Slider: an Efficient Incremental Reasoner

Jules Chevalier
jules.chevalier@univ-st-etienne.fr

Laboratoire Hubert Curien, Télécom Saint Etienne, Université Jean Monnet

March 2015

Supervisors :
Fréférique Laforest
Christophe Gravier
Julien Subercaze

Summary

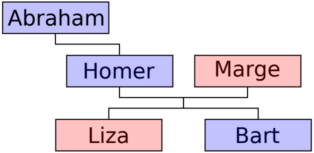
Semantic Web

- Formalises concepts to represent them
- Standardizes this representation
- Makes it readable for both humans and computers
- Links these data together
- Allows automatic operations on these data
  - Integrity constraint validation
  - Query the knowledge base
  - Extraction of implicit data

Semantic Web

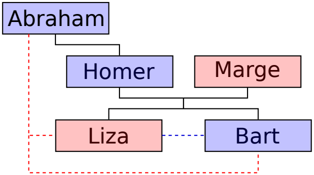
- Formalises concepts to represent them
- Standardizes this representation
- Makes it readable for both humans and computers
- Links these data together
- Allows automatic operations on these data
  - Integrity constraint validation
  - Query the knowledge base
  - Extraction of implicit data = **Reasoning**

Reasoning : Forward Chaining VS Backward Chaining



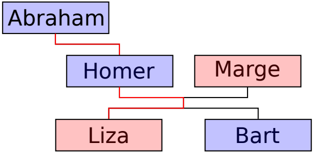
- What we know :
  - Abraham father Homer
  - Homer father Liza
  - Homer father Bart
  - Marge mother Liza
  - Marge mother bart

Reasoning : Forward Chaining VS Backward Chaining



- What we know :
  - Abraham father Homer
  - Homer father Liza
  - Homer father Bart
  - Marge mother Liza
  - Marge mother bart
- What **Forward Chaining** do :
  - Abraham grandfather Liza
  - Abraham grandfather Bart
  - ...
  - Abraham grandfather Liza ? $\rightarrow$ yes

Reasoning : Forward Chaining VS Backward Chaining



- What we know :
  - Abraham father Homer
  - Homer father Liza
  - Homer father Bart
  - Marge mother Liza
  - Marge mother bart
- What **Forward Chaining** do :
  - Abraham grandfather Liza
  - Abraham grandfather Bart
  - ...
  - Abraham grandfather Liza ? $\rightarrow$ yes
- What **Backward Chaining** do :
  - Abraham grandfather Liza ?
  - Abraham father X & X father Liza ?
  - Abraham father Homer & Homer father Liza $\rightarrow$ yes

Rule-based Reasoning

Rules

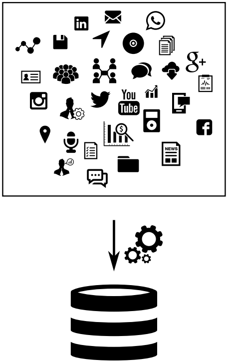
- An *antecedent*: Allows the rule to be executed
- A *consequent*: The statement inferred

$$\frac{c_1 \ \texttt{subClassOf} \ c_2, \quad x \ \texttt{type} \ c_1}{x \ \texttt{type} \ c_2} \ \text{(cax-sco)}$$

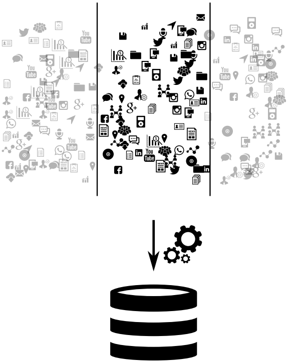
Fragments

- A fragment is a set of inference rules
- Semantic Web standards suggest different pre defined fragments (RDFS, OWL Lite, OWL Full, OWL DL, ...)
- The more they have a high expressivity, the more the operations are complex (from P to NEXPTIME)
- Choosing one fragment is trade off between expressivity and computational complexity

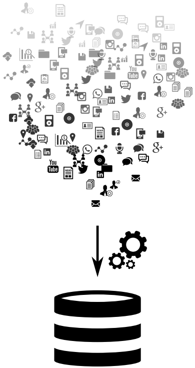
Reasoning kinds



Classical
Reasoning



Streaming
Reasoning



Incremental
Reasoning

Problematic

What we want to do

- Efficient and scalable incremental forward-chaining reasoning

Problematic

What we want to do

- Efficient and scalable incremental forward-chaining reasoning

What are the problems

- Rules form a cyclic graph
  - Complexity depends on the fragment !
- The amount of triples generated is quite unpredictable
  - The complexity also depends on data !
- Big Data is not static
  - We need to handle data streams !

Summary

Batch reasoning approaches

WebPie : a Web-scale Parallel Inference Engine

- 2009 - Jacopo Urbani Thesis [7]
  - Uses MapReduce for OWL Horst and RDFS reasoning
- 2011 - Fix some issues to improve OWL Horst reasoning [8]
  - Duplicates limitation
  - Indexation for `sameAs`
  - *Greedy* scheduling
  - Cleaner Job after some rules, or at the end

MapResolve [6]

- Based on WebPie to provide $\mathcal{EL}+$ classification
- Use 3 sets for triples : usable, used, inferred
- Limits overheads, optimise
- Points out MapReduce limitations

Analysis : MapReduce approaches

MapReduce Framework

- Allows to implement distributed tasks
- The Hadoop framework
- Best suited to batch process huge amounts of data

- MapReduce requires an acyclic dataflow
- Jobs run in isolation
- Not suitable network shuffling
- Hadoop distributed file system

WebPie and MapResolve Contributions

- Only provide batch reasoning
- Nodes must wait for each other
- Generate a lot of duplicates
- Fragment dependant
- Naive partitioning

- Critical letter for WebPie [5]

Incremental solutions

History Matters: Incremental Ontology Reasoning Using Modules [3]

- Maintains classification of ontologies as they evolve
- Provides encouraging results
- Not viable for static hierarchy of ontologies
- Not adapted on high number of nominals

Incremental Reasoning in OWL EL without Bookkeeping [4]

- Handles both addition and deletion of knowledge
- Incremental classification of TBox
- Limited to the classification on the TBox
- Dedicated to the $\mathcal{EL}+$ fragment

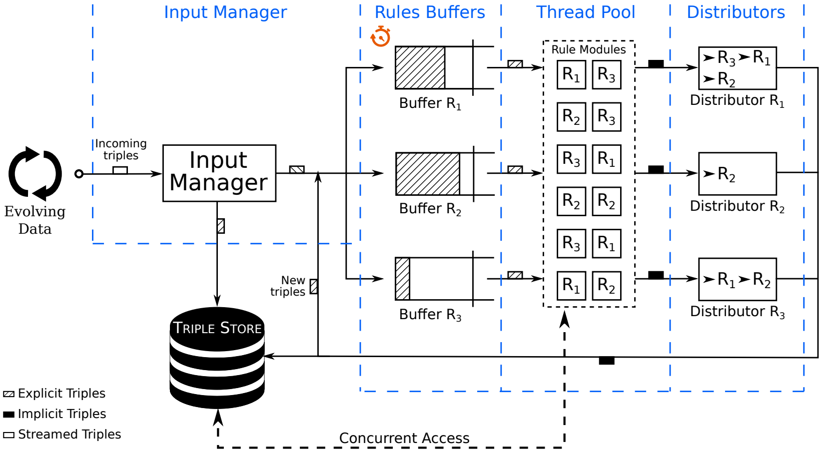
Summary

Proposed solution

Slider

- **Parallel and Scalable Execution**
  - Rules mapped to independent modules
  - Multiple rule instances allowed to run in parallel
- **Duplicates Limitation**
  - Shared triple store
  - Vertical partitioning [1] and multiple indexing
- **Data Stream Support**
  - Streamed architecture
  - Parallel parsing/reasoning
- **Fragment's Customization**
  - Dynamic support of ruleset
  - ρdf and RDFS natively supported
  - Extendible to any other fragment

Architecture



Architecture

Input Manager

- Receives incoming triples
- Sends them to
  - The triple store
  - The rules buffers

Rules Buffers

- A buffer for each rule
- Run the rule when full
- Run the rule when timed-out
- Ensures completeness

Thread Pool

- Manages a pool instances
- Ensures scalability

Rule instance

- Execute the inference
- Access concurrently the triple store

Distributor

- Stores inferred triples
- Dispatches them to the buffers

Inference: `cax-sco`

$$\frac{c_1 \; \texttt{subClassOf} \; c_2, \quad x \; \texttt{type} \; c_1}{x \; \texttt{type} \; c_2} \text{ (cax-sco)}$$

Algorithm 1 cax-sco

Require: tripleStore, newTriples, outputTriples

```
for all triple1 in TripleStore with predicate subClassOf do
   for all triple2 in newTriples with predicate type do
      if triple1.subject = triple2.object then
         output ← (triple2.subject,type,triple1.object)
         outputTriples ← outputTriples ∪ {output}
      end if
   end for
end for

for all triple1 in newTriples with predicate subClassOf do
   for all triple2 in TripleStore with predicate type do
      if triple1.subject = triple2.object then
         output ← (triple2.subject,type,triple1.object)
         outputTriples ← outputTriples ∪ {output}
      end if
   end for
end for
```

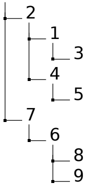
Triple Store

Vertical Partitioning

TRIPLES

(1,2,3)
(4,2,5)
(6,7,8)
(6,7,9)

ENCODING



Near-optimal indexing

- Indexing by predicates, subjects and objects
- Best trade-off for nearly all rules from the OWL fragments

Concurrent Access

- `ReentrantReadWriteLocks` ensure concurrency
- **Write** lock to add triples
- **Read** lock for other methods

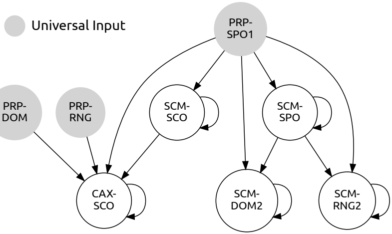
Duplicates Elimination

- `HashMap` of `MultiMaps`*
- Bans duplicates
- Ensures uniqueness of triples

* Google's Guava libraries

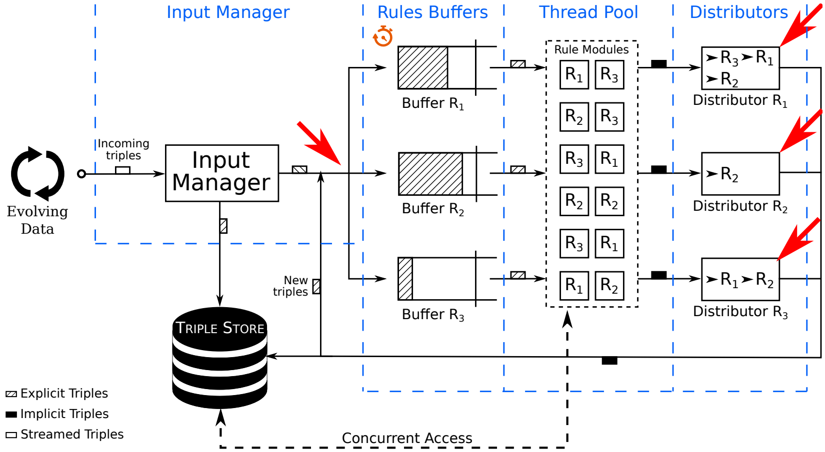
Rules Dependency Graph

- Directed graph
- Edges represent rules
- $A \rightarrow B$: B can use the output of A
- Created at initialisation time
- Used to route new triples by
  - The input manager
  - The distributors



Rules Dependency Graph for ρdf

Architecture



Summary

Experimentations

Baseline

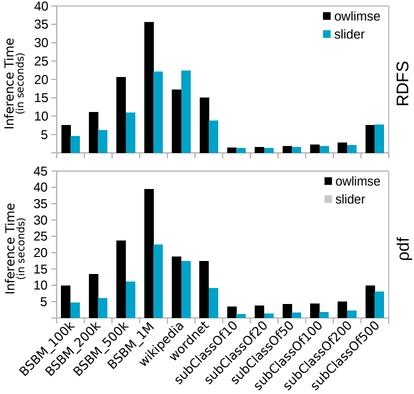
- OWLIM-SE (Standard Edition)
- Semantic repository with reasoning features
- Fastest reasoner available to the best of our knowledge
- Outperforms Jena and Sesame
- Natively supports RDFS, custom rule configuration for ρdf

Dataset

- 13 ontologies from 3 sets:
  - 2 Real life ontologies: WordNet and Wikipedia
  - 5 generated by BSBM, from 100,000 to 5 million triples
  - 6 subClassOf ontologies (closure computation, duplicates intensive)

Experiments

	ρdf reasoning		RDFS reasoning	
Ontology	**OWLIM**	**Slider**	**OWLIM**	**Slider**
BSBM_100k	9.907s	**4.636s**	7.487s	**4.558s**
BSBM_200k	13.338s	**6.059s**	11.064s	**6.198s**
BSBM_500k	23.595s	**11.133s**	20.580s	**10.984s**
BSBM_1M	39.364s	**22.357s**	35.602s	**22.192s**
BSBM_5M	170.151s	**126.292s**	160.699s	**127.037s**
wikipedia	18.802s	**17.422s**	**17.186s**	22.443s
wordnet	-	-	15.075s	**8.828s**
subClassOf10	3.507s	**1.209s**	1.423s	**1.216s**
subClassOf20	3.730s	**1.316s**	1.536s	**1.330s**
subClassOf50	4.159s	**1.615s**	1.865s	**1.583s**
subClassOf100	4.397s	**1.827s**	2.242s	**1.805s**
subClassOf200	4.962s	**2.210s**	2.837s	**2.170s**
subClassOf500	9.862s	**8.102s**	**7.584s**	7.625s

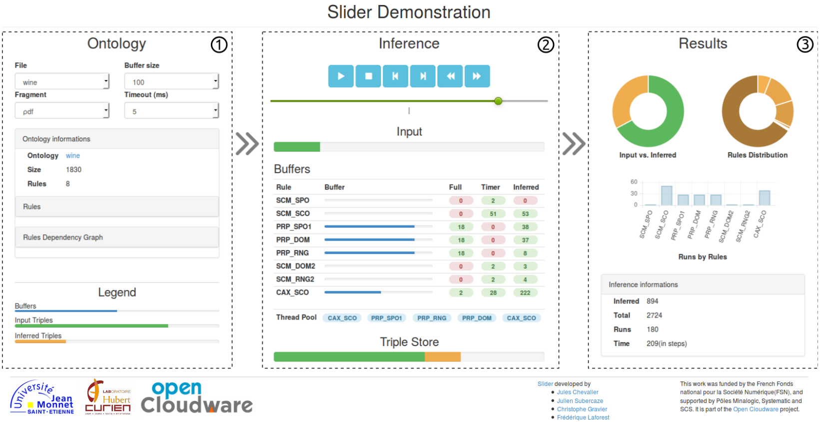


Inference time for Slider and OWLIM-SE on ρdf and RDFS

Improvement

- Average **71.47%**
- RDFS **36.08%**
- ρdf **106.86%**

Demonstration



[2] J Chevalier, J Subercaze, C Gravier, F Laforest. *Slider: an Incremental EfficientReasoner*, SIGMOD 2015

Summary

Conclusion and Future Work

Slider

- Efficient incremental rule-based reasoning
- Fragment agnocism
- Data streams support
- Improvement of **71.47%** in average against baseline

Future Work

- Timeout and buffer size cutomisable by rule
- Implementation of new rulesets
- Just-in-time optimisation of rules scheduling
- Use of historical statistics for adaptation

Bibliography I

[1] Abadi, D. J., Marcus, A., Madden, S. R., and Hollenbach, K.
Scalable semantic web data management using vertical partitioning.
In *Proceedings of the 33rd International Conference on Very Large Data Bases* (2007), VLDB '07, VLDB Endowment, pp. 411–422.

[2] Chevalier, J., Subercaze, J., Gravier, C., and Laforest, F.
Slider, an Efficient Incremental Reasoner.
SIGMOD (2015).

[3] Cuenca Grau, B., Halaschek-Wiener, C., and Kazakov, Y.
History matters: Incremental ontology reasoning using modules.
In *The Semantic Web*. 2007.

[4] Kazakov, Y., and Klinov, P.
Incremental reasoning in owl el without bookkeeping.
In *ISWC 2013*. 2013.

[5] Patel-Schneider, P.
Letter: Comments on WebPIE: A Web-scale parallel inference engine using MapReduce.
Web Semantics: Science, Services and Agents on ... 15 (Sept. 2012), 69–70.

[6] Schlicht, A., and Stuckenschmidt, H.
Mapresolve.
Web Reasoning and Rule Systems 6902 (2011), 294–299.

[7] Urbani, J.
RDFS/OWL reasoning using the MapReduce framework.
PhD thesis, Vrije Universiteit - Faculty of Sciences, 2009.

[8] Urbani, J., Kotoulas, S., Maassen, J., van Harmelen, F., and Bal, H. E.
WebPIE: A Web-scale parallel inference engine using MapReduce.
J. Web Sem. 10 (2012), 59–75.

Thank you for your attention

jules.chevalier@univ-st-etienne.fr

`julesschevalier.github.io/slider`

`demo-satin.telecom-st-etienne.fr/slider`