OpenRuleBench: An Analysis of the Performance of Rule Engines

Senlin Liang, Paul Fodor, Hui Wan, Michael Kifer

Department of Computer Science
State University of New York at Stony Brook

sliang@cs.sunysb.edu

April 24, 2009

OpenRuleBench: http://rulebench.projects.semwebcentral.org
Motivations

- Semantic web is HOT!
- Rules are powerful in processing semantic information.
- How rule technologies perform on the Web scale?
- Previous comparisons were superficial [Bishop 2008, Sure 2002].


Outline

1 Introduction

2 Methodology

3 Results and Analysis

4 Conclusion
OpenRuleBench

- Five technologies:
  - Prolog-based
  - Deductive database
  - Production rules
  - Triple engines
  - General knowledge base

- Twelve systems:
  - XSB, Yap, SWI
  - DLV, IRIS, Ontobroker
  - Drools, Jess, Prova
  - Jena, SwiftOWLIM
  - CYC

Senlin Liang, Paul Fodor, Hui Wan, Michael Kifer

OpenRuleBench: http://rulebench.projects.semwebcentral.org
OpenRuleBench

- Five packages:
  - Large joins
  - Datalog recursion
  - Default negation
  - Dynamic indexing
  - Database interfaces

- Open community resource:
  - Programs
  - Scripts
  - Results
  - Manuals
Outline

1. Introduction
2. Methodology
3. Results and Analysis
4. Conclusion
Test Principles & System Capabilities

- Test principles:
  - Loading vs. inference times
  - Using the best settings for each system

- System capabilities:
  - Predicate arity constraints
  - Negation handling
  - Automatic optimizations:
    Cost-based optimizations, Subgoal reordering, Query filtering, Magic Sets, Indexing

Senlin Liang, Paul Fodor, Hui Wan, Michael Kifer
Stony Brook University
OpenRuleBench: http://rulebench.projects.semwebcentral.org
WWW 2009, Madrid, Spain
Large Joins

- Join1 and Join2:

**Fig. 1 Join1**

**Fig. 2 Join2**

Senlin Liang, Paul Fodor, Hui Wan, Michael Kifer

OpenRuleBench: http://rulebench.projects.semwebcentral.org

Stony Brook University

WWW 2009, Madrid, Spain
3 queries from LUBM: [Guo 2005].

Mondial: a geographical database.

DBLP: a publication database.

query(Id, T, A, Y, M) :- att(Id, title, T), att(Id, year, Y),
att(Id, author, A), att(Id, month, M).

LUBM: A Benchmark for OWL Knowledge Base Systems.
Datalog Recursion

- **Transitive closure:**
  ancestry(X, Y) :- parent(X, Y).
  ancestry(X, Y) :- parent(X, Z), ancestry(Z, Y).

  queries: ancestry(X, Y), ancestry(1, X), and ancestry(X, 1).

- **Same generation:**
  sg(X, Y) :- sib(X, Y).
  sg(X, Y) :- par(X, Z), sg(Z, Z1), par(Y, Z1).

  queries: sg(X, Y), sg(1, X), and sg(X, 1).

- **WordNet tests:** hypernyms, hyponyms, etc.

- **Wine ontology:** many mutually recursive rules.

---

Senlin Liang, Paul Fodor, Hui Wan, Michael Kifer
OpenRuleBench: http://rulebench.projects.semwebcentral.org
Stony Brook University
WWW 2009, Madrid, Spain
Default Negation

- **Modified same generation:**
  
  \[
  \text{non} \_ \text{sg}(X, Y) \leftarrow \text{ancestor}(X, Y).
  
  \text{non} \_ \text{sg}(X, Y) \leftarrow \text{ancestor}(Y, X).
  
  \text{sg2}(X, Y) \leftarrow \text{sg}(X, Y), \text{not} \ \text{non} \_ \text{sg}(X, Y).
  
  \]

- **Win-not-win:**

  \[
  \text{win}(X) \leftarrow \text{move}(X, Y), \text{not} \ \text{win}(Y).
  
  \]
Default Negation Cont’d

- A complex program from [Balbin 2008].

  \[
  \text{fb}(X) :- \text{magicfb}(X), d(X), \text{not} \text{ ab}(X), \\
  \hspace{1cm} h(X,Y), \text{ab}(Y).
  \]

  \[
  \text{ab}(X) :- \text{magicab}(X), g(X).
  \]

  \[
  \text{ab}(X) :- \text{magicab}(X), b(X,Y), \text{ab}(Y).
  \]

  \[
  \text{magicab}(Y) :- \text{magicab}(X), b(X,Y).
  \]

  \[
  \text{magicab}(Y) :- \text{magicfb}(X), d(X), \text{not} \text{ ab}(X), \\
  \hspace{1cm} h(X,Y).
  \]

  \[
  \text{magicab}(X) :- \text{magicfb}(X), d(X).
  \]

Efficient bottom-up computation of queries on stratified databases.
Miscellaneous Tests

- 16-Puzzle
- N-Queens
- Bitrev
- Dynamic indexing
- Database interfaces
Outline

1. Introduction
2. Methodology
3. Results and Analysis
4. Conclusion
Results Summary

No system was the best for all the tests.

- Three overall winners: Yap, XSB, and Ontobroker.
- DLV was also close.

No optimization was the best for all tests.

Promising technologies:

- Tabling Prolog technology: XSB and Yap.
- Deductive database technology: Ontobroker and DLV.

Scalability and performance issues:

- Indexing.
- Memory management.
- Query optimization.

Senlin Liang, Paul Fodor, Hui Wan, Michael Kifer

OpenRuleBench: http://rulebench.projects.semwebcentral.org

Stony Brook University

WWW 2009, Madrid, Spain
Results Summary

No system was the best for all the tests.

- Three overall winners: Yap, XSB, and Ontobroker.
- DLV was also close.

No optimization was the best for all tests.

Promising technologies:

- Tabling Prolog technology: XSB and Yap.
- Deductive database technology: Ontobroker and DLV.

Scalability and performance issues:

- Indexing.
- Memory management.
- Query optimization.

Senlin Liang, Paul Fodor, Hui Wan, Michael Kifer

OpenRuleBench: http://rulebench.projects.semwebcentral.org
Results Summary

No system was the best for all the tests.

- Three overall winners: Yap, XSB, and Ontobroker.
- DLV was also close.

No optimization was the best for all tests.

Promising technologies:

- Tabling Prolog technology: XSB and Yap.
- Deductive database technology: Ontobroker and DLV.

Scalability and performance issues:

- Indexing.
- Memory management.
- Query optimization.

Senlin Liang, Paul Fodor, Hui Wan, Michael Kifer

OpenRuleBench: http://rulebench.projects.semwebcentral.org
Results Summary

No system was the best for all the tests.

- Three overall winners: Yap, XSB, and Ontobroker.
- DLV was also close.

No optimization was the best for all tests.

Promising technologies:

- Tabling Prolog technology: XSB and Yap.
- Deductive database technology: Ontobroker and DLV.

Scalability and performance issues:

- Indexing.
- Memory management.
- Query optimization.
The Effect of Indexing and Tabling

<table>
<thead>
<tr>
<th>system</th>
<th>XSB</th>
<th>Yap</th>
<th>Ontobroker</th>
<th>DLV</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>0.004</td>
<td>0.037</td>
<td>0.042</td>
<td>1.045</td>
</tr>
</tbody>
</table>

Table: Mondial (Fully Optimized)

Case study: XSB

- Fully optimized: tabling and manual indexing.
- NO tabling: 1.713 seconds. (400 times slower!)
- NO tabling or manual indexing: 129.89 seconds (30,000 times slower!)
The Effect of Join Strategies
—Indexed-Nested-Loop vs. Sort-Merge

<table>
<thead>
<tr>
<th>query</th>
<th>a(X,Y)</th>
<th></th>
<th>b1(X,Y)</th>
<th></th>
<th>b2(X,Y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>size</td>
<td>50K</td>
<td>250K</td>
<td>50K</td>
<td>250K</td>
<td>50K</td>
</tr>
<tr>
<td>ontobroker</td>
<td>4.089</td>
<td>28.385</td>
<td>0.213</td>
<td>4.806</td>
<td>0.019</td>
</tr>
<tr>
<td>xsb</td>
<td>12.774</td>
<td>timeout</td>
<td>0.122</td>
<td>14.920</td>
<td>0.013</td>
</tr>
<tr>
<td>yap</td>
<td>10.534</td>
<td>timeout</td>
<td>0.109</td>
<td>12.123</td>
<td>0.013</td>
</tr>
<tr>
<td>dlv</td>
<td>85.459</td>
<td>838.781</td>
<td>7.177</td>
<td>60.239</td>
<td>0.820</td>
</tr>
</tbody>
</table>

Table: Join1, no query bindings

- Sort-merge (Ontobroker): scales better.
- Indexed-nested-loop (XSB and Yap): low overhead.

Senlin Liang, Paul Fodor, Hui Wan, Michael Kifer
Stony Brook University
OpenRuleBench: http://rulebench.projects.semwebcentral.org
WWW 2009, Madrid, Spain
Join1 & Join2

Fig. 1 Join1

Fig. 2 Join2
The Effect of Subgoal Reordering

<table>
<thead>
<tr>
<th>query</th>
<th>a(1,Y)</th>
<th>b1(1,Y)</th>
<th>b2(1,Y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>size</td>
<td>50K</td>
<td>250K</td>
<td>50K</td>
</tr>
<tr>
<td>ontobroker</td>
<td>0.035</td>
<td>0.038</td>
<td>0.013</td>
</tr>
<tr>
<td>xsb</td>
<td>0.013</td>
<td>35.990</td>
<td>0.000</td>
</tr>
<tr>
<td>yap</td>
<td>0.021</td>
<td>30.233</td>
<td>0.007</td>
</tr>
<tr>
<td>dlv</td>
<td>0.287</td>
<td>6.014</td>
<td>0.014</td>
</tr>
</tbody>
</table>

**Table**: Join1, 1st argument bound.

- Subgoal reordering (Ontobroker): scales better, but has initial overhead.
Cartesian Product

<table>
<thead>
<tr>
<th>system</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>2.087</td>
<td>2.092</td>
<td>11.935</td>
<td>44.692</td>
</tr>
</tbody>
</table>

Table: Times for Join2.

- Database technology (Ontobroker) cannot do much for Cartesian products.
- The tabled SLG-WAM (XSB and Yap) has low overhead.
Naive Select-Join

<table>
<thead>
<tr>
<th>system</th>
<th>ontobroker</th>
<th>xsb</th>
<th>yap</th>
<th>drools</th>
<th>dlv</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>1.602</td>
<td>1.752</td>
<td>2.447</td>
<td>0.186</td>
<td>2.201</td>
</tr>
</tbody>
</table>

Table: Times for DBLP

- query(Id,T,A,Y,M) :- att(Id,title,T), att(Id,year,Y),
  att(Id,author,A), att(Id,month,M).

- Drools: select, build indexing, and join.
Datalog Recursion

<table>
<thead>
<tr>
<th>size</th>
<th>50K</th>
<th>50K</th>
<th>500K</th>
<th>500K</th>
</tr>
</thead>
<tbody>
<tr>
<td>cyclic data</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>ontobroker</td>
<td>6.129</td>
<td>19.145</td>
<td>49.722</td>
<td>182.633</td>
</tr>
<tr>
<td>dlv</td>
<td>19.655</td>
<td>73.837</td>
<td>148.740</td>
<td>900.773</td>
</tr>
<tr>
<td>xsb</td>
<td>2.725</td>
<td>7.081</td>
<td>35.036</td>
<td>88.028</td>
</tr>
<tr>
<td>yap</td>
<td>2.066</td>
<td>13.026</td>
<td>33.128</td>
<td>82.900</td>
</tr>
</tbody>
</table>

Table: Transitive closure, no query bindings.

- Transitive closure: XSB and Yap perform the best.
Datalog Recursion Cont’d

No obvious overall winner.

- **Same generation**: Ontobroker performs and scales better.
- **Wordnet**: Yap performs significantly better than others.
- **Wine ontology**: XSB and Ontobroker perform better.
Default Negation

<table>
<thead>
<tr>
<th>test</th>
<th>win-not-win</th>
<th>modified same gen</th>
</tr>
</thead>
<tbody>
<tr>
<td>size</td>
<td>100K</td>
<td>500K</td>
</tr>
<tr>
<td>ontobroker</td>
<td>1.327</td>
<td>9.988</td>
</tr>
<tr>
<td>dlv</td>
<td>0.691</td>
<td>3.554</td>
</tr>
<tr>
<td>xsb</td>
<td>0.231</td>
<td>1.218</td>
</tr>
<tr>
<td>yap</td>
<td>0.103</td>
<td>0.654</td>
</tr>
</tbody>
</table>

Table: Locally- and predicate-stratified negation.

<table>
<thead>
<tr>
<th>test</th>
<th>win-not-win</th>
<th>[Balbin 2008]</th>
</tr>
</thead>
<tbody>
<tr>
<td>size</td>
<td>50K</td>
<td>250K</td>
</tr>
<tr>
<td>ontobroker</td>
<td>0.419</td>
<td>3.754</td>
</tr>
<tr>
<td>dlv</td>
<td>0.344</td>
<td>1.879</td>
</tr>
<tr>
<td>xsb</td>
<td>0.339</td>
<td>1.416</td>
</tr>
</tbody>
</table>

Table: Locally non-stratified rule sets.

- Top-down SLG-resolution (XSB and Yap) performs and scales better than bottom-up alternating fixed point computation (Ontobroker and DLV).
## Outline

1. Introduction
2. Methodology
3. Results and Analysis
4. Conclusion

---

Senlin Liang, Paul Fodor, Hui Wan, Michael Kifer

OpenRuleBench: [http://rulebench.projects.semwebcentral.org](http://rulebench.projects.semwebcentral.org)

Stony Brook University

WWW 2009, Madrid, Spain
Conclusions

- Open community resource: OpenRuleBench.
- Identified two promising rule technologies:
  - Tabling Prolog
  - Deductive database
- Identified several important issues:
  - Indexing
  - Memory management
  - Query optimization
- Future work: more systems and tests.
Thank you!