Processing OWL2 ontologies using Thea: An application of logic programming

Vangelis Vassiliadis[1]
Jan Wielemaker[2]
Chris Mungall[3]

[1] semanticweb.gr
[2] VU University Amsterdam
[3] Lawrence Berkeley National Laboratory

What is Thea

- Prolog library for querying and processing OWL2 Ontologies.
- OWL2 axioms as Prolog facts based on the OWL functional syntax.
- Use of Prolog as an application programming language (host language), rather than as an OWL reasoning engine
- Extensions / libraries to support:
  - java OWL API
  - OWLLink servers
  - SWRL
  - translation to DLP

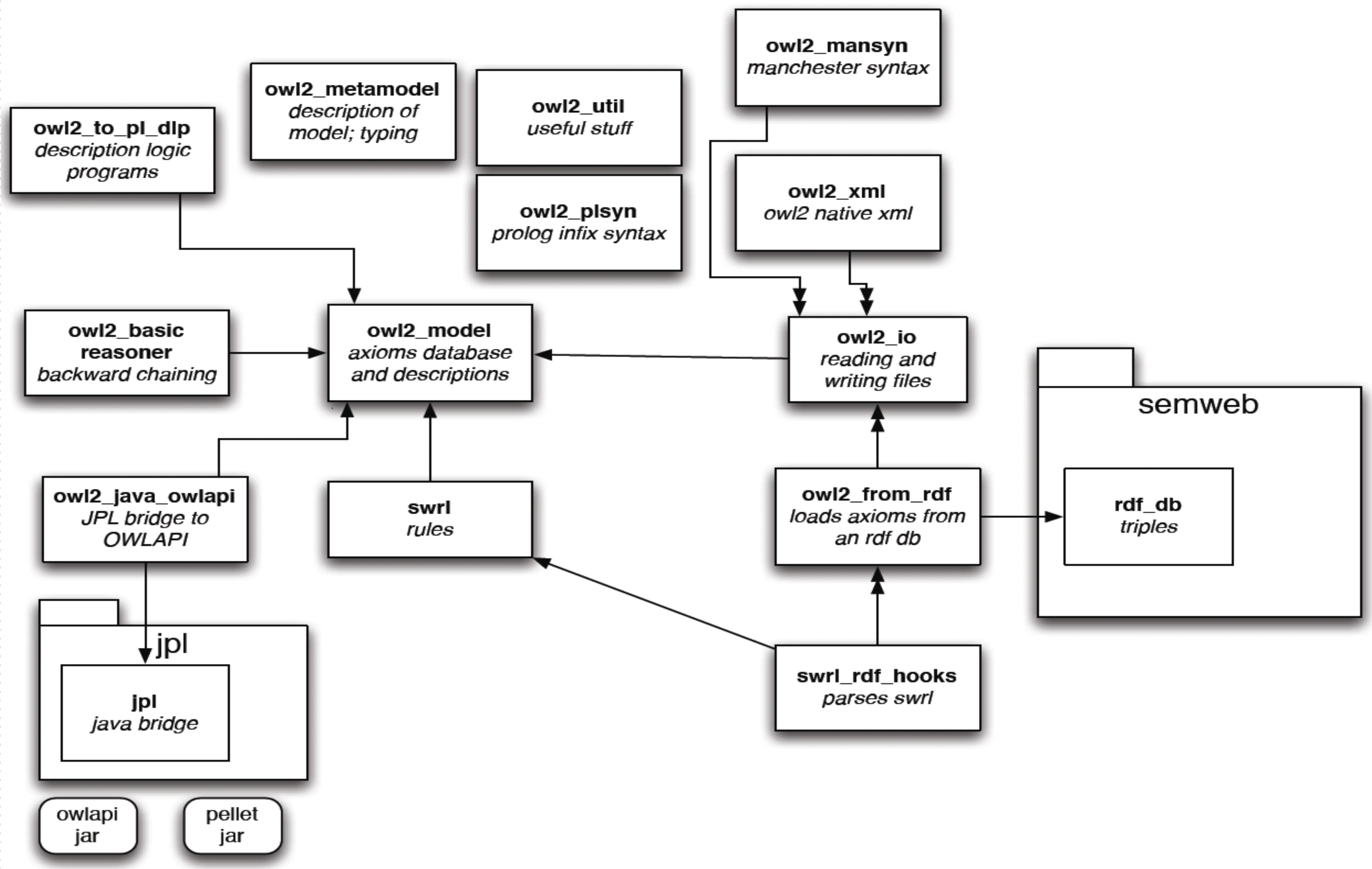
Motivation

- Sophisticated Ontology engineering environments.
- Powerful reasoning servers.
- RDF centric tools ‘triple-focus’ not suitable for complex T-Boxes heavily axiomatized.
- BUT… need for Easy programmatic access to Ontologies or Knowledge bases.
  - Querying
  - Scripting operations
  - Build applications

Why Prolog?

- Declarative features, pattern matching.
  - SLD resolution, backward chaining.
- Use as a Rule-based system.
- Thea uses Prolog as a Host programming language, not as a reasoning system.
- SWI-Prolog implementation, semweb package, efficient RDF library

Thea library



Thea library: Model & I/O

- Model
  - Directly corresponds to the OWL2 structural syntax:

```
subClassOf('http://example.org#Human','http://example.org#Mammal').

OWL2
EquivalentClasses(forebrain_neuron
             intersectionOf(neuron
                            someValuesFrom(partOf forebrain)))
Prolog
equivalentClasses([forebrain_neuron,
               intersectionOf([neuron,
                              someValuesFrom(partOf, forebrain)])]).
```

- I/O (Parsing – Serialization)
  - RDF/XML
  - OWL2 XML
  - Manchester syntax

Thea library: Reasoning

Description Logic Programs	`equivalentClasses([only_has_part_a, allValuesFrom(has_part, a)])`
	
	`a(Y):- has_part(X,Y),only_has_part_a(X).`

SWRL	`?- prolog_clause_to_swrl_rule((hasUncle(X1,X3):- hasParent(X1,X2),hasBrother(X2,X3)),SWRL),swrl_to_owl_axioms(SWRL,Axiom).`
	`X1 = v(1), X3 = v(2), X2 = v(3),`
	`SWRL = implies(['_d:hasParent'(v(1), v(3)), '_d:hasBrother'(v(3), v(2))], '_d:hasUncle'(v(1), v(2))),`
	`Axiom = [subPropertyOf(propertyChain(['_d:hasParent', '_d:hasBrother']), '_d:hasUncle')].`

Backward chaining	`subclass1(_X,'http://www.w3.org/2002/07/owl#Thing').`
	
	`subclass1(X,Y) :- subclassOf(X,intersectionOf(Z)), member(Y,Z).`

Thea library: External reasoners

- OWLAPI through SWI's JPL package)
- OWLLink (RacerPro):

```
// Requests
owl_link(URL,[createKB([kb='http://owllink.org/examples/KB_1'],[]),
                  tell('http://owllink.org/examples/KB_1',
                       [subClassOf('B','A'),
                        subClassOf('C','A'),
                        equivalentClasses(['D','E']),
                        classAssertion('A','iA'),
                        subClassOf('C','A')]),
                   getAllClasses('http://owllink.org/examples/KB_1',
                   getEquivalentClasses('http://owllink.org/examples/KB_1','D'),
                   getSubClasses('http://owllink.org/examples/KB_1','C'),
                                  …
```

```
// Responses
[kb(http://owllink.org/examples/KB_1, []),
 syntaxError(Ignored non-valid OWLlink Tell requests: ((ClassAssertion (Class A) (Class
iA)))),
 setOfClasses([], [owl:Thing, C, B, E, A, D]),
 setOfClasses([], [E, D]),
 element(SetOfClassSynsets, [], []),]
```

Applications

<table>
<tr>
<td>

Ontology Querying

Least common ancestor

Count # of class members

</td>
<td>

```
common_ancestor(X,Y,A) :-
          entailed(subClassOf(X,A)),
          entailed(subClassOf(Y,A)).

least_common_ancestor(X,Y,A) :-
          common_ancestor(X,Y,A),
          \+ ((common_ancestor(X,Y,A2), A2\=A,
          entailed(subClassOf(A2,A)))).
% ---
class(C),aggregate(count,I,classAssertion(C,I),Num).
```

</td>
</tr>
<tr>
<td>

Ontology Processing

Enforce disjointUnion with exceptions

</td>
<td>

```
class(Y),
setof(X,
      (subClassOf(X,Y),
       \+ annotationAssertion(status,X,unvetted)
      ),
      Xs),
assert_axiom(disjointUnion(Y,Xs))
```

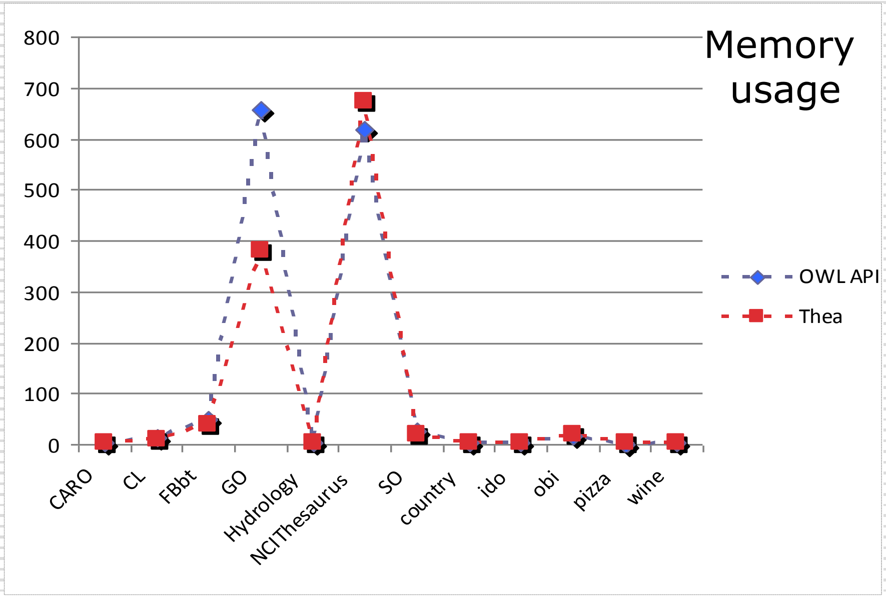
</td>
</tr>
</table>

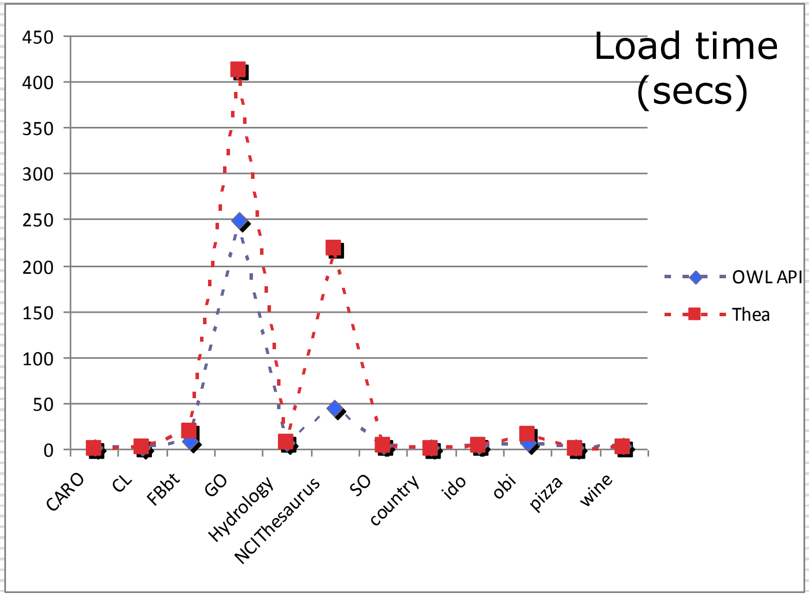
Comparison with other systems

- SPARQL
  - No means of updating data
  - – Too RDF-centric for querying complex TBoxes
  - – Lack of ability to name queries (as in relational views)
  - – Lack of aggregate queries
  - – Lack of programmability
  - But … extensions (SPARQL update)
- OPPL:
  - Simple, SQL – like
  - In Protégé…
  - Thea offers a complete programming language.

Comparison with OWLAPI

- OWLAPI:
  - Full featured.
  - Mature.
  - Java API (OO language)
- Thea:
  - declarative.
  - offers bridge via JPL.
  - easy scripting





Conclusions and Next steps

- OWL2 support within Prolog
- Full support of OWL2 structural syntax
- Easy programmatic access to query and process Ontologies within Prolog.
- Import and export to different formats
- Modules for external reasoning support
- Next Steps
  - Portability (other Prolog systems)
  - Improvements in efficiency..
  - Complete modules (other I/Os, Reasoners etc)
  - Use and feedback from the community…

thank you.

- http://github.com/vangelisv/thea