Practical Reasoning with OWL and DL-Safe Rules

Peter Haase[1] Pascal Hitzler[1] Markus Krötzsch[1]

Jürgen Angele[2] Rudi Studer[123]

[1] Institute AIFB, Universität Karlsruhe
[2] ontoprise GmbH, Karlsruhe
[3] FZI Karlsruhe

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, . . .)

$\rightsquigarrow$ Here: focus on OWL DL

OWL DL

- OWL DL is based on the description logic $\mathcal{SHOIN}(D)$
- typical reasoning tasks for OWL DL are decidable
  $\rightsquigarrow$ relevant for many applications
- implementing OWL DL reasoning is difficult
- largely compatible with RDFS
  (but not containing all of RDFS)

Overview

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 $\rightsquigarrow$ suitable as ontology languages
- formal semantics allows for unambiguous interpretation
- reasoning with DLs typically decidable $\rightsquigarrow$ 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:

1. Individuals, describing single elements of the domain. E.g. *eswc2006*, *markus*, ...
2. Concepts, describing sets of individuals. E.g. *Conference*, *Human*, ...
3. Roles, describing binary relations between individuals. E.g. *loves*, *participates_in*, ...

Simple assertions about individuals, e.g.

$$\mathit{eswc2004} : \mathit{Conference}$$
$$(\mathit{markus}, \mathit{eswc2004}) : \mathit{participates\_in}$$

$\rightsquigarrow$ ABox of assertional axioms

Applications involve complex classes and relationships between them $\rightsquigarrow$ combine roles and classes with logical operators

A simple DL: $\mathcal{ALC}$

- $\mathcal{ALC}$: "Attribute Language with Complement"
- Building concepts:

$C \sqcap D$	individuals in 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 individucals of C and D are the same

$\rightsquigarrow$ TBox of terminological axioms

Knowledge base = ABox + TBox

Example

$\mathit{Conference} \sqsubseteq \mathit{Event}$	Every conference is an event.
$\mathit{Conference} \sqsubseteq \forall \mathit{participant}.\mathit{Person}$	Everybody who participates in a conference is a person.
$\mathit{Person} \sqsubseteq \mathit{Female} \sqcup \mathit{Male}$	Persons are female of male.
$\mathit{eswc2006} : \mathit{Conference}$	ESWC2006 is a conference.
$(\mathit{eswc2006}, \mathit{markus}) : \mathit{participant}$	Markus participates in ESWC.

We would like to conclude also that

$\mathit{markus} : \mathit{Person}$	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 satisfibility. Does the definition of concept C allow any instances of this concept?
- Knowledge base satisfibility. 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)

Example – drawing conclusions

$Conference \sqsubseteq Event$	Every conference is an event.
$Conference \sqsubseteq \forall participant.Person$	Everybody who participates in a conference is a person.
$markus : Person$	Markus is a person.
$(markus, escw2006) : participant$	Oops! ESWC participates in Markus?

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

$eswc2006 : Person$	Uh oh . . . ESWC is a person.

What is missing?

$Person \sqsubseteq \neg Events$	Persons are not events.
$eswc2006 : Conference$	ESWC is a conference.

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 $\rightsquigarrow$ determine whether knowledge base is satisfiable
- possible results:
  1. model constructed successfully $\rightsquigarrow$ satisfiable
  2. *all* attempts of model construction fail $\rightsquigarrow$ unsatisfiable
  3. the algorithms fails to halt

The tableau calculus

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

Can we derive $eswc : Event$?

$eswc : \neg Event$
$eswc : Conference$

→ $eswc : \neg conference$ ■

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

→ $eswc : \neg\exists progChair.\top$
$eswc : \forall progChair.\bot$
$x : \bot$ ■

→ $eswc : Event$ ■

Termination

> $markus : Person$
> $Person \sqsubseteq \exists parent.Person$

$markus : Person$

$markus : \neg Person$ ■

$markus : \exists parent.Person$
$(markus, x) : parent$
$x : Person$

$x : \neg Person$ ■

$x : \exists parent.Person$
$(x, x_2) : parent$
$x_2 : Person$

$x : \neg Person$ ■

∞

$\rightsquigarrow$ sophisticated blocking techniques needed to ensure termination

More expressive DLs

Concepts

$\mathcal{ALC}$	Boolean operators, modalities: $\sqcap, \sqcup, \neg, \forall R, \exists R$	
$\mathcal{N}$	Number restrictions	≥ 2 *has_child* ≤ 1 *has_mother*
$\mathcal{Q}$	Qualified number restr.	≥ 2 *has_child.Doctor*
$\mathcal{O}$	Nominals	$\{peter, pascal, markus\}$

Individuals

$\approx$	Same	$markus \approx m\_krötzsch$
$\not\approx$	Different	$peter \not\approx markus$

Roles

$\mathcal{H}$	Subrole hierarchy	$has\_mother \sqsubseteq has\_ancestor$
$\mathcal{I}$	Inverse roles	$has\_ancestor^- \sqsubseteq has\_offspring$
$\mathcal{S}$	($\mathcal{ALC}$ +) role transitivity	$\mathrm{Trans}(has\_ancestor)$

OWL DL: $\mathcal{SHOIN}$ + concrete domains (datatypes)

Complexity and efficiency

Recall: P $\subseteq$ NP $\subseteq$ PSPACE $\subseteq$ EXPTIME $\subseteq$ NEXPTIME

Reasoning with (many) DLs is computationally hard:

- $\mathcal{ALC}$ with empty TBox: PSPACE
- $\mathcal{ALC}$: EXPTIME
- $\mathcal{SHOIN}$ (OWL DL): NEXPTIME

However, worst case $\neq$ average case!

- Highly optimized (practically efficient) reasoners exist.
- Some more restricted DLs are tractable (= decidable in P)

Back to OWL DL

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

is expressed in OWL/RDF as:

```
<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 → Disjunctive datalog engine

OWL DL TBox (w/o nominals) → Transformation to disjunctive datalog program → Disjunctive datalog engine

OWL DL query → Transformation to disjunctive datalog program

OWL DL query → Disjunctive datalog engine

Datalog can be cached and reused for various queries

OWL as a fragment of FOL

OWL can be translated into first-order logic:

$$\begin{array}{rcl} C \sqsubseteq D & \mapsto & \forall x.C(x) \rightarrow D(x) \\ a : \exists R.C & \mapsto & \exists x.(R(a,x) \wedge C(x)) \\ a : \leq 1\, R & \mapsto & \forall x.\forall y.((R(a,x) \wedge R(a,y)) \rightarrow x \approx y) \\ \ldots & \mapsto & \ldots \end{array}$$

$\rightsquigarrow$ application of first-order reasoning techniques possible

KAON2 transformation in detail

OWL DL
$\mathcal{SHIQ}$ TBox (qualified number restr., no nominals)

→

Elimination of Transitivity Axioms
Trans(R) $\Rightarrow$ $\forall R.C \sqsubseteq \forall R.\forall R.C$

→

Translation into clauses
$\exists R.C(a) \Rightarrow R(a, f(a)) \wedge C(f(a))$

↓

Saturation
Use resolution to add infered clauses, ExpTime!

→

Eliminate function symbols
Add constants: $f(x) \Rightarrow x_f$

→

Disjunctive datalog
Negation- & function-free logic program

Example

$$
\begin{aligned}
\text{Person} &\sqsubseteq \exists \text{childOf.Person} \\
\text{NoSiblings} &\sqsubseteq \text{Person} \sqcap \forall \text{childOf.} \leq 1 \text{hasChild.}\top \\
\text{Parent} &\equiv \exists \text{hasChild.}\top \\
\text{childOf} &\equiv \text{hasChild}^{-1}
\end{aligned}
$$

$$
\begin{aligned}
\text{childof}(X, X_{f0}) &:- \text{person}(X), \text{kaon2s}_{f0}(X, X_{f0}). \\
\text{person}(X_{f0}) &:- \text{person}(X), \text{kaon2s}_{f0}(X, X_{f0}). \\
\text{person}(X) &:- \text{nosiblings}(X). \\
\text{kaon2equal}(Y_1, Y_2) &:- \text{nosiblings}(X), \text{childof}(X, Z), \\
&\qquad \text{haschild}(Z, Y_1), \text{haschild}(Z, Y_2). \\
\text{haschild}(X, X_{f1}) &:- \text{parent}(X), \text{kaon2s}_{f1}(X, X_{f1}). \\
\text{parent}(X) &:- \text{haschild}(X, Y). \\
\text{haschild}(Y, X) &:- \text{childof}(X, Y). \\
\text{childof}(Y, X) &:- \text{haschild}(X, Y).
\end{aligned}
$$

Complexity and efficiency

1. Process query and TBox to obtain disjunctive datalog ⇝ EXPTIME
2. Add ABox
3. Use Datalog reasoner for query answering ⇝ 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 (EXPTIME)

KAON2: strengths and limitations

Problem: comparison to other reasoners difficult

due to different architectures, caching mechanisms, pre-processing, ...

- best results for large ABoxes, medium complexity TBoxes
  ⇝ generally superior to tableaux algorithms
- still able to solve non-trivial TBox problems
  ⇝ 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

$\exists y, z : \mathit{Conference}(x) \wedge \mathit{location}(x, y) \wedge \mathit{weather}(y, z) \wedge \neg \mathit{rainy}(z)$:

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

⇝ 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.

$\rightsquigarrow$ Rules might add additional expressiveness:

$$parent(x, y) \wedge brother(y, z) \rightarrow uncle(x, z)$$

$\rightsquigarrow$ Semantic Web Rule Language (SWRL)

DL-safe rules

Problem

OWL DL + SWRL is not decidable anymore.

$\rightsquigarrow$ restriction of SWRL rules

Safety condition

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

Example:

$$O(x), O(y), O(z), parent(x, y) \wedge brother(y, z) \rightarrow uncle(x, z)$$

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

DL-safe rules in KAON2

$O(x), O(y), O(z), parent(x, y) \wedge brother(y, z) \rightarrow uncle(x, z)$

What means $O(x)$?

$\rightsquigarrow$ 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
  $\rightsquigarrow$ additional expressivity with similar complexity
  `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`
- Tractable fragments of OWL: intersting DLs with polynomial descision problems $\rightsquigarrow$ e.g. Horn-$\mathcal{SHIQ}$, EL++, DL-Lite, ...
  `http://www.w3.org/2005/rules/wg.html`

Practical Reasoning with OWL and DL-Safe Rules Part II

Tools and Applications

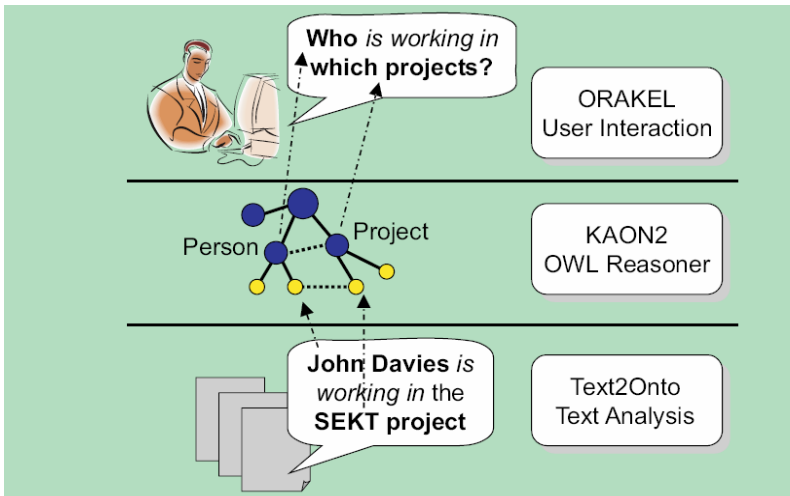
Required Software (on CD)

- 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/

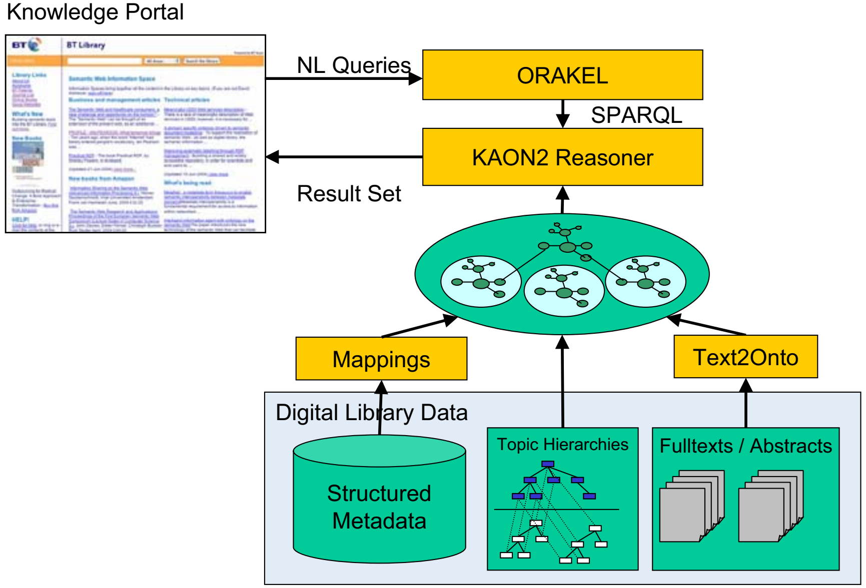
Agenda

- 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

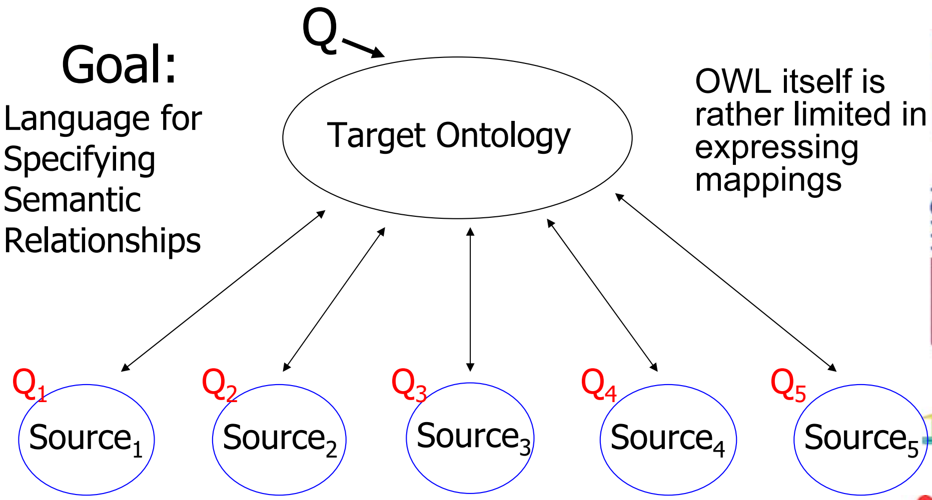


Conceptual Architecture



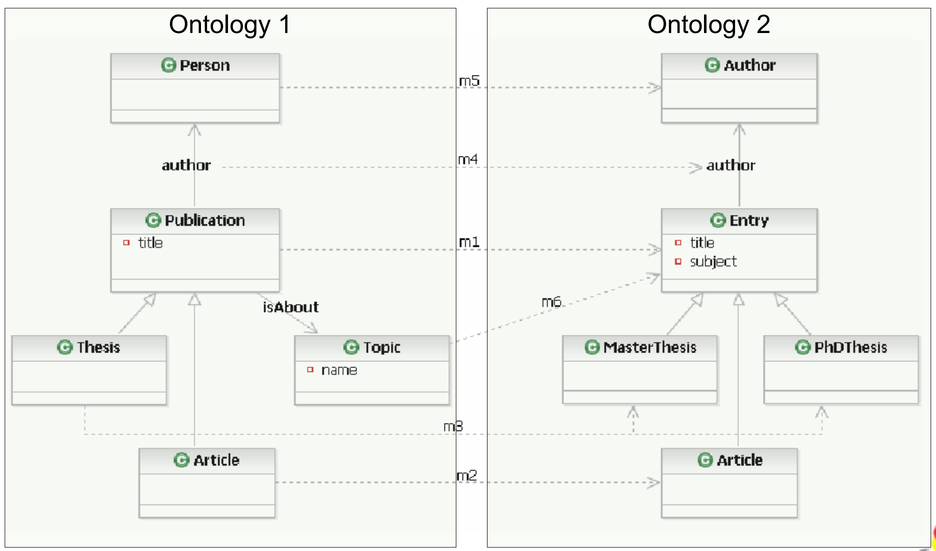
Mapping Systems for Ontology Integration

Goal:
Language for Specifying Semantic Relationships



OWL itself is rather limited in expressing mappings

Sample Mapping



OWL DL Mapping System

- An OWL DL mapping system is a triple ($\mathcal{S}$, $\mathcal{T}$, $\mathcal{M}$), where
  - $\mathcal{S}$ is the source OWL DL ontology
  - $\mathcal{T}$ is the target OWL DL ontology
  - $\mathcal{M}$ is the mapping between $\mathcal{S}$ and $\mathcal{T}$
- Mapping: set of assertions
  - $q_S \sqsubseteq q_T$ (sound mapping)
  - $q_S \sqsupseteq q_T$ (complete mapping)
  - $q_S \equiv q_T$ (exact mapping)
  - where q_S and q_T are conjunctive queries over $\mathcal{S}$ and $\mathcal{T}$, respectively, with the same set of distinguished variables
- Semantics defined via translation into FOL, computing answers against $\mathcal{S} \cup \mathcal{T} \cup \mathcal{M}$

Query Answering in the Mapping System

- A mapping $q_S \sqsubseteq q_T$ is equivalent to an axiom
  $\forall\, \mathbf{x} : q_T(\mathbf{x},\mathbf{y}_T) \leftarrow q_S(\mathbf{x},\mathbf{y}_S)$
- Query answering undecidable with general implication mappings
- Decidable query answering:
  - Disallow non-distinguished variables in q_T to obtain safe rules:
    - $\forall\, x : \mathbf{q_T}(x) \leftarrow \mathbf{q_S}(x,y_S)$
    - **These rules directly correspond to SWRL rules**
  - Require q_S 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 http://www.cs.man.ac.uk/~sattler/reasoners.html

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

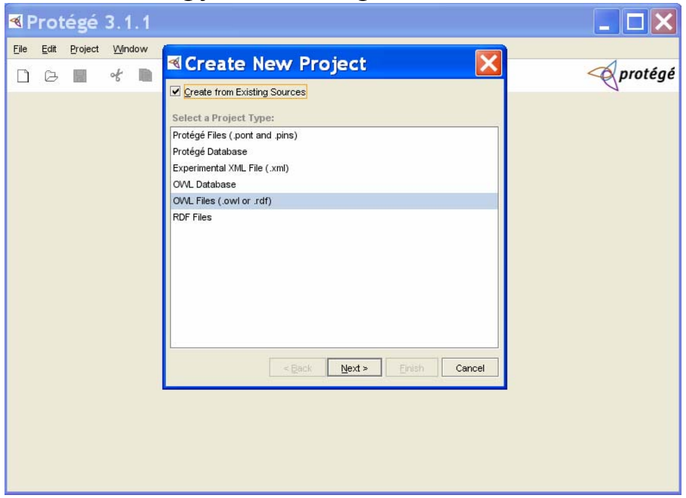
API Layer	OWL API	Logic API
Reduction Layer	Theorem Prover for BS	
	Reduction to DD	
Datalog Layer	Disjunctive Datalog Engine	
Data Layer	Files	Rel. Databases

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

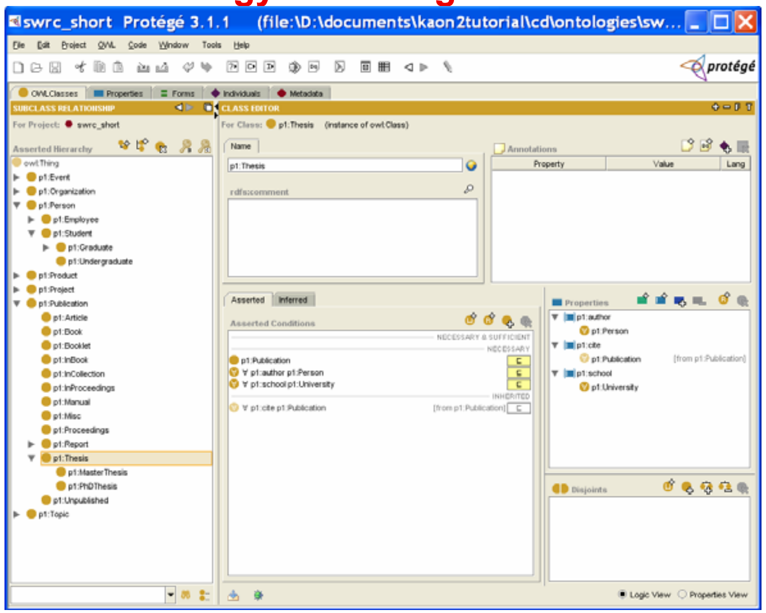
- Create new project from existing source: SWRC ontology in ontologies/swrc.owl



SWRC Ontology Overview
Semantic Web for Research Communities

- Classes (53)
  - Person
  - Publication
    - Article
    - Book
    - …
  - Project
  - Event
  - Topic
  - …
- Object properties (42)
  - affiliation
  - is about
  - works at
  - …
- Inverse object properties (10)
- Additional annotation information

SWRC Ontology in Protege



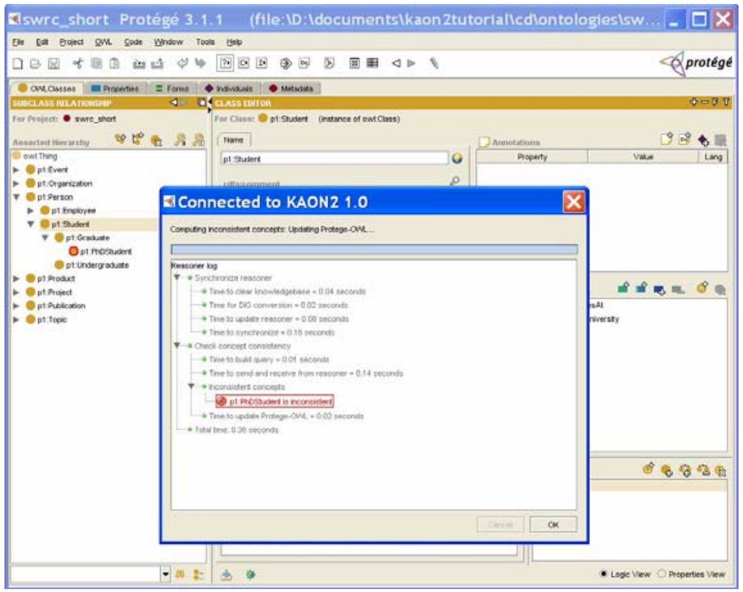
The DIG interface

- 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



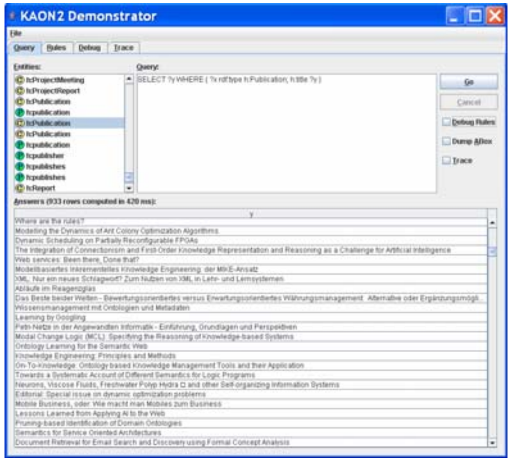
Reasoning Tasks

- 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

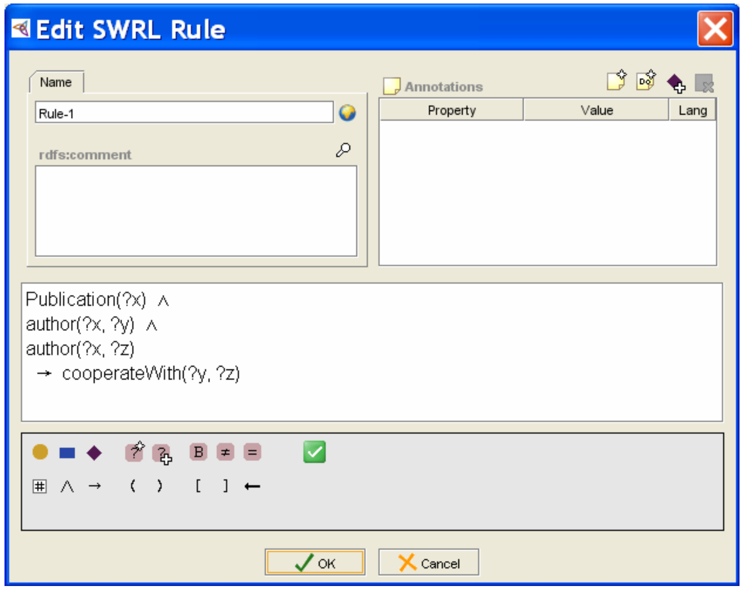
Return the titles of all publications and the names of their authors

$$Q(t,n) := Publication(x) \wedge title(x,t) \wedge author(x,y) \wedge name(y,n)$$

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

select ?t ?n
where
{ ?x rdf:type :Publication . ?x :title ?t . ?x :author ?y  . ?y :name ?n }
```


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
  - **owlrdf2owlxml**: serializes an ontology in OWL/XML presentation syntax
  - **owlxml2owlrdf**: 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/

Practical Reasoning with OWL and DL-Safe Rules • ESWC06 • June 2006

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

Practical Reasoning with OWL and DL-Safe Rules • ESWC06 • June 2006

KAON2 API Packages

- **org.semanticweb.kaon2.api**
- **org.semanticweb.kaon2.api.flogic**
- **org.semanticweb.kaon2.api.logic**
- **org.semanticweb.kaon2.api.owl.axioms**
- **org.semanticweb.kaon2.api.owl.elements**
- **org.semanticweb.kaon2.api.reasoner**

Practical Reasoning with OWL and DL-Safe Rules • ESWC06 • June 2006

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 hiearchy

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

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 1/2

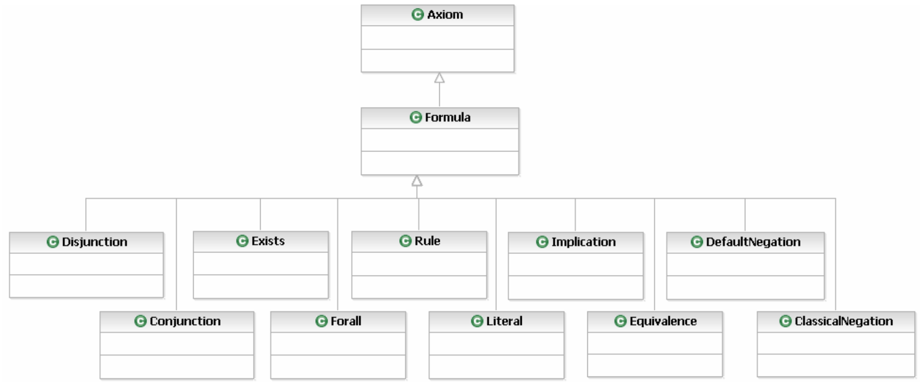
- **SubClassOf**
  - a subclass axiom in an ontology.
- **DisjointClasses**
  - an axiom specifying that descriptions are disjoint.
- **EquivalentClasses**
  - an axiom specifying that descriptions are equivalent.

- **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.

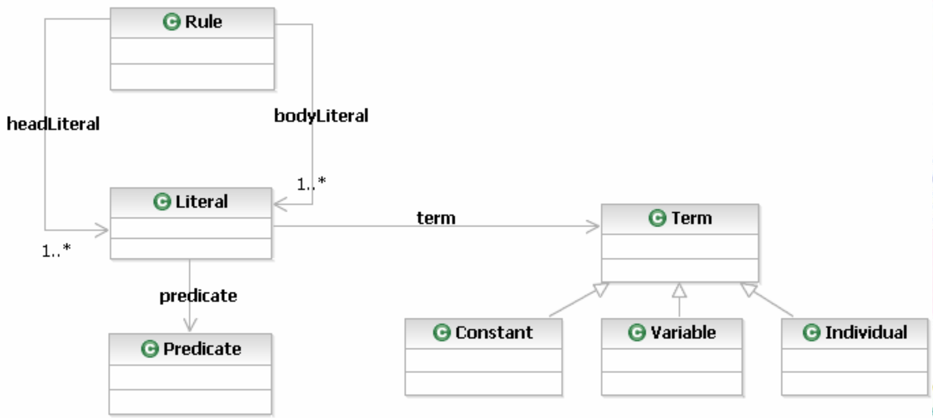
Data Model: Axioms 2/2

- **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, …)

KAON2 Logic API



Rules in the KAON2 Logic API



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 builtin 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:

$$\text{expertOn}(x,z) \leftarrow \text{worksAtProject}(x,y) \wedge \text{isAbout}(y,z)$$

- 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!