Running Example – Data Model

Observation

Sensor

Person

Room

where

observes

subClassOf

subPropOf

1

2

3

subClassOf

subPropOf

discuss

who

isWith

isConnectedTo

isIn
Add reasoning in window-based RSPs

Naïve solution: materialize everything, every time

But windows slide:

- The materialisation is executed every time the window updates
- Only part of data changes at each window update
- Materialisation is (usually) an expensive task
Naïve solution: an example
Incremental maintenance

- Adopt an incremental approach
- Compute only the differences that should be removed and added from the materialization
Incremental maintenance: an example

TBOX

\[ \text{dom(}x\text{)} \subseteq \text{Person} \]
\[ \text{rng(}y\text{)} \subseteq \text{Speech} \]

To be deleted

To be renewed

To be added

http://streamreasoning.org/sr4ld2013
The common problem in designing incremental maintenance techniques is in the management of deletions.

In general it is not possible to foresee the statement deletions:
- DRed works with random insertions and deletions.

In our setting it is possible:
- The window operator allow us to determine when statements will be removed.
Variation of DRed for RDF streams

It pushes the maintenance algorithm in the window operator

An IMaRS window is a sliding window with four parameters:
- $\omega$: the size of the window
- $\beta$: the slide of the window
- $T$: the TBox that describes the data model
- $M$: the maintenance program

One of the central IMaRS concepts is the expiration time
Expiration time

- Every time a statement is added to the window, it is annotated with an **expiration time**
- The expiration time indicates when the statement should be removed from the materialization

**TBOX**

\[
\text{dom}(\cdot) \subseteq \text{dom}(\cdot) \subseteq \text{dom}(\cdot)
\]

\[
\text{rng}(\cdot) \subseteq \text{rng}(\cdot) \subseteq \text{rng}(\cdot)
\]

**Current time**

10

**Window (2,1)**

- The inferred statements expire
- The inferred statements will exit at 13
- The statement will exit at 13

[Diagram showing the expiration time and materialization process]
Expiration time generation

- At each window update

- The computation is done through the execution of a **maintenance program**

```plaintext
Statement deletion

Expiration time assignment

Expiration time update

To be renewed

Inferred statements

Explicit statements

To be added

To be removed

Expiration time assignment
```
The maintenance program computes the delta sets $\Delta^+$ and $\Delta^+$.

- It is a **logic program**

The program is executed every time the content changes.

- In our context, the program is executed every time the window slides.

The program is composed by **maintenance rules**.

- A maintenance rule adds a statement in a set **(context)** if the preconditions are satisfied.
The maintenance program uses four contexts to build the delta sets $\Delta^{\uparrow-}$ and $\Delta^{\uparrow+}$

- *Mat*: the current materialization
- *Ins*: the input that enters the stream and the related inferred statements

Additionally, two support sets are used:

- *New*: statements to be added to the materialization
- *Ren*: renewed statements

The new materialization is computed as

$$Mat \cup \Delta^{\uparrow+} \setminus \Delta^{\uparrow-}$$
IMaRS maintenance rules

- Two examples of maintenance rules:

\[ \Delta^{-}(s, p, o)[e] \leftarrow Mat(s, p, o)[e] \cdot e < now \]

A triple is removed by the materialization when its expiration time expires

\[ Ins(x, p, z)[e] \leftarrow New(x, p, y)[e1] \cdot Ins(x, p, y)[e2] \cdot Ins(p, isA, TransitiveProperty)[e3] \cdot e = \min\{e1, e2, e3\} \]

When a triple \( <s, p, o> \) enters the window, \( p \) is transitive and there is a triple \( <o, p, k> \) in the \( Ins \) context, then the triple \( <s, p, k> \) is a candidate for the addition in the materialization
The maintenance program is composed by two sets of maintenance rules:
- One set of fixed maintenance rules
- One dependent on the ontological language

The ontological language should be expressed as a set of inference rules, e.g.
\[
T(?x, ?p, ?z) :- \\
T(?p, rdf:type, owl:TransitiveProperty), \\
T(?x, ?p, ?y), T(?y, ?p, ?z)
\]

It does not depend on the TBox!
Generation of the maintenance program

Rewriting functions

Maintenance program generator

Ontological language

Maintenance program

TBox

IMaRS Window \((\omega, \beta)\)
Example: DRed

TBOX
\[ \text{tr}(\text{msg}) \]

Current time
12

Window (3,1)

Delete
Rederive

http://streamreasoning.org/sr4ld2013
Example: IMaRS

TBOX
tr(\text{rng}(\ldots))

Current time
12

Window (3,1)

<table>
<thead>
<tr>
<th>3</th>
<th>←</th>
<th>1</th>
</tr>
</thead>
</table>

http://streamreasoning.org/sr4ld2013