Supporting Open and Closed World Reasoning in the Semantic Web

C. V. Damásio, A. Analyti, G. Antoniou, G. Wagner
Overview

- Motivation
- Open and Closed World Reasoning
- Building blocks
- Knowledge in the Semantic Web
- Composing modules together
- Transformational Semantics
- Conclusions
Motivation

- Merging knowledge in the Semantic Web is one fundamental unsolved problem
- The need of combining closed and open world reasoning is desirable
- The adopted mechanisms should be modular
- The solution should be independent of the semantics adopted
- Use of nonmonotonic reasoning in the Semantic Web should be carefully controlled
Approach

- Open and closed world assumptions can be already combined in extended logic programming!
- It is required two forms of negation:
  - strong or explicit
  - weak, default or as failure
- The two forms of negation are available in
  - Well-founded semantics with explicit negation (WFSX)
  - Answer Set Semantics (AS)
- The proposed solution uses the same program transformation for both semantics
- The user should have an easy syntactic mechanisms to specify the use of nonmonotonic reasoning constructs
Open and Closed World reasoning

- **Open World Reasoning**
  - Founded on First Order Logic
  - Adopted in Description Logics, OWL and SWRL
  - Appropriate for the Semantic Web
  - Sometimes too conservative

- **Closed World Reasoning**
  - Founded on Nonmonotonic Logics
  - Adopted in Logic Programming and WRL
  - Appropriate for (Deductive) Databases
  - Sometimes too brave
Example

- Consider the following list of facts

  % All current EU countries
  CountryEU(Austria) … CountryEU(UK)

  % Some non EU countries (not all…)
  🔹 CountryEU(China)
  🔹 CountryEU(Djibuti)
A little geography...

- Is Austria a EU country?
  - YES, because it appears the fact CountryEU(Austria) in the knowledge base

- Is China a EU country?
  - NO, because it is expressed that \( \neg \) CountryEU(China)

- Is Montenegro a EU country?
  - NO, because it is not listed there and the list is complete (CLOSED WORLD REASONING)
  - DON’T KNOW, since it is not listed then it might be or not (OPEN WORLD REASONING)
The help of extended LP

- **Closed world reasoning:**

  \[ \neg \text{CountryEU}(?C) \leftarrow \neg \text{CountryEU}(?C) \]

- **Open world reasoning:**

  \[ \neg \text{CountryEU}(?C) \leftarrow \neg \text{CountryEU}(?C) \]
  \[ \text{CountryEU}(?C) \leftarrow \neg \neg \text{CountryEU}(?C) \]
A syntactic detour

- Rule bases are sets of rules of the form
  - $L_0 \leftarrow L_1, \ldots, L_m, \neg L_{m+1}, \ldots, \neg L_n$

- Each $L_i$ ($0 \leq i \leq n$) is an objective literal, i.e.
  - An atom $A(t)$, or
  - The strong negation of an atom $\neg A(t)$

- The symbol $\neg$ represents nonmonotonic weak negation, and cannot occur in the head
- The symbol $\neg$ represents monotonic strong negation, and can occur in the head and in the body of rules
- The discussion is restricted to the DATALOG case, i.e. no function symbols in the language
Putting weak negation on the leash

- The following predicate types are proposed
  - Definite or objective predicates
  - Open predicates
  - Closed predicates
  - Normal or unrestricted predicates

- Definite, open and closed predicates are limited to be defined by rules without weak negation

- Normal predicates can use the full language
Definite Predicates

- **Similar to** Definite Logic Programming, but allowing for explicit negation in the head and body of rules
- There can exist information gaps: predicates are partial
- Reasoning is purely monotonic
- Reasoning is polynomial on the size of the ground rule base and can be readily implemented in Prolog
Open Predicates

- Rules are like in the previous case, but additionally it is added the following pair of rules for each open predicate $A$ with arity $n$
  - $\neg A(x_1, \ldots, x_n) \leftarrow \sim A(x_1, \ldots, x_n)$
  - $A(x_1, \ldots, x_n) \leftarrow \neg \sim A(x_1, \ldots, x_n)$

- Reasoning is monotonic

- Reasoning is polynomial for WFSX and co-NP complete for AS

- Can be implemented with XSB or any answer set programming system like DLV, Smoodels, etc.
Closed Predicates

- Rules are like in the previous case, but it is added only one of the following pair of rules for closed predicate $A$ with arity $n$
  - $\neg A(?x_1,\ldots,?x_n) \leftarrow \neg A(?x_1,\ldots,?x_n)$
  - $A(?x_1,\ldots,?x_n) \leftarrow \neg A(?x_1,\ldots,?x_n)$
- Reasoning is nonmonotonic
- Conclusions obtained by objective predicates are also obtained by closed ones (common safe knowledge)
Normal Predicates

- Full syntax of extended logic programming
- Nonmonotonic
- No guarantees...
- Sometimes it is required
Entailment of Objective Literals

- Predicates are all definite or open, except varying A

Answer Set Semantics  
A is closed  
A is open  
A is definite

Well-founded Semantics with Explicit Negation  
A is closed  
A is definite  
A is open
Monotonicity of Reasoning

- All predicates in rule bases $P$ and $Q$ are either definite or open
Particularities of the Semantic Web

- Rule bases cannot be seen isolated
- Modularity, encapsulation, information hiding and access control mechanisms are required
- Have to deal with four levels of context
  - Semantic Web context
  - Application context
  - Rule base context
  - Predicate context
- IRIs should be used in names of rule bases and predicates
Requirements

- Applications loading/asserting knowledge need mechanisms to express that
  - Nonmonotonic reasoning is allowed or inhibited
  - Force to use only safe knowledge

- Producers of knowledge need mechanisms to
  - Declare that a predicate cannot be redefined
  - Declare hidden predicates not visible in the Semantic Web
  - Use all available knowledge in the application context, or get it explicitly from particular rule bases

- Monotonic reasoning is the default, not the exception
The DEFINES declaration states:
- The predicates defined in a rule base and their type
- The predicates exported and their scope

The USES declaration states:
- The predicates imported, and from where
- Reasoning mode to be used
DEFINES

[RuleBaseIRI] (Absolute IRI, default is the rule base where it occurs)
defines

[ScopeDecl] global | local | internal

PredDeclList (objective | open | closed [↑] | normal) AbsIRI [/N], ...

[visible to RuleBaseList] (list of Absolute IRIs, if omitted, visible everywhere)
USES declaration

\[[RuleBaseIRI]\] (Absolute IRI, default is the rule base where it occurs)
uses
\[PredDeclList\] (objective | open | closed [?] | normal) \[AbsIRI [/N], …
[from
\[RuleBaseList]\] (list of Abs. IRIs, by default uses from any available rulebase)

NOTE: The scope of an imported predicate is given by a corresponding defines declaration. If absent, the predicate is global and open; the defaults adopted!
Combining reasoning forms

<table>
<thead>
<tr>
<th>uses (importer)</th>
<th>normal</th>
<th>objective</th>
<th>open</th>
<th>closed</th>
<th>normal</th>
</tr>
</thead>
<tbody>
<tr>
<td>closed</td>
<td>objective</td>
<td>open</td>
<td>closed</td>
<td>error</td>
<td></td>
</tr>
<tr>
<td>open</td>
<td>objective</td>
<td>open</td>
<td>open</td>
<td>error</td>
<td></td>
</tr>
<tr>
<td>objective</td>
<td>objective</td>
<td>objective</td>
<td>objective</td>
<td>error</td>
<td></td>
</tr>
<tr>
<td>objective</td>
<td>objective</td>
<td>open</td>
<td>closed</td>
<td>normal</td>
<td></td>
</tr>
</tbody>
</table>

defines (exporter)
Defining and using the same predicate

<table>
<thead>
<tr>
<th>uses</th>
<th></th>
<th></th>
<th>defines</th>
</tr>
</thead>
<tbody>
<tr>
<td>global</td>
<td>allowed</td>
<td>error</td>
<td>allowed</td>
</tr>
<tr>
<td>local</td>
<td>error</td>
<td>error</td>
<td>allowed</td>
</tr>
<tr>
<td>internal</td>
<td>error</td>
<td>error</td>
<td>error</td>
</tr>
<tr>
<td>global</td>
<td>local</td>
<td>internal</td>
<td></td>
</tr>
</tbody>
</table>
Example

\[\text{d}_{\text{http://www.eu.int#CountryEU}}(X) \iff -\text{o}_{\text{http://www.eu.int#CountryEU}}(X)\]
\[\text{o}_{\text{http://www.eu.int#CountryEU}}(X) \iff \text{~d}_{\text{http://www.eu.int#CountryEU}}(X)\]
\[\text{c}_{\text{http://www.eu.int#CountryEU}}(X) \iff \text{~n}_{\text{http://www.eu.int#CountryEU}}(X)\]
\[\text{n}_{\text{http://www.eu.int#CountryEU}}(X) \iff \text{~c}_{\text{http://www.eu.int#CountryEU}}(X)\]

\[\text{d}_{\text{http://www.eu.int#CountryEU}}(\text{Austria}).\]
\[\text{o}_{\text{http://www.eu.int#CountryEU}}(\text{Austria}).\]
\[\text{c}_{\text{http://www.eu.int#CountryEU}}(\text{Austria}).\]
\[\text{n}_{\text{http://www.eu.int#CountryEU}}(\text{Austria}).\]
Example

- `<http://security.int>`

defines global open sec#citizenOf/2.

```
sec: citizenOf(Anne, Austria).
sec: citizenOf(Boris, Bulgaria).
sec: citizenOf(Chen, China).
sec: citizenOf(Dil, Djibuti).

o_<http://security.int><http://security.int#citizenOf>(X,Y) :-
  ~ - o_<http://security.int><http://security.int#citizenOf>(X,Y).
-o_<http://security.int><http://security.int#citizenOf>(X,Y) :-
  ~ o_<http://security.int><http://security.int#citizenOf>(X,Y).

c_<http://security.int><http://security.int#citizenOf>(X,Y) :-
  ~ - c_<http://security.int><http://security.int#citizenOf>(X,Y).
-c_<http://security.int><http://security.int#citizenOf>(X,Y) :-
  ~ c_<http://security.int><http://security.int#citizenOf>(X,Y).

n_<http://security.int>n_citizenOf(X,Y) :-
  ~ - n_<http://security.int><http://security.int#citizenOf>(X,Y).
-n_<http://security.int>n_citizenOf(X,Y) :-
  ~ n_<http://security.int><http://security.int#citizenOf>(X,Y).
```
Example

\[
\text{d_} \langle \text{http://gov.country}\rangle \langle \text{http://security.int#citizenOf}\rangle(X,Y) : - \\
\text{-d_} \langle \text{http://gov.country}\rangle \langle \text{http://security.int#citizenOf}\rangle(X,Y).
\]

\[
\text{o_} \langle \text{http://gov.country}\rangle \langle \text{http://security.int#citizenOf}\rangle(X,Y) : - \\
\text{-o_} \langle \text{http://gov.country}\rangle \langle \text{http://security.int#citizenOf}\rangle(X,Y).
\]

\[
\text{c_} \langle \text{http://gov.country}\rangle \langle \text{http://security.int#citizenOf}\rangle(X,Y) : - \\
\text{-c_} \langle \text{http://gov.country}\rangle \langle \text{http://security.int#citizenOf}\rangle(X,Y).
\]

\[
\text{n_} \langle \text{http://gov.country}\rangle \langle \text{http://security.int#citizenOf}\rangle(X,Y) : - \\
\text{-n_} \langle \text{http://gov.country}\rangle \langle \text{http://security.int#citizenOf}\rangle(X,Y).
\]

d_<http://gov.country><http://gov.country#Enter>(P) :-
  d_<http://gov.country><http://www.eu.int#CountryEU>(C),

o_<http://gov.country><http://gov.country#Enter>(P) :-
  o_<http://gov.country><http://www.eu.int#CountryEU>(C),

c_<http://gov.country><http://gov.country#Enter>(P) :-
  c_<http://gov.country><http://www.eu.int#CountryEU>(C),

n_<http://gov.country><http://gov.country#Enter>(P) :-
  n_<http://gov.country><http://www.eu.int#CountryEU>(C),
Summary of the program transformation

- Each rule is translated into four different rules, one for each reasoning mode
  - Definite (prefix d)
  - Open (prefix o)
  - Closed (prefix c)
  - Normal (prefix n)

- The predicate name is obtained by composition of the IRI of the Rule base plus prefix and the IRI of the predicate name

- The rules for open and closed predicates are as well introduced.

- Global and local predicates introduce a rule with prefix plus the IRI of the predicate, in order to make it visible everywhere

- USES declarations are introduced by respecting combination of reasoning forms in the table.
Problems to be addressed

- Implicit Domain Closure Assumption
- Unique Names Assumption
- Expressing the domain of predicates in order to avoid floundering, using for instance `rdf:domain` and `rdf:range`
- Optimisation of the program transformation: too many repeated rules
- Handling disjunction and paraconsistency
Conclusions

- Modular approach to mixing open and closed world reasoning
- Users have mechanisms to control the use of nonmonotonic reasoning in the Semantic Web
- Defines the notion of scope of predicates
- Captures the intuitions of knowledge merging in the Semantic Web
- Solution based on widely accepted semantics
- Polynomial program transformation for WFSX and AS