Incremental rule-based reasoning for streaming data

Rehab Albeladi
Introduction

- **Aim:**
  - To enable rule-based reasoning for RDF streaming data efficiently

- **Design decisions:**
  - Data-driven incremental reasoning using Rete [1] networks
  - Native support for RDF streams
  - Support RIF rules
**Approach**

**Figure 1**: System architecture
Approach

- Example Rete network for rdfs9 and rdfs10
- Alpha and beta memories are managed using time windows
- Entailed triples are assigned time validity as the intersection of the time intervals of the triples that produced it

**rdfs9**: IF u rdfs:subClassOf x AND v rdf:type u THEN v rdf:type x

**rdfs10**: IF u rdf:type rdfs:Class THEN u rdfs:subClassOf u

![Example Rete network](Figure 2: Example Rete network)
Experiments

- **Dataset**: Real-world RDF streams from the sensorGrid4Env [2] project

- **Functionality tests**: we tested a set of RIF rules ranging in complexity

- **Performance**:
  - We first compared the processing time against the Jena static reasoner
  - Currently working on comparing our system against Etalis [3] and Sparkwave [4]
Experiments

Figure 3: Comparing against Jena
Optimisation

- Static optimisation:
  - Based on [5], the initial Rete network is built following this simple heuristic: 
    \[(s, p, o) < (s, \ ?, o) < (\ ?, p, o) < (s, p, \ ?) < (\ ?, \ ?, o) < (s, \ ?, \ ?) < (\ ?, p, \ ?) < (\ ?, \ ?, \ ?)\]

- Adaptive optimisation:
  - We opt for a coarse adaptivity at an intra-query level as the per-tuple routing (e.g. Eddies) might be too expensive for the RDF model
  - The adaptive optimiser re-arrange the order of join nodes based on collected statistics
References


