Imperial College London

SPOWL: Spark-based OWL 2 Reasoning Materialisation

Yu Liu and Peter McBrien

Department of Computing, Imperial College London

Table of Contents

Introduction

SPOWL Overview

SPOWL Features

Evaluation

Summary

Imperial College London

Table of Contents

Introduction

SPOWL Overview

SPOWL Features

Evaluation

Summary

Reasoning materialisation for OWL 2 ontologies

- LUBM T-Box:

$$\text{Student} \sqsubseteq \text{Person} \quad (1)$$

$$\text{Student} \sqsubseteq \exists \text{takesCourse}.\text{Course} \quad (2)$$

- LUBM A-Box:

$$\text{Student}(\text{John}) \quad (3) \qquad \text{Person}(\text{Lewis}) \quad (5)$$

$$\text{Student}(\text{Tom}) \quad (4) \qquad \text{Person}(\text{Mary}) \quad (6)$$

- Reasoning materialisation:

$\text{Student} := \{\text{John}, \text{Tom}\}$; $\text{Person} := \{\text{Lewis}, \text{Mary}, \text{John}, \text{Tom}\}$

$\text{takesCourse} := \{(\text{John}, ?\text{C1}), (\text{Tom}, ?\text{C2})\}$; $\text{Course} := \{?\text{C1}, ?\text{C2}\}$

- Querying the ontology:
  - Not only explicit but also implicit facts will be returned.

Reasoning materialisation for OWL 2 ontologies

Materialising reasoning results:

$$\text{Student} := \{\text{John}, \text{Tom}\}$$
$$\text{Person} := \{\text{Lewis}, \text{Mary}, \text{John}, \text{Tom}\}$$
$$\text{takesCourse} := \{(\text{John}, ?\text{C2}), (\text{Tom}, ?\text{C2})\}$$
$$\text{Course} := \{?\text{C1}, ?\text{C2}\}$$

- Queries directly read the materialised results.
- Faster query processing and larger space required.
- Maintenance of the materialisation is difficult.
- Ideal case: queries are much more frequent than updates.
- Example systems: SPOWL, Oracle's RDF Store, WebPIE, etc.

Rule evaluation for reasoning materialisation

- Rule format: **if** $\langle$antecedent$\rangle$ **then** $\langle$consequent$\rangle$:

  Example: **if** $C \sqsubseteq D, C(x)$ **then** $D(x)$

  $\Longrightarrow$ **if** $\mathsf{Student} \sqsubseteq \mathsf{Person}, \mathsf{Student}(x)$ **then** $\mathsf{Person}(x)$
- Well-known rulesets:
  - RDFS entailment rules.
  - OWL ter Horst rules.
  - OWL 2 RL/RDF rules.
- Limitations:
  - No use of tableaux reasoners (e.g. Pellet and Hermit).
  - Reasoning relies on which set of entailment rules is chosen.
  - Inefficient rule matching process.

Table of Contents

Introduction

SPOWL Overview

SPOWL Features

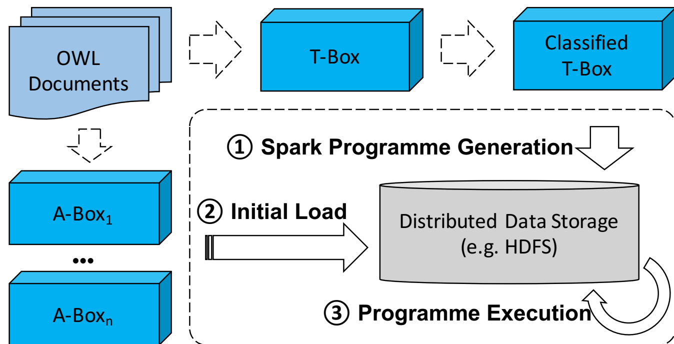
Evaluation

Summary

Imperial College London

SPOWL architecture

- T-Box is small enough for tableaux reasoners.
- The number of queries is much larger than the number of updates.



SPOWL overview

1. Classes & properties to Spark RDDs:

$$C \rightsquigarrow C_{rdd}(id) \qquad P \rightsquigarrow P_{rdd}(domain, range)$$

2. T-Box axioms are mapped to entailment rules $\mathcal{R}_{axiom}$:

$$C \sqsubseteq D \rightsquigarrow \mathcal{R}_{C \sqsubseteq D} ::= \textbf{if } C_{rdd}(x) \textbf{ then } D_{rdd}(x)$$

3. $\mathcal{R}_{axiom}$ are further implemented as Spark programmes $\mathcal{P}_{axiom}$:

$$\mathcal{R}_{C \sqsubseteq D} \rightsquigarrow \mathcal{P}_{C \sqsubseteq D} ::= D_{rdd} = D_{rdd}.\text{union}(C_{rdd})$$

4. $\mathcal{P}_{axiom}$ are iteratively executed to build up the RDDs.

Table of Contents

Introduction

SPOWL Overview

SPOWL Features

Evaluation

Summary

SPOWL uses tableaux reasoner

- More complete T-Box reasoning:

  e.g. classifying $\begin{array}{c} C \sqsubseteq D \sqcup E \\ C \sqcap D \sqsubseteq \bot \end{array}$ gives us $C \sqsubseteq E$

- Entailment rules are specific to the A-Box data:
  - No need to evaluate rules that are irrelevant to the ontological data.

SPOWL partitions reasoning materialisation

- Data of each class or property is stored separately in HDFS:
  $C \rightsquigarrow$ `hdfs://${C_PATH}/` $P \rightsquigarrow$ `hdfs://${P_PATH}/`
- A variant of the vertical partitioning model.
  - Only the partitions storing the relevant data need to be accessed.
    e.g. $\text{Student}_{rdd} = \text{sc.textfile}($`"hdfs://${Student_PATH}/"`$)$
  - Otherwise, the whole ontology should be read and a fragment of it should be filtered out.

SPOWL handles axioms beyond OWL 2 RL

- SomeValuesFrom forms a superclass expression (i.e. $C \sqsubseteq \exists P.D$)

$$\text{e.g. Student} \sqsubseteq \exists \text{takesCourse.Course} \quad (2)$$

- Non-deterministic reasoning (OWL 2 RL Interpretation $\mathcal{I}$):

$$\mathcal{I} \models C \sqsubseteq \exists P.D \text{ iff } C^{\mathcal{I}} \subseteq \{x \mid \exists y : \langle x, y \rangle \in P^{\mathcal{I}} \text{ and } y \in D^{\mathcal{I}}\}$$

- Entailment rule $\mathcal{R}_{C \sqsubseteq \exists P.D}$:

$$\textbf{if } C_{rdd}(x), \neg P_{rdd}(x, y) \textbf{ then } P_{rdd}(x, null)$$

- Spark programme $\mathcal{P}_{C \sqsubseteq \exists P.D}$:

$$P_{rdd} = P_{rdd}.\text{union}(C_{rdd}.\text{subtract}(P_{rdd}.\text{map}(\text{lambda } (x, y) : x)).\text{map}(\text{lambda } x : (x, null)))$$

The advantage of using Spark (1)

Spark caches RDDs in distributed memory as much as possible:

- reduce the needs to write/read intermediate results to/from disk.
- reduce I/O overhead.
- suitable for iterative computation (e.g. computing transitive closure).

Data caching in distributed memory

Iterative computation:

- TransitiveProperty P ($P \circ P \sqsubseteq P$).

$$\text{subOrganisationOf} \circ \text{subOrganisationOf} \sqsubseteq \text{subOrganisationOf} \quad (7)$$

- Entailment rule $\mathcal{R}_{P \circ P \sqsubseteq P}$:

$$\textbf{if } P_{rdd}(x, y), P_{rdd}(y, z) \textbf{ then } P_{rdd}(x, z)$$

- Spark programme $\mathcal{P}_{P \circ P \sqsubseteq P}$:

```
while True do
    P_tmp = P_rdd.map(lambda (x_p, y_p) : (y_p, x_p)).join(P_rdd)
                  .map(lambda (y_k, (x_p, z_p)) : (x_p, z_p))

    if P_tmp.isEmpty() then break
    P_rdd = P_rdd.union(P_tmp)
end
```

Data caching in distributed memory

Iterative computation:

- TransitiveProperty P ($P \circ P \sqsubseteq P$).

$$\text{subOrganisationOf} \circ \text{subOrganisationOf} \sqsubseteq \text{subOrganisationOf} \quad (7)$$

- Entailment rule $\mathcal{R}_{P \circ P \sqsubseteq P}$:

$$\textbf{if } P_{rdd}(x, y), P_{rdd}(y, z) \textbf{ then } P_{rdd}(x, z)$$

- Spark programme $\mathcal{P}_{P \circ P \sqsubseteq P}$:

```
while True do
    P_tmp = P_rdd.map(lambda (x_p, y_p) : (y_p, x_p)).join(P_rdd)
                  .map(lambda (y_k, (x_p, z_p)) : (x_p, z_p))
    P_tmp.cache()
    if P_tmp.isEmpty() then break
    P_rdd = P_rdd.union(P_tmp)
end
```

Imperial College London

Data caching in distributed memory

- $\text{GraduateStudent}_{rdd}$ will be used three times:

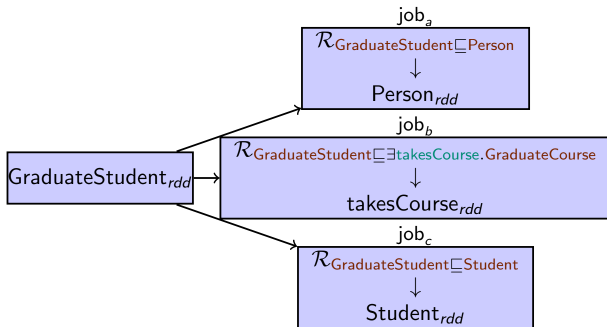


Figure: Caching $\text{GraduateStudent}_{rdd}$ for Repeated Usage

Imperial College
London

The advantage of using Spark (2)

More flexible job scheduling as compared to Hadoop:

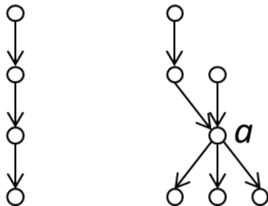


Figure: Job Scheduling between Hadoop (left) and Spark (right)

Imperial College London

DAG for parallelising reasoning

Consider $\text{Person} \sqcap \exists \text{takesCourse}.\text{Course} \sqsubseteq \text{Student}$:

- $\mathcal{R}_{\text{Person} \sqcap \exists \text{takesCourse}.\text{Course} \sqsubseteq \text{Student}}$:

  $$\textbf{if } \text{Person}_{rdd}(x),\ \text{takesCourse}_{rdd}(x, y),\ \text{Course}_{rdd}(y)$$
  $$\textbf{then } \text{Student}_{rdd}(x)$$

- $\mathcal{P}_{\text{Person} \sqcap \exists \text{takesCourse}.\text{Course} \sqsubseteq \text{Student}}$:

$$\begin{aligned}
\text{Student}_{tmp_1} &= \text{takesCourse}_{rdd}.\text{map(lambda } (x_t, y_t) : (y_t, x_t)) \\
&\quad .\text{join(Course}_{rdd}.\text{map(lambda } y_c : (y_c, y_c))) \\
&\quad .\text{map(lambda } (y_k, (x_t, y_c)) : x_t)) \\
\text{Student}_{tmp_2} &= \text{Student}_{tmp_1}.\text{intersection(Person}_{rdd}) \\
\text{Student}_{rdd} &= \text{Student}_{rdd}.\text{union(Student}_{tmp_2})
\end{aligned}$$

DAG for parallelising reasoning

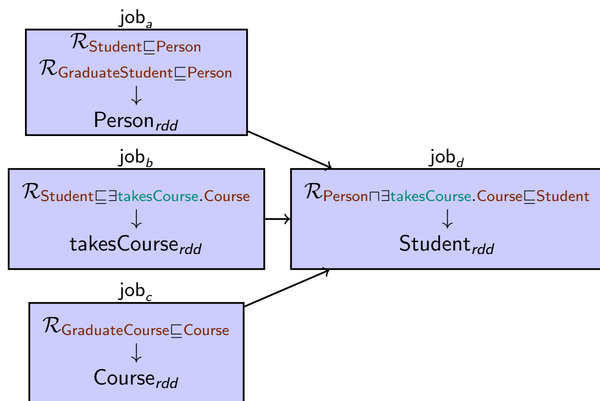


Figure: DAG Scheduling for $\mathcal{R}_{\text{Person} \sqcap \exists \text{takesCourse}.\text{Course} \sqsubseteq \text{Student}}$

Optimising programme execution order

Executing job_a, job_b and job_c before job_d is the best order.

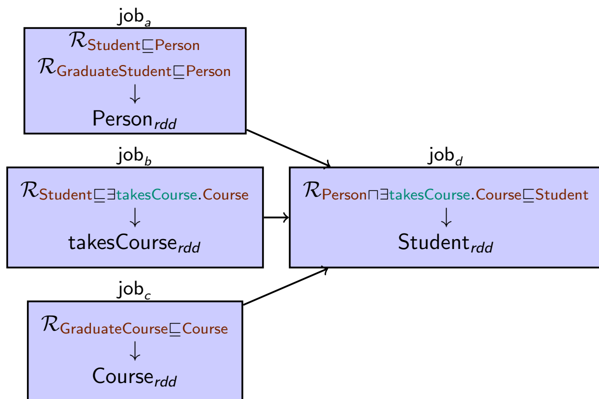


Figure: DAG Scheduling for $\mathcal{R}_{Person \sqcap \exists takesCourse.Course \sqsubseteq Student}$

Imperial College London

Ordering Spark Programmes

Consider $P_1 \sqsubseteq P_2$, $P_2 \circ P_2 \sqsubseteq P_2$ and $P_2 \sqsubseteq P_3$:

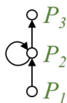


Figure: Acyclic property hierarchy

How about considering an addition axiom $P_3 \equiv P_1{}^-$?

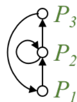


Figure: Cyclic property hierarchy

Table of Contents

Introduction

SPOWL Overview

SPOWL Features

Evaluation

Summary

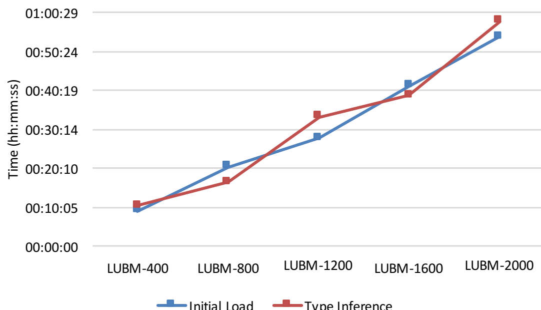
Evaluating SPOWL of reasoning materialisation

- Evaluation environment
  - A cluster of 9 machines running on a private cloud environment.
  - Each node with CPU @ 2.5GHz, 4 Cores, and 16 GB of Memory.
- Benchmarking dataset LUBM
  - LUBM-2000: about 270 million A-Box facts and 44GB in size.
- Comparison system: WebPIE
  - Using MapReduce as the computation framework.
  - Not using tableaux reasoners.
  - Not partitioning reasoning materialisation.
  - Compressing data before reasoning materialisation.

Performance of reasoning materialisation

- Reasoning materialisation by SPOWL

SPOWL	LUBM-400	LUBM-800	LUBM-1200	LUBM-1600	LUBM-2000
Initial Load	9m08s	20m30s	27m50s	41m20s	54m10s
Reasoning	10m19s	16m28s	33m20s	38m58s	58m08s
Total Time	19m27s	36m58s	1h01m10s	1h20m18s	1h52m18s



Performance of reasoning materialisation

- Reasoning materialisation by SPOWL

SPOWL	LUBM-400	LUBM-800	LUBM-1200	LUBM-1600	LUBM-2000
Initial Load	9m08s	20m30s	27m50s	41m20s	54m10s
Reasoning	10m19s	16m28s	33m20s	38m58s	58m08s
Total Time	19m27s	36m58s	1h01m10s	1h20m18s	1h52m18s

- Reasoning materialisation by WebPIE

WebPIE	LUBM-1000	LUBM-2000	LUBM-3000	LUBM-4000
compress	29m04s	59m37s	1h31m52s	2h01m59s
reasoning	30m36s	46m02s	58m27s	70m13s
decompress	14m03s	28m35s	49m16s	1h03m7s
Total	1h13m43s	2h14m14s	3h19m35s	4h15m19s

Imperial College London

Table of Contents

Introduction

SPOWL Overview

SPOWL Features

Evaluation

Summary

Summary

- SPOWL: a compiler for translating OWL axioms to Spark programmes.
  - Combine tableaux reasoning and rule-based reasoning.
  - Partition reasoning materialisation.
  - Use Spark to implement entailment rules.
  - Optimise the order of executing Spark programmes.
  - Preliminary evaluation over LUBM datasets.