Database Backend for Description Logics

Yang Chen
yang@cise.ufl.edu

Computer and Information Science and Engineering
University of Florida

Aug 30, 2013
Outline

1. Introduction
2. Description Logic
3. Implementation
Conversation Understanding

Ken 7 to 14 year olds need plenty of computer counseling.

Jen Yeah, in another job that requires people skills!

Jen Thats true Ken- a benefit.

Ken Who knows... perhaps they need help with their spreadsheets near income tax time!

Lance Actually, a computer expert at the Y would be nice... I tried to change my address to one outside Albany the other day but the guy at the desk said the computer system refused the zip code, heh.

Jen Hehe.

Jen I think the job descrip is lacking tonight.

Jen It doesn't ally say what you'd be doing-counselor, etc. Just says working with 7-14.
Conversation Understanding

- We can extract topics and polarities:
  computer conseling \{Positive, negative, neutral\}
- Using background knowledge, it’s possible to learn more about the topic and speakers’ attitude.
  - WordNet example: something that provides direction or advice as to a decision or course of action
  - Using synonym(computer, IT), we may find related IT counseling companies.
  - Limited use cases for separate conversations; Aggregation may produce better results\(^1\).

\(^1\)http://www.technologyreview.com/view/421201/how-to-use-twitter-for-personal-data-mining
Conversation Understanding

{"conversation":{
  "name":"Interview Discussion",
  "id":"http://CUBISM/api/1.0/conversations/5.json",
  "topics":[
    {
      "id":"http://CUBISM/api/1.0/conversations/5/topics/1.json",
      "name":"topic A",
      "belief":"belief of system about topic A",
      "viewpoints":[
        {
          "speaker-id":"http://CUBISM/api/1.0/speakers/2.json",
          "belief":"belief of speaker 2 about topic A",
          "viewpoints":[
            {
              "speaker-id":"http://CUBISM/api/1.0/speakers/3.json",
              "belief":"belief of speaker two about speaker 3’s belief about topic A"
            }
          ]
        }
      ]
    }
  ]
}}
Outline

1. Introduction
2. Description Logic
3. Implementation
Description Logic
ABox axioms capture knowledge about named individuals (facts).

**Example**

parentOf(Julia,John)
bornIn(John,LA)

```plaintext
Declaration( NamedIndividual( :John ) )
Declaration( NamedIndividual( :Julia ) )
Declaration( NamedIndividual( :LA ) )
Declaration( ObjectProperty( :parentOf ) )
Declaration( ObjectProperty( :bornIn ) )

ObjectPropertyAssertion( :parentOf :Julia :John )
ObjectPropertyAssertion( :bornIn :John :LA )
```
Description Logic

TBox Axioms

TBox axioms describe relationships between concepts (rules).

Example

$$\text{Man}(x) \land \text{hasBrother}(x, y) \land \text{hasChild}(y, z) \rightarrow \text{Uncle}(x)$$

$$\text{Man} \sqcap \exists \text{hasBrother.}\exists \text{hasChild.}\top \sqsubseteq \text{Uncle}$$

```
Declaration( Class( :Man ) )
Declaration( Class( :Uncle ) )
Declaration( ObjectProperty( :hasBrother ) )
Declaration( ObjectProperty( :hasChild ) )

SubClassOf(
    ObjectIntersectionOf(
        :Man
        ObjectSomeValuesFrom(
            :hasBrother
            ObjectSomeValuesFrom(
                :hasChild
                owl:Thing
            )
        )
    )
    :Uncle
)
```
Description Logic

SWRL Rules

\[ \text{Person}(?x), \text{hasParent}(?x, ?y), \text{hasParent}(?x, ?z), \text{hasSpouse}(?y, ?z) \rightarrow \text{ChildOfMarriedParents}(?x) \]

SWRL can be used to model such rules. It supports general *Horn rules*:

\[ \text{hasParent}(?x_1, ?x_2), \text{hasBrother}(?x_2, ?x_3) \rightarrow \text{hasUncle}(?x_1, ?x_3) \]
\[ \text{Student}(?x_1) \rightarrow \text{Person}(?x_1) \]
\[ \text{Artist}(?x), \text{artistStyle}(?x, ?y), \text{Style}(?y), \text{creator}(?z, ?x) \rightarrow \text{style/period}(?z, ?y) \]
\[ \text{Artist}(?x), \text{artistStyle}(?x), \text{creator}(?z, ?x) \rightarrow \text{style/period}(?z) \]
The HermiT Reasoner

- HermiT is a reasoner for ontologies written in OWL.
- ABox assertions:
  
  \[
  \text{Man}(\text{Bob}), \text{Man}(\text{Ted}), \text{Man}(\text{Tom}), \\
  \text{hasBrother}(\text{Bob}, \text{Ted}), \text{hasChild}(\text{Ted}, \text{Tom})
  \]

- DL Rule:
  \[
  \text{Man} \sqcap \exists \text{hasBrother}. \exists \text{hasChild}. \top \sqsubseteq \text{uncleOf}
  \]
Outline

1. Introduction
2. Description Logic
3. Implementation
Implementation Overview

Missing interface?
String-Based Implementation

\[ \text{Man}(x) \land \text{hasBrother}(x, y) \land \text{hasChild}(y, z) \rightarrow \text{Uncle}(x) \]

\[
\text{Man} \sqcap \exists \text{hasBrother}. \exists \text{hasChild}. \top \sqsubseteq \text{Uncle}
\]

**Table: DL Axioms**

<table>
<thead>
<tr>
<th>propId</th>
<th>axiom</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>\text{SubClassOf}( \text{ObjectIntersectionOf}( \text{Man} \text{ObjectSomeValuesFrom}( \text{hasBrother} \text{ObjectSomeValuesFrom}( \text{hasChild} \text{owl:Thing})) \text{Uncle}))</td>
</tr>
</tbody>
</table>
Triple-Based Implementation

\[
\text{Man}(x) \land \text{hasBrother}(x, y) \land \text{hasChild}(y, z) \rightarrow \text{Uncle}(x)
\]

\[
\text{Man} \sqcap \exists \text{hasBrother}.\exists \text{hasChild}.\top \sqsubseteq \text{Uncle}
\]

**Table: DL Axioms**

<table>
<thead>
<tr>
<th>propId</th>
<th>axiom</th>
<th>x</th>
<th>y</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>someValuesFrom</td>
<td>t1</td>
<td>hasChild</td>
<td>OWL:Thing</td>
</tr>
<tr>
<td>1</td>
<td>someValuesFrom</td>
<td>t2</td>
<td>hasBrother</td>
<td>t1</td>
</tr>
<tr>
<td>1</td>
<td>classIntersection</td>
<td>t3</td>
<td>Man</td>
<td>t2</td>
</tr>
<tr>
<td>1</td>
<td>subClassOf</td>
<td>t3</td>
<td>Uncle</td>
<td>null</td>
</tr>
</tbody>
</table>
Belief Modeling

Env (id INT,
    parent_id INT REFERENCES Env(id),
    label TEXT);

Axioms (id INT, type INT,
    x INT, y INT, z INT);

Axioms (id INT, label TEXT);

Beliefs (believer INT REFERENCES Env(id),
    belief INT REFERENCES Axioms(id));
Object-Relational Mapping (ORM)

- Object-relational mapping maps database rows directly to OOP objects.
- The ORM libraries manage SQL queries that implement the mapping.

```
Object-Relational Mapping (ORM)
```

```
- Object-relational mapping maps database rows directly to OOP objects.
- The ORM libraries manage SQL queries that implement the mapping.
```
ORM-Based Architecture

Figure: ORM-Based Architecture
Evaluation

- **String-Based implementation:**
  - Easy to implement;
  - More generalizable;
  - Hard to query; need external parsers.

- **Triple-Based implementation:**
  - Support more queries;
  - Not convenient for complex rules.
Questions?

Thank you!