Web Ontology Language OWL

- W3C recommendation since April 2004
- conceived as extension of RDFS
- three different flavours:
  - OWL Lite: based on a fragment of first-order logic (FOL)
  - OWL DL: larger fragment of FOL
  - OWL Full: “combination” of OWL DL with all features of RDFS (reification, ...)

⇝ Here: focus on OWL DL

Overview

1. Description logics
2. The KAON2 reasoner
3. Conjunctive queries
4. Semantic Web rules
5. Outlook
Description logics (DLs)

DLs are logical formalisms that correspond to certain fragments of first-order logic
- used for describing knowledge in a precise and well-defined way
  ⇝ suitable as ontology languages
- formal semantics allows for unambiguous interpretation
- reasoning with DLs typically decidable
  ⇝ fully automatic processing of knowledge
- there are many DLs, defined by their expressive features: expressivity vs. complexity

Simple description logics

Three types of modelling primitives:
- **Individuals**, describing single elements of the domain.
  E.g. eswc2006, markus, ...
- **Concepts**, describing sets of individuals.
  E.g. Conference, Human, ...
- **Roles**, describing binary relations between individuals.
  E.g. loves, participates_in, ...

Simple assertions about individuals, e.g.

\[
\text{eswc2004 : Conference} \\
\text{(markus, eswc2004) : participates_in}
\]

⇝ ABox of assertional axioms

Applications involve complex classes and relationships between them
⇝ combine roles and classes with logical operators

A simple DL: ALC

**ALC**: “Attribute Language with Complement”

- Building concepts:
  - \( C \sqcap D \) individuals in both \( C \) and \( D \)
  - \( C \sqcup D \) individuals in \( C \) or \( D \)
  - \( \neg C \) individuals not in \( C \)
  - \( \exists R.C \) individuals with some relation \( R \) to \( C \)
  - \( \forall R.C \) individuals with all relations \( R \) to \( C \)

- Stating relationships between concepts
  - \( C \sqsubseteq D \) all individuals of \( C \) are also in \( D \)
  - \( C \equiv D \) the individuals of \( C \) and \( D \) are the same

⇝ TBox of terminological axioms

Knowledge base = ABox + TBox

Example

\[
\text{Conference} \sqsubseteq \text{Event} \\
\text{Conference} \sqsubseteq \forall \text{participant}.\text{Person} \\
\text{Person} \sqsubseteq \text{Female} \sqcup \text{Male} \\
\text{eswc2006 : Conference} \\
\text{(eswc2006, markus) : participant}
\]

We would like to conclude also that
\[
\text{markus : Person}
\]

Markus is a person.
Reasoning tasks

Classical tasks for reasoning with description logics:
- **Instance checking.** Is individual $a$ in concept $C$?
- **Concept subsumption.** Is concept $C$ more general than $D$?
- **Concept satisfiability.** Does the definition of concept $C$ allow any instances of this concept?
- **Knowledge base satisfiability.** Is the combined knowledge of TBox and ABox free of contradictions?

Checking knowledge base satisfiability suffices

- Individual $a$ in concept $C$, if $a : \neg C$ leads to a contradiction
- $C$ is more general than $D$, if $x : C \sqcap \neg D$ leads to a contradiction
- $C$ is unsatisfiable if $x : C$ leads to a contradiction (with $x$ is some hitherto unused individual)

Reasoning

Most common reasoning method: **tableau calculus**
- closely related to tableaux in FOL and modal logics
- approach: try to construct a valid model for a knowledge base
  - determine whether knowledge base is satisfiable
- possible results:
  - model constructed successfully $\rightsquigarrow$ satisfiable
  - all attempts of model construction fail $\rightsquigarrow$ unsatisfiable
  - the algorithms fails to halt

Example – drawing conclusions

- **Conference $\sqsubseteq$ Event**
- **Conference $\sqsubseteq \forall$ participant.Person**
- **markus : Person**
- **(markus, eswc2006) : participant**

Isn’t this a contradiction since ESWC is no person?

- **eswc2006 : Person**

What is missing?

- **Person $\sqsubseteq \neg$Events**
- **eswc2006 : Conference**

Persons are not events.
ESWC is a conference.

The tableau calculus

- **eswc : Conference**
- **Conference $\sqsubseteq \exists$progChair.Person**
- **\exists$progChair.\top \sqsubseteq$ Event**

Can we derive **eswc : Event**?

- **eswc : \neg Event**
- **eswc : Conference**

- **eswc : \neg$conference**

- **eswc : \exists$progChair.Person**
  - **(eswc, x) : progChair**
  - **x : Person**

- **eswc : \neg$\exists$progChair.\top**

- **eswc : \neg$\forall$progChair.\bot**

- **eswc : \neg x : \bot**

- **eswc : Event**
Termination

More expressive DLs

Complexity and efficiency

Back to OWL DL

Recall: $P \subseteq NP \subseteq PSPACE \subseteq \text{EXPTIME} \subseteq \text{NEXPTIME}$

Reasoning with (many) DLs is computationally hard:
- $\mathcal{ALC}$ with empty TBox: $PSPACE$
- $\mathcal{ALC}$: $\text{EXPTIME}$
- $\mathcal{SHOIN}^+$ (OWL DL): $\text{NEXPTIME}$

However, worst case $\neq$ average case!
- Highly optimized (practically efficient) reasoners exist.
- Some more restricted DLs are tractable (= decidable in $P$)

$\text{Conference} \sqsubseteq \forall \text{participant. (Female} \sqcup \text{Male)}$

is expressed in OWL/RDF as:

```xml
<owl:Class rdf:ID="Conference">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="participant"/>
      <owl:allValuesFrom>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="Female"/>
          <owl:Class rdf:about="Male"/>
        </owl:unionOf>
      </owl:allValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```
KAON2 ontology management suite

KAON2: OWL reasoner and ontology management API

KAON2 reasoner:
- not based on tableau methods
- based on first-order resolution calculus
- goal: efficient reasoning for large ABoxes
- binaries available from http://kaon2.semanticweb.org

Reasoning in KAON2: overview

KAON2 answers queries in two processing steps:

OWL DL ABox

Transformation to disjunctive datalog program

Disjunctive engine

OWL DL TBox (w/o nominals)

OWL DL query

Datalog can be cached and reused for various queries

OWL as a fragment of FOL

OWL can be translated into first-order logic:

\[ C \sqsubseteq D \rightarrow \forall x. C(x) \rightarrow D(x) \]

\[ a : \exists R.C \rightarrow \exists x. (R(a,x) \land C(x)) \]

\[ a : \leq 1 R \rightarrow \forall x. \forall y. ((R(a,x) \land R(a,y)) \rightarrow x \approx y) \]

\[ \cdots \rightarrow \cdots \]

\[ \Rightarrow \text{application of first-order reasoning techniques possible} \]

KAON2 transformation in detail

OWL DL

Elimination of Transitivity Axioms

Translation into clauses

Disjunctive datalog

Eliminate function symbols

Saturation

Negation- & function-free logic program

Use resolution to add inferred clauses, EXPTIME!
Example

\[
\text{Person} \subseteq \exists \text{childOf}. \text{Person} \\
\text{NoSiblings} \subseteq \text{Person} \cap \forall \text{childOf}. \leq 1\text{hasChild}. \top \\
\text{Parent} \equiv \exists \text{hasChild}. \top \\
\text{childOf} \equiv \text{hasChild}^{-1}
\]

\[
\text{childof}(X, X_{i0}) :\sim \text{person}(X), \text{kaon2s}_i(X, X_{i0}). \\
\text{person}(X_{i0}) :\sim \text{person}(X), \text{kaon2s}_i(X, X_{i0}). \\
\text{person}(X) :\sim \text{noSiblings}(X). \\
\text{kaon2equal}(Y_1, Y_2) :\sim \text{noSiblings}(X), \text{childof}(X, Z). \\
\text{haschild}(Z, Y_1), \text{haschild}(Z, Y_2). \\
\text{haschild}(X, X_{i1}) :\sim \text{parent}(X), \text{kaon2s}_i(X, X_{i1}). \\
\text{parent}(X) :\sim \text{haschild}(X, Y). \\
\text{haschild}(Y, X) :\sim \text{childof}(X, Y). \\
\text{childof}(Y, X) :\sim \text{haschild}(X, Y).
\]

Complexity and efficiency

- Process query and TBox to obtain disjunctive datalog $\sim$ \text{ExpTime}
- Add ABox
- Use Datalog reasoner for query answering $\sim$ NP

Features

- TBox translation not necessary for every query
- Datalog-reasoning exploits well-known optimisation strategies (e.g. magic sets)
- Data complexity is NP
- Overall algorithm is worst-case optimal (\text{ExpTime})

KAON2: strengths and limitations

Problem: comparison to other reasoners difficult

- best results for large ABoxes, medium complexity TBoxes
  $\sim$ generally superior to tableaux algorithms
- still able to solve non-trivial TBox problems
  $\sim$ generally inferior to tableaux algorithms
- no support for nominals
- additional reasoning features: see below
- powerful API and ontology management tools: see second half of tutorial

Conjunctive queries

The knowledge base can be queried for conjunctions of
- terms $A(x)$ and $\neg A(x)$ where $A$ is a concept name, and
- terms $R(x, y)$ where $R$ is a simple role (one without transitive subroles).

The conjunctive query asks for concrete individuals that are valid fillers for the distinguished variables.

Example

\[
\exists y, z : \text{Conference}(x) \land \text{location}(x, y) \land \text{weather}(y, z) \land \neg \text{rainy}(z):
\]

"Find known conferences at some (possibly unknown) location where the weather is no rainy."

$\sim$ additional query expressivity to extend DL reasoning.
Semantic Web rules

Expressiveness of OWL is limited

**Example: uncles in OWL**

Given DL roles `parent`, `brother`, and `uncle`, one cannot describe their exact relationship, i.e.

“Someones uncle is the brother of her parent”
cannot be expressed in OWL.

⇝ Rules might add additional expressiveness:

```
parent(x, y) ∧ brother(y, z) → uncle(x, z)
```

⇝ Semantic Web Rule Language (SWRL)

DL-safe rules

**Problem**

OWL DL + SWRL is not decidable anymore.

⇝ restriction of SWRL rules

**Safety condition**

Every variable appears in a non-DL atom in the rule condition.

Example:

```
O(x), O(y), O(z), parent(x, y) ∧ brother(y, z) → uncle(x, z)
```

where `O` is not a concept from the DL knowledge base.

DL-safe rules in KAON2

```
O(x), O(y), O(z), parent(x, y) ∧ brother(y, z) → uncle(x, z)
```

What means `O(x)`?

⇝ Add fact `O(a)` for every known individual `a`.

Intuition: DL-safe rules are SWRL rules that are restricted to known individuals.

In KAON2:
- DL-safe rules can be added to the disjunctive datalog output.
- No additional pre-processing required.
- Complexity of Datalog reasoning still NP.

OWL: future development

- **Further extension of OWL DL**: OWL 1.1
  ⇝ additional expressivity with similar complexity
  [http://www-db.research.bell-labs.com/user/pfps/owl/overview.html](http://www-db.research.bell-labs.com/user/pfps/owl/overview.html)

- **Rule languages**: W3C working group “RIF”
  [http://owl-workshop.man.ac.uk/Tractable.html](http://owl-workshop.man.ac.uk/Tractable.html)

- **Tractable fragments of OWL**: interesting DLs with polynomial decision problems ⇝ e.g. Horn-\(SHIQ\), EL++, DL-Lite, ...
Practical Reasoning with OWL and DL-Safe Rules
Part II

Tools and Applications

Required Software (on CD)

- Java SDK 1.5
- Protege Version 3.1.1
  http://protege.stanford.edu/
  - Install with Option „Everything“
- KAON2
  http://kaon2.semanticweb.org/
- KAON2 OWL Tools
  http://owltools.ontoware.org/

Agenda

- Sample application: Question Answering
- KAON2 Overview
  - Protege and KAON2
  - KAON2 Demonstrator
  - KAON2 OWL Tools (command line tools)
  - KAON2 Java API
- Hands-on session

Sample Application:
Question Answering over Heterogeneous Data Sources
**Conceptual Architecture**

Knowledge Portal

- NL Queries
- ORAKEL
- KAON2 Reasoner
- SPARQL
- Result Set

Digital Library Data

- Structured Metadata
- Fulltexts / Abstracts
- Topic Hierarchies

Mappings

- Text2Onto

**Mapping Systems for Ontology Integration**

**Goal:**

Language for Specifying Semantic Relationships

- Target Ontology
- Source1
- Source2
- Source3
- Source4
- Source5

- Q1
- Q2
- Q3
- Q4
- Q5

OWL itself is rather limited in expressing mappings

**Sample Mapping**

**OWL DL Mapping System**

- An OWL DL mapping system is a triple \( (S, T, M) \), where
  - \( S \) is the source OWL DL ontology
  - \( T \) is the target OWL DL ontology
  - \( M \) is the mapping between \( S \) and \( T \)

- Mapping: set of assertions
  - \( q_S \subseteq q_T \) (sound mapping)
  - \( q_S \supseteq q_T \) (complete mapping)
  - \( q_S \equiv q_T \) (exact mapping)
  - where \( q_S \) and \( q_T \) are conjunctive queries over \( S \) and \( T \), respectively, with the same set of distinguished variables

- Semantics defined via translation into FOL, computing answers against \( S \cup T \cup M \)
Query Answering in the Mapping System

- A mapping \( q_0 \subseteq q_1 \) is equivalent to an axiom
  \[ \forall x : q_1(x, y) \leftarrow q_0(x, y) \]

- Query answering undecidable with general implication mappings

- Decidable query answering:
  - Disallow non-distinguished variables in \( q_1 \) to obtain safe rules:
    - \[ \forall x : q_1(x) \leftarrow q_0(x) \]
    - These rules correspond to SWRL rules
  - Require \( q_0 \) to be DL-safe:
    - Each variable in a DL-atom must also occur in a non-DL atom (makes queries applicable only to explicitly introduced individuals)

Ontology Engineering and Reasoning Tools

- Ontology Engineering Tools
  - Protege
  - Ontostudio
  - TopBraid
  - ...

- Ontology Reasoners
  - KAON2
  - Pellet
  - RacerPro, FaCT(++)
  - See also

OWL Reasoner: KAON2

- Language support: OWL, SWRL
  - More precisely: SHIQ(D), DL-safe rules

- Features
  - OWL parser and serializer for OWL RDF, OWL XML, Abstract Syntax
  - Interfaces for manipulation of ontologies
  - a stand-alone server providing access to ontologies in a distributed manner,
  - an inference engine for answering queries (including support for SPARQL),
  - efficient access to instances via relational databases

- Download (free for research purposes)
  - http://kaon2.semanticweb.org/

KAON2 Architecture

<table>
<thead>
<tr>
<th>API Layer</th>
<th>OWL API</th>
<th>Logic API</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reduction Layer</td>
<td>Theorem Prover for BS</td>
<td>Reduction to DD</td>
</tr>
<tr>
<td>Datalog Layer</td>
<td>Disjunctive Datalog Engine</td>
<td></td>
</tr>
<tr>
<td>Data Layer</td>
<td>Files</td>
<td>Rel. Databases</td>
</tr>
</tbody>
</table>
Interfaces to KAON2

- KAON2 in server mode via DIG and RMI
- KAON2 Demonstrator
- KAON2 Java API
- KAON2 OWL Tools (Command Line)

Getting started with Protege

- Create new project from existing source:
  SWRC ontology in ontologies/swrc.owl
The DIG interface

- Defined by the Description Logic Implementation Group
- Standardized interface for interaction with a DL reasoner
- Contains XML Schema for a DL concept language along with ask/tell functionality
- HTTP as underlying transfer protocol
- Reasoning tasks
  - Primitive Concept Retrieval
  - Satisfiability
  - Concept Hierarchy
  - Role Hierarchy
  - Individual Queries
- http://dig.sourceforge.net/

KAON2 Server mode

- `java -cp kaon2.jar org.semanticweb.kaon2.server.ServerMain -registry -rmi -ontologies server_root -dig -digport 8088`
- Make sure Protege uses the same DIG Port (Menu: OWL / Preferences)

Connecting to KAON2

Reasoning Tasks

- Check consistency of SWRC ontology …
- Create an inconsistency
  - `SubClassOf(PhDStudent, Employee)`
  - `DisjointClasses(Student, Employee)`
- Check consistency again
KAON2 Demonstrator

- Simple interface to query and ontologies and debug reasoning processes

• To start double-click kaon2.jar
  or start from command line: java –jar kaon2.jar

SPARQL

- Actually a query language for RDF...
- Operates on RDF graphs (i.e. sets of triples)
- Core concepts: Graph pattern matching

  • SQL-like syntax (similar to RDQL and SeRQL)
  • Semantics more or less up to the implementation...
  • In KAON2: SPARQL encodes conjunctive queries
    - Supports ABox queries only!

  http://www.w3.org/TR/rdf-sparql-query/

SPARQL Syntax for Conjunctive Queries

Return the titles of all publications and the names of their authors

\[ Q(t,n) := \text{Publication}(x) \land \text{title}(x,t) \land \text{author}(x,y) \land \text{name}(y,n) \]

prefix : <http://swrc.ontoware.org/ontology#>

select ?t ?n
where

SWRL Rules
KAON2 OWL Tools

• Convenient Command Line Access to KAON2

• Available Commands:
  – dump: prints the axioms (or entities) of the ontology
  – latex: prints out the axioms ready for inclusion in a latex document
  – filter: remove all the axioms of a special kind
  – count: counts the number of axioms (or entities) in the ontology
  – diff: returns all axioms in one ontology missing from the other
  – merge: returns an ontology containing all axioms from two input ontologies
  – owl2owlxml: serializes an ontology in OWL/XML presentation syntax
  – owlxml2owl2rdf: serializes an ontology in standard OWL/RDF syntax
  – deo: weakens the ontology by replacing nominals with simple classes
  – ded: removes all concrete domains from the ontology
  – populate: populates an ontology randomly with instances
  – dlpconvert: converts an DL ontology to rules
  – screech: creates a split program out of a DL ontology
  – satisfiable: checks the satisfiability of an ontology
  – shell: offers a shell to work with OWL ontologies.

• Available at http://owltools.ontoware.org/

KAON2 API

• Java Interface to the KAON2 reasoner

• Interfaces for
  – OWL parser and serializer for
    • OWL RDF
    • OWL XML
  – Manipulation of ontologies
  – Reasoning Capabilities
  – Query Capabilities

• Centered around ontologies as sets of axioms

KAON2 API Packages

• org.semanticweb.kao2.api
• org.semanticweb.kao2.api.flogic
• org.semanticweb.kao2.api.logic
• org.semanticweb.kao2.api.owl.axioms
• org.semanticweb.kao2.api.owl.elements
• org.semanticweb.kao2.api.reasoner

API: Important concepts

• Ontology
  – Represents a DL ontology
• Axiom
  – Represents an axiom in the ontology
• Entity
  – Represents a entity in a DL ontology

• OntologyResolver
  – The resolver for ontology parameters
• KAON2Connection
  – The connection to KAON2
• KAON2Factory
  – The factory for KAON2 objects
API: Reasoner

- **Reasoner**
  - Provides methods to answer queries over an ontology
- **Query**
  - Represents a conjunctive DL-safe query over an ontology
  - Can be constructed manually or via SPARQL
- **SubsumptionHierarchy**
  - Represents a subsumption hierarchy for atomic classes in an ontology
- **SubsumptionHierarchy.Node**
  - Represents a node in the subsumption hierarchy

Data Model: Elements

- **OWLEntity** Represents an entity in an OWL ontology.
  - **OWLClass** Represents a class in an ontology.
  - **Individual** Represents an individual in an ontology.
  - **Datatype** Represents a datatype in an ontology.
  - **ObjectProperty** Represents an object property in an ontology.
  - **DataProperty** Represents a data property in an ontology
  - **AnnotationProperty** Represents an annotation property in an ontology

Data Model: Axioms 1/2

- **SubClassOf**
  - a subclass axiom in an ontology.
- **DisjointClasses**
  - an axiom specifying that descriptions are disjoint.
- **EquivalentClasses**
  - an axiom specifying that descriptions are equivalent.
- **ClassMember**
  - an axiom specifying that an individual is a member of a description.
- **DifferentIndividuals**
  - an axiom specifying that individuals are different.
- **SameIndividual**
  - an axiom specifying that individuals are the same.

OWL Class Descriptions

- **Description** a class description in the ontology.
  - **OWLClass** Represents a class in an ontology.
  - **ObjectCardinality** A description specifying the cardinality of some object property.
  - **ObjectAll** A description specifying the all values of the object property are from a description.
  - **ObjectSome** A description specifying the some values of the property are from a description.
  - **ObjectHasValue** A description specifying the value of some object property.
  - **ObjectOneOf** An description built by enumerating individuals.
  - **ObjectAnd** A description specifying intersection of descriptions.
  - **ObjectOr** A description specifying union of descriptions.
  - **ObjectNot** A description specifying a complement of a description.
  - (Almost) the same for data properties
Data Model: Axioms 2/2

- **SubObjectPropertyOf**
  - specifies that one object property is a subproperty of another property.
- **EquivalentObjectProperties**
  - specifies that the object properties are equivalent.
- **InverseObjectProperties**
  - specifies that one property is an inverse property of the other property.
- **ObjectPropertyDomain**
  - specifies the domain of an object property.
- **ObjectPropertyRange**
  - specifies the range of an object property.
- **ObjectPropertyMember**
  - specifies the value of an object property for an individual.
- **ObjectPropertyAttribute**
  - specifies that an object property has some attribute (transitive, functional, ...)

---

**Rules in the KAON2 Logic API**

```
Axiom

Conjunction

Disjunction

Exists

ForAll

Literal

Equivalence

DefaultNegation
```

---

**KAON2 Logic API**

```
Formula

Obligation

Tautology

Rule

Inference

DefaultNegation
```

---

**KAON2 Examples Overview**

1. Load an ontology and print subclasses of some classes
2. Create and save an ontology
3. Retrieve elements in an ontology
4. Create a simple ontology containing rules, run queries
5. Use KAON2 built-in predicates
6. Extend KAON2 with new datatypes and with new built-in functions
7. Access the ontology server
8. Create an ontology-based view over an existing relational database
9. Invoke built-in database functions from KAON2
Hands-On Exercise

• Create a rule in the SWRC ontology:

\[
\text{expertOn}(x,z) \leftarrow \text{worksAtProject}(x,y) \land \text{isAbout}(y,z)
\]

• Write a SPARQL query that asks for names of people who are an expert on „semantic web“

Open Hands-On Exercise

• Using Protege and KAON2 Demonstrator
  – Create the rule using the SWRL rules editor, save it
  – Load ontology with rule in KAON2 Demonstrator
  – Perform SPARQL query

• Or Using KAON2 API
  – Investigate Example 4 of KAON2
  – Create, compile and run own Java program

Thank You!