**Incremental Reasoning on RDFS**

**Jules Chevalier, Julien Subercaze, Christophe Gravier, Frédérique Laforest**

Université de Lyon, F-42023, Saint-Étienne, France,
CNRS, UMR5516, Laboratoire Hubert Curien, F-42000, Saint-Étienne, France,
Université de Saint-Etienne, Jean Monnet, F-42000, Saint-Étienne, France.

{jules.chevalier,julien.subercaze,christophe.gravier,fr Edrique.laforest}@univ-st-etienne.fr

---

**Context**

The Semantic Web enables:

- to describe knowledge from data
- to leverage implicit knowledge through reasoning algorithms

The main limitations of current reasoning methods are:

- lack of scalability for large datasets
- inability to reason over knowledge from evolving data

We contribute to solving these problems by introducing Slider, an efficient incremental reasoner.

---

**Main Features**

- **Parallel and Scalable Execution**: Each inference rule is mapped to an independent module, receiving intended triples and later distributing them to other modules for further processing.
- **Duplicates Limitation**: Vertical partitioning [1] and multiple indexing limit the production of duplicates and avoid unnecessary computation.
- **Data Stream Support**: Slider can handle both dynamic triple streams and static triples sets by employing parallel architecture.
- **Fragment’s Customization**: Slider natively support both RDFS [4] and ρdf [5] fragments, and can be extended to any other fragments.

---

**Architectural Overview**

- **Input Manager**
  - Buffers
  - Thread Pool
  - Distributors
- **General Distributor**
- **Evolving Data**

---

**Experiments**

- Comparison with OWLIM-SE [2]
- Inference on both RDFS and ρdf
- 13 different ontologies
  - 5 generated with BSBM [3]
  - 2 from real-word datasets
  - 6 subClassOf ontologies
- 106.86% improvement for ρdf
- 36.08% improvement for RDFS
- 71.47% improvement in average

---

**References**


---

**Future Work**

- Implementation of more complex inference rules, to provide reasoning over more complex fragments.
- Just-in-time optimisations of the rules execution’s scheduling.
- Use of previous runs informations to adapt and be more reactive.

**Source Code and Demo**

The source code is available here: [https://github.com/juleschevalier/slider](https://github.com/juleschevalier/slider)

A demo can be found here: [http://demo-satin.telecom-st-etienne.fr/slider/](http://demo-satin.telecom-st-etienne.fr/slider/)

**Acknowledgement**

This work is supported by the OpenCloudware project. OpenCloudware is funded by the French FSN (Fond national pour la Société Numérique), and is supported by Pôles Minalogic, Systematic and SCS.