Mandarax RDF / RDFS / OWL Extension

Agenda

- Mandarax RDF
- Mandarax OWL
- OWL2Prova / Hybrid Description Logic Programs
- Key Findings and Discussion

Adrian Paschke (TUM)

AORML Workshop Cottbus, 2006

© Internet-based Information Systems
Dept. of Informatics, TU München
Mandarax RDF
Mandarax RDF Extension – Basic Concept

Mandarax Knowledge Base

Rules

Facts

Mandarax RDF ClauseSet

Mandarax RDF ClauseSet

Mandarax RDF ClauseSet

Parser / Query (Jena API)

RDF Triple Model

RDF Triple Model

RDF Triple Model

Logical Facts

RDF Triple Statements

VCard

RSS

Dublin Core

User Defined

RDF/XML

N-Triple

N3
Mandarax RDF Integration

- RDF integration based on Mandarax Clause Sets: “RDFClauseSet”

- Iterator based: “RDFClauseSetIterator”
  - Wraps Jena RDF Interface
  - RDF model with Statements “S P O” is built at query-time via Jena
  - RDF model is queried via Jena “listStatement” to select predicates
  - Result “StmtIterator” is iterated
  - During iteration a Jena statement is mapped to a fact “p(S,O)”.

- Wrapper for Predicates: RDFPredicate
  - Structure {Object, Object}
  - Slots (Subject, Predicate)
  - Key: NS+local Name

```
1. Create model from URL
2. Select Statements → StmtIterator
3. Iterate StmtIterator
   - has more statements
4. Access next clause
   - map statement to fact
5. Release model
```
RDF Mapping - Example

**RDF / XML Data**

```
<rdf:Description rdf:about="http://ibis.in.tum.de/staff/paschke/rbsla/index.htm">
  <pstcn:author>Adrian Paschke</pstcn:author>
  <pstcn:title>Rule based Service Level Agreements</pstcn:title>
  <pstcn:series rdf:resource="http://ibis.in.tum.de/staff/paschke/rbsla/index.htm"/>
  <pstcn:contains>Information on the contract management project</pstcn:contains>
  <pstcn:alsoContains>Implementations</pstcn:alsoContains>
</rdf:Description>
```

**Tripel Statements “Subject Predicate Object”**

- [http://ibis.in.tum.de/staff/paschke/rbsla/index.htm, http://burningbird.net/postcon/elements/1.0/author, "Adrian Paschke"]
- [http://ibis.in.tum.de/staff/paschke/rbsla/index.htm, http://burningbird.net/postcon/elements/1.0/title, "Rule based Service Level Agreements"]
- [http://ibis.in.tum.de/staff/paschke/rbsla/index.htm, http://burningbird.net/postcon/elements/1.0/contains, "Information on the contract management project"]
- [http://ibis.in.tum.de/staff/paschke/rbsla/index.htm, http://burningbird.net/postcon/elements/1.0/alsoContains, "Implementations"]

**Mandarax Facts “Predicate(Subject,Object)***

- http://burningbird.net/postcon/elements/1.0/author ( http://ibis.in.tum.de/staff/paschke/rbsla/index.htm, "Adrian Paschke")
- http://burningbird.net/postcon/elements/1.0/title ( http://ibis.in.tum.de/staff/paschke/rbsla/index.htm, "Rule based Service Level Agreements")

**Rules / Queries**

http://burningbird.net/postcon/elements/1.0/author ( http://ibis.in.tum.de/staff/paschke/rbsla/index.htm, Author ) ?

...
Usage Example

// Set Up KB and add RDF ClauseSet

KnowledgeBase kb = new AdvancedKnowledgeBase();
URL url = this.getClass().getResource("articles.rdf");
RDFClauseSet clauseSet = new RDFClauseSet(url);
kb.add(clauseSet);

// Query KB with RDF data

Query query1 = lfs.query(
    lfs.fact(RDFLib.pred(RDFLib.NS("pstcn"),"author"),
    lfs.variable("Article", Object.class),
    lfs.variable("Name", Object.class)),"query");
ResultSet rs = ie.query(query1,kb,InferenceEngine.ALL,InferenceEngine.BUBBLE_EXCEPTIONS);

// Print Result Set

while (rs.next()) System.out.println(
    "Author of "+rs.getResult(Object.class,"Article")+" is "+ rs.getResult(Object.class,"Name"));
Properties of the Mandarax RDF API (1)

- Pre-fetching of predicate keys
  - Needed for indexing of clause sets
  - Done at construction time of Clause Set
  - Problem: large RDF models

- Containers (SEQ, ALT, BAG) and Collections (Linked Lists)
  - Several contents
  - Triple representation with bNodes

```xml
<rdf:Description rdf:about="http://burningbird.net/earthstars/contest.htm">
  <pstcn:photos>
    <rdf:Bag>
      <rdf:li rdf:resource="http://burningbird.net/earthstars/capo.jpg"/>
      <rdf:li rdf:resource="http://burningbird.net/earthstars/baritea.jpg"/>
      <rdf:li rdf:resource="http://burningbird.net/earthstars/cfluorite.jpg"/>
    </rdf:Bag>
  </pstcn:photos>
</rdf:Description>
```
Problem querying Containers/Collections

- bNode IDs change
- unordered triples

Solution:

- Wrapper for Collections and Containers: RDFContainers, RDFCollections
- Lazy initialization during RDF2Fact mapping
- Overwrite “equals” method to compare contents at query-time
Properties of the Mandarax RDF API (2)

- **Reification**
  - Reified statement: “(Subject Property Object) Property2 Object2”
  - Translation into 5 facts
    
    ```
    type(StmtID, Statement)
    subject(StmtID,Subject)
    object(StmtID,Object)
    property(StmtID,Property)
    property2(StmtID,Object2)
    ```

- **Namespaces**
  - Fully supported
  - Predefined namespaces in RDFConstants (RDF, XMLS, DC, VCARD, RMS …)
  - Namespace Map + mapping method: RDFLib.NS(prefix) returns namespace
    
    ```
    RDFLib.NS("rdf") + "type" expands to "http://www.w3.org/1999/02/22-rdf-syntax-ns#type"
    ```

- **RDFLib (under rdf.lib)**
  - Factory methods for creating RDF content: subj(URI), obj(URI), pred(URI), lit(value)

- **Caching of Facts and Models** (cache timeout can be set)

- **TestCases + Examples** (test.org.mandarax.rdf / org.mandarax.rdf.examples)
Mandarax OWL
Mandarax OWL

- Provides inference support for RDFS, OWL (Lite, DL), DAML, SWRL??
- Build on top of Mandarax RDF
- Two basic approaches:
  1. External reasoner (Jena) to infer model and answer queries
  2. Built-In inference rules on top of Mandarax RDF facts
- OWLClauseSet and OWLClauseIterator wrap RDFClauseSet and RDFClauseIterator
Approach 1: External Reasoner

Mandarax Knowledge Base

- Rules
- Facts
  - Mandarax OWL ClauseSet
  - Mandarax OWL ClauseSet
  - Mandarax OWL ClauseSet

Reasoner (Jena, Pellet, Racer ...)

- Inferred Model
- Inferred Model
- Inferred Model

Extended RDFClauseSet

- Parser (Jena API)
  - RDF Triple Model
  - RDF Triple Model
  - RDF Triple Model

External Data

- RDF/XML
- N-Triple
- N3
Mandarax OWL with External Reasoner

Mandarax OWL support:
- Transitive reasoner: traversing class and property lattices
- RDFS reasoner: RDFS entailments
- OWL reasoners: OWL Lite entailments
- OWL DL (via Pellet)
- SWRL (planned within Pellet)

Inferred model is build on top of default RDF model at query time

Caching of inferred models supported and useful !!!!
- Each subgoal derivation creates new Iterator with new inferred model
- Needs efficient garbage collector in resolution implementation
- Close() releases resources

Special treatment of properties needed for reasoning over user-defined properties
- Mandarax RDF transformation “Class1 Property Class1” => property(Class, Class) excludes free property queries: “c1 Property? C2”
- Solution: Additional mapping in OWLClauseterator to “property(P, C1, C2)”
- Note:
  - property(p,C1,C2)? queries are internally transferred to normal queries p(C1,C2)?, i.e. restricted search space
  - property(P,C1,C2)? queries with free property variable are applied on the complete model, i.e. more expensive

Problem: RDFPredicates in rule heads not entailed in original model
Approach 2: Built-In Inference Rules

Mandarax Knowledge Base

Inference Rules

RDFS / OWL Axioms / Closure Rules

Logical Facts

Mandarax RDF ClauseSet

Mandarax RDF ClauseSet

Mandarax RDF ClauseSet

Parser (Jena API)

RDF Triple Model

RDF Triple Model

RDF Triple Model

RDF Triple Statements

RDF/XML

N-Triple

N3

VCARD

RSS

Dublin Core

User Defined
Approach 2: Built-In Inference Rules

- Based on a plug-in module concept
  - A set of inference rules is defined as a module which can be plugged-in the KB
  - Helps to control the level of inference

- Implemented in OWLLib (owl.lib)
  - Defines reusable RDF/RDFS/OWL predicates: TYPE, SUBCLASSOF, DOMAIN etc.
  - Factory methods for creating RDF/RDFS/OWL content
    - `instance()` create instances / individual
    - `property()` create properties
    - `clazz()` create classes

- Currently 5 plug-ins:
  - Transitive Inference Rules (subclassOf, subPropertyOf)
  - RDFS Axioms
  - RDFS Inference Rules
  - OWL Axioms
  - OWL Inference Rules
Inference Rules - Examples

Inference Rules

- RDFS Inference

\[
\text{rdfs:domain}(P, C), \text{property}(P, X, Y) \rightarrow \text{rdf:type}(X, C) \\
\text{rdfs:range}(P, C), \text{property}(P, X, Y) \rightarrow \text{rdf:type}(Y, C) \\
\text{rdfs:subPropertyOf}(A, B), \text{rdfs:subPropertyOf}(B, C) \rightarrow \text{rdfs:subPropertyOf}(A, C) \\
\text{rdfs:subClassOf}(A, B), \text{rdfs:subClassOf}(B, C) \rightarrow \text{rdfs:subClassOf}(A, C) \\
\text{rdfs:subPropertyOf}(P, Q), \text{property}(P, A, B) \rightarrow \text{property}(Q, A, B)
\]

- OWL Inference

\[
\text{owl:equivalentClass}(P, Q) \rightarrow \text{rdfs:subClassOf}(P, Q) \\
\text{owl:equivalentClass}(P, Q) \rightarrow \text{rdfs:subClassOf}(Q, P) \\
\text{rdfs:subClassOf}(P, Q), \text{rdfs:subClassOf}(Q, P) \rightarrow \text{owl:equivalentClass}(P, Q)
\]

\[
\text{owl:disjointWith}(C, D), \text{rdf:type}(X, C), \text{rdf:type}(Y, D) \rightarrow \text{owl:differentFrom}(X, Y)
\]

\[
\text{rdf:type}(P, \text{owl:SymmetricProperty}), \text{property}(P, X, Y) \rightarrow \text{property}(P, Y, X)
\]

\[
\text{owl:equivalentProperty}(P, Q) \rightarrow \text{rdfs:subPropertyOf}(P, Q) \\
\text{owl:equivalentProperty}(P, Q) \rightarrow \text{rdfs:subPropertyOf}(Q, P) \\
\text{rdfs:subPropertyOf}(P, Q), \text{rdfs:subPropertyOf}(Q, P) \rightarrow \text{owl:equivalentProperty}(P, Q)
\]

\[
\ldots
\]
Axioms - Examples

■ RDFS Axioms

- \( \texttt{rdf:type} \ \texttt{rdfs:range} \ \texttt{rdfs:Class} \).
- \( \texttt{rdfs:subPropertyOf} \ \texttt{rdfs:domain} \ \texttt{rdf:Property} \).
- \( \texttt{rdfs:Resource} \ \texttt{rdf:type} \ \texttt{rdfs:Class} \).
- \( \texttt{rdfs:Literal} \ \texttt{rdf:type} \ \texttt{rdfs:Class} \).

... 

■ OWL Axioms

- \( \texttt{owl:equivalentProperty} \ \texttt{rdf:type} \ \texttt{owl:SymmetricProperty} \).
- \( \texttt{owl:equivalentProperty} \ \texttt{rdf:type} \ \texttt{owl:TransitiveProperty} \).
- \( \texttt{owl:equivalentClass} \ \texttt{rdf:type} \ \texttt{owl:SymmetricProperty} \).
- \( \texttt{owl:equivalentClass} \ \texttt{rdf:type} \ \texttt{owl:TransitiveProperty} \).
- \( \texttt{owl:sameIndividualAs} \ \texttt{rdf:type} \ \texttt{owl:SymmetricProperty} \).
- \( \texttt{owl:sameIndividualAs} \ \texttt{rdf:type} \ \texttt{owl:TransitiveProperty} \).

...
Key findings: Built-in inference rules

- Loop Checking needed
  - SLD (with negation as finite failure): subclassOf(A,B), subclassOf(B,C) → subclassOf(A,C)
  - Manadarax LoopChecker does not work ??
  - Simple loop checker implemented in OWLLib
  - Better solution: extend resolution, other semantics, e.g. well-founded

- Performance most of the time slower than with external reasoner
  - Mandarax resolution needs to be improved

- Currently: Incomplete implementation in particular for OWL
  - First test with some inference rules;

- Useful for transitive reasoning: transitive plug-in
  - e.g. typed logic, i.e. using RDFS taxonomies as type system

- Useful for reasoning of derived RDF predicates and updates
  - e.g. subclassOf(A,B) :- body.

- Useful for combination with other logics, e.g. defeasible priority reasoning in case of ontology conflicts.

- Better control of inference layer

- Further optimization can be applied, e.g.
  - Bound testing and rule variants
    - bound(C) rdfs:domain(P, C), property(P, X, Y) -> rdf:type(X, C)
    - free(C) property(P, X, Y), rdfs:domain(P, C) -> rdf:type(X, C)
  - Description logic programs (DLP)
    - subclassOf(C,D) maps to C(x) -> D(x) => reduces search space
  - ....

- Problems: functional properties, existential restrictions, cardinality restrictions
  - Needs rules with equality in the rule heads and existential restrictions and counters of variable bindings
OWL2Prova
Flexible framework to reason over RDFS / OWL data

- Precompilation
  - Converters are used to translate RDF statements into arbitrary output facts
  - SimpleConverter translates to arbitrary outputs according to user-defined patterns, e.g:
    - ["predicate","subject","object"] => predicate(subject,object)
    - ["rdf", "subject","object"] => rdf(subject,object)
    - ["rdfTriple","subject","predicate","object"] => rdfTriple(subject,predicate,object)
  - DLPConverter translates to description logic programs:
    - C(X) → D(x) class C is subclassOf class D
    - P(x,y) → Q(x,y) property P is a subproperty of property Q
    - P(x,y) → C(y) range of property P is class C
    - C(X)→D(X)
    - D(X)→C(X) Class C and Class D are equivalent
    - etc.

  - **Needs loop checker** – OWL2Prova adds a kind of memoization of critical subgoals in order to prevent loops

- Further user-defined converters can be easily added, extending the Converter interface

- Supports different reasoners such as Transitive, RDFS, OWL, OWL Mini etc. to infer a model (precompile) and translate it

- The precompiled and translated models can be consulted into an existing KB
Special dynamic query predicate

“rdf(file, reasoner, subject, predicate, object)"
“rdf(url, subject, predicate, object)"

Wraps external reasoner (Jena / Pellet) which is used for query-answering

Result list with variable bindings can be iterated and used in the further resolution process

Advantages

- Variables over predicates / properties possible ➔ enables reasoning
- No need to pre-fetch predicate names for indexing
- Only one RDF predicate must be indexed – no need for indexing (prefetching) predicate keys
- Note: Search space will be constrained by subject, predicate, object

Might be useful for Mandarax RDF / OWL  ???
Typed Hybrid Description Logic Programs

- Build on OWL2Prova Integration of Jena API and Pellet Reasoner

- Hybrid Description Logic Programs
  - LP reasoner for hybrid DL-typed rules
  - External reasoner (DL reasoner Pellet)

- Types Systems: Order Sorted OWL / RDFS Type Systems (Ontologies)

- Polymorphic Order-Sorted Unification Algorithm in ContractLog KR

- Terms are typed with type from Semantic Web ontology

- Subsumption reasoning for type checking

- Instance reasoning for Individual → Class / Type reasoning

- Decidable with Datalog restriction

- Exptime (OWL-Lite) resp. NEXPTIME (OWL-DL)

- More Information:
Mandarax RDF / OWL enables rules on top of Semantic Web Data

- Combine rules and description logics
  - Negation
  - Semantic Web taxonomies for term typing in rules
  - Use logical concepts to handle conflicts and incomplete knowledge, e.g. defeasible logic
  - Etc.

Mandarax RDF

- Provides basic RDF integration
- Optimizations possible

Mandarax OWL

- Provides built-in inference rules and external reasoner
- Various optimization can be applied
- Test cases are needed (e.g. W3C tests)
- External “reasoner approach” more efficient
Thank you for attention !!!!

Discussion