On Computing Deltas of RDF/S Knowledge Bases

Dimitris Zeginis, Yannis Tzitzikas and Vassilis Christophides

Institute of Computer Science (ICS)
Foundation for Research and Technology Hellas (FORTH), Crete, GREECE

Ontology Evolution

- Ontologies change over time because:
  - The real world is dynamic (or changes in the domain of interest)
  - Users’ perspective, needs or desires evolve (or changes in the conceptualizations)
  - Adaptations to the employed ontology language (or changes in specification)
  - Incomplete, faulty or inaccurate conceptualizations

- Two setting in which changes occur:
  - Controlled Evolution (e.g. Ontology Editors)
  - Open environments (i.e. the Semantic Web)
What is Change?

Real World + Ontologies and KBs = Ontology Evolution Algorithm

Capture Changes
Build Update Request

Add Class(...)
Populate Class(...)
Add Relationship(...)

What is Diff?

Real World + Ontologies and KBs = Comparison Algorithm (diff)

Add Class(...)
Populate Class(...)
Add Relationship(...)

...
RDF/S based Ontologies

K

\[
\begin{aligned}
&\text{Person} \\
&\text{Address [Literal]} \\
&\text{TA} \\
&\text{Student} \\
&\text{Jim}
\end{aligned}
\]

K'

RDF Knowledge Bases K, K'

\[
\begin{aligned}
&\text{Person} \\
&\text{Address [Literal]} \\
&\text{Student} \\
&\text{TA} \\
&\text{Jim}
\end{aligned}
\]

\[
\begin{aligned}
\{ & \text{(Person type Class),} \\
& \text{(Student type Class),} \\
& \text{(TA type Class),} \\
& \text{(Student subClassOf Person),} \\
& \text{(TA subClassOf Person),} \\
& \text{(Address type Property),} \\
& \text{(Address domain Student),} \\
& \text{(Address range Literal),} \\
& \text{(Jim type Student),} \\
& \text{(Jim type Person)}\}
\end{aligned}
\]

• K and K' are equivalent (K ~ K') if C(K)=C(K')

RDF KB Closures C(K), C(K')

Unique RDF/S KB Reductions

K

\[
\begin{aligned}
&\text{A} \\
&\text{B} \\
&\text{C}
\end{aligned}
\]

K_1

\[
\begin{aligned}
&\text{A} \\
&\text{B} \\
&\text{C}
\end{aligned}
\]

K_2

\[
\begin{aligned}
&\text{A} \\
&\text{B} \\
&\text{C}
\end{aligned}
\]

• K, K_1 and K_2 are equivalent
• K_1 and K_2 are redundancy free, but K_1 ≠ K_2.

• Reduction of a KB R(K) is smallest in size set of triples such that C(R(K))=C(K)

• K is redundancy free KB RF(K)) if it does not contain explicit triples which can be inferred from K
Distinctions of RDF/S Triple Sets

- **K** explicit triples
- **C(K)** inferred triples
- **K-R(K)** Redundant explicit triples
- **C(K)-R(K)** Redundant triples

Comparing RDF/S KBs

- Which operations can transform K to K'?
  - • The output of a comparison function is a set of primitive change operations of the form **Add(t)**, **Del(t)** where t is a triple of K and K'  
    - • We call such a set **Delta** \(\Delta(K \rightarrow K')\)
  - • Deltas can be exploited for providing **Versioning** and **Synchronization** services

Therefore their **size** (i.e. number of change ops) is crucial to reduce the amount of data that have to be stored or exchanged over the network
Limits of Text-based Diffs

Why don’t we use CVS as we do for text?
• RDF/S KBs have not a unique syntax

```xml
<rdfs:Class rdf:ID="Class1"/>
</rdfs:Description>
```

• RDF/S KBs (i.e. graphs) have not a unique serialization

• RDF/S KBs imply inferred facts w.r.t. semantics of RDFS

```
K
  Student
    ↓
  Jim

K'
  Student
    ↓
  Jim
```

Outline

• Comparison functions
  – Size of Deltas
• Executing Deltas
  – Change Operation Semantics
  – Properties of RDF Deltas and Change Operation Semantics
• Synopsis of Results
• Related work
• Concluding remarks
Comparison Function $\Delta_e(K \rightarrow K')$

- **Delta Explicit ($\Delta_e$):** takes into account only explicit triples
  
  \[ \Delta_e(K \rightarrow K') = \{\text{Add}(t) \mid t \in K' - K\} \cup \{\text{Del}(t) \mid t \in K - K'\} \]

Comparison Function $\Delta_c(K \rightarrow K')$

- **Delta Closure ($\Delta_c$):** takes also into account inferred triples
  
  \[ \Delta_c(K \rightarrow K') = \{\text{Add}(t) \mid t \in C(K') - C(K)\} \cup \{\text{Del}(t) \mid t \in C(K) - C(K')\} \]
Comparison Function $\Delta_d(K \rightarrow K')$

- **Delta Dense ($\Delta_d$):** return the explicit triples of one KB that does not exist at the closure of the other KB
  
  $\Delta_d(K \rightarrow K') = \{\text{Add}(t) \mid t \in K' - C(K)\} \cup \{\text{Del}(t) \mid t \in K - C(K')\}$

Comparison Function $\Delta_{dc}(K \rightarrow K')$

- **Delta Dense & Closure ($\Delta_{dc}$):** resembles $\Delta_d$ regarding additions and $\Delta_c$ regarding deletions
  
  $\Delta_{dc}(K \rightarrow K') = \{\text{Add}(t) \mid t \in K' - C(K)\} \cup \{\text{Del}(t) \mid t \in C(K) - C(K')\}$
Comparison Functions: Example

\[ \Delta_e(K \rightarrow K') = \{ \text{Del(TA subClassOf Person), Del(Address domain Student), } \]
\[ \text{Del(Jim type Student), Add(TA subClassOf Student), } \]
\[ \text{Add(Address domain Person), Add(Jim type Person) } \} \]

\[ \Delta_c(K \rightarrow K') = \{ \text{Del(Jim type Student), Add(TA subClassOf Student), } \]
\[ \text{Add(Address domain Person), Add(Address domain TA) } \} \]

\[ \Delta_d(K \rightarrow K') = \{ \text{Del(Jim type Student), Add(TA subClassOf Student), } \]
\[ \text{Add(Address domain Person) } \} \]

\[ \Delta_{dc}(K \rightarrow K') = \{ \text{Del(Jim type Student), Add(TA subClassOf Student), } \]
\[ \text{Add(Address domain Person) } \} \]

\[ \Delta_e(K' \rightarrow K) = \{ \text{Del(TA subClassOf Student), Del(Address domain Student), } \]
\[ \text{Del(Jim type Person), Add(TA subClassOf Person), } \]
\[ \text{Add(Address domain Student), Add(Jim type Student) } \} \]

\[ \Delta_c(K' \rightarrow K) = \{ \text{Del(TA subClassOf Student), Del(Address domain Person), } \]
\[ \text{Add(Jim type Student) } \} \]

\[ \Delta_d(K' \rightarrow K) = \{ \text{Del(TA subClassOf Student), Del(Address domain Person), } \]
\[ \text{Add(Jim type Student) } \} \]

\[ \Delta_{dc}(K' \rightarrow K) = \{ \text{Del(TA subClassOf Student), Del(Address domain Person), } \]
\[ \text{Add(Jim type Student) } \} \]
**Size of Deltas**

**Existing Approaches**
- $\Delta_{\text{closure}}$  
- $\Delta_{\text{explicit}}$  
- $\Delta_{\text{dense \& closure}}$

**Our Approach**
- $\Delta_{\text{dense}}$  

**On Executing Deltas**

- *We need to execute the Change operations returned by the Delta functions in order to transform one KB to another*
  - $\Delta_x(K \rightarrow K') (K)$ ?
  - $\Delta_x(K' \rightarrow K) (K')$ ?
  where $x \in \{e, c, d, dc\}$
- A KB $K$ satisfies a set of $\text{Add}(t)$ and $\text{Del}(t)$ operations $\Delta_x$ iff:
  - $\Delta_x = \{ \text{Add}(t) \}$ and $t \in C(K)$
  - $\Delta_x = \{ \text{Del}(t) \}$ and $t \notin C(K)$
  - $K$ satisfies every element of $\Delta_x$

- **Two Important Questions:**
  - *Does the resulting, KB $\Delta_x(K \rightarrow K') (K)$, satisfy the Delta $\Delta_x(K \rightarrow K)$ for every $x \in \{e, c, d, dc\}$?*
  - *Does the order of execution of change operations matters?*
**Change Operation Semantics**

- **What is the semantics of the Delta change operations?**
  - Adequate Pre and Post conditions of \( Add(t) \) and \( Del(t) \) operations

- Two alternative change operation semantics:
  - \( \mathcal{U}_p \) (\( p \) from plain): based on plain set theoretic semantics (additions and deletions of triples)
  - \( \mathcal{U}_r \) (\( ir \) from inference & reduction): incurs inference and redundancy elimination

\[
\Delta_e(K \rightarrow K') = \{ \text{Del(TA subClassOf Person), Del(Jim type Student), Add(TA subClassOf Student), Add(Jim type Person), Del(Address domain Student), Add(Address domain Person)} \}
\]

<table>
<thead>
<tr>
<th>Oper.</th>
<th>Precond.</th>
<th>Postcond.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add(t)</td>
<td>( t \in K )</td>
<td>( K' = K )</td>
</tr>
<tr>
<td></td>
<td>( t \in C(K) - K )</td>
<td>( K' = K \cup {t} )</td>
</tr>
<tr>
<td></td>
<td>( t \not\in C(K) )</td>
<td>( K' = K \cup {t} )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Oper.</th>
<th>Precond.</th>
<th>Postcond.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Del(t)</td>
<td>( t \in K )</td>
<td>( K' = K - {t} )</td>
</tr>
<tr>
<td></td>
<td>( t \in C(K) - K )</td>
<td>( K' = K )</td>
</tr>
<tr>
<td></td>
<td>( t \not\in C(K) )</td>
<td>( K' = K )</td>
</tr>
</tbody>
</table>
\[
\mathcal{U}_T - \text{semantics}
\]

<table>
<thead>
<tr>
<th>Oper.</th>
<th>Precond.</th>
<th>Postcond.</th>
</tr>
</thead>
<tbody>
<tr>
<td>\text{Add}(t)</td>
<td>\text{t} \in \text{K}</td>
<td>\text{K'} = \text{K}</td>
</tr>
<tr>
<td></td>
<td>\text{t} \in \text{C(K)} - \text{K}</td>
<td>\text{K'} = \text{K}</td>
</tr>
<tr>
<td></td>
<td>\text{t} \notin \text{C(K)}</td>
<td>\text{K'} = \text{R(K + {\text{t}})}</td>
</tr>
</tbody>
</table>

\[
\Delta_c(K \rightarrow K') = \{ \text{Del(Jim type Student)}, \text{Add(TA subClassOf Student)}, \text{Add(Address domain Person)} \}
\]

\[
\Delta_c(K' \rightarrow K) = \{ \text{Del(TA subClassOf Student)} \}
\]

The execution of $\Delta_c$ w.r.t. $\mathcal{U}_P$ is not always correct.
Properties of \((\Delta_x, \cup_y)\) Pairs

- **Which combinations of Change op semantics and Delta functions satisfy the following properties?**
  - **Correctness:** A pair \((\Delta_x, \cup_y)\) is correct if for any \(K\) and \(K'\), the KB satisfying \(\Delta_x(K \rightarrow K')\) (or \(\Delta_x(K \rightarrow K')\)) under \(\cup_y\) semantics is equivalent to \(K'\) (or \(K\))
    - \(\Delta_x(K \rightarrow K')^{\cup_y} \equiv K'\)
    - \(\Delta_x(K' \rightarrow K)^{\cup_y} \equiv K\)
    where \(x \in \{e,c,d,dc\}\) and \(y \in \{p, ir\}\)
  - **Non Redundancy:** the execution of \(\Delta_x(K \rightarrow K')\) upon \(K\) assuming \(\cup_y\) semantics results to a redundancy free KB
  - **Semantic Identity:** Delta is empty when its operands KBs are equivalent
    - If \(K \equiv K'\) then \(\Delta_x(K \rightarrow K') = \emptyset\)

**Correctness of \((\Delta_d, \cup_{ir})\)**

\[
\Delta_d(K' \rightarrow K) = \{\text{Del}(TA \text{ subClassOf Student}), \text{Del}(\text{Address domain Person}) \}
\text{Add}(\text{Jim type Student})
\]

<table>
<thead>
<tr>
<th>Precond.</th>
<th>Postcond.</th>
</tr>
</thead>
<tbody>
<tr>
<td>(t \in K')</td>
<td>(K = R(C(K') \setminus {t}))</td>
</tr>
<tr>
<td>(t \in C(K'))</td>
<td>(K = K')</td>
</tr>
<tr>
<td>(t \in C(K'))</td>
<td>(K = K')</td>
</tr>
</tbody>
</table>

The execution of \(\Delta_d\) w.r.t. \(\cup_{ir}\) is not always correct
Correctness of \(\Delta_{dc} \cup_{ir} \)

\[ \Delta_{dc}(K' \rightarrow K) = \{\text{Del}(TA \text{ subClassOf Student}), \text{Del}(Address \text{ domain Person}), \text{Del}(Address \text{ domain TA}), \text{Add}(Jim \text{ type Student})\} \]

Precond. | Postcond.  
---|---
\(t \in K'\) | \(K = R(C(K') - \{t\})\)
\(t \in C(K') \rightarrow K'\) | \(K = K'\)
\(t \not\in C(K')\) | \(K = K'\)

The execution of \(\Delta_{dc} \) w.r.t. \(\psi_{ir}\) is correct

Semantics Identity for \(\Delta_e\)

\[ \Delta_e(K' \rightarrow K'') = \{\text{Add}(TA \text{ subClassOf Person})\} \]

Although \(K' \sim K''\) we have \(\Delta_e(K' \rightarrow K'') \neq \emptyset\)

- So \(\Delta_e\) does not satisfy semantic identity
  (although used by most existing systems)
- However \(\Delta_c, \Delta_d\) and \(\Delta_{dc}\) satisfy semantic identity
Order of Change Op Execution

\[ \Delta_{dc}(K \rightarrow K') = \{ \text{Del(Student subClassOf Person)}, \text{Del(TA subClassOf Person)} \} \]

- Precond.: \( t \in K \)  
- Postcond.: \( K' = R(C(K) - \{t\}) \)

- Precond.: \( t \in C(K) - K \)  
- Postcond.: \( K' = K \)

- Precond.: \( t \in C(K) \)  
- Postcond.: \( K' = K \)

Multi-pass Execution Mode

- To avoid nondeterminism and to ensure correctness, a multi-pass execution mode is needed for:
  - \( \Delta_c, \Delta_d, \Delta_{dc} \) under \( U_{ir} \) semantics
- The algorithm repeats a loop until every change operation is satisfied
  Execute \((K, \Delta_x)\) where \( x \in \{c,d,dc\}\)
  Repeat
  Pick an element \( \delta \in \Delta_x \) such that is not satisfied by \( K \)
  \( K = \delta^{U_{ir}}(K) \) i.e. apply on \( K \) the appropriate post-conditions of \( \delta \) w.r.t \( U_{ir} \)-semantics
  // Note that \( K \) may still not satisfy \( \delta \)
  Until \( \{\delta | \delta \in \Delta_x \text{ and not satisfied by } K\} = \text{Empty} \)

- We have proved that the loop always terminates
  - Finally all the operations of the delta will be satisfied
Synopsis of Results

<table>
<thead>
<tr>
<th>Pair</th>
<th>Correct</th>
<th>Exec Mode</th>
<th>Semantic ID</th>
<th>Non redund.</th>
</tr>
</thead>
<tbody>
<tr>
<td>( (\Delta_s, \psi_s) )</td>
<td>Y</td>
<td>S</td>
<td>Y if ( RF(K) ) &amp; ( RF(K') )</td>
<td>Y if ( RF(K') )</td>
</tr>
<tr>
<td>( (\Delta_j, \psi_j) )</td>
<td>Y if ( K ) compl.</td>
<td>S</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>( (\Delta_i, \psi_i) )</td>
<td>N</td>
<td>-</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>( (\Delta_i, \psi_i) )</td>
<td>N</td>
<td>-</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>( (\Delta_i, \psi_i) )</td>
<td>N</td>
<td>-</td>
<td>Y if ( RF(K) ) &amp; ( RF(K') )</td>
<td>Y</td>
</tr>
<tr>
<td>( (\Delta_i, \psi_i) )</td>
<td>Y in ( \Psi^1 )</td>
<td>M</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>( (\Delta_i, \psi_i) )</td>
<td>Y</td>
<td>M</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>( (\Delta_i, \psi_i) )</td>
<td>Y in ( \Psi )</td>
<td>M</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>

1 Knowledge Base with unique reduction

2 If (a) \( K \) is complete or (b) \( C(K) - K \subseteq C(K') \)

Comparison With Related Work

<table>
<thead>
<tr>
<th>System</th>
<th>Comparison Function</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>CWM (w3c)</td>
<td>( \Delta_s(K \rightarrow K') )</td>
<td>( \psi_r )</td>
</tr>
<tr>
<td>RDF-Utils</td>
<td>( \Delta_a(K \rightarrow K') )</td>
<td>( \psi_r )</td>
</tr>
<tr>
<td>SemVersion</td>
<td>( \Delta_a(K \rightarrow K') )</td>
<td>( \psi_r )</td>
</tr>
<tr>
<td></td>
<td>( \Delta_a(K \rightarrow K') )</td>
<td>( \psi_r )</td>
</tr>
<tr>
<td>PromptDiff</td>
<td>Heuristic matchers</td>
<td>-</td>
</tr>
<tr>
<td>Ontoview</td>
<td>Rules for changes</td>
<td>-</td>
</tr>
<tr>
<td>SWKM</td>
<td>( \Delta_d(K \rightarrow K'), \Delta_d(K \rightarrow K') )</td>
<td>( \psi_p, \psi_r )</td>
</tr>
<tr>
<td></td>
<td>( \Delta_d(K \rightarrow K'), \Delta_d(K \rightarrow K') )</td>
<td>( \psi_p, \psi_r )</td>
</tr>
</tbody>
</table>

Existing systems do not use \( \Delta_d, \Delta_{dc} \) and \( \psi_r \)
Delta Algorithms

- The implementation of the comparison functions is based in the main memory representation of RDF graphs.
- The algorithm compares these two graphs in the following order:
  - Classes
  - Properties
  - Resources
  - Containers
- For each of the above kinds of elements of the first graph it finds the corresponding (mapped) element of the second graph and compares it accordingly.

Complexity of Delta Algorithms

<table>
<thead>
<tr>
<th>$K_1 \rightarrow K_2$</th>
<th>(i) all stored</th>
<th>(ii) only K's stored</th>
<th>(iii) only K's stored and labeling</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Delta_e$</td>
<td>$O(N)$</td>
<td>$O(N)$</td>
<td>$O(N)$</td>
</tr>
<tr>
<td>$\Delta_c$</td>
<td>$O(L)$</td>
<td>$O(N^2)$</td>
<td>$O(N^2)$</td>
</tr>
<tr>
<td>$\Delta_d$</td>
<td>$O(N)$</td>
<td>$O(N^2)$</td>
<td>$O(N)$</td>
</tr>
<tr>
<td>$\Delta_{dc}$</td>
<td>$O(\max(N_2, L_1))$</td>
<td>$O(N^2)$</td>
<td>$O(\max(N_2, N_1^2))$</td>
</tr>
</tbody>
</table>

- Notations used:
  - $N_1 = |K_1|$, $N_2 = |K_2|$, $L_1 = |C(K_1)|$, $L_2 = |C(K_2)|$
  - $N = \max(N_1, N_2)$ and $L = \max(L_1, L_2)$
- Three different settings:
  (i) $K_1, K_2, C(K_1)$ and $C(K_2)$ are stored in a hashtable
    - Decision if t belongs to that set in $O(1)$
  (ii) Only $K_1$ and $K_2$ are stored
    - The cost to compute $C(K_1)$ from $K_1$ is in $O(L_1) = O(N_1^2)$
  (iii) Only $K_1$ and $K_2$ are stored and subsumption is labeled
    - checking whether t in $C(K)$ in is almost constant
Inference Strength and Delta Sizes

• The inference strength of a knowledge base \( K \) denoted by \( is(K) \) is defined as:
  \[ is(K) = \frac{|C(K)| - |K|}{|K|} \]
  - If \( K = C(K) \) then \( is(K) = 0 \)
• The less this factor is, the less the execution times and delta sizes of the four comparison functions differ.
  - For \( \Delta_e \) all kinds of changes affect in the same way the delta size.
  - For \( \Delta_a \) additions or deletions that occur highly at the subsumption hierarchy affect more the delta size.
  - For \( \Delta_d \) all kinds of changes affect in the same way the delta size.
  - For \( \Delta_{dc} \) all additions have the same impact on the result. But deletions that occur highly at the subsumption hierarchy affect more the size of the produced delta.

Experiments Datasets

• Biological Data
  - Schema of the Gene ontology is very simple. It contains only one metaclass and all classes are instances of this metaclass.
  - Uses many blank nodes (~ 50%)
• Synthetic Data set
  - We created a sequence of four KBs: \( K_1, K_2, K_3, K_4 \) with
    • 100, 200, 300, and 400 classes respectively
    • 300, 600, 900, 1200 properties respectively
  - For each class, 10 instances were created,
  - For each property, 10 instances were created
  - All classes, property and their instances in \( K_i \) are also present in \( K_{i+1} \); their structuring may be different
  - The depth of the subclassOf hierarchy in each schema is 7.
Real Dataset: GO Ontology

- **is(K) is very small in this data set**

| $K_1$ | $K_2$ | $|\Delta (K_1 \rightarrow K_2)|$ Size of $\Delta (K_1 \rightarrow K_2)$ |
|-------|-------|----------------------------------|
| KBytes | Triples | KBytes | Triples | (T1) of $(\Delta (K_1 \rightarrow K_2))$ | $\Delta_e$ | $\Delta_d$ | $\Delta_c$ | $\Delta_a$ |
| 197    | 2898   | 0.225 | 201    | 2964   | 0.228 | 0.329 | 0.296 | 0.312 | 92    | 107   | 92    | 92    |
| 306    | 4496   | 0.225 | 331    | 4816   | 0.225 | 0.594 | 0.765 | 0.625 | 0.648 | 527   | 590   | 527   | 528   |
| 413    | 5994   | 0.218 | 418    | 6068   | 0.218 | 0.967 | 1.046 | 0.975 | 0.998 | 110   | 125   | 110   | 110   |
| 507    | 7340   | 0.206 | 512    | 7399   | 0.205 | 1.119 | 1.354 | 1.133 | 1.149 | 81    | 90    | 81    | 81    |
| 624    | 9020   | 0.196 | 638    | 9217   | 0.196 | 1.665 | 1.763 | 1.698 | 1.718 | 311   | 349   | 311   | 313   |
| 753    | 10800  | 0.182 | 757    | 12777  | 0.182 | 1.973 | 2.090 | 1.979 | 2.042 | 71    | 83    | 71    | 71    |
| 815    | 11680  | 0.179 | 822    | 11779  | 0.179 | 2.181 | 2.300 | 2.187 | 2.249 | 135   | 152   | 135   | 135   |

<table>
<thead>
<tr>
<th>Inference strength</th>
<th>Inference strength</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small variations in delta sizes</td>
<td></td>
</tr>
</tbody>
</table>

Synthetic Dataset: RDFS Schemas

- **is(K) is higher in this data set**

| $K_1$ | $K_2$ | $|\Delta (K_1 \rightarrow K_2)|$ Size of $\Delta (K_1 \rightarrow K_2)$ |
|-------|-------|----------------------------------|
| KBytes | Triples | KBytes | Triples | (T1) of $(\Delta (K_1 \rightarrow K_2))$ | $\Delta_e$ | $\Delta_d$ | $\Delta_c$ | $\Delta_a$ |
| 408    | 5162   | 0.821 | 919    | 10267  | 0.789 | 0.806 | 0.570 | 0.552 | 0.594 | 0.718 | 13999 | 22837 | 13864 | 16362 |
| 919    | 10267  | 0.789 | 1232   | 13589  | 0.806 | 1.449 | 1.387 | 1.388 | 23185 | 36449 | 23063 | 27553 |
| 1232   | 15389  | 0.806 | 1642   | 20460  | 0.826 | 2.217 | 1.623 | 1.702 | 32214 | 50415 | 32067 | 38941 |
| 408    | 5162   | 0.821 | 1642   | 20460  | 0.826 | 1.077 | 1.624 | 1.109 | 1.097 | 24004 | 41951 | 23937 | 26433 |

| • $\Delta_a$ is 63% bigger than $\Delta_e$ |
| • $\Delta_{dc}$ is 16% bigger that $\Delta_e$ |
| • satisfies semantic identify |
| • $\Delta_d$ is 1% smaller than $\Delta_e$ |

Notable variations in delta sizes (is(K) is higher for this data set)
Summary

- Four comparison functions were introduced and analyzed
  - Delta explicit ($\Delta_\epsilon$),
  - Delta closure ($\Delta_c$),
  - Delta dense ($\Delta_d$)
  - Delta dense & closure ($\Delta_{dc}$)

- Two change operation semantics were introduced and analyzed
  - One plain set theoretic $\mathcal{U}_p$
  - One that involves inference and redundancy elimination $\mathcal{U}_{ir}$

- We identified the always correct pairs: ($\Delta_\epsilon$, $\mathcal{U}_p$), ($\Delta_{dc}$, $\mathcal{U}_{ir}$)

- ($\Delta_{dc}$, $\mathcal{U}_{ir}$) is most promising:
  - Returns empty result if $K$, $K'$ are semantically equivalent
  - Resulting KB is redundancy free when applying $\Delta_{dc}(K \rightarrow K')$ on $K$
  - Gives the smallest result if $K'$ is an extension of $K$

Thanks for your attention!

Any questions?
Correctness of $\Delta_{d_r} \cup_{ir}$

$\Delta_{d}(K \rightarrow K') = \{ \text{Del(Student subClassOf Person)} \}$

**Precond.** $t \in K$  
**Postcond.** $K' = R(C(K) - \{t\})$

$t \in C(K) \rightarrow K' = K$

The execution of $\Delta_{d}$ w.r.t. $\cup_{ir}$ is not always correct

Correctness of $\Delta_{dc} \cup_{ir}$

$\Delta_{d}(K' \rightarrow K) = \{ \text{Del(Student subClassOf Person), Del(TA subClassOf Person)} \}$

**Precond.** $t \in K$  
**Postcond.** $K' = R(C(K) - \{t\})$

$t \in C(K) \rightarrow K' = K$

The execution of $\Delta_{dc}$ w.r.t. $\cup_{ir}$ is correct
Note on Closures

• $\forall K, K' \subseteq K \Rightarrow C(K') \subseteq C(K)$ and $K \subseteq C(K)$ (monotonicity)
  - $K \text{Del} \subseteq K$, $K-K\text{Del} \subseteq K$, $C(K\text{Del}) \subseteq C(K)$, $K\text{Del} \subseteq C(K\text{Del})$, $K\text{Del} \subseteq C(K)$
  - $K \text{Del} \subseteq C(K) - C(K\text{Del})$
  - $C(K) - C(K\text{Del}) \subseteq C(K) - K\text{Del}$
  - $C(K - K\text{Del}) \subseteq C(K) - K\text{Del}$
  - $C(K - K\text{Del}) \subseteq C(K) - C(K\text{Del})$
  - $K - K\text{Del} \subseteq C(K - K\text{Del}) \subseteq C(K) - C(K\text{Del}) \subseteq C(K)$

Comparison Functions: Example

RDFS Standard

$\Delta_a(K \rightarrow K') = \{ \text{Del}(\text{TA subClassOf Person}), \text{Del}(\text{Address domain Student}), \text{Del}(\text{Jim type Student}), \text{Add}(\text{TA subClassOf Student}), \text{Add}(\text{Address domain Person}), \text{Add}(\text{Jim type Person}) \}$

$\Delta_c(K \rightarrow K') = \{ \text{Del}(\text{Jim type Student}), \text{Del}(\text{Address domain Student}), \text{Add}(\text{TA subClassOf Student}), \text{Add}(\text{Address domain Person}) \}$

$\Delta_d(K \rightarrow K') = \{ \text{Del}(\text{Jim type Student}), \text{Del}(\text{Address domain Student}), \text{Add}(\text{TA subClassOf Student}), \text{Add}(\text{Address domain Person}) \}$

$\Delta_{dc}(K \rightarrow K') = \{ \text{Del}(\text{Jim type Student}), \text{Del}(\text{Address domain Student}), \text{Add}(\text{TA subClassOf Student}), \text{Add}(\text{Address domain Person}) \}$
Comparison Functions: Example

RDFS Standard

$\Delta_e(K' \rightarrow K) = \{\text{Del}(\text{TA subClassOf Student}), \text{Del}(\text{Address domain Person}), \text{Del}(\text{Jim type Person}) \text{ Add}(\text{TA subClassOf Person}), \text{Add}(\text{Address domain Student}), \text{Add}(\text{Jim type Student})\}$

Size=6

$\Delta_c(K' \rightarrow K) = \{\text{Del}(\text{TA subClassOf Student}), \text{Del}(\text{Address domain Person}), \text{Add}(\text{Address domain Student}), \text{Add}(\text{Jim type Student})\}$

Size=4

$\Delta_d(K' \rightarrow K) = \{\text{Del}(\text{TA subClassOf Student}), \text{Del}(\text{Address domain Person}), \text{Add}(\text{Address domain Student}), \text{Add}(\text{Jim type Student})\}$

Size=4

$\Delta_{dc}(K' \rightarrow K) = \{\text{Del}(\text{TA subClassOf Student}), \text{Del}(\text{Address domain Person}), \text{Add}(\text{Address domain Student}), \text{Add}(\text{Jim type Student})\}$

Size=4