Concept Learning in the ALC Description Logic

Jens Lehmann² and Pascal Hitzler²

1 Department of Computer Science, Universität Leipzig, Germany – lehmann@informatik.uni-leipzig.de
2 AIFB Institute, Universität Karlsruhe, Germany – hitzler@aifb.uni-karlsruhe.de

Motivation:
- The ontology language OWL DL is based on Description Logics (DLs) and is widely used in many application areas.
- Developing ontologies is still a burdensome task (knowledge engineers and domain experts required).
- The application of Inductive Logic Programming (ILP) methods to Description Logics has not been fully researched yet – both theoretical and practical.

Solution Approach:
- Transfer ILP methodology, in particular refinement operators, to DLs.
- Provide solid theoretical foundations for using refinement operators.
- Create an algorithm for learning concepts from positive and negative examples to enable creation, extension, and analysis of OWL ontologies.
- System provides for the ALC description logic but extensible to more expressive languages.
- Implemented system DL-Learner: top-down algorithm, redundancy checks, handles infinite operator, DIG interface for reasoner communication.

Algorithm

Input: horizonExpFactor in [0, 1]
ST (search tree) is set to the tree consisting only of the root node
(T, 0, q, T), false

minHorizExp = 0
while ST does not contain a correct concept do
  choose N = (C, n, q, b) with highest fitness in ST
  expand N up to length n + 1, i.e.:
    begin
      add all nodes (D, n−, checkRef(ST, D)) with D ∈ trans(ρ(C)) and
      |D| = n + 1 as children of N
    end
  evaluate created non-redundant nodes
  change N to (C, n + 1, q, b)
end
minHorizExp = max(minHorizExp, [horizonExpFactor * (n + 1)])
while there are nodes with defined quality and horizon expansion
  smaller minHorizExp do
    expand these nodes up to minHorizExp
  return a correct concept in ST

Theoretical Foundations
- We researched the properties (completeness, properness, redundancy, finiteness, minimality) of refinement operators
- Which properties can be combined?
- We obtained general results for any sufficiently expressive description language L (i.e., L allows to express ⊤, ⊥, conjunction, disjunction, universal quantification, and existential quantification).

Property Theorem
Considering the properties completeness, weak completeness, properness, finiteness, and non-redundancy the following are maximal sets of properties (in the sense that no other of the mentioned properties can be added) of L refinement operators:
1. (weakly complete, complete, finite)
2. (weakly complete, complete, proper)
3. (weakly complete, non-redundant, finite)
4. (weakly complete, non-redundant, proper)
5. (non-redundant, finite, proper)

Operator

\[ p(C) = \begin{cases} \{\top\} \cup p'(C) & \text{if } C = \top \\ \emptyset & \text{if } C = \bot \\ \{C_i \cup \ldots \cup C_n \mid C_i \in \{M (1 \leq i \leq n)\}\} & \text{if } C = A \in \mathcal{N}_C \\ \{\neg A \mid A \in \mathcal{N}_C\} & \text{if } C = \neg A \in \mathcal{N}_C \\ \{\forall E \mid E \in \mathcal{N}_E\} \cup (\exists R \mathcal{R} E) & \text{if } C = \exists R \mathcal{R} D \\ \{\forall R \mathcal{R} E\} \cup (\exists R \mathcal{R} E) & \text{if } C = \forall R \mathcal{R} D \\ \end{cases} \]

Contributions & Future Work

Contributions to the State of the Art:
- Full analysis of properties of refinement operators in DLs.
- A refinement operator conforming to the theoretical findings.
- Algorithm handling the unavoidable limitations of the operator.
- Provision of a preliminary evaluation.

Future Work:
- More evaluation examples (performance on noisy or inconsistent data).
- Create (more) benchmarks to assess scalability and enable comparison of different algorithms.
- Tests on real world data, e.g., DBpedia.
- Embed learning algorithm in ontology editor, e.g., OntoWiki.
- Extend operator to other description languages and OWL (nominals, cardinality restrictions, datatye intengs).
- Algorithm performance improvements: using domain/range of properties and subproperties.

Evaluation

<table>
<thead>
<tr>
<th>Problem</th>
<th>ALC-Learner</th>
<th>YinYang</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trains</td>
<td>124.0 1.51 2.35</td>
<td>30.0 1.51 100%</td>
</tr>
<tr>
<td>Arches</td>
<td>71.0 5.19 5.0</td>
<td>4.6 100% 28 100%</td>
</tr>
<tr>
<td>Moral (simple)</td>
<td>217.0 4.41 4.41</td>
<td>3.75 100% 205.3 69 69.4%</td>
</tr>
<tr>
<td>Moral (complex)</td>
<td>219.7 2.35 3.35</td>
<td>3.75 100% 205.3 69 69.4%</td>
</tr>
<tr>
<td>Poker (pair)</td>
<td>1335.0 2.63 30.0</td>
<td>7.75 100% 17.3 43 100%</td>
</tr>
<tr>
<td>Poker (straight)</td>
<td>1419.0 2.63 30.0</td>
<td>3.75 100% 205.3 69 69.4%</td>
</tr>
</tbody>
</table>

Contributions & Future Work

Contributions to the State of the Art:
- Full analysis of properties of refinement operators in DLs.
- A refinement operator conforming to the theoretical findings.
- Algorithm handling the unavoidable limitations of the operator.
- Provision of a preliminary evaluation.

Future Work:
- More evaluation examples (performance on noisy or inconsistent data).
- Create (more) benchmarks to assess scalability and enable comparison of different algorithms.
- Tests on real world data, e.g., DBpedia.
- Embed learning algorithm in ontology editor, e.g., OntoWiki.
- Extend operator to other description languages and OWL (nominals, cardinality restrictions, datatye intengs).
- Algorithm performance improvements: using domain/range of properties and subproperties.