

Mandarax RDF / Mandarax OWL
23.2.2006

IBIS, TU München
+49 89 289 17534
<http://ibis.in.tum.de>



Mandarax RDF / RDFS / OWL Extension

Agenda

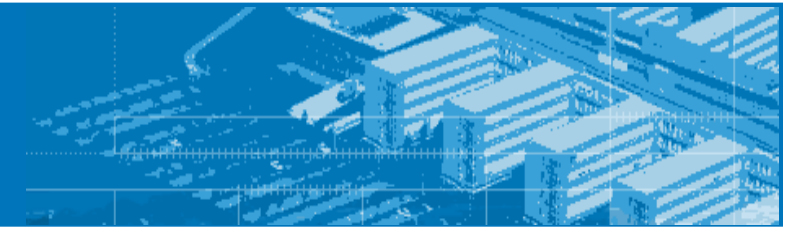
- Mandarax RDF
- Mandarax OWL
- OWL2Prova / Hybrid Description Logic Programs
- Key Findings and Discussion

Adrian Paschke (TUM)

AORML Workshop Cottbus, 2006

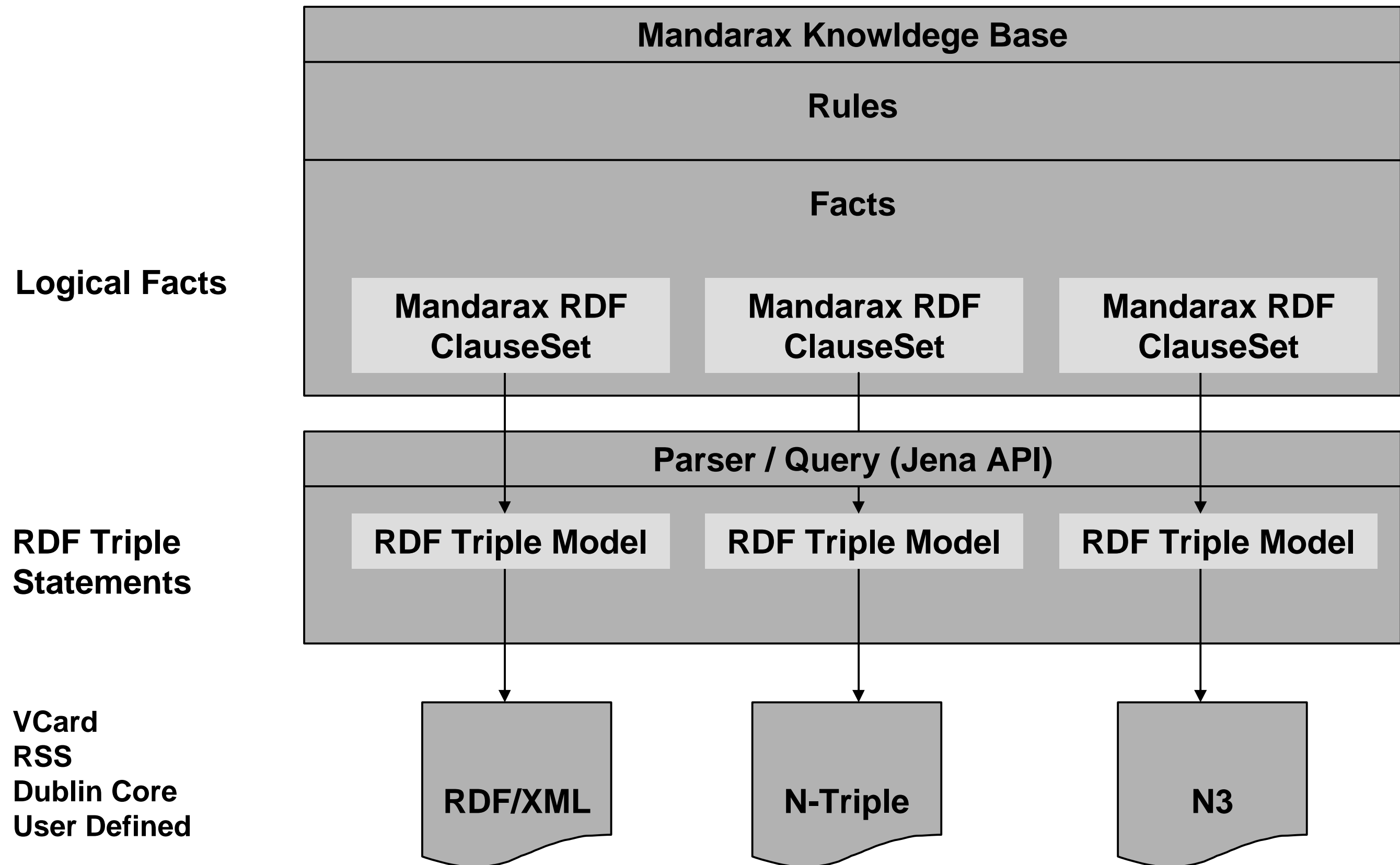
© Internet-based Information Systems
Dept. of Informatics, TU München

IBISTUM



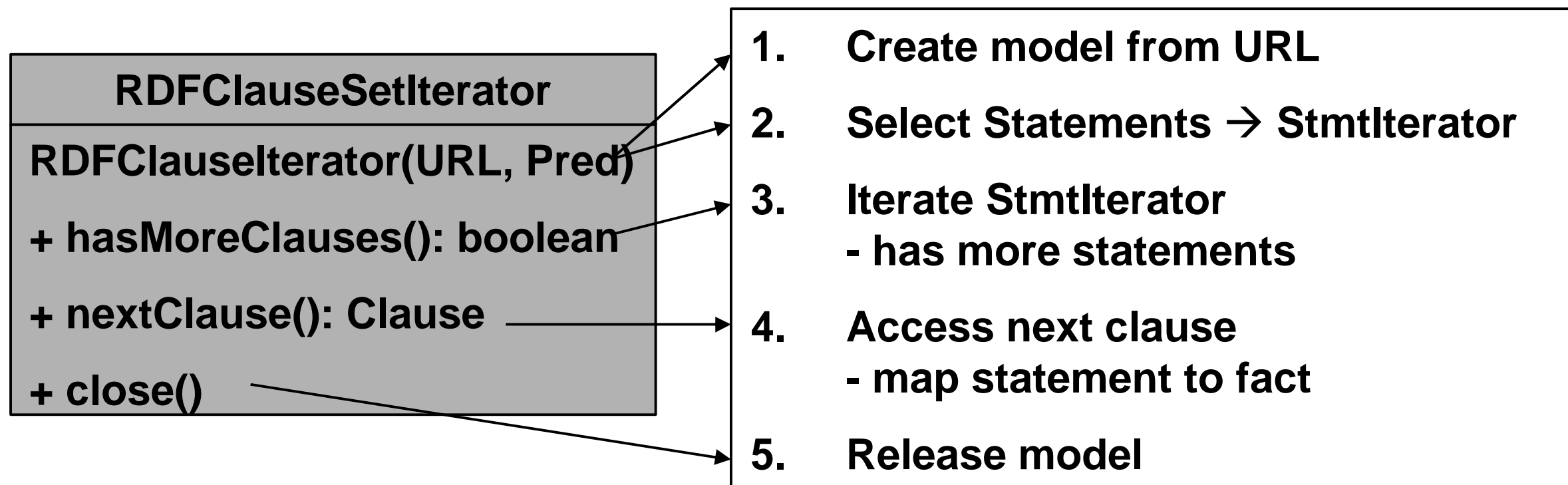
Mandarax RDF

Mandarax RDF Extension – Basic Concept



Mandarax RDF Integration

- RDF integration based on Mandarax Clause Sets: “RDFClauseSet”
- Iterator based: “RDFClauseIterator”
 - Wraps Jena RDF Interface
 - RDF model with Statements “S P O” is build at query-time via Jena
 - RDF model is queried via Jena “listStatement” to select predicates
 - Result “StmntIterator” is iterated
 - During iteration a Jena statement is mapped to a facts “p(S,O)”.
- Wrapper for Predicates: RDFPredicate
 - Structure {Object, Object}
 - Slots (Subject, Predicate)
 - Key: NS+local Name



RDF Mapping - Example

Rules / Queries

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

...

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")`

...

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/series, http://ibis.in.tum.de/staff/paschke/rbsla/index.htm]`

`[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"]`

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>
```

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 = ifs.query(  
    ifs.fact(RDFLib.pred(RDFLib.NS("pstcn"), "author"),  
    ifs.variable("Article", Object.class),  
    ifs.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

```
<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:li rdf:resource="http://burningbird.net/earthstars/ccinnibar.jpg" />
      <rdf:li rdf:resource="http://burningbird.net/earthstars/baryto.jpg" />
      <rdf:li rdf:resource="http://burningbird.net/earthstars/cbarite2a.jpg" />
    </rdf:Bag>
  </pstcn:photos>
</rdf:Description>
```

Containers / Collections (2)

[9f7e9a5:1098836ea34:-8000, http://www.w3.org/1999/02/22-rdf-syntax-ns#_2, <http://burningbird.net/earthstars/baritea.jpg>]

[9f7e9a5:1098836ea34:-8000, <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>, <http://www.w3.org/1999/02/22-rdf-syntax-ns#Bag>]

[<http://burningbird.net/earthstars/contest.htm>, <http://burningbird.net/postcon/elements/1.0/photos>, 9f7e9a5:1098836ea34:-8000]

[9f7e9a5:1098836ea34:-8000, http://www.w3.org/1999/02/22-rdf-syntax-ns#_1, <http://burningbird.net/earthstars/capo.jpg>]

■ 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

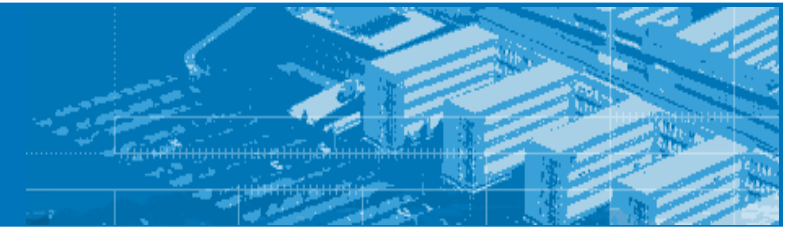
- Fully supported
- Predefined namespaces in RDFConstants (RDF, XMLS, DC, VCARD, RMS ...)
- Namespace Map + mapping method: RDFLib.NS(prefix) returns namespace
RDFLib.NS(“rdf”)+”type” expands to “http://www.w3.org/1999/02/22-rdf-syntax-ns#type”

■ RDFLib (under rdf.lib)

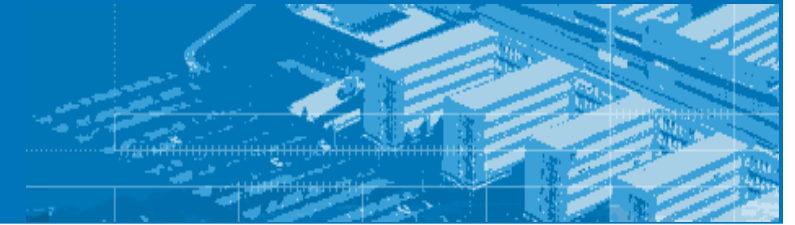
- Factory methods for creating RDF content: subj(URI),obj(URI),pred(URI),lit(value)

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

■ TestCases + Examples (test.org.mandarax.rdf / org.mandarax.rdf.examples)

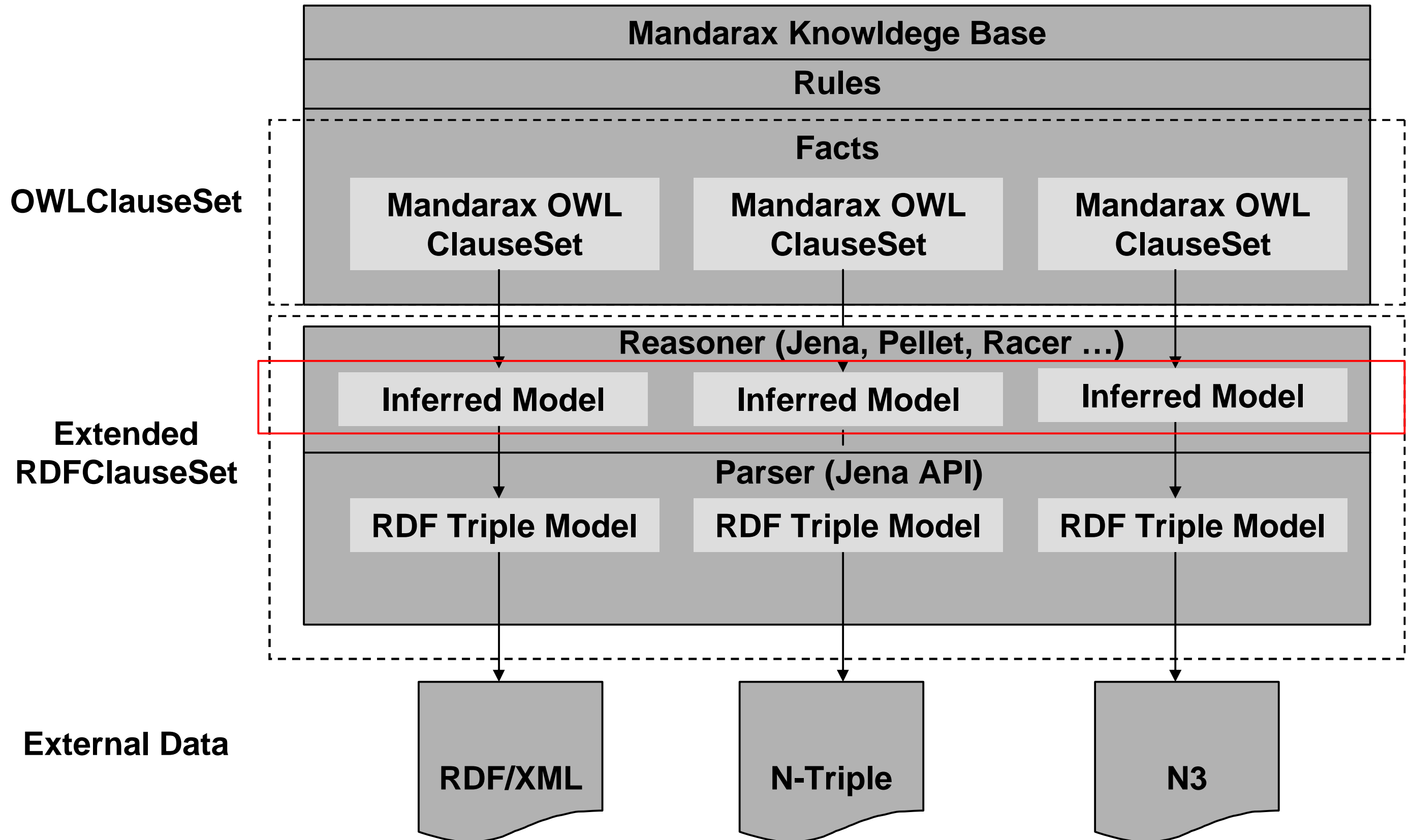


Mandarax OWL



- 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

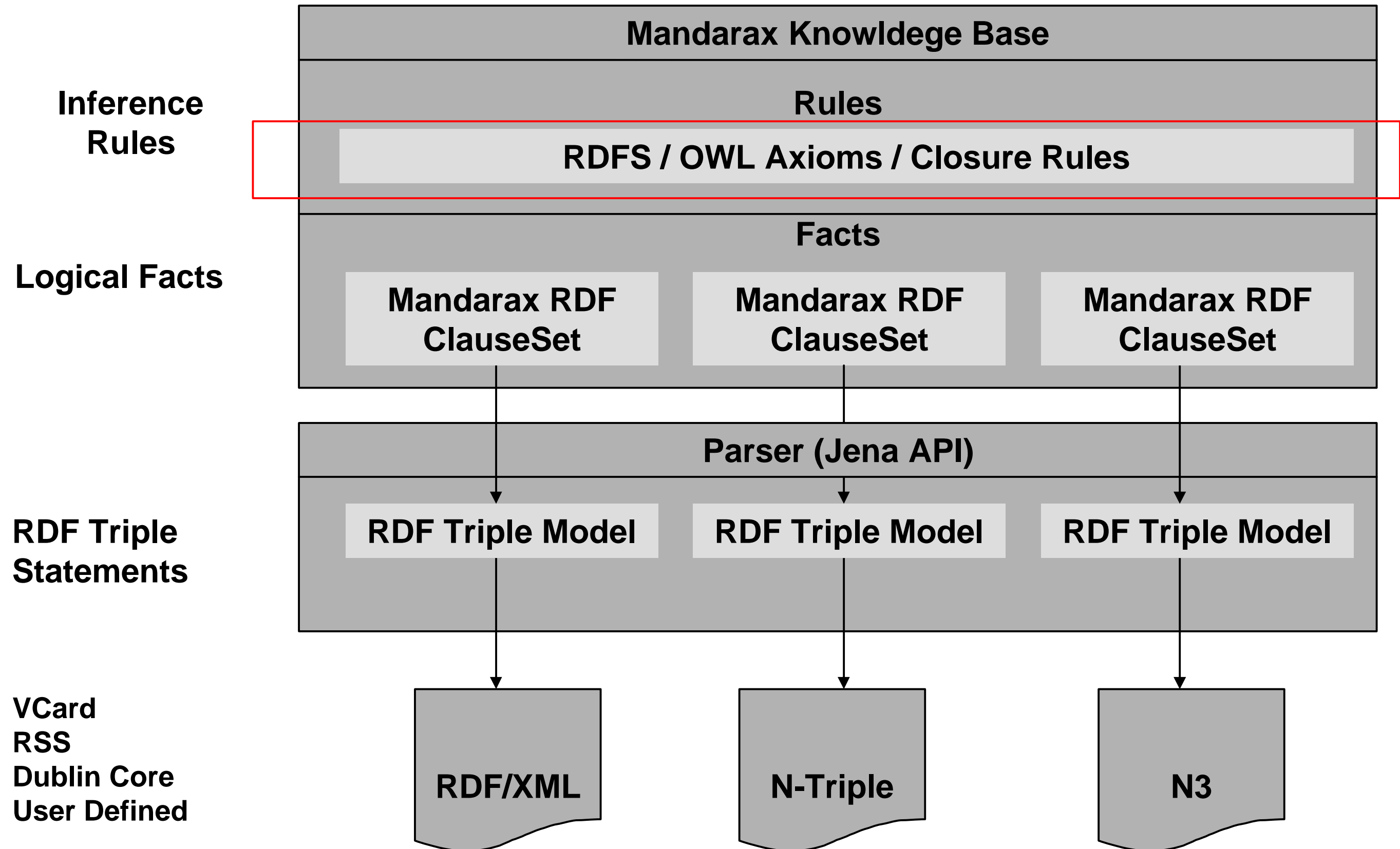
Approach 1: External Reasoner



Mandarax 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 OWLClauseIterator 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



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

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

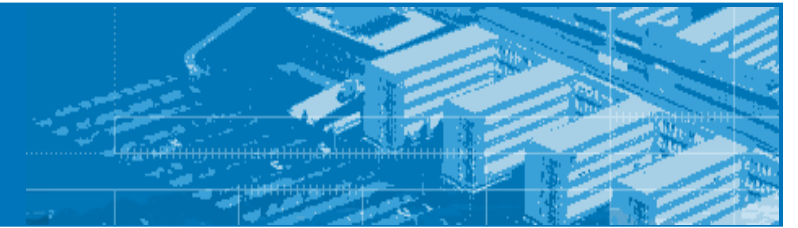
 $\text{owl:disjointWith}(C, D), \text{rdf:type}(X, C), \text{rdf:type}(Y, D) \rightarrow \text{owl:differentFrom}(X, Y)$

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

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

...

Axioms - Examples



■ RDFS Axioms

- > (rdf:type rdfs:range rdfs:Class).
- > (rdfs:subPropertyOf rdfs:domain rdf:Property).
- > (rdfs:Resource rdf:type rdfs:Class).
- > (rdfs:Literal rdf:type rdfs:Class).

...

■ OWL Axioms

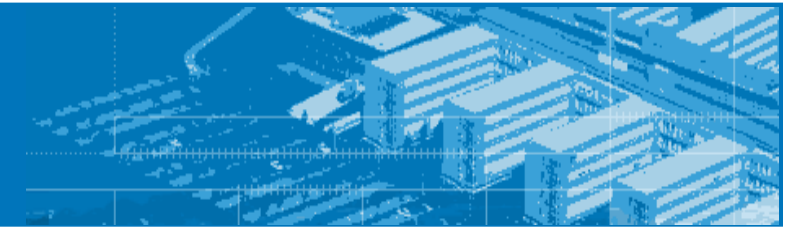
- > (owl:equivalentProperty rdf:type owl:SymmetricProperty).
- > (owl:equivalentProperty rdf:type owl:TransitiveProperty).
- > (owl:equivalentClass rdf:type owl:SymmetricProperty).
- > (owl:equivalentClass rdf:type owl:TransitiveProperty).
- > (owl:sameIndividualAs rdf:type owl:SymmetricProperty).
- > (owl:sameIndividualAs rdf:type owl:TransitiveProperty).

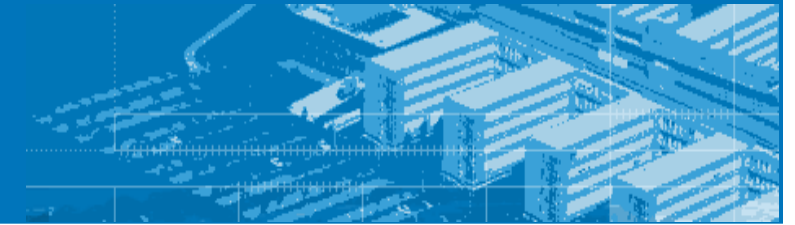
...

Key findings: Built-in inference rules

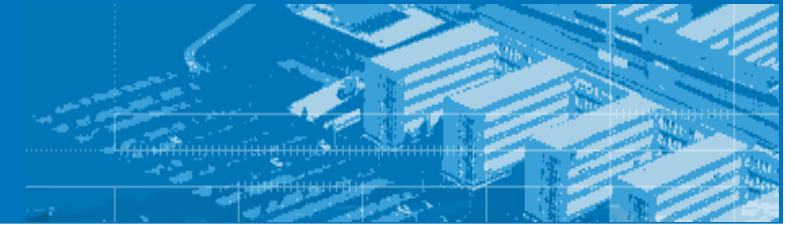
- Loop Checking needed
 - SLD (with negation as finite failure): $\text{subclassOf}(A,B), \text{subclassOf}(B,C) \rightarrow \text{subclassOf}(A,C)$
 - Mandarax LoopChecker does not work ??
 - Simple loop checker implemented in OWLLib
 - Better solution: extend resolution, other semantics, e.g. well-founded
- 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. $\text{subclassOf}(A,B) :- \text{body}.$
- 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
 - ◆ $\text{bound}(C) \text{ rdfs:domain}(P, C), \text{property}(P, X, Y) \rightarrow \text{rdf:type}(X, C)$
 - ◆ $\text{free}(C) \text{ property}(P, X, Y), \text{rdfs:domain}(P, C) \rightarrow \text{rdf:type}(X, C)$
 - Description logic programs (DLP)
 - ◆ $\text{subclassOf}(C,D) \text{ maps to } C(x) \rightarrow D(x) \Rightarrow \text{reduces search space}$
 -
- 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)
 - DLPCConverter translates to description logic programs:
 - ◆ $C(X) \rightarrow D(x)$ class C is subclassOf class D
 - ◆ $P(x,y) \rightarrow Q(x,y)$ property P is a subproperty of property Q
 - ◆ $P(x,y) \rightarrow C(y)$ range of property P is class C
 - ◆ $C(X) \rightarrow D(X)$
 - ◆ $D(X) \rightarrow C(X)$ 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

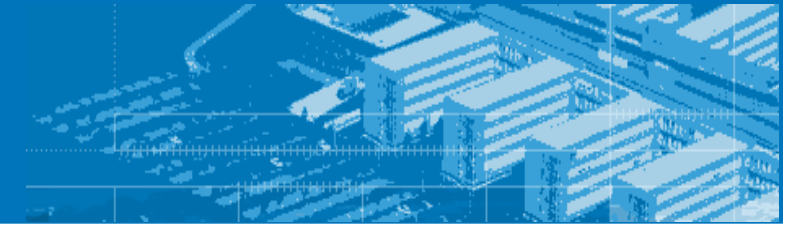


- Special dynamic query predicate
 - “*rdf(file, reasoner, subject, predicate, object)*”
 - “*rdf(url, subject, predicate, object)*”
- Wraps external reasoner (Jena / Pellet) which is used for query-answering
- Result list with variable bindings can be iterated and used in the further resolution process
- 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

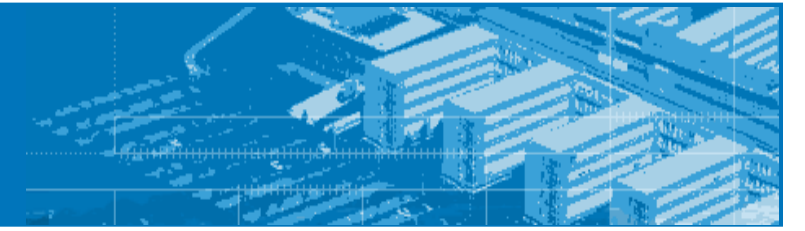
- Build on OWL2Prova Integration of Jena API and Pellet Reasoner
- Hybrid Description Logic Programs
 - LP reasoner for hybrid DL-typed rules
 - External reasoner (DL reasoner Pellet)
- Types Systems: Order Sorted OWL / RDFS Type Systems (Ontologies)
- Polymorphic Order-Sorted Unification Algorithm in ContractLog KR
- Terms are typed with type from Semantic Web ontology
- Subsumption reasoning for type checking
- Instance reasoning for Individual \rightarrow Class / Type reasoning
- Decidable with Datalog restriction
- Exptime (OWL-Lite) resp. NEXPTIME (OWL-DL)
- More Information:
 - Paschke, A.: Typed Hybrid Description Logic Programs with Order-Sorted Semantic Web Type Systems based on OWL and RDFS, Internet Based Information Systems, Technical University Munich, Technical Report 12/05, 2005.



- **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