..."));
OWLClass course=df.getOWLClass(IRI.create("..."));
OWLObjectProperty takes
    =df.getOWLObjectProperty(IRI.create("..."));
OWLClassExpression sup
    =df.getOWLObjectSomeValuesFrom(takes, course);
OWLAxiom ax=df.getOWLSubClassOfAxiom(student, sup);
manager.addAxiom(ont, axiom);
```

Using a Reasoner

- the class `OWLReasoner` is an interface, that is implemented by most of the reasoners
- contains methods, to interact with the reasoner

```
import org.semanticweb.owlapi.reasoner.OWLReasoner;
import org.semanticweb.HermiT.Configuration;
import org.semanticweb.HermiT.Reasoner.ReasonerFactory;
...
OWLReasonerFactory reasonerFactory
    =new Reasoner.ReasonerFactory(); // HermiT factory
OWLReasoner reasoner
    =reasonerFactory.createReasoner(ont);
System.out.println(reasoner.isConsistent());
System.out.println(reasoner.isSatisfiable(student));
hermit.precomputeInferences(
    InferenceType.CLASS_HIERARCHY);
```

Visitor patterns in the OWL API

- **visitor**: Encapsulate an operation that is to be performed on the elements of an object structure as an object. The visitor pattern allows you to define a new operation without changing the classes of the targeted data structure.
  [Design Patterns, Gamma et al.]
- allows for diverse operations over the data structures without overloading them with application-specific code
- good as long as the data structures don't change
- changes in the OWL standards (very) rare

Example ClassExpressionVisitor

- visitors implement the interface `OWLClassExpressionVisitor` which needs a methof `visit()` per OWL class object
- all implementations of the interface `OWLClassExpression` must implement a method `accept(OWLClassExpressionVisitor visitor)`

```
public interface OWLClassExpressionVisitor {
    void visit(OWLObjectUnionOf ce);
    ...  }
public interface OWLObjectUnionOf
                  extends OWLClassExpression {
    public void accept(OWLClassExpressionVisitor v) {
        visitor.visit(this);  }
    ...  }
```

Applications for Visitors

allow for working with arbitrary OWL objects, for instance

- conversion of classes or axioms into NNF
- simplification of axioms
- serialization into diverse formats

Agenda

- How can we work with an ontology in a programming environment?
- How to work with an ontology via a graphical user interface?

Ontology Editing with Protege

- load mad-cow-start.owl in Protege
- familiarize yourself with the ontology

Ontology Editing mit Protege

given the following text, extend the ontology by the concepts pet+owner, dog+owner and vegetarian:

Pet owners are persons who own a pet. Dog owners are special pet owners in that they own a dog.

Vegetarians are animals that do not eat other animals and neither do they eat parts of other animals.

Ontology Editing with Protege

- classify the ontology
- you should obtain an unsatisfiable class mad+cow