On Computing Deltas of RDF/S Knowledge Bases

DIMITRIS ZEGINIS
Computer Science Department, University of Crete, GREECE
YANNIS TZITZIKAS
Computer Science Department, University of Crete, GREECE and Institute of Computer Science, FORTH-ICS, GREECE
VASSILIS CHRISTOPHIDES
Computer Science Department, University of Crete, GREECE and Institute of Computer Science, FORTH-ICS, GREECE

The ability to compute the differences that exist between two RDF/S knowledge bases (KB) is an important step to cope with the evolving nature of the Semantic Web (SW). In particular, RDF/S deltas can be employed to reduce the amount of data that need to be exchanged and managed over the network in order to build SW synchronization and versioning services. By considering deltas as sets of change operations, in this paper we introduce various RDF/S differential functions which take into account inferred knowledge from an RDF/S knowledge base. We first study their correctness in transforming a source to a target RDF/S knowledge base in conjunction with the semantics of the employed change operations (i.e. with or without side-effects on inferred knowledge). Then we formally analyze desired properties of RDF/S deltas such as, size minimality, semantic identity, redundancy elimination, reversibility and composability, as well as, identify those RDF/S differential functions that satisfying them. Subsequently, we experimentally evaluate the computing time and size of the produced deltas over real and synthetic RDF/S knowledge bases.

Categories and Subject Descriptors: H.0 [**Information Systems**]: ; I.2.4 [**Knowledge Representation Formalisms and Methods**]: Semantic networks; H.3.5 [**Online Information Services**]: Web-based services

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Semantic Web, RDF, Delta

1. INTRODUCTION

In order to cope with the evolving nature of the Semantic Web (SW) we need effective and efficient support for building adequate synchronization and versioning services. RDF/S deltas, reporting the differences that exist between two RDF/S

Accepted for publication at ACM Transactions on the Web (TWEB).
This is the authors version of the work. It is posted here by permission of ACM for your personal use. Not for redistribution. The definitive version was published in PUBLICATION, VOL#, ISS#, (DATE) http://doi.acm.org/10.1145/nnnnnn.nnnnnn.

Knowledge Bases (KB) have been proven to be crucial in order to reduce the amount of data that needs to be exchanged in this respect over the network [Heflin et al. 1999; Klein et al. 2002; Berners-Lee and Connolly 2004; Cloran and Irwin 2005].

Although RDF/S KBs can be serialized in various text formats (e.g. N-Triples[1], Trix[2], or XML[3]), existing delta-based support for software code, such as RCS[4] or CVS[5] [Berliner 1990], or for semi-structured XML data, such as [Marian et al. 2001], [Cobena et al. 2001], is not suited for RDF/S. This is mainly due to the fact that RDF/S KBs represent essentially graphs which (a) may feature several possible serialization orders, and (b) can be enriched with the semantics of RDFS specification [Brickley and Guha 2004] (e.g. edges can be also inferred from transitive relations such as class or property subsumption). Even through a standard serialization order among the triples of a RDF/S KB can be imposed (i.e. sorting by *subject*, *predicate* or *object*) existing comparison tools for plain text or XML documents clearly fail to consider knowledge inferred from the RDFS schemas associated with the RDF/S KBs. For these reasons, several tools have been recently developed for comparing RDF/S KBs evolving autonomously on the SW, for example, SemVersion [Volkel et al. 2005], PromptDiff [Noy and Musen 2002], Ontoview [Klein et al. 2002], RDF-Utils [Ding et al. 2005] and CWM [Berners-Lee and Connolly 2004].

However, existing RDF differential tools focus almost exclusively on the generation of deltas that will be exploited mainly by humans. More precisely, SemVersion [Volkel et al. 2005] proposes two Diff algorithms: (a) one *structure-based* which returns a set-based difference of the triples explicitly forming the two graphs, and (b) one *semantic-aware* which also takes into account the triples inferred by the associated RDFS schemas. PromptDiff [Noy and Musen 2002; Noy et al. 2006] is an ontology-versioning environment, that includes a differential algorithm based on a heuristic matching of ontology concepts [Noy and Musen 2004], while the particular attention is given on the visualization of the computed deltas [Noy et al. 2004]. Furthermore, Ontoview [Klein et al. 2002] is an ontology management system, able to compare two ontology versions and recognize their differences. Notably, it allows users to specify the conceptual relations (i.e. equivalence, subsumption) that exist between the two versions of an ontology concept. Moreover, [Ding et al. 2005; Tummarello et al. 2004] introduce the notion of RDF/S molecules as the finest components to be used when comparing RDF/S KBs (especially when anonymous nodes are used). Finally, tracking the evolution of ontologies when changes can be synchronously monitored, e.g. in collaborative environments, has been addressed in [Klein and Noy 2003; Plessers and Troyer 2005; Zhang et al. 2003].

In contrast to previous related work, in this paper we are interested in differential functions that produce RDF/S deltas exhibiting interesting formal properties (e.g. regarding their size and execution semantics) so they can be interpreted both by programs and humans. In particular, we are interested in RDF/S deltas as sets of change operations (i.e. insertions and removals) that enable us to successfully

[1] www.w3.org/2001/sw/RDFCore/ntriples/
[2] www.w3.org/2004/03/trix/
[3] www.w3.org/TR/rdf-syntax-grammar/
[4] www.gnu.org/software/rcs
[5] www.cvshome.org

transform one RDF/S KB into another by taking into account or not inferred knowledge (i.e. with or without side-effects). This functionality is crucial for SW search engines[6] hosting multiple versions of RDF/S KBs. Rather than storing complete snapshots of all RDF/S KBs, we can archive only one version and be able to restore any version of interest by executing consecutive deltas. If a delta's target version is newer (vs. older) than its source version in the archive, then it is called a *forward delta* (*backward delta*). If we consider, for example, the two RDF/S KBs K and K' of Figure 1 represented as sets of explicitly defined triples, we would like to compute $\Delta(K, K')$ that correctly transform K to K' (or vice versa $\Delta(K', K)$).

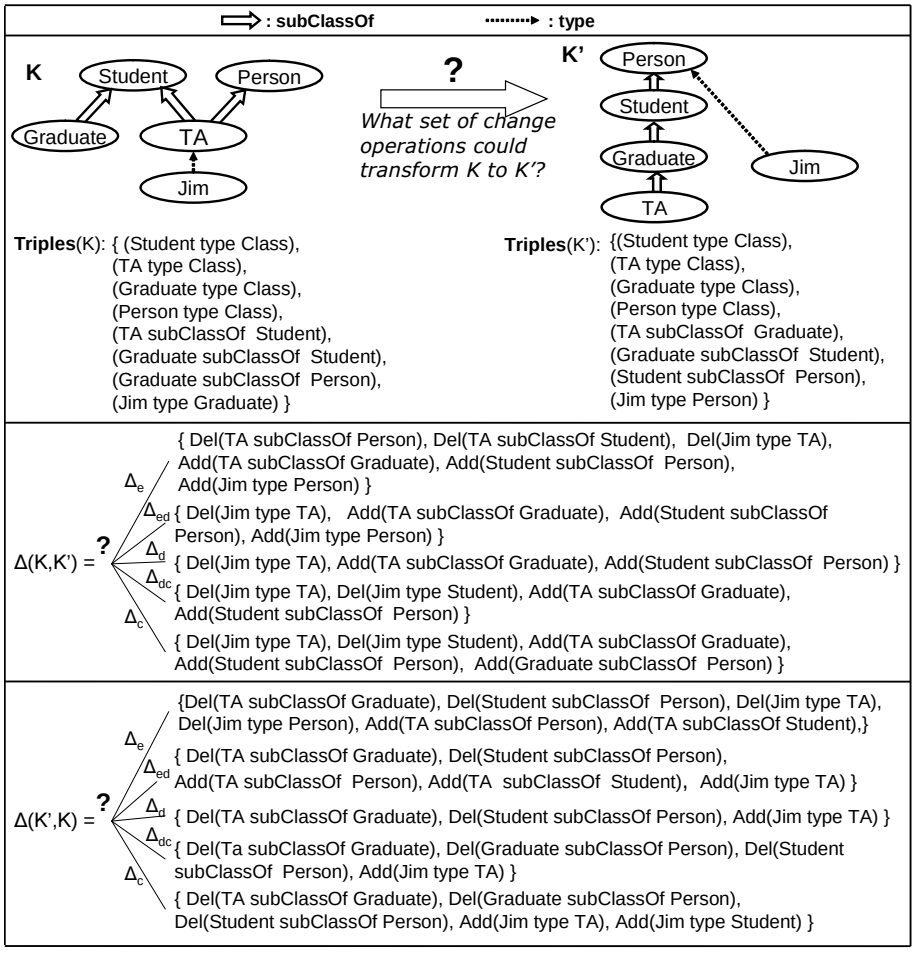


Fig. 1. Transforming K to K' and vice versa

In the simplest case, RDF/S deltas can be executed over a source KB version using plain-set theoretic semantics: to obtain the target KB, a triple t (involving any RDF/S predicate) can be added ($Add(t)$) to or deleted ($Del(t)$) from the set of explicit triples forming the source KB. Going back to the example of Figure 1 the

[6]See for instance watson.kmi.open.ac.uk and swoogle.umbc.edu

target K' can be obtained by the following set Δ_e (e stands for $e$xplicit) of change operations executed without any side-effect:

$$\begin{aligned}\Delta_e \ = \ \{&Del(TA \ subClassOf \ Person), \ Del(TA \ subClassOf \ Student),\\ &Del(Jim \ \ type \ \ TA), \ Add(TA \ subClassOf \ Graduate),\\ &Add(Student \ subClassOf \ Person), \ Add(Jim \ \ type \ \ Person)\}\end{aligned}$$

Δ_e is actually provided by the majority of existing RDF/S differential tools [Berners-Lee and Connolly 2004; Volkel et al. 2005; Ding et al. 2005] and consists of insertions and removals of triples explicitly defined in K and K'. However, by assuming the closure of RDF/S KBs [Brickley and Guha 2004] we can observe that deleting from K the triple (TA *subClassOf* Person) is not necessary since it can still be inferred by K (due to subsumption transitivity). Analogously, inserting in K the triple (Graduate *subClassOf* Person) is redundant since it can be inferred by K. This brings us to a change operation semantics with side-effects over the inferred triples of the involved RDF/S KBs. More precisely, under this semantics the target KB will not contain any explicit triple that can be inferred while preserves as much as possible the knowledge of source KB. For instance in Figure 1 the target K' can be obtained by the following set Δ_c (c stands for $c$losure [Volkel et al. 2005]) of change operations executed with side-effects:

$$\begin{aligned}\Delta_c \ = \ \{&Del(Jim \ \ type \ \ TA), Del(Jim \ \ type \ \ Student), \ Add(TA \ subClassOf \ Graduate),\\ &Add(Student \ subClassOf \ Person), \ Add(Graduate \ subClassOf \ Person)\}\end{aligned}$$

The question that naturally arises is whether unnecessary deletions or redundant insertions can be avoided as early as possible i.e. during delta computation instead of delta execution. This is the objective of the other three differential functions illustrated in Figure 1. Consider for instance the following set Δ_d (d stands for $d$ense) of change operations:

$$\begin{aligned}\Delta_d \ = \ \{&Del(Jim \ \ type \ \ TA), \ Add(TA \ subClassOf \ Graduate),\\ &Add(Student \ subClassOf \ Person)\}\end{aligned}$$

Δ_d is the smallest in size delta with only three change operations compared to Δ_e that has six and Δ_c that has five. However, Δ_d cannot always successfully transform one RDF/S KB to another. In our example, Δ_d cannot be used to migrate backwards from K' to K (under both execution semantics) since $Del(Graduate \ subClassOf \ Person)$ is an update not included in Δ_d. For this reason, we need to consider two more RDF/S differential functions also involving inferred triples such as Δ_{dc} (dc stands for $d$ense & $c$losure) illustrated in Figure 1. Still the resulting sets of operations have at most the same size as those returned by Δ_c. Finally, we consider one more RDF/S differential function named Δ_{ed} (ed stands for $e$xplicit & $d$ense) illustrated in Figure 1. Still the resulting sets of operations have at most the same size as those returned by Δ_e.

Clearly, *small sized* deltas yielding as few as possible change operations are quite beneficial in various backward and forward versioning policies if of course can be successfully executed under one of the previous change operation semantics. Deltas should not report any change between two *semantically equivalent* RDF/S graphs

(i.e. with the same set of explicit and inferred triples). Additionally, when executed, deltas should at best result in a KB that is *redundancy free*. Furthermore, when we want to propagate changes across distributed RDF/S graph versions (i.e. synchronization), we also need deltas that can be *reversed* and *composed* without materializing the involved intermediate RDF/S graph versions. In response to the above requirements, the main contributions of our work are:

(a) We analyze five differential functions returning sets of change operations, namely, *explicit* (Δ_e), *closure* (Δ_c), *dense* (Δ_d), *dense & closure* (Δ_{dc}), and *explicit & dense* (Δ_{ed}) .

(b) We consider two change operations semantics: one *plain* set-theoretic ($\mathcal{U}_p$), and another that involves *inference* and *redundancy elimination* ($\mathcal{U}_{ir}$).

(c) We study which combinations of change operation semantics and differential functions can be employed to *correctly* transform a source to a target RDF/S KB. In addition, we are interested in other useful properties such as semantic identity, non redundancy, reversibility and composition of RDF/S deltas.

(d) We experimentally investigate the impact of the inferential potential of RDF KBs to delta computation times and sizes for real (from e-Science and e-Culture applications) and synthetic data sets.

This paper is an extended version of the work originally presented in [Zeginis et al. 2007]. The rest of this paper is organized as follows. Section 2 introduces briefly RDF/S knowledge bases. Section 3 defines formally five RDF/S differential functions and discusses their size and time complexity. Section 4 elaborates on the change operations and their semantics, while Section 5 studies the properties of the introduced RDF/S deltas. Subsequently, Section 6 reports comparative experimental results over various data sets. Finally, Section 7 concludes the paper and identifies issues for future research. All proofs of propositions and theorems are given in the Appendix.

2. PRELIMINARIES: RDF/S KNOWLEDGE BASES

In general, an RDF/S Knowledge Base (KB) is defined by a set of triples of the form (subject, predicate, object). Let $\mathcal{T}$ be the set of all possible triples that can be constructed from a countably infinite set of URIs (for resources, classes and properties) as well as literals (e.g. strings, integer, real numbers) [Gutierrez et al. 2004]. Then, a KB can be seen as a finite subset K of $\mathcal{T}$, i.e. $K \subseteq \mathcal{T}$. Apart from the explicitly specified triples of a K, other triples can be inferred based on the RDF/S semantics [Hayes 2004]. For this reason, we introduce the notion of closure and reduction of RDF/S KBs.

Let K be a knowledge base, the *closure* of a K, denoted by $C(K)$, is the set of all triples that either are explicitly asserted or can be inferred from K by taking into account class or property assertions made by the associated RDFS schemas. Thus, we can consider that $C(K)$ is defined (and computed) by taking the transitive closures of RDFS binary relations such as `subClassof` and `type`. Essentially, the following rules are used: (i) if *a subClassOf b*, and *b subClassOf c* in K, then *a subClassOf c* in $C(K)$, (ii) if p *subPropertyOf* p', and p' *subPropertyOf* p'' in K,

then p *subPropertyOf* p'' in $C(K)$, and (iii) if i *type* a, and a *subClassOf* b in K, then i *type* b in $C(K)$.

We do not deal with blank nodes in this paper. It should be stressed that our work is orthogonal to the consequence operator of logic theories actually employed to define the closure operator C[7]. Specifically, if P denotes the powerset of $\mathcal{T}$, then a closure operator can be defined as any function $C : P \rightarrow P$ that satisfies the following properties:

—$K \subseteq C(K)$ for all K, i.e. C is *extensive*
—If $K \subseteq K'$ then $C(K) \subseteq C(K')$, i.e. C is *monotonically increasing*
—$C(C(K)) = C(K)$ for all K, i.e. C is an *idempotent* function

If it holds $C(K) = K$, then we will call K *closed*. The elements of K will be called *explicit* triples, while the elements of $C(K) - K$ will be called *inferred*. We can now define an equivalence relation between two knowledge bases.

DEF. 1. Two knowledge bases K and K' are *equivalent*, denoted by $K \sim K'$, iff $C(K) = C(K')$.

Let K be a knowledge base, the *reduction* of a K, denoted by $R(K)$, is the smallest in size set of triples such that $C(R(K)) = C(K)$. Let Ψ denote the set of all knowledge bases that have a unique reduction.

PROP. 1. If $K \in \Psi$ then $R(K) \subseteq K$

Independently of whether the reduction of a K is unique or not, we can characterize a K as (semantically) *redundancy free*, and we can write $RF(K) = True$ (or just $RF(K)$), if it does not contain explicitly defined triples which can be inferred from other statements in K. Formally, K is redundancy free if there is not any proper subset K' of K (i.e. $K' \subset K$) such that $K \sim K'$.

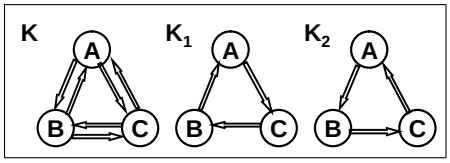


Fig. 2. KB without unique reduction

As stated earlier, the reduction of a K is not always unique. In general, uniqueness of the transitive reduction of a binary relation R is guaranteed only when R is antisymmetric and finite. Unfortunately, this is not the case for RDF/S KBs allowing cycles in the subsumption relations. For example, in Figure 2 we have $K \sim K_1 \sim K_2$, moreover $RF(K_1), RF(K_2)$, but $K_1 \neq K_2$.

3. RDF/S KNOWLEDGE BASES DELTAS

In this section, we formally define the five differential functions of RDF/S KBs introduced in Figure 1, namely, Δ_e, Δ_c, Δ_d, Δ_{dc} and Δ_{ed}.

[7]Unlike our work, belief contraction-revision theories (e.g. [Gärdenfors 1992; Konieczny and Perez 2005; Flouris 2006]) consider KBs as logic theories and focus on what the result of applying a

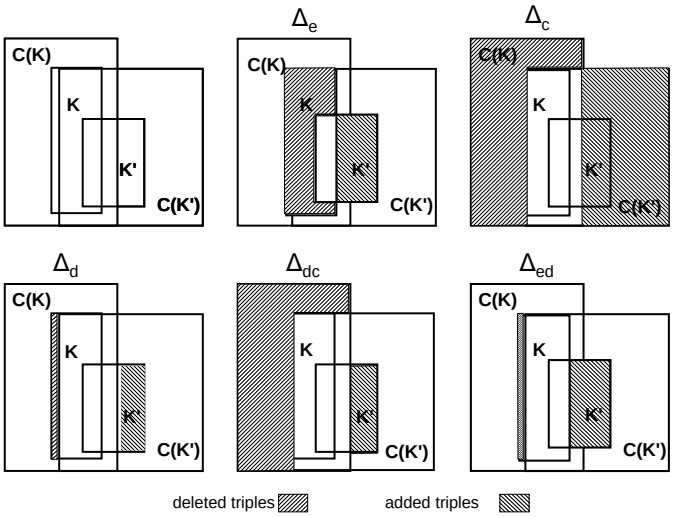


Fig. 3. RDF/S KBs differential functions

$$
\begin{aligned}
\Delta_e(K, K') &= \{Add(t) \mid t \in K' - K\} \cup \\
&\quad \{Del(t) \mid t \in K - K'\} \\
\Delta_c(K, K') &= \{Add(t) \mid t \in C(K') - C(K)\} \cup \\
&\quad \{Del(t) \mid t \in C(K) - C(K')\} \\
\Delta_d(K, K') &= \{Add(t) \mid t \in K' - C(K)\} \cup \\
&\quad \{Del(t) \mid t \in K - C(K')\} \\
\Delta_{dc}(K, K') &= \{Add(t) \mid t \in K' - C(K)\} \cup \\
&\quad \{Del(t) \mid t \in C(K) - C(K')\} \\
\Delta_{ed}(K, K') &= \{Add(t) \mid t \in K' - K\} \cup \\
&\quad \{Del(t) \mid t \in K - C(K')\}
\end{aligned}
$$

Δ_e (where e stands for *e*xplicit) actually returns the set difference over the explicitly asserted triples, while Δ_c (where c stands for *c*losure) returns the set difference by also taking into account the inferred triples. As we mentioned in Section 1, existing approaches (e.g. [Volkel et al. 2005]) are based on Δ_e and Δ_c. However, as we are interested in small sized deltas, we introduce three novel differential functions namely Δ_d (where d comes from *d*ense), Δ_{dc} (dc comes from *d*ense & *c*losure) and Δ_{ed} (ed comes from *e*xplicit & *d*ense). Δ_d contains add operations for those triples which are explicit in K' and do not belong to the closure of K, and delete operations for those triples which are explicit in K and do not belong to the closure of K'. It is not hard to see that Δ_d produces the smallest in size set of change operations. Figure 3 shows the Venn diagrams of the corresponding sets of triples to be added and deleted, in the general case of overlapping K and K'. Unfortunately,

---

contraction/revision operation on a KB should be. In our setting, the target K' is known, so the focus is given on identifying the changes between K and K'.

and as we will see at Section 5, Δ_d can be actually applied to transform K to K' only under certain conditions. For this reason, we additionally consider Δ_{dc} and Δ_{ed} yielding smaller in size deltas than Δ_c and Δ_e respectively. Δ_{dc} resembles to Δ_d regarding additions and to Δ_c regarding deletions, while Δ_{ed} resembles to Δ_e regarding additions and to Δ_d regarding deletions.

3.1 RDF/S Delta Containment and Size

To keep notations simple we shall also use Δ_x to denote $\Delta_x(K, K')$ for any $x \in \{e, c, d, dc, ed\}$. In general, for any pair of knowledge bases K and K' it holds:

$$\Delta_d \subseteq \Delta_{ed} \subseteq \Delta_e$$
$$\Delta_d \subseteq \Delta_{dc} \subseteq \Delta_c$$

In a nutshell, Δ_d gives always the smallest deltas, Δ_{dc} gives smaller deltas than Δ_c, Δ_{ed} gives smaller deltas than Δ_e, while Δ_{dc} is incomparable to Δ_{ed}. Figure 4(a) illustrates the Hasse diagram of the ordering relation of delta sizes in the general case. Below we compare the sizes in certain conditions.

If $K \subseteq K'$ then the following relationships hold:

$$\Delta_e = \Delta_{ed}$$
$$\Delta_d = \Delta_{dc}$$

This case corresponds to the quite frequent scenario where K is stored in KB (and therefore it might be also redundancy free), and K' is derived from K by adding new triples. Here Δ_d and Δ_{dc} give the same deltas, as it also happens for Δ_e and Δ_{ed}. Figure 4(b) shows the corresponding Hasse diagram.

If $C(K) \subseteq C(K')$ then the following relationship holds:

$$\Delta_d = \Delta_{dc}$$

This case corresponds to the scenario where K' is backwards compatible to K (note that it is not necessary to hold $K \subseteq K'$). Here Δ_d and Δ_{dc} give the same deltas.

If $K \supseteq K'$ then the following relationships hold:

$$\Delta_d = \Delta_{ed}$$
$$\Delta_c = \Delta_{dc}$$

This case corresponds to the scenario where K' is generated only by deleting triples from K. Here Δ_d and Δ_{ed} give the same deltas. Δ_c and Δ_{dc} are equal too. Figure 4(c) shows the corresponding Hasse diagram.

If $K = C(K)$ then the following relationships hold:

$$\Delta_d = \Delta_{dc} = \Delta_{ed}$$

This case corresponds to the scenario where either the subsumption hierarchy of K consists of only one level (i.e. no triples can be inferred) or the closure of K is already materialized. Thus, Δ_d, Δ_{dc} and Δ_{ed} give the same deltas. Figure 4(d) shows the corresponding Hasse diagram.

If $K' = C(K')$ then the following relationships hold:

$$\Delta_c \ = \ \Delta_{dc}$$
$$\Delta_e \ = \ \Delta_{ed}$$

This case corresponds to the scenario where either the subsumption hierarchy of K' consists of only one level (i.e. no triples can be inferred) or the closure of K' is already materialized. Thus, Δ_c and Δ_{dc} give the same deltas, as also happens for Δ_e and Δ_{ed}. Figure 4(e) shows the corresponding Hasse diagram.

If $K = C(K')$ then the following relationships hold:

$$\Delta_d = \Delta_{dc} \ = \ \Delta_c = \Delta_{ed} = \emptyset$$

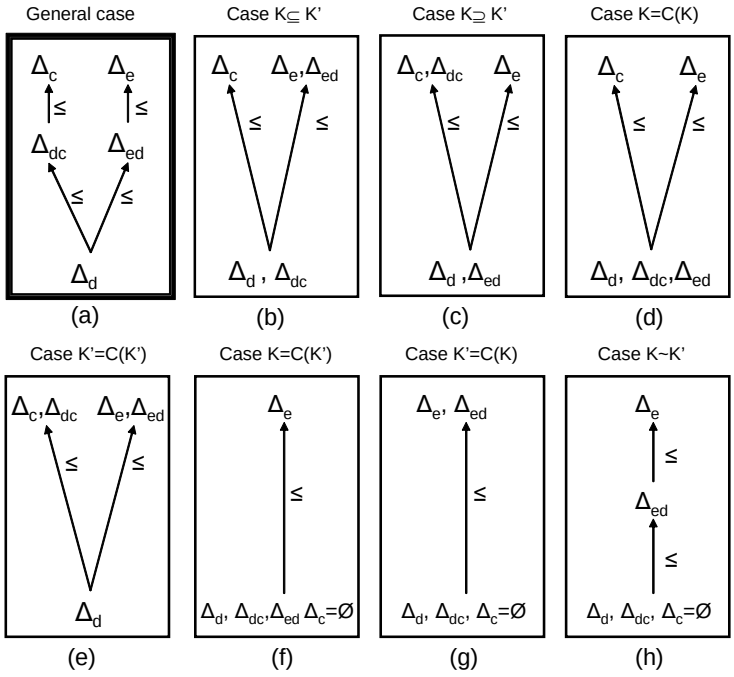


Fig. 4. Ordering Relation of RDF/S delta Sizes

In this case K is semantically equivalent to K'. This case arise when K contains explicitly defined triples that became implicit in K'. Hence, unlike Δ_e, the differential function Δ_d, Δ_{dc}, Δ_c and Δ_{ed} give an empty result. Note that if $K = C(K')$ then the set K is closed since the closure operator C is idempotent (i.e. $C(C(K')) = C(K')$). It follows that $C(K) = K$, that's why Δ_{dc} and Δ_c are empty. Figure 4(f) shows the corresponding Hasse diagram.

If $K' = C(K)$ then the following relationships hold:

$$\Delta_d = \Delta_{dc} \ = \ \Delta_c = \emptyset$$
$$\Delta_e = \Delta_{ed}$$

In this case K' contains explicitly defined triples that were implicit in K. Thus, Δ_d, Δ_{dc} and Δ_c give an empty result, while Δ_e and Δ_{ed} give the same non-empty result. Figure 4(g) shows the corresponding Hasse diagram.

If $K \sim K'$ then the following relationships hold:

$$\Delta_d = \Delta_{dc} = \Delta_c = \emptyset$$
$$\Delta_{ed} \subseteq \Delta_e$$

In this case K and K' are equivalent. Thus, Δ_c, Δ_{dc} and Δ_c give an empty result, while Δ_e and Δ_{ed} give a non-empty result with Δ_{ed} producing a smaller in size result. Figure 4(h) shows the corresponding diagram.

3.2 RDF/S Deltas Complexity

Consider two knowledge bases K_1 and K_2. Below we shall use the following notations:

$N_1 = |K_1|$ $\quad N_2 = |K_2|$ $\quad N = \max(N_1, N_2)$
$C_1 = |C(K_1)|$ $\quad C_2 = |C(K_2)|$ $\quad C = \max(C_1, C_2)$

Regarding the size of closures, the transitive closure of a graph with n nodes can have at most n^2 edges (recall that $C(K)$ completes K with triples induced due to the transitivity of the `subclassOf` relation) and its cost is in $O(n^3)$ [Warshall 1962; Ullman and Yannakakis 1991; Nuutila 1995]. Below we study the complexity of computing $\Delta_x(K_1, K_2)$ for each $x \in \{e, c, d, dc, ed\}$, in the following settings:

(i) $C(K_1)$ and $C(K_2)$ are precomputed and stored along with K_1, K_2 in corresponding hashtables. Thus, we can decide in $O(1)$ whether a triple t belongs to any of these collections.

(ii) Only K_1 and K_2 are stored in hashtables, while $C(K_1)$ and $C(K_2)$ are computed on the fly using an interval-based labeling scheme [Christophides et al. 2003] of transitive RDFS relations (i.e., subclassof and subpropertyof).

Whenever a differential function has an expression of the form $\forall\ t \in X - C(K)$, where X is a set of triples, we do not necessarily have to compute $C(K)$. For instance, suppose that we have stored only K and we have a method with which we can decide whether $t \in C(K)$ without computing the entire $C(K)$. In this case to compute the expression $\forall\ t \in X - C(K)$ for each element of X we can exploit this method instead of computing the entire $C(K)$. Table I summarizes for each setting the time complexity of Δ_x. In a nutshell, Δ_c is the most expensive to compute delta, followed by Δ_{dc}, then by Δ_d and Δ_{ed} and finally by Δ_e.

$K_1 \rightarrow K_2$	(i) all stored	(ii) K's & labels
Δ_e	$O(N)$	$O(N)$
Δ_c	$O(C)$	$O(C)$
Δ_d	$O(N)$	$O(N)$
Δ_{dc}	$O(\max(C_1, N_2))$	$O(\max(C_1, N_2))$
Δ_{ed}	$O(N)$	$O(N)$

Table I. Time Complexity of RDF/S Deltas

There are essentially two major cost factors: (a) whether a Δ_x requires computing the closure of the operand KBs, and (b) whether it has to scan the involved KB

closures. Only Δ_c requires to scan both KB closures: Δ_d and Δ_{ed} do not imply any scan while Δ_{dc} scans only $C(K_1)$ (as depicted in column i of Table I). This is the reason why even when labels are used (see column ii of Table I), Δ_c and Δ_{dc} are more expensive to compute. The interval-based labels (that are used in setting ii) essentially encode the transitive (class/property) subsumption relationships of the two KBs. The label of a class is composed by two numbers denoted by $start(c)$ and $end(c)$. A class c is subsumed by a class c' iff $start(c') < start(c)$ and $end(c) < end(c')$. Thus checking whether a class/property c is direct or indirect subclass/subproperty of a class/property c') can be performed in $O(1)$ just by comparing the involved labels (no transitive closure computation is required). The same is true for instance checking (i.e. to check whether a resource r is direct or indirect instance of a class c). However, if the subsumption hierarchy is a directed acyclic graph (DAG) rather than a tree, more than one labels need to be generated and compared[8]. Summarizing, we can check whether $t \in C(K_1)$ (or $t \in C(K_2)$) with a cost analogous to the average number of labels that have to be compared. Comparative experimental results for all differential functions are given in Section 6.

4. RDF/S KNOWLEDGE BASE CHANGE OPERATIONS SEMANTICS

In this work, we focus on two *basic* change operations allowing to transform one KB to another, namely triple *addition* $Add(t)$ and *deletion* $Del(t)$ where $t \in T$. The five differential functions presented in Section 3, yield essentially *sets of atomic change operations*. For a $\Delta_x(K, K')$ where $x \in \{e, c, d, dc, ed\}$, we shall hereafter use Δ_x^+ to denote the corresponding set of triple additions (i.e. *incremental* changes), and Δ_x^- to denote the set of triple deletions (i.e. *decremental* changes). Obviously, Δ_x contains only sets of consistent change operations reflecting the net effect of successive modifications over the same (explicit or inferred) triple of two KB versions, i.e. $\Delta_x(K, K')$ is *consistent* if it does not contain both $Add(t)$ and $Del(t)$ operations for a given $t \in T$.

By defining RDF/S deltas as sets of atomic change operations, we avoid to specify an execution order as in an edit-script (i.e. a sequence of triple additions or deletions). This design choice amends to a simpler computation of RDF/S deltas (see Section 3.2) while provides the opportunity of applying alternative semantics of changes when transforming a source to a target KB (i.e. with or without side-effects on the KB closure). In the sequel, we will formally introduce the *semantics* of basic $Add(t)$ and $Del(t)$ operations (i.e. the exact pre and post-conditions of each operation) while in Section 5.6 we will investigate how a set of change operations can be transformed to an edit-script amendable to a sequential execution.

4.1 Semantics of Change Operations

Table II defines two alternative semantics for triple additions and deletions, namely, $\mathcal{U}_p$ (p comes from $p$lain), and $\mathcal{U}_{ir}$ (ir comes from $i$nference & $r$eduction).

—Under $\mathcal{U}_p$-semantics, change operations capture essentially plain set theoretic additions and deletions of triples. This implies that only the explicitly defined

[8]Several labels also need to be compared when resources are classified under several unrelated through subsumption classes.

triples are taken into account while inferred ones are ignored (as in a standard database context).

—Under $\mathcal{U}_{ir}$-semantics, change operations incur also interesting side-effects such as redundancy elimination and knowledge preservation. This implies that the updated KB will not contain any explicit triple which can be inferred, while preserves as much of the knowledge expressed in K as possible (reminiscent of the postulates of the AGM theory [Alchourrón et al. 1985] regarding contraction, and compliant with the semantics of the RUL update language [Magiridou et al. 05]).

$\mathcal{U}_p$-semantics is straightforward. We will illustrate in the sequel $\mathcal{U}_{ir}$-semantics using the example of Figure 1. If we apply on K the set Δ_{dc} under $\mathcal{U}_{ir}$-semantics, then we will indeed get K'. The insertion of $(Student\ subClassOf\ \ Person)$ makes the triple $(TA\ subClassOf Person)$ redundant, so the execution of $Add(Student\ subClassOf\ Person)$ will remove $(TA\ subClassOf\ \ Person)$ from the KB. Analogously, the insertion of $(TA\ subClassOf\ \ Graduate)$ makes the triple $(TA\ \ subClassOf\ \ Student)$ redundant, while the deletion of the triple $(Jim\ \ type\ \ TA)$ will add the triples $(Jim\ \ type\ \ Person)$ and $(Jim\ \ type\ \ Student)$. Finally, the deletion of the triple $(Jim\ \ type\ \ Student)$ will not have any side effects.

Change Operation Semantics $\mathcal{U}_p$			
Operation		Pre-condition	Post-condition
$Add(t)$	1	$t \in K$	$K' = K$
	2	$t \in C(K) - K$	$K' = K \cup \{t\}$
	3	$t \notin C(K)$	$K' = K \cup \{t\}$
$Del(t)$	4	$t \in K$	$K' = K - \{t\}$
	5	$t \in C(K) - K$	$K' = K$
	6	$t \notin C(K)$	$K' = K$
Change Operation Semantics $\mathcal{U}_{ir}$			
$Add(t)$	7	$t \in K$	$K' = K$
	8	$t \in C(K) - K$	$K' = K$
	9	$t \notin C(K)$	$K' = R(K \cup \{t\})$
$Del(t)$	10	$t \in K$	$K' = R(C(K) - \{t\})$
	11	$t \in C(K) - K$	$K' = K$
	12	$t \notin C(K)$	$K' = K$

Table II. The Change Operations Semantics $\mathcal{U}_p$ and $\mathcal{U}_{ir}$

Returning to Table II, for every incremental ($Add(t)$) or decremental ($Del(t)$) operation three different, and mutually exclusive, pre-conditions are examined, namely $t \in K$, $t \in C(K) - K$ and $t \notin C(K)$. The post-conditions of each case are specified. K (resp. K') denotes the knowledge base before (resp. after) the execution of a change operation. Notice that post-conditions define exactly what K' will be[9], unless the reduction is not unique.

In particular, let t be a triple whose addition is requested. If $t \in K$, then under both $\mathcal{U}_p$ and $\mathcal{U}_{ir}$ semantics no change will be made i.e. $K' = K$ (recall that K is a

[9]One could consider the rows of Table II as ECA (Event-Condition-Action) rules where the Events correspond to column "Operation", the Conditions correspond to column "Pre-Condition" and the Actions correspond to column "Post-condition".

set of triples). If $t \in C(K) - K$, then under $\mathcal{U}_p$-semantics, K' will indeed contain t, however under $\mathcal{U}_{ir}$-semantics we will have $K' = K$ because every triple that exists at $C(K) - K$ can be inferred (and $\mathcal{U}_{ir}$ aims at redundancy-free KBs). Finally, when requesting the addition of a triple $t \notin C(K)$ under $\mathcal{U}_p$, K' will contain t. Under $\mathcal{U}_{ir}$, K' will contain only the triples remaining after the elimination of redundant ones (i.e. those that can be inferred) once t is added to K.

Let's now consider the deletion of a triple t. If t belongs to K, then K' will not contain t under $\mathcal{U}_p$-semantics. Under $\mathcal{U}_{ir}$, K' will contain the triples that remain after deleting t from $C(K)$ and eliminating the redundant ones (note that $C(K)$ is used in order to preserve as much knowledge as possible). Now if $t \in C(K) - K$, then this deletion request has no effect under both semantics. This means that under both semantics, *only explicitly defined triples can be deleted*. This relieves us from having to decide which of the (possibly several) policies need to be adopted for reaching a K' whose closure does not contain t. For example, if (*Crete subClassOf Europe*) is a triple inferred from the triples (*Crete subClassOf Greece*) and (*Greece subClassOf Europe*), then to delete (*Crete subClassOf Europe*) from the closure requires either deleting (*Crete subClassOf Greece*), or (*Greece subClassOf Europe*), or both. In general, we would have to find the minimal set(s) of triples that have to be deleted so that t is no longer an inferred triple. This problem is similar in spirit with the task of schema evolution (for more see [Konstantinidis et al. 2008]). Finally, if $t \notin C(K)$, then this deletion request has no effect as t is already out of K.

$\mathcal{U}_{ir}$ is computationally more expensive than $\mathcal{U}_p$ as it requires computing the closure and the reduction of a KB. As it was shown in [Aho et al. 1972], the algorithms for transitive reduction have the same time complexity as algorithms for transitive closure. Algorithms for keeping a (graph-based) knowledge base in a redundancy-free state include [Poutre' and van Leeuwen 1987], while a similar in spirit approach for RDF/S has already been implemented for the RUL language [Magiridou et al. 05].

We can now define when a KB K satisfies a basic change operation under $\mathcal{U}_p$ and $\mathcal{U}_{ir}$-semantics. If a KB satisfies an operation u, then this means that the re-application of u on KB will be void.

DEF. 2. We will say that K *satisfies* under $\mathcal{U}_p$-semantics:
(a) an operation $Add(t)$, iff $t \in K$,
(b) an operation $Del(t)$, iff $t \notin K$

DEF. 3. We will say that K *satisfies* under $\mathcal{U}_{ir}$-semantics:
(a) an operation $Add(t)$, iff $t \in C(K)$,
(b) an operation $Del(t)$, iff $t \notin C(K)$

Notice that the above definitions stem directly from the pre and post conditions of the operations depicted in Table II. $\mathcal{U}_p$-satisfaction could also be called *plain satisfaction*, while $\mathcal{U}_{ir}$-satisfaction could be called *semantic satisfaction*.

In general, we can say that a K satisfies an RDF/S delta Δ under $\mathcal{U}_y$, where $y \in \{p, ir\}$, iff K satisfies *every* addition of Δ^+ and *every* deletion of Δ^-. Recall at this point that the RDF/S deltas considered in our work consist only of consistent change operations.

If $\mathcal{U}$ is a symbol that denotes the semantics of a particular change operation (i.e. $\mathcal{U}_p$,$\mathcal{U}_{ir}$), then we will use $\Delta^{\mathcal{U}}$ to denote the result of applying Δ to K under $\mathcal{U}$ semantics. Note that the result of applying a change operation is unique under $\mathcal{U}_p$-semantics. This is true also for $\mathcal{U}_{ir}$ if we are in Ψ (KBs with unique reduction).

DEF. 4. A delta $\Delta(K, K')$ can be applied on K as follows:
(a)Under $\mathcal{U}_p$, $\Delta^{\mathcal{U}_p}(K, K') = (K - \Delta^-) \cup \Delta^+$
(b)Under $\mathcal{U}_{ir}$, $\Delta^{\mathcal{U}_{ir}}(K, K') = R((C(K) - \Delta^-) \cup \Delta^+)$

PROP. 2. If $\{K, K'\} \subseteq \Psi$ then for every $x \in \{e, ed\}$ and $y \in \{\Delta_d, \Delta_{dc}, \Delta_c\}$, $\Delta_x^{\mathcal{U}_p}(K, K')$ and $\Delta_y^{\mathcal{U}_{ir}}(K, K')$ satisfy all operations of Δ_x under $\mathcal{U}_p$ and Δ_y under $\mathcal{U}_{ir}$ respectively.

However, as we will see in the next section, the target KB is not always a correct transformation of the source KB.

5. FORMAL PROPERTIES OF RDF/S DELTAS

In this section, we investigate which of the five RDF/S differential functions (introduced in Section 3) actually produce *correct* deltas and under what semantics of change operations (presented in Section 4). In addition, we are interested in other useful properties such as *semantic identity*, *non-redundancy*, *reversibility* and *composability* of RDF/S deltas. Finally, we elaborate on the execution semantics of RDF/S deltas as SW update programs (i.e. sequences of change operations).

5.1 Properties of RDF/S deltas

Let Δ_x be a differential function, and let $\mathcal{U}_y$ be a change operation semantics. Obviously, a pair (Δ_x,$\mathcal{U}_y$) can be used for versioning or synchronizing RDF/S KBs only if it is correct.

DEF. 5. A pair (Δ_x,$\mathcal{U}_y$) is *correct* if for any pair of knowledge bases K and K', it holds $\Delta_x^{\mathcal{U}_y}(K, K') \sim K'$.

Apart from correctness, RDF/S deltas may also satisfy the following properties.

Semantic Identity (P1): It is desirable to have a Δ_x that reports an empty result if its operands are semantically equivalent: If $K \sim K'$ then $\Delta_x(K, K') = \emptyset$.

Non-Redundancy (P2): It is desirable that the execution of a Δ_x will result in a KB that is always redundancy free (independently of whether K and K' are redundancy free or not). This is denoted by $RF(\Delta_x^{\mathcal{U}_y}(K, K'))$.

Weaker Non-Redundancy (P2.1): If $RF(K')$ then $RF(\Delta_x^{\mathcal{U}_y}(K, K'))$. Notice that P2.1 is weaker than P2: if P2 holds then P2.1 holds too.

Reversibility (P3): The inverse of an atomic change operation is defined as: $Inv(Add(t)) = Del(t)$ and $Inv(Del(t)) = Add(t)$. A set of change operations U can be reversed as follows: $Inv(U) = \cup\{\ Inv(u) \mid u \in U\}$. A Δ_x is reversible if: $Inv(\Delta_x(K, K')) = \Delta_x(K', K)$.

The reversibility of a delta is a useful property for moving forward and backward across versions (recall the discussion on forward/backward delta at Section 1).

Composition (P4): Consider two deltas $\Delta_1 = \Delta_1^+ \cup \Delta_1^-$ and $\Delta_2 = \Delta_2^+ \cup \Delta_2^-$. Their composition, denoted by $\Delta_1 \circ \Delta_2$, is a delta $\Delta = \Delta^+ \cup \Delta^-$ where:

- $\Delta^+ = (\Delta_1^+ \cup \Delta_2^+) - (\Delta_1^- \cup \Delta_2^-)$
- $\Delta^- = (\Delta_1^- \cup \Delta_2^-) - (\Delta_1^+ \cup \Delta_2^+)$

A differential function Δ_x satisfies composability if:

$$\Delta_x(K_1, K_2) \circ ... \circ \Delta_x(K_{n-1}, K_n) = \Delta_x(K_1, K_n)$$

We call this composition because it allows composing a number of consecutive deltas and then executing the operations of the composed delta, instead of having to execute each particular delta. The former can be more efficient because the constituent deltas can contain compensating operations which are eliminated in the composed delta, i.e. $|\Delta_x(K_1, K_n)| \leq \sum_{i=1}^{n-1} |\Delta_x(K_i, K_{i+1})|$.

5.2 Correctness of $(\Delta_x, \mathcal{U}_y)$-pairs

For identifying the pairs that are correct, Table III depicts for each of our four examples the result of applying Δ_d, Δ_c, Δ_e, Δ_{dc} and Δ_{ed} in both directions: $K \to K'$ and $K' \to K$. Then, for each of the two update semantics:

—column $\mathcal{U}_p$: is marked with Y (from Yes) when $\Delta_x(K, K')^{\mathcal{U}_p}(K) \sim K'$, i.e. if the transformation is *correct* using $\mathcal{U}_p$. Otherwise, the column is marked with N (from No). If the respective pair is correct and the result is *redundancy free* it is also mentioned in the table with the label *RF*.

—column $\mathcal{U}_{ir}$: is marked with Y (from Yes) when $\Delta_x(K, K')^{\mathcal{U}_{ir}}(K) \sim K'$, i.e if the transformation is *correct* using $\mathcal{U}_{ir}$. Otherwise the column is marked with N. Since by definition in this case the resulting KB is always in a redundancy free state, we do not mark it as *RF*.

Those pairs that are marked with "N" w.r.t. correctness, essentially constitute a proof by counterexample of their incorrectness.

THEOREM 1. For any pair of knowledge bases K and K' it holds:

$$\Delta_e^{\mathcal{U}_p}(K, K') \sim \Delta_{ed}^{\mathcal{U}_p}(K, K') \sim$$
$$\Delta_c^{\mathcal{U}_{ir}}(K, K') \sim \Delta_{dc}^{\mathcal{U}_{ir}}(K, K') \sim K'$$

THEOREM 2. $\Delta_d^{\mathcal{U}_{ir}}(K, K') \sim K'$ iff $C(K) - K \subseteq C(K')$.

THEOREM 3. $\Delta_c^{\mathcal{U}_p}(K, K') \sim K'$ if K is closed.

The proofs of the above theorems are given in the Appendix. An interesting remark regarding Theorem 2 is that if $C(K) \subseteq C(K')$, then theorem's condition holds. This implies that we could use the pair $(\Delta_d, \mathcal{U}_{ir})$ in cases we know that $C(K) \subseteq C(K')$. For example if K is an ontology O and K' is an ontology O' that specializes O, then we are sure that $C(K) \subseteq C(K')$. In such cases we can use Δ_d (or alternatively Δ_{dc}) which gives the smallest in size delta (Δ_{dc} returns the same delta). Another common case where the condition of Theorem 2 holds is when K

is closed (i.e. $K = C(K)$). This implies that the pair $(\Delta_d, \mathcal{U}_{ir})$ can be used at the cases that the closure of K has been materialized.

As we have seen, although Δ_d produces the smallest in size delta, it is not always correct (assuming both $\mathcal{U}_{ir}$ and $\mathcal{U}_p$ semantics). It is worth investigating the reason of incorrectness of Δ_d. An example demonstrating the problem is shown at Figure 5: (a) demonstrates why the application of $\Delta_d(K, K')$ to K assuming $\mathcal{U}_{ir}$ semantics can lead to wrong results, while (b) demonstrates why the application of $\Delta_d(K, K')$ to K assuming $\mathcal{U}_p$ semantics can lead to wrong results.

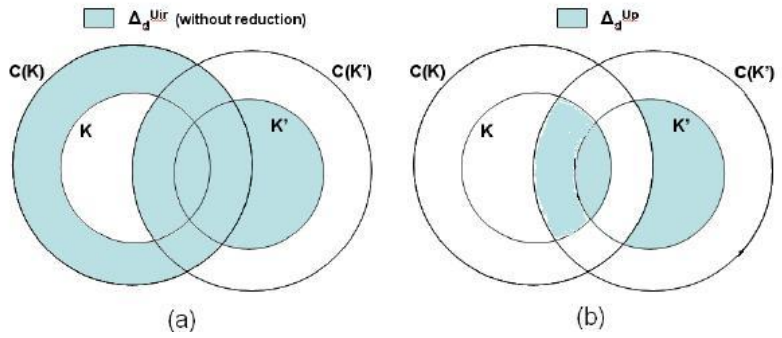


Fig. 5. (a) $\Delta_d^{U_{ir}}$ without reduction (b)$\Delta_d^{U_P}$

5.3 Semantic Identity and RF of $(\Delta_x, \mathcal{U}_y)$-pairs

PROP. 3. If $K \sim K'$ then $\Delta_d(K, K') = \Delta_c(K, K') = \Delta_{dc}(K, K') = \emptyset$.

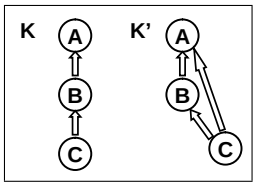
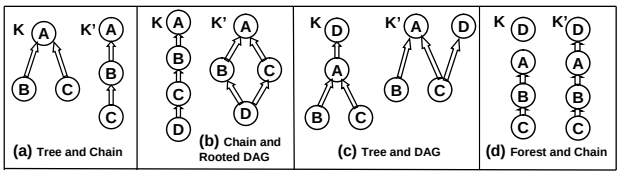


Fig. 6. Two Equivalent KBs

Note that Δ_e is not included in Prop. 3 because even if $K \sim K'$, it may be $K = K'$, $K \subset K'$, $K' \subset K$, or $K \not\subseteq K'$ and $K' \not\subseteq K$. In the example of Figure 6 we get $\Delta_e(K, K') = \{Add(C\ subClassOf\ A)\}$ although $K \sim K'$.

PROP. 4. If $K \sim K'$, $\{K, K'\} \subseteq \Psi$ and $RF(K), RF(K')$ then $\Delta_e(K, K') = \emptyset$

PROP. 5. If $K \sim K'$, and (a) $K' \subseteq K$ or (b) $\{K, K'\} \subseteq \Psi$ and $RF(K), RF(K')$ then $\Delta_{ed}(K, K') = \emptyset$



(a) Tree and Chain							
delta	$K \to K'$	$\mathcal{U}_p$	$\mathcal{U}_{ir}$	delta	$K' \to K$	$\mathcal{U}_p$	$\mathcal{U}_{ir}$
Δ_e	$\{Add(C\ subClassOf\ B),$ $Del(C\ subClassOf\ A)\}$	Y,RF	Y	Δ_e	$\{Add(C\ subClassOf\ A),$ $Del(C\ subClassOf\ B)\}$	Y,RF	Y
Δ_c	$\{Add(C\ subClassOf\ B)\}$	Y	Y	Δ_c	$\{Del(C\ subClassOf\ B)\}$	**N**	Y
Δ_d	$\{Add(C\ subClassOf\ B)\}$	Y	Y	Δ_d	$\{Del(C\ subClassOf\ B)\}$	**N**	Y
Δ_{dc}	$\{Add(C\ subClassOf\ B)\}$	Y	Y	Δ_{dc}	$\{Del(C\ subClassOf\ B)\}$	**N**	Y
Δ_{ed}	$\{Add(C\ subClassOf\ B)\}$	Y	Y	Δ_{ed}	$\{Add(C\ subClassOf\ A),$ $Del(C\ subClassOf\ B)\}$	Y, RF	Y
(b) Chain and Rooted DAG							
Δ_e	$\{Add(C\ subClassOf\ A),$ $Add(D\ subClassOf\ B),$ $Del(C\ subClassOf\ B)\}$	Y, RF	Y	Δ_e	$\{Add(C\ subClassOf\ B),$ $Del(C\ subClassOf\ A),$ $Del(D\ subClassOf\ B)\}$	Y, RF	Y
Δ_c	$\{Del(C\ subClassOf\ B)\}$	**N**	Y	Δ_c	$\{Add(C\ subClassOf\ B)\}$	Y	Y
Δ_d	$\{Del(C\ subClassOf\ B)\}$	**N**	Y	Δ_d	$\{Add(C\ subClassOf\ B)\}$	Y	Y
Δ_{dc}	$\{Del(C\ subClassOf\ B)\}$	**N**	Y	Δ_{dc}	$\{Add(C\ subClassOf\ B)\}$	Y	Y
Δ_{ed}	$\{Add(C\ subClassOf\ A),$ $Add(D\ subClassOf\ B),$ $Del(C\ subClassOf\ B)\}$	Y, RF	Y	Δ_{ed}	$\{Add(C\ subClassOf\ B)\}$	Y	Y
(c) Tree and DAG							
Δ_e	$\{Add(C\ subClassOf\ D),$ $Del(A\ subClassOf\ D)\}$	Y, RF	**N**	Δ_e	$\{Add(A\ subClassOf\ D),$ $Del(C\ subClassOf\ D)\}$	Y, RF	Y
Δ_c	$\{Del(A\ subClassOf\ D),$ $Del(B\ subClassOf\ D)\}$	**N**	Y	Δ_c	$\{Add(A\ subClassOf\ D),$ $Add(B\ subClassOf\ D)\}$	Y	Y
Δ_d	$\{Del(A\ subClassOf\ D)\}$	**N**	**N**	Δ_d	$\{Add(A\ subClassOf\ D)\}$	Y	Y
Δ_{dc}	$\{Del(A\ subClassOf\ D),$ $Del(B\ subClassOf\ D)\}$	**N**	Y	Δ_{dc}	$\{Add(A\ subClassOf\ D)\}$	Y	Y
Δ_{ed} s	$\{Add(C\ subClassOf\ D),$ $Del(A\ subClassOf\ D)\}$	Y, RF	**N**	Δ_{ed}	$\{Add(A\ subClassOf\ D)\}$	Y	Y
(d) Forest and Chain							
Δ_e	$\{Add(A\ subClassOf\ D)\}$	Y, RF	Y	Δ_e	$\{Del(A\ subClassOf\ D)\}$	Y, RF	**N**
Δ_c	$\{Add(A\ subClassOf\ D),$ $Add(B\ subClassOf\ D),$ $Add(C\ subClassOf\ D)\}$	Y	Y	Δ_c	$\{Del(A\ subClassOf\ D),$ $Del(B\ subClassOf\ D),$ $Del(C\ subClassOf\ D)\}$	Y, RF	Y
Δ_d	$\{Add(A\ subClassOf\ D)\}$	Y, RF	Y	Δ_d	$\{Del(A\ subClassOf\ D)\}$	Y, RF	**N**
Δ_{dc}	$\{Add(A\ subClassOf\ D)\}$	Y, RF	Y	Δ_{dc}	$\{Del(A\ subClassOf\ D),$ $Del(B\ subClassOf\ D),$ $Del(C\ subClassOf\ D)\}$	Y, RF	Y
Δ_{ed}	$\{Add(A\ subClassOf\ D)\}$	Y, RF	Y	Δ_{ed}	$\{Del(A\ subClassOf\ D)\}$	Y, RF	**N**

Table III. Examples of Deltas and of their Properties in Various KBs

5.4 Reversibility of Δ_x

It is not hard to see that this property is satisfied only by those differential functions defining symmetric set differences on operand KBs (and their closures). This is the case of Δ_e, Δ_c and Δ_d. It should be stressed that the reverse of a delta $Inv(\Delta_x(K, K'))$ is correct only when the $\Delta_x(K', K)$ is correct (see Section 5.2)

PROP. 6. For every pair of knowledge bases K, K' it holds:

$$Inv(\Delta_e(K, K')) = \Delta_e(K', K)$$
$$Inv(\Delta_c(K, K')) = \Delta_c(K', K)$$
$$Inv(\Delta_d(K, K')) = \Delta_d(K', K)$$

Δ_{dc} and Δ_{ed} do not satisfy (P3). For example in the case (a) of Table III $\Delta_{ed}(K, K') = \{Add(C\ subClassOf\ B)\}$, so $Inv(\Delta_{ed}(K, K')) = \{Del(C\ \ subClassOf\ \ B)\}$, while $\Delta_{ed}(K', K) = \{Add(C\ subClassOf\ A), Del(C\ subClassOf\ B)\}$. This means that:

$$Inv(\Delta_{ed}(K, K')) \neq \Delta_{ed}(K', K)$$

In the case (d) of Table III $\Delta_{dc}(K, K') = \{Add(A\ subClassOf\ D)\}$, so $Inv(\Delta_{dc}(K, K')) = \{Del(A\ subClassOf\ D)\}$, while $\Delta_{dc}(K', K) = \{Del(A\ \ subClassOf\ \ D), Del(B\ subClassOf\ D), Del(C\ subClassOf\ D)\}$. Thus:

$$Inv(\Delta_{dc}(K, K')) \neq \Delta_{dc}(K', K)$$

Generally Δ_{dc} and Δ_{ed} do not satisfy reversibility, unless K' is closed (i.e. $K' = C(K')$). This is due to the fact that when $K' = C(K')$, Δ_{dc} yields the same result as Δ_c and Δ_{ed} as Δ_e (see Figure 4.e).

5.4.1 *Reversibility and delta size.* In order to speed-up access to forward and backward RDF/S KB versions we have to compute and store non-reversible deltas (such as Δ_{ed} and Δ_{dc}) in both directions. However, since the only reversible deltas are Δ_e and Δ_c whose size is bigger than Δ_{ed} and Δ_{dc}, it will be interesting to examine whether it is more beneficial to store Δ_e and Δ_c assuming only one direction, or Δ_{ed} and Δ_{dc} in both directions. The following properties show that the former option is better than the latter. For any pair of knowledge bases K and K' it holds:

$$|\Delta_e(K, K')| \leq |\Delta_{ed}(K, K')| + |\Delta_{ed}(K', K)|$$
$$|\Delta_c(K, K')| \leq |\Delta_{dc}(K, K')| + |\Delta_{dc}(K', K)|$$

5.5 Composition of Δ_x

The RDF/S deltas that can be always composed are Δ_e and Δ_c. Note that the composition of a differential function $\Delta_x(K_1, K_2) \circ ... \circ \Delta_x(K_{n-1}, K_n) = \Delta_x(K_1, K_n)$ is correct only when the $\Delta_x(K_1, K_n)$ is correct (see Section 5.2)

PROP. 7. For any knowledge bases $K_1, K_2, K_3, ..., K_n$ it holds:

$$\Delta_e(K_1, K_2) \circ ... \circ \Delta_e(K_{n-1}, K_n) = \Delta_e(K_1, K_n)$$
$$\Delta_c(K_1, K_2) \circ ... \circ \Delta_c(K_{n-1}, K_n) = \Delta_c(K_1, K_n)$$

It is not hard to see that this property is not satisfied by Δ_d, Δ_{dc} and Δ_{ed}. For example, for the KBs of the Figure 7 (a) for Δ_d we have:

$$\Delta_d(K_1, K_2) = \{Add(B\ subClassOf\ A)\}$$
$$\Delta_d(K_2, K_3) = \{Del(B\ subClassOf\ A)\}$$
$$\Delta_d(K_1, K_3) = \{Add(C\ subClassOf\ A)\}$$

If we compose the first two deltas we get the empty result, i.e. $\Delta_d(K_1, K_2) \circ \Delta_d(K_2, K_3) = \emptyset$, although $\Delta_d(K_1, K_3) \neq \emptyset$.

The same result holds also for Δ_{dc}.

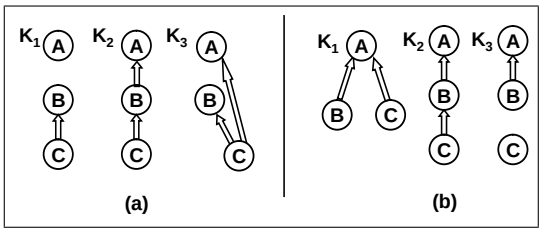


Fig. 7. Composition of Δ_x

In Figure 7 (b) for Δ_{ed} we have:

$$\begin{aligned}\Delta_{ed}(K_1, K_2) &= \{Add(C\ subClassOf\ B)\}\\ \Delta_{ed}(K_2, K_3) &= \{Del(C\ subClassOf\ B)\}\\ \Delta_{ed}(K_1, K_3) &= \{Del(C\ subClassOf\ A)\}\end{aligned}$$

We can observe that the composition of the two deltas yields an empty set, i.e. $\Delta_{ed}(K_1, K_2) \circ \Delta_{ed}(K_2, K_3) = \emptyset$, while $\Delta_{ed}(K_1, K_3)$ is not empty.

Generally speaking, Δ_{dc} and Δ_{ed} can not be composed unless K' is closed. This is due to the fact that when $K' = C(K')$, Δ_{dc} yields the same result as Δ_c and Δ_{ed} as Δ_e (see Figure 4.e).

However, we have to note that in case of non composable deltas, stream management methods (like those described in [Tzitzikas and Kotzinos 2007]) could be used to reduce the size of a stream of change operations before executing it.

5.6 Streaming Execution of RDF/S deltas

Regarding the execution of deltas, Section 4 (specifically Def. 4) provided a method to apply a delta to a KB. We can call this method "batch" because the execution of a set of operations (and thus of a delta) requires applying the closure and the reduction operation just once. An alternative method would be to execute each operation in Δ^+ and Δ^- individually (according to the pre/post-conditions defined in Table II). A question that rises is whether these two alternative execution modes yield equivalent KBs. It is not hard to see that this is true only for $\mathcal{U}_p$ semantics.

As expected, the order of execution of change operations affect the resulting KB under $\mathcal{U}_{ir}$ semantics. In particular the resulting KB may not satisfy all change operations returned by a differential function (see Def. 3). For instance, for the KBs of Table III (d) we get $\Delta_{dc}(K', K) = \{Del(A\ subClassOf\ D), Del(B\ subClassOf\ D), Del(C\ subClassOf\ D)\}$. If the operations are executed in the order $\langle Del(A\ subClassOf\ D), Del(B\ subClassOf\ D), Del(C\ subClassOf\ D)\rangle$ under $\mathcal{U}_{ir}$ semantics, then all of them will be satisfied and the result will be equivalent to K. Now consider the following execution order: $\langle Del(B\ subClassOf\ D), Del(C\ subClassOf\ D), Del(A\ subClassOf\ D)\rangle$. In this case the operation $Del(B\ subClassOf\ D)$ does not change the K as it requests the deletion of an inferred triple and according to $\mathcal{U}_{ir}$ semantics an inferred triple cannot be deleted. The same will happen with the operation $Del(C\ subClassOf\ D)$. Finally, the operation $Del(A\ subClassOf\ D)$ will be executed and will cause the addition of the triple $(B\ subClassOf\ D)$. It is obvious that the operation $Del(B\ subClassOf\ D)$ is not

```
Alg.1 Execute(K, M)
(1) UnSat = M
(2) repeat
(3)     Sat = ∅     //the set of all satisfied operations
(4)     for   i=1  to  |UnSat|
(5)           u = UnSat[i]     // gets the next operation
(6)           K_t = u^{U_ir}(K)     // applies the operation u using U_ir
(7)           if K_t satisfies u then // according to U_ir
(8)                 Sat = Sat ∪ {u}
(9)                 K = K_t
(10)          endif
(11)    endfor
(12)    UnSat = UnSat − Sat
(13) until UnSat = ∅
```

satisfied by the resulting KB because it contains the triple (B *subClassOf* D). The same problem occurs when K' contains a redundant triple e.g. (B *subClassOf* D). A similar situation is encountered with Δ_c when K and K' are not redundancy free.

For these reasons, we devise a loop-based algorithm (`Alg.1`) that takes as input a set of change operations M, and executes them (under $\mathcal{U}_{ir}$ semantics) in a way that guarantees that all operations in M are satisfied. From the structure of the algorithm, it is clear that it will not terminate unless all delete operations in M are satisfied. It is proved that if M is derived from one of Δ_c, Δ_{dc}, Δ_d, and we are in Ψ then the algorithm terminates.

PROP. 8. For every set of change operations M produced by Δ_{dc}, Δ_c and Δ_d in Ψ, Alg. 1 terminates.

The crux of the proof (the complete proof is found in the Appendix) is that $UnSat$ keeps decreasing.

If we are not in Ψ, i.e. if circles exist, then Alg.1 may not terminate. For example consider the two KBs shown at Figure 8. In this case we have $\Delta_{dc} = \Delta_c = \{Del(A\ subClassOf\ B), Del(A\ \ subClassOf\ C)\}$. None of these two operations can be satisfied under $\mathcal{U}_{ir}$ semantics. It follows that Alg. 1 will not terminate. However, in batch execution mode both operations will be satisfied.

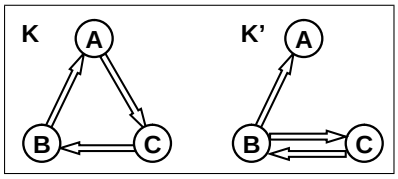


Fig. 8. Case where Alg.1 does not Terminate in Ψ

THEOREM 4. If $\{K, K'\} \subseteq \Psi$ then for every delta produced by $\{\Delta_d, \Delta_{dc}, \Delta_c\}$ and $\{\Delta_{ed}, \Delta_e\}$ there exists at least one sequence of change operations that when executed under $\mathcal{U}_{ir}$ and $\mathcal{U}_p$ respectively the result is equivalent to the execution of the corresponding *set* of change operations. Alg. 1 produces such a sequence.

5.7 Summarizing the Results

The pairs that are always correct are: $(\Delta_e,\mathcal{U}_p)$, $(\Delta_{ed},\mathcal{U}_p)$, $(\Delta_c,\mathcal{U}_{ir})$, $(\Delta_{dc},\mathcal{U}_{ir})$. The pair $(\Delta_c,\mathcal{U}_p)$ is correct if K is closed, while the pair $(\Delta_d,\mathcal{U}_{ir})$ is correct only in the cases specified in Theorem 2. The deltas that always satisfy *semantic identity* are: Δ_c, Δ_{dc} and Δ_d, while Δ_e and Δ_{ed} satisfy semantic identity only in the cases specified in Propositions 4 and 5 respectively. Pairs that rely on $\mathcal{U}_{ir}$ semantics always result in *non-redundant* RDF/S KBs, while the pairs $(\Delta_e,\mathcal{U}_p)$ and $(\Delta_{ed},\mathcal{U}_p)$ satisfy only *weaker non-redundancy*. The pair $(\Delta_c,\mathcal{U}_p)$ does not satisfy neither of the two properties. The deltas that are always *reversible* are Δ_e, Δ_c and Δ_d, while Δ_{dc} and Δ_{ed} are reversible only if K' is closed. Finally, Δ_e and Δ_c can be always *composed*, while Δ_{dc} and Δ_{ed} only in the case where K' is closed. Table IV synopsizes the results. Note that the properties (P1)-(P4) are examined only for the pairs $(\Delta_x,\mathcal{U}_y)$ that are always correct.

$(\Delta_x,\mathcal{U}_y)$	Correctness	(P1)	(P2)	(P2.1)	(P3)	(P4)
$(\Delta_e,\mathcal{U}_p)$	**Y**	Prop. 4	N	Y	Y	Y
$(\Delta_c,\mathcal{U}_p)$	Y [1]	Y	N	N	Y	Y
$(\Delta_d,\mathcal{U}_p)$	N	-	-	-	-	-
$(\Delta_{dc},\mathcal{U}_p)$	N	-	-	-	-	-
$(\Delta_{ed},\mathcal{U}_p)$	**Y**	Prop. 5	N	Y	Y [2]	Y [2]
$(\Delta_e,\mathcal{U}_{ir})$	N	-	-	-	-	-
$(\Delta_c,\mathcal{U}_{ir})$	**Y**	Y	Y	Y	Y	Y
$(\Delta_d,\mathcal{U}_{ir})$	Th.2	Y	Y	Y	Y	N
$(\Delta_{dc},\mathcal{U}_{ir})$	**Y**	Y	Y	Y	Y [2]	Y [2]
$(\Delta_{ed},\mathcal{U}_{ir})$	N	-	-	-	-	-

(1) if $K = C(K)$ (2) if $K' = C(K')$

Table IV. Synopsis of the Properties of $(\Delta_x,\mathcal{U}_y)$ Pairs

6. EXPERIMENTAL EVALUATION

In this Section we experimentally measure the time required to compare real and synthetic RDF/S KBs of variable size and structure. The objectives of this evaluation is to:

—measure and compare delta computation times over real data sets,

—measure and compare the delta sizes for identifying relationships that hold in various data sets (apart from the analytically derived relationships which are depicted at Figure 4), and investigating factors which affect the differences of deltas sizes. Our hypothesis is that significant variations are expected when the compared KBs have a rich subsumption structure. To quantify the latter we introduce a metric that measures the inference potential of a set of triples K.

DEF. 6. *The* inference strength *of a knowledge base K, denoted by $is(K)$, is defined as:*

$$is(K) = \frac{|C(K)| - |K|}{|K|}$$

Clearly, if $K = C(K)$ then its inference strength is zero, i.e. $is(K) = 0$, while the greater this factor is, the greater is the number of new inferred triples in $C(K)$ w.r.t. K.

Finally, another objective is to compare the sizes of reversible deltas (Δ_e, Δ_c) with respect to the sum of both forward and backward deltas of the non reversible ones $(\Delta_{ed}, \Delta_{dc})$.

For our experiments we have implemented the five differential functions and the two update semantics in the context of SWKM (Semantic Web Knowledge Middleware)[10]. For our measurements both KBs have been loaded in main memory, along with their labels, and thus only the net differential times are reported. All experiments were carried out in a computer with processor Intel Core2 Quad CPU Q6600 2.4 Ghz, 8 GB Ram, running Ubuntu 9.10 kernel 2.6.31-21-generic.

6.1 Testbed

Our testbed comprises *real* and *synthetic* datasets:

Real Data Sets

—We used the RDFS versions of CIDOC Conceptual Reference Model (ISO 21127) which is a core ontology describing the underlying semantics of data schemata and structures from museum disciplines and archives [Lin et al. 2008]. It is result of long-term interdisciplinary work and agreement and it has been derived by integrating (in a bottom-up manner) hundreds of metadata schemas. In our experiments we used the following sequence of versions: V3.2.1, V3.3.2, V3.4.9, V4.2 and V5.0.1.

—We used the RDF/S dumps from the Gene Ontology (GO) project[11] as a representative of large-scale evolving Semantic Web data. This data set contains 27,640 classes, and 1,359 property instances and uses 126 properties to describe genes. In particular, this dataset consists of 8 versions (v.25-11-2008, v.16-12-2008, v.24-3-2009, v.5-5-2009, v.26-5-2009, v.16-6-2009, v.22-9-2009, v.20-4-2010). It is worth noticing that GO curators usually reclassify terms as obsolete and thus they are not always deleted; this is done so that existing biological annotations do not have dangling references during the time lag between the term being made obsolete and the reannotation of the entity.

Synthetic Data Sets

We created and used two synthetic data sets. Specifically, using the synthetic KB generator described in [Theoharis et al. 2007], we created a sequence of four KBs, $K_1, \ldots, K_4$, with 100, 200, 300, and 400 classes respectively. The number of properties in each KB is $3 * N$ where N is the number of classes. The subsumption relation follows a power law distribution. Specifically, we used the metrics reported in [Theoharis et al. 2008]. We set the power-law exponent to 0.5 for the total-

[10] http://athena.ics.forth.gr:9090/SWKM

[11] www.geneontology.org/. Data were downloaded from ftp://ftp.uniprot.org/pub/databases/uniprot/previous_major_releases/ on Feb 2010.

degree VR[12] function of the property graph and to 1.7 for the PDF[13] function of the descendants distribution. The depth of the class subsumption hierarchy in each schema is 7.

At the **small** case, for each class 10 instances were created, while at the **large** one 100 instances were created. In both data sets, for each property 10 property instances were created among randomly selected instances of the corresponding domain and range class. To simulate their evolution, we adopted a naming convention such that all classes, properties, and their instances in K_i are also present in K_{i+1} for each i=1..3. However, their subsumption structure may differ: classes which are higher in the subsumption hierarchy in version K_i may be found at a lower level in version K_{i+1}. This results to both additions and deletions of explicitly defined triples (i.e. "subClassof" relationships). The depth of the class subsumption hierarchy in each schema is 7, and no redundant triple is contained in the synthetic data set.

6.2 Experimental Results

Real Data Sets

Regarding **CIDOC CRM**, Table V reports delta times in seconds. For each KB involved, the table shows its size (in triples) and its inference strength. Figure 9 shows delta sizes where the height of each bar indicates the delta size. The white part of each bar corresponds to deletions, and the black part to additions. We report delta sizes for both forward and backward directions. At this data set the produced deltas have similar sizes for Δ_e, Δ_d, Δ_{ed}, while the sizes of Δ_c and Δ_{dc} have small variations. The small differences in delta sizes are due to the small inference factor, whose range is [0.262, 0.357].

Table V. Delta Times for the CIDOC CRM Data Set

K			K'			Time of($\Delta(K, K')$)				
KB	Triples	$is(K)$	KB	Triples	$is(K')$	Δ_e	Δ_{ed}	Δ_c	Δ_d	Δ_{dc}
K_{c1}	952	0.262	K_{c2}	1,081	0.309	0.062	0.073	0.103	0.079	0.098
K_{c2}	1,081	0.309	K_{c3}	1,110	0.302	0.069	0.084	0.98	0.071	0.094
K_{c3}	1,110	0.302	K_{c4}	1,254	0.267	0.087	0.106	0.172	0.103	0.124
K_{c4}	1,254	0.267	K_{c5}	1,318	0.357	0.093	0.098	0.189	0.096	0.156
K_{c1}	952	0.262	K_{c5}	1,318	0.357	0.094	0.094	0.140	0.097	0.110

[12]VR stands for Value vs Rank. It measures the relationship between the i^{th} biggest value and its rank (in descending order), i.

[13]PDF stands for Probability Density Function.

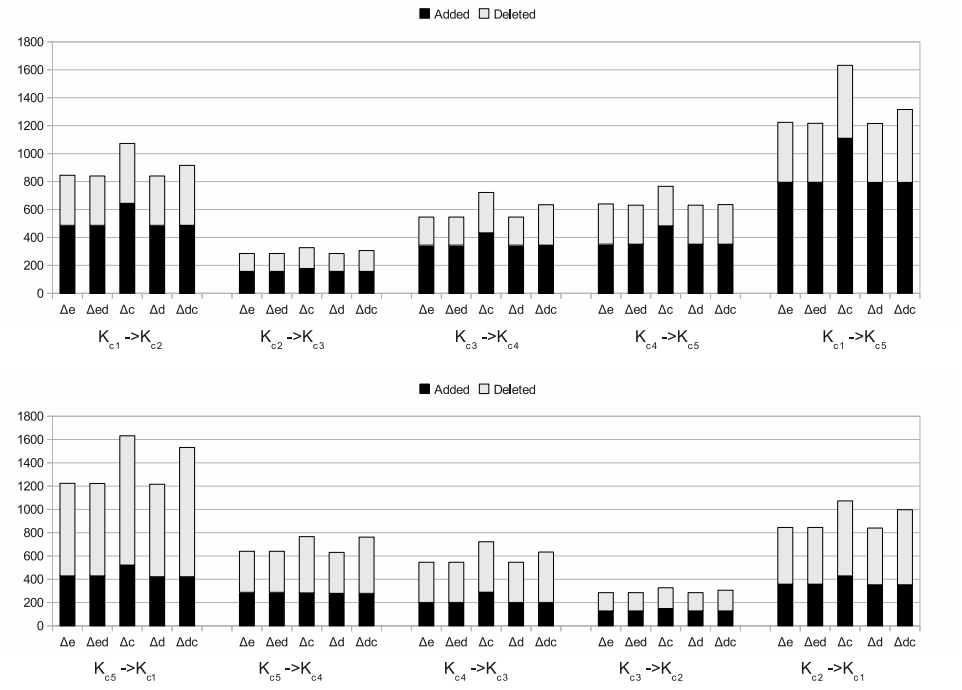


Fig. 9. Delta sizes for the CIDOC CRM Data Set

Regarding the **Biological** data set, Table VI shows the delta computing time for the 8 versions.

Table VI. Delta times for the Biological Data Set

K			K'			Time of $(\Delta(K, K'))$				
KB	Triples	$is(K)$	KB	Triples	$is(K')$	Δ_e	Δ_{ed}	Δ_c	Δ_d	Δ_{dc}
K_{b1}	184,563	1.631	K_{b2}	185,837	1.652	7.864	7.913	14.691	8.043	10.843
K_{b2}	185,837	1.652	K_{b3}	189,401	1.664	8.041	8.353	14.951	8.209	11.223
K_{b3}	189,401	1.664	K_{b4}	191,230	1.681	8.109	8.157	14.873	8.235	11.252
K_{b4}	191,230	1.681	K_{b5}	192,550	1.696	8.198	8.228	14.383	8.306	11.410
K_{b5}	192,550	1.696	K_{b6}	193,038	1.698	8.212	8.235	14.519	8.229	11.431
K_{b6}	193,038	1.698	K_{b7}	195,148	1.752	8.356	8.419	14.655	8.764	11.374
K_{b7}	195,148	1.752	K_{b8}	210,099	1.818	8.127	8.165	14.856	9.578	11.296

Figure 10 shows the delta sizes. As we can observe, the produced deltas exhibit important differences at the sizes, especially the size of Δ_c and Δ_{dc}. The bigger difference of sizes is due to the fact that the inference factor (i.e. is) is bigger in this dataset, specifically its range is $[1.631, 1.818]$.

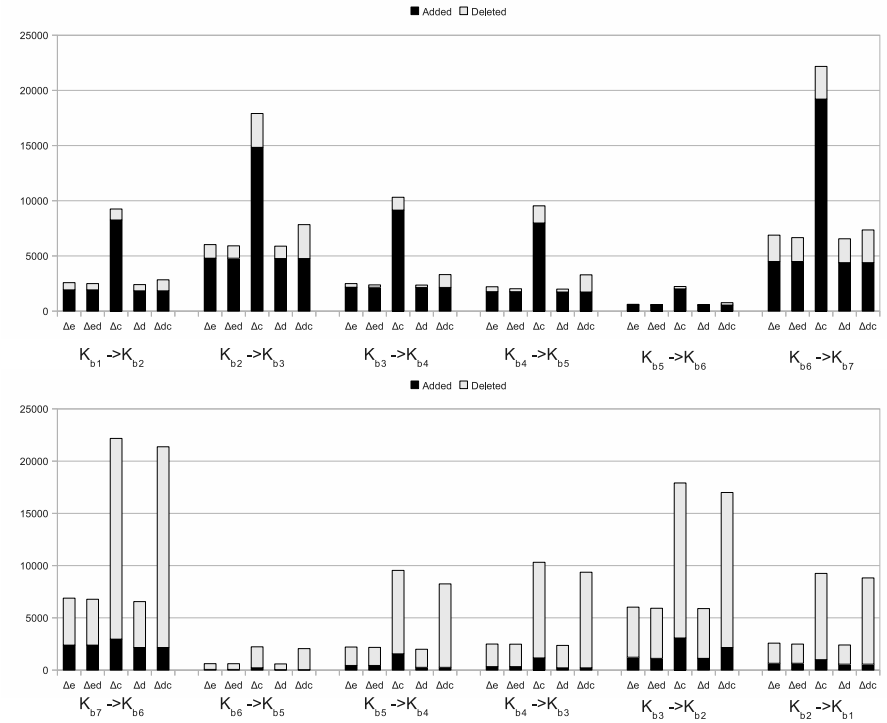


Fig. 10. Delta sizes for the Biological Data Set

Synthetic Data Sets

Table VII and Figure 11 report respectively, the delta computing times and sizes for the four consequent versions of the **small** Synthetic data set. Here the range of the inference factor is [0.789, 0.826] (greater than that of CIDOC CRM but less than that of GO). On average, the deltas produced by Δ_c are 63% bigger than those of Δ_e. The deltas of Δ_{dc} are 16% bigger that those of Δ_e. Note however, that Δ_c Δ_d and Δ_{dc} satisfy properties like semantic identity and also non-redundancy when used with $\mathcal{U}_{ir}$-semantics. The delta of Δ_d is 1% smaller than the size of Δ_e and the delta of Δ_{ed} is 0.5% smaller than the size of Δ_e. The minor differences among $|\Delta_e|$, $|\Delta_{ed}|$ and $|\Delta_d|$ are due to the fact that most changes occur at the explicit graph i.e. only few triples that are inferred by K became explicit in K' and vice versa. On the other hand, the significant divergences between $|\Delta_e|$ and $|\Delta_c|$ are due to changes occurring higher at the subsumption hierarchy. For the same reason, the deletions reported by Δ_{dc} are more than those reported by Δ_e, Δ_{ed} and Δ_d. In general, for Δ_e, Δ_{ed} and Δ_d all kinds of changes affect in the same way the delta size, while for Δ_c additions or deletions that occur highly at the subsumption hierarchy affect more the delta size. Finally, for Δ_{dc} all additions have the same impact, but deletions that occur highly at the subsumption hierarchy affect more the size of the produced delta. Our experimental findings are summarized in Figure 12 which shows the delta times and sizes in relation to the size (in triples) of the compared KBs (each value of the X-axis corresponds to the sum $|K_1| + |K_2|$). Apart from the ordering of deltas sizes that was illustrated in Figure 4 (and of course hold here as well), we observe that in most cases it holds $|\Delta_e| < |\Delta_c|$ and $|\Delta_{ed}| < |\Delta_{dc}|$. Recall that Δ_e produces a big in size result if $K = C(K')$, while Δ_e and Δ_{ed} produce a big in size result if $K' = C(K')$.

Table VII. Delta Times for the Small Synthetic Data Set

K			K'			Time of $(\Delta(K, K'))$				
KB	Triples	$is(K)$	KB	Triples	$is(K')$	Δ_e	Δ_{ed}	Δ_c	Δ_d	Δ_{dc}
K_{s1}	5,162	0.821	K_{s2}	10,267	0.789	0.570	0.583	0.952	0.594	0.718
K_{s2}	10,267	0.789	K_{s3}	15,389	0.806	0.889	0.903	1.499	1.187	1.388
K_{s3}	15,389	0.806	K_{s4}	20,460	0.826	1.296	1.352	2.217	1.623	1.702
K_{s1}	5,162	0.821	K_{s4}	20,460	0.826	1.077	1.088	1.624	1.109	1.092
K_{s4}	20,460	0.826	$C(K_{s4})$	37,369	0	1.225	1.358	1.782	1.462	1.639

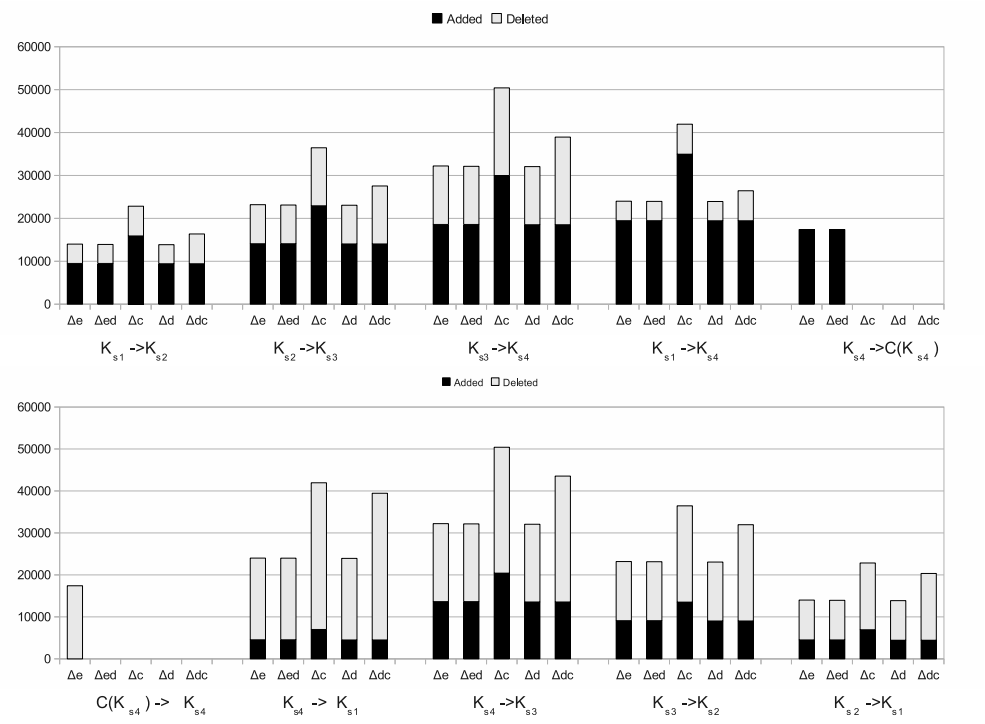


Fig. 11. Delta sizes for Small Synthetic Data Set

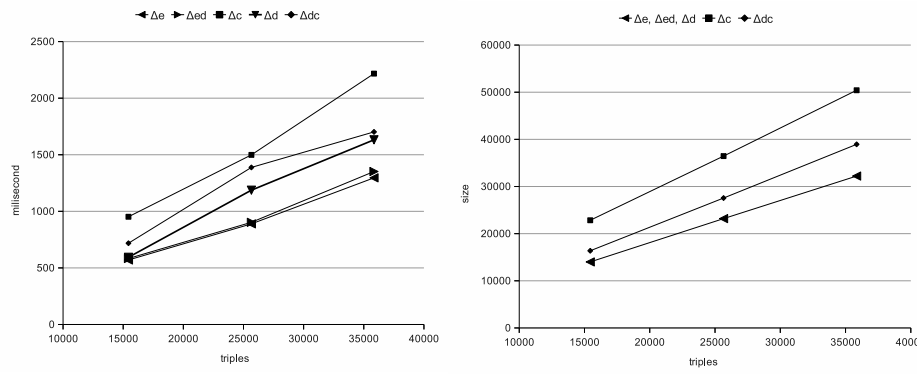


Fig. 12. Delta time/size: small Synthetic Data

Table VIII and Figure 13 report respectively, the delta computing times and sizes for the four successive versions of the **large** Synthetic data set that has a bigger inference strength, specifically it ranges [2.868, 2.987], which is the biggest in comparison to the previous data sets. In this case, significant differences in delta sizes are observed. Specifically, on average Δ_c is 212% bigger than Δ_e. Δ_{dc} is 57% bigger than Δ_e. Δ_d is 1.3% smaller than Δ_e. Δ_{ed} is 0.8% smaller than Δ_e. Significant divergences also observed in delta computing time. Figure 14 shows the delta times and sizes in relation to the size (in triples) of the compared KBs.

Table VIII. Delta Times for the Large Synthetic Data Set

K			K'			Time of $(\Delta(K, K'))$				
KB	Triples	$is(K)$	KB	Triples	$is(K')$	Δ_e	Δ_{ed}	Δ_c	Δ_d	Δ_{dc}
K_{l1}	14,250	2.970	K_{l2}	28,355	2.868	1.333	1.339	5.382	1.844	2.590
K_{l2}	28,355	2.868	K_{l3}	42,477	2.923	3.266	2.511	9.302	3.413	5.201
K_{l3}	42,477	2.923	K_{l4}	56,546	2.987	3.443	3.643	13.442	5.071	7.329
K_{l1}	14,250	2.970	K_{l4}	56,546	2.987	2.131	2.192	7.741	2.205	3.405

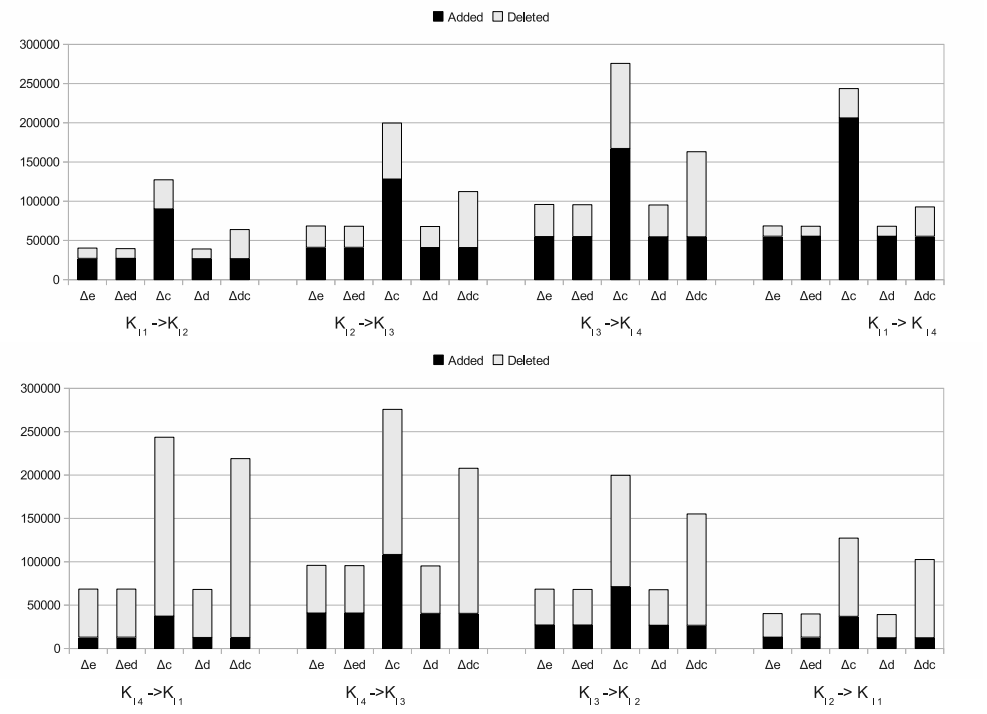


Fig. 13. Delta Sizes for the Large Synthetic Data Set

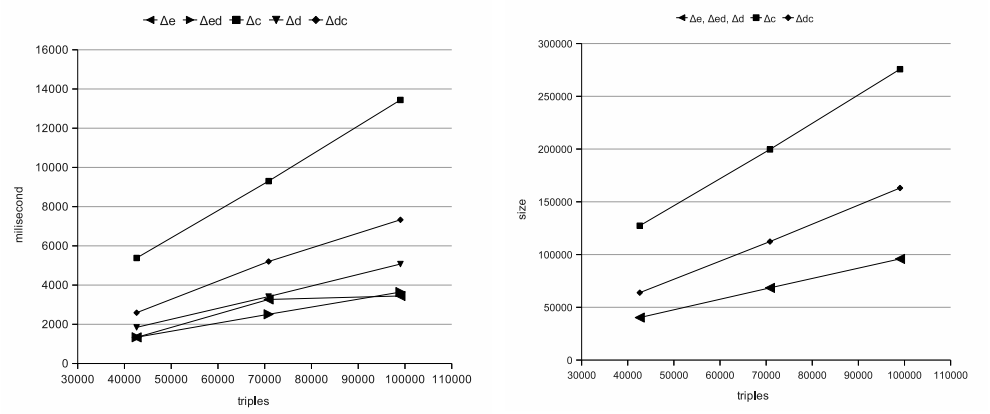


Fig. 14. Delta time/size: large Synthetic Data

Aggregated Results

Here we present some aggregated results based on all the four previously presented data sets.

Figure 15 (left) shows how inference strength affects the differences of the delta sizes returned by the five differential functions The figure shows the % difference of the delta size of each function with respect to the delta size returned by Δ_e, i.e. it depicts the quantity $\frac{|\Delta_x(K,K')|-|\Delta_e(K,K')|}{|\Delta_e(K,K')|} * 100$. The size of Δ_e is considered as the baseline that's why it is not shown in the figure. We can see that the more the inference strength is, the more the delta sizes differ, and this affects more the delta sizes of Δ_{dc} and Δ_c. The sizes of Δ_{ed} and Δ_d, are very close to the size of Δ_e, specifically their plot is slightly negative (since they are less than Δ_e).

Figure 15 (right) illustrates how the inference strength affects the difference at the size of deltas required in order to be able to move both forward and backward to KB versions. Since Δ_e and Δ_c are reversible only the size of delta at one direction is considered (i.e. $|\Delta(K, K')|$), while for Δ_{dc} and Δ_{ed} the bisectional delta size is considered i.e. $|\Delta(K, K')| + |\Delta(K', K)|$. Again the plots assume the delta size of Δ_e as baseline. Δ_e appears to be the cheaper in size solution, while Δ_{dc} is the most expensive in size solution. Notice that although in one direction Δ_c yields the biggest delta (Figure 15 (left)), if both directions are considered, then Δ_c is not the most expensive, it is Δ_{dc} that yields the biggest (bisectional) delta. Regarding the delta sizes of Δ_c and Δ_{ed} we can see that Δ_c is cheaper in size than Δ_{ed} if the inference strength is less than a value around 1, while Δ_{ed} is cheaper for higher values of inference strength.

To understand the junction of the $|\Delta_c(K_1, K_2)|$ and $|\Delta_{ed}(K_1, K_2)|+|\Delta_{ed}(K_2, K_1)|$ plots, let us investigate the hypotheses that

$$|\Delta_c(K_1, K_2)| \leq |\Delta_{ed}(K_1, K_2)| + |\Delta_{ed}(K_2, K_1)| \quad (1)$$

By unfolding the definitions of the differential functions, the inequality becomes:
$|\Delta_c(K_1, K_2)| \leq |\Delta_{ed}(K_1, K_2)| + |\Delta_{ed}(K_2, K_1)| \Leftrightarrow$
$|C(K_2) - C(K_1)| + |C(K_1) - C(K_2)| \leq$

$$|K_2 - K_1| + |K_1 - K_2| + |K_1 - C(K_2)| + |K_2 - C(K_1)| =$$
$$|\Delta_e(K_1, K_2)| + |\Delta_d(K_1, K_2)|.$$

By replacing
$|C(K_2) - C(K_1)|$ by $|K_2 - C(K_1)| + |(C(K_2) - K_2) - C(K_1)|$ and
$|C(K_1) - C(K_2)|$ by $|K_1 - C(K_2)| + |(C(K_1) - K_1) - C(K_2)|$
the above inequality becomes:

$$|(C(K_2) - K_2) - C(K_1)| + |(C(K_1) - K_1) - C(K_2)| \leq |K_2 - K_1| + |K_1 - K_2| \quad (2)$$

Notice that if the inference strengths of the KBs (i.e. $is(K_1)$ and $is(K_2)$) are small, then $(C(K_2) - K_2)$ and $(C(K_1) - K_1)$ become small. In case they are zero (i.e. $is(K_1) = is(K_2) = 0$) these quantities are equal to zero and the inequality (1) certainly holds (since it reduces to $0 \leq |K_2 - K_1| + |K_1 - K_2|$). It follows that the plot of $|\Delta_c|$ is certainly lower than the plot of bisectional Δ_{ed} for zero or small inference strengths. We conclude that for small inference strengths, Δ_c is cheaper than the bisectional Δ_{ed}.

We can also observe that the bisectional Δ_{ed} has a rather straight plot. This is justified from the fact that $|\Delta_{ed}(K_1, K_2)| + |\Delta_{ed}(K_2, K_1)| = |\Delta_e(K_1, K_2)| + |\Delta_d(K_1, K_2)|$, and as we can see the plot of Δ_d (at Figure 15 (left)) is quite close to the baseline Δ_e, therefore the plot of the bisectional Δ_{ed} (at Figure 15 (right)) is just below 100% for all values of inference strength.

Returning to inequality (2) we can also see that its right part is independent of the inference strength, while its left part depends on inference strength. The left part becomes bigger than the right part at some point as it is shown in the plot (the exact value cannot be derived analytically since it depends on the overlap between $C(K_1)$ and $C(K_2)$).

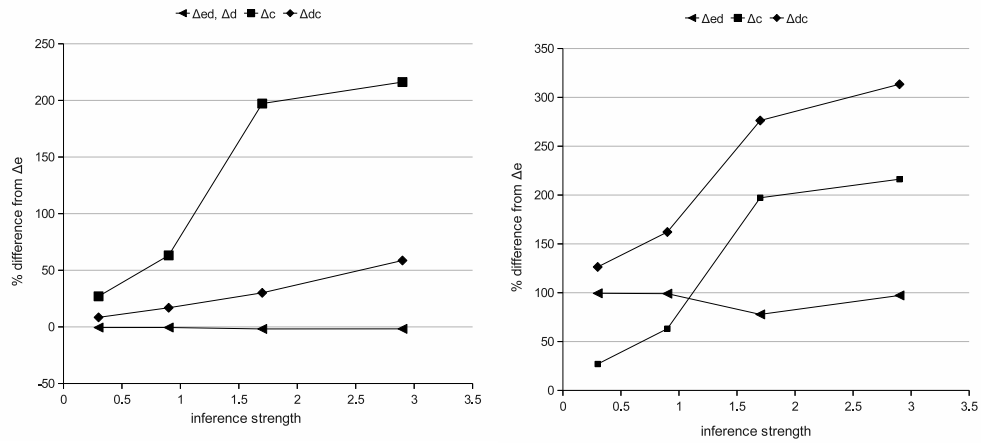


Fig. 15. % difference from Δ_e size at (a) Forward Deltas (b)Forward/Backward Deltas

7. CONCLUDING REMARKS

Most of the existing RDF/S differential tools [Berners-Lee and Connolly 2004; Volkel et al. 2005; Ding et al. 2005] rely on the $(\Delta_e, \mathcal{U}_p)$ pair. Semversion [Volkel et al. 2005] offers also $(\Delta_c, \mathcal{U}_p)$ which (as we have proved) yields correct results if

K is closed. None of the works (theoretical or practical) has used Δ_d, Δ_{dc} or Δ_{ed}, nor studied formally properties as their size, correctness and non-redundancy w.r.t. the semantics of the change operations, as well as semantic identity, reversibility and composability.

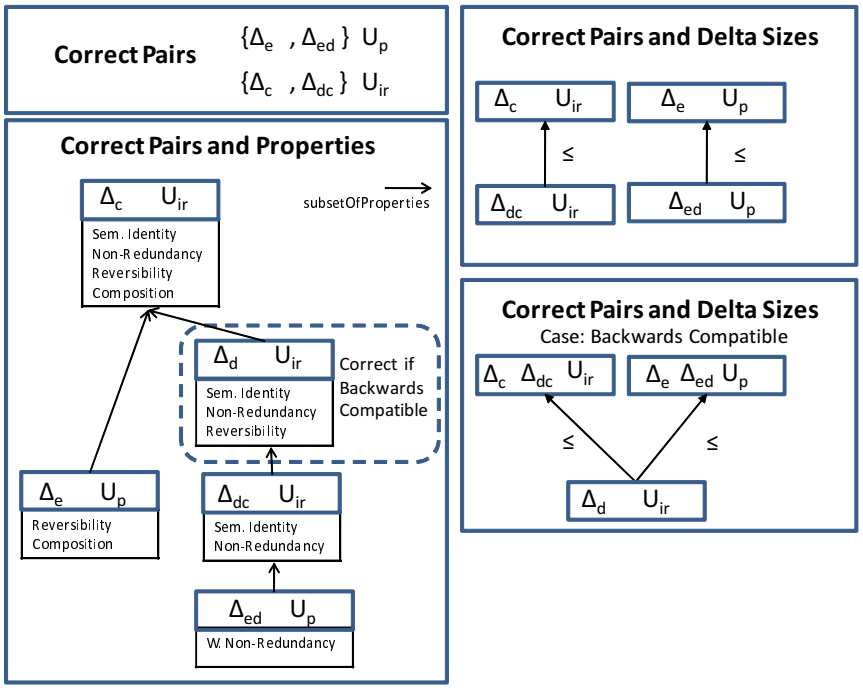


Fig. 16. Synopsis of the Results Regarding the Properties and Sizes of $(\Delta_x, \mathcal{U}_y)$ pairs

Figure 16 synopsizes the main results of this paper. The pairs that are always correct are: $(\Delta_e, \mathcal{U}_p)$, $(\Delta_{ed}, \mathcal{U}_p)$, $(\Delta_{dc}, \mathcal{U}_{ir})$ and $(\Delta_c, \mathcal{U}_{ir})$ (shown at the upper left part of the figure). We have also identified special cases where $(\Delta_d, \mathcal{U}_{ir})$ is correct given that it is the smallest in size delta. The diagram at the lower right part of the figure shows the properties that each pair satisfies, and the pairs are ordered according to the properties that they have (a pair p is child of a p', if p satisfies a subset of the properties of p').

Regarding delta sizes, at the general case it holds $|\Delta_{dc}| < |\Delta_c|$ and $|\Delta_{ed}| < |\Delta_e|$. We have experimentally verified that Δ_e produces smaller in size results than Δ_c and that in most practical cases it holds that $|\Delta_{ed}| < |\Delta_{dc}|$ (as long as the KBs inference strength increases). In case the two KBs are backward compatible (i.e. $C(K) \subseteq C(K')$) then Δ_d and Δ_{dc} produce the smallest in size result.

Our experimental evaluation showed that the computation time for Δ_c is always greater that the other ones. Furthermore, the computation times of Δ_e, Δ_d and Δ_{ed} are very close and clearly smaller than Δ_{dc}. Furthermore, Δ_e and Δ_{ed} require less time than Δ_c and Δ_{dc}, respectively. However, if $K \sim K'$ then Δ_e and Δ_{ed} may return a very big in size result while Δ_{dc} and Δ_c yield always an empty result.

The cost of executing correctly the deltas depends on the semantics of the change operations, and $\mathcal{U}_{ir}$ is more expensive as it requires computing the closure and the

reduction of a KB. This is the price to pay for achieving correctness with certain differential functions and for keeping the resulting KBs in a redundancy-free state.

In addition, we have proved that only $(\Delta_e, \mathcal{U}_p)$ and $(\Delta_c, \mathcal{U}_{ir})$ can be reversed and composed to implement various forward/backward version policies over distributed RDF/S KB archives. Δ_e appears to be the cheaper in size solution, while the bisectional Δ_{dc} is the most expensive in size solution. Notice that although in one direction Δ_c yields the biggest delta, if both directions are considered, then Δ_c is not the most expensive, it is Δ_{dc} that yields the biggest (bisectional) delta. Regarding the delta sizes of Δ_c and the bisectional Δ_{ed}, Δ_c is cheaper in size than Δ_{ed} if the inference strength is less than a value around 1, while Δ_{ed} is cheaper for higher values of inference strength.

Implementation details, as well as issues regarding the peculiarities of RDF including blank nodes identification and containers (e.g. Bag, Sequence, Alternative), go beyond the scope of this work and are issues for future research. For instance, blank node matching is worth investigating in order to further reduce the size of deltas.

Acknowledgements. This work was partially supported by the EU projects CASPAR (FP6-2005-IST-033572) and KP-Lab (FP6-2004-IST-4). Many thanks to Yannis Theoharis for providing us the synthetic datasets and to Vicky Papavassileiou for helping us in the experimental evaluation.

REFERENCES

Aho, A., Garey, M., and Ullman, J. 1972. The transitive reduction of a directed graph. *SIAM Journal on Computing 1*, 131.

Alchourrón, C. E., Gärdenfors, P., and Makinson, D. 1985. On the Logic of Theory Change: Partial Meet Contraction and Revision Functions. *Journal of Symbolic Logic 50,* 2, 510–530.

Berliner, B. 1990. CVS II: Parallelizing Software Development. In *Proceedings of the USENIX Winter 1990 Technical Conference*. Berkeley, CA, 341–352.

Berners-Lee, T. and Connolly, D. 2004. Delta: An Ontology for the Distribution of Differences Between RDF Graphs. http://www.w3.org/DesignIssues/Diff (version: 2006-05-12).

Brickley, D. and Guha, R. V. 2004. RDF Vocabulary Description Language 1.0: RDF Schema, W3C Recommendation. http://www.w3.org/TR/rdf-schema/.

Christophides, V., Plexousakis, D., Scholl, M., and Tourtounis, S. 2003. On Labeling Schemes for the Semantic Web. In *Proceedings of WWW'03*. Budapest, Hungary, 544–555.

Cloran, R. and Irwin, B. 2005. Transmitting RDF Graph Deltas for a Cheaper Semantic Web. In *Proceedings of Southern African Telecommunications Networks and Applications Conference (SATNAC'05)*. South Africa.

Cobena, G., Abiteboul, S., and Marian, A. 2001. Detecting Changes in XML Documents. In *Proceedings of the International Conference on Data Engineering (ICDE)*. Heidelberg, Germany.

Ding, L., Finin, T., Joshi, A., Peng, Y., da Silva, P., and McGuinness, D. 2005. Tracking RDF Graph Provenance using RDF Molecules. In *Proceedings of International Semantic Web Conference (ISWC'05)*. Galway,Ireland.

Flouris, G. 2006. On Belief Change and Ontology Evolution. Ph.D. thesis, Computer Science Department, University of Crete, Greece.

Gärdenfors, P. 1992. Belief Revision: An Introduction. In *Belief Revision*. Cambridge University Press, 1–20.

Gutierrez, C., Hurtado, C., and Mendelzon, A. 2004. Foundations of Semantic Web Databases. In *Proceedings 23 ACM Symposium on Principles of Database Systems (PODS) 2004*.

Hayes, P. 2004. RDF Semantics, W3C Recommendation. http://www.w3.org/TR/rdf-mt/.

Heflin, J., Hendler, J., and Luke, S. 1999. Coping with Changing Ontologies in a Distributed Environment. In *Proceedings of Association for the Advancement of Artificial Intelligence (AAAI'99) Workshop on Ontology Management*. Florida.

Klein, M., Fensel, D., Kiryakov, A., and Ognyanov, D. 2002. Ontology Versioning and Change Detection on the Web. In *Proceedings of International Conference on Knowledge Engineering and Knowledge Management Knowledge Patterns (EKAW'02)*. Siguenza, Spain, 197–212.

Klein, M. and Noy, N. 2003. A Component-based Framework for Ontology Evolution. In *Workshop on Ontologies and Distributed Systems at International Joint Conferences on Artificial Intelligence (IJCAI'03)*. Acapulco, Mexico.

Konieczny, S. and Perez, R. P. 2005. Propositional Belief Base Merging or How to Merge Beliefs/Goals Coming from Several Sources and Some Links With Social Choice Theory. *European Journal of Operational Research 160,* 3, 785–802.

Konstantinidis, G., Flouris, G., Antoniou, G., and Christophides, V. 2008. A formal approach for rdf/s ontology evolution. In *Procs. of ECAI'08*. Patras, Greece, 70–74.

Lin, C., Hong, J., and Doerr, M. 2008. Issues in an inference platform for generating deductive knowledge: a case study in cultural heritage digital libraries using the CIDOC CRM. *International Journal on Digital Libraries 8,* 2, 115–132.

Magiridou, M., Sahtouris, S., Christophides, V., and Koubarakis, M. 05. RUL: A Declarative Update Language for RDF. In *Proceedings of International Semantic Web Conference (ISWC'05)*. Galway, Ireland, 506–521.

Marian, A., Abiteboul, S., Cobena, G., and Mignet, L. 2001. Change-Centric Management of Versions in an XML Warehouse. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*. Roma, Italy, 581–590.

Noy, N., Chugh, A., Liu, W., and Musen, M. 2006. A framework for ontology evolution in collaborative environments. In *International Semantic Web Conference*. Athens, GA, USA, 544–558.

Noy, N., Kunnatur, S., Klein, M., and Musen, M. A. 2004. Tracking Changes During Ontology Evolution. In *Proceedings of International Semantic Web Conference (ISWC'04)*. Hisroshima, Japan, 259–273.

Noy, N. and Musen, M. 2002. PromptDiff: A Fixed-point Algorithm for Comparing Ontology Versions. In *Proceedings of Association for the Advancement of Artificial Intelligence (AAAI'02)*. Edmonton, Alberta, 744–750.

Noy, N. and Musen, M. 2004. Ontology Versioning in an ontology Management framework. *IEEE Intelligent Systems 19,* 4, 6–13.

Nuutila, E. 1995. *Efficient Transitive Closure Computation in Large Digraphs*. Acta Polytechnica Scandinavica, Mathematics and Computing in Engineering Series No. 74, Finnish Academy of Technology.

Plessers, P. and Troyer, O. D. 2005. Ontology Change Detection Using a Version Log. In *Proceedings of International Semantic Web Conference (ISWC'05)*. Galway, Ireland, 578–592.

Poutre', J. A. L. and van Leeuwen, J. 1987. Maintenance of Transitive Closures and Transitive Reductions of Graphs. In *Proceedings of the Intern. Workshop on Graph-Theoretic Concepts in Computer Science*. 106–120.

Theoharis, Y., Georgakopoulos, G., and Christophides, V. 2007. On the Synthetic Generation of Semantic Web Schemas. In *Proceedings of the SWDB-ODBIS07: Joint ODBIS & SWDB workshop on Semantic Web, Ontologies, Databases. Colocated with VLDB'07*. Vienna, Austria.

Theoharis, Y., Tzitzikas, Y., Kotzinos, D., and Christophides, V. 2008. On graph features of semantic web schemas. *IEEE Trans. on Knowl. and Data Eng. 20,* 5, 692–702.

Tummarello, G., Morbidoni, C., Petersson, J., Piazza, F., and Puliti, P. 2004. RDFGrowth, a P2P Annotation Exchange Algorithm for Scalable Semantic Web Applications. In *1st Annual International Conference on Mobile and Ubiquitous Systems MobiQuitous*. Boston, MA.

Tzitzikas, Y. and Kotzinos, D. 2007. (Semantic Web) Evolution through Change Logs: Problems and Solutions. In *Proceedings of Artificial Intelligence and Applications (AIA'07)*. Innsbruck, Austria.

Ullman, J. and Yannakakis, M. 1991. The input/output complexity of transitive closure. *Annals of Mathematics and Artificial Intelligence 3,* 2, 331–360.

Volkel, M., Winkler, W., Sure, Y., Kruk, S. R., and Synak, M. 2005. SemVersion: A Versioning System for RDF and Ontologies. In *Proceedings of European Semantic Web Conference (ESWC'05)*. Heraklion, Crete.

Warshall, S. 1962. A Theorem on Boolean Matrices. *J. ACM 9,* 1, 11–12.

Zeginis, D., Tzitzikas, Y., and Christophides, V. 2007. On the Foundations of Computing Deltas Between RDF Models. In *Proceedings of the 6th International Semantic Web Conference (ISWC'07)*. Busan, S. Korea.

Zhang, Z., Zhang, L., Lin, C., Zhao, Y., and Yu, Y. 2003. Data Migration for Ontology Evolution. In *Poster Proceedings International Semantic Web Conference (ISWC'03)*. Sanibel Island, Florida, USA.

8. APPENDIX: PROOFS

At first we list a number of properties (the proofs of some of these are trivial and are therefore omitted) which we exploit subsequently for proving a number of propositions and theorems.

(a) $C(R(K)) = C(K)$
(b) $A - (B - D) = (A - B) \cup (A \cap D)$
(c) If $A \subseteq D$ then $(A - B) \cup (B \cap D) = A \cup (B \cap D)$

Proof of (b)
$x \in A - (B - D) \Leftrightarrow x \in A$ and $x \notin (B - D) \Leftrightarrow$
$x \in A$ and not $(x \in B$ and $x \notin D) \Leftrightarrow x \in A$ and $(x \notin B$ or $x \in D) \Leftrightarrow$
$(x \in A$ and $x \notin B)$ or $(x \in A$ and $x \in D) \Leftrightarrow x \in (A - B)$ or $x \in (A \cap D) \Leftrightarrow$
$x \in (A - B) \cup (A \cap D)$ $\diamond$

Proof of (c)
$A \subseteq D \Rightarrow A \cap B \subseteq B \cap D$ (F)
$(A - B) \cup (B \cap D) = (A - (A \cap B)) \cup (B \cap D) \overset{(F)}{=} A \cup (B \cap D)$ $\diamond$

PROP. 1 If $K \in \Psi$ then $R(K) \subseteq K$

Proof:
We will prove this inductively with respect to $s = |K|$. Obviously, If $s = 0$ then $K = C(K) = R(K) = \emptyset$, so $R(K) \subseteq K$ holds. Let's now make the inductive hypothesis that for an $s = n$ it holds: if $|K| \leq n$ then $R(K) \subseteq K$. We will show that this holds also for $s = n + 1$.

Consider a K' such that $|K'| = n+1$. This means that we can write $K' = K \cup \{t_x\}$ for a K such that $|K| = n$. If t_x can be inferred from K (i.e. $t_x \in C(K)$), then t_x cannot belong to $R(K')$ because the set $R(K)$ is a reduction also for $K \cup \{t_x\} = K'$. The fact that t_x can be inferred from K means that $K' \subseteq C(K)$ and recall that the reduction of $C(K)$ is unique as we are in Ψ. So $R(K') = R(K) \subseteq K \subseteq K'$.

Now suppose that $K' = K \cup \{t_x\}$ but t_x cannot be inferred from K (i.e. $t_x \notin$

$C(K)$ and thus $t_x \notin C(R(K)))$. Let $B = R(K) \cup \{t_x\}$. Obviously $B \subseteq K'$ and $C(B) = C(K')$. To prove that B is the reduction of K' we also need to prove that the size of B is minimal. If the size of B is indeed the minimal, then it is the reduction, so we have $R(K') = B \subseteq K'$. If the size of B is not minimal then this means that the addition of t_x (to $R(K)$) has turned one or more elements of $R(K)$ redundant. So let $B' = B - S$ where S are the redundant (inferred) triples (of course $|S| \geq 1$) . In this case we have $|B'| < |B| \leq n + 1$, so $|B'| \leq n$, but in that case $R(B') \subseteq B'$ due to the inductive hypothesis. It follows that $R(B') \subseteq B' \subseteq B \subseteq K'$, so $R(K') \subseteq K'$ as we have shown that $R(K')$ is either the set B or the set B'. $\diamond$

PROP. 2 If $\{K,K'\} \subseteq \Psi$ then for every $x \in \{e, ed\}$ and $y \in \{\Delta_d, \Delta_{dc}, \Delta_c\}$, $\Delta_x^{\mathcal{U}_p}(K, K')$ and $\Delta_y^{\mathcal{U}_{ir}}(K, K')$ satisfy all operations of Δ_x under $\mathcal{U}_p$ and Δ_y under $\mathcal{U}_{ir}$ respectively.

Proof:

For $\mathcal{U}_p$ semantics it is obvious that all operations will be satisfied. For $\mathcal{U}_{ir}$ semantics all additions will be satisfied since after the union operation only the reduction operation takes places (which does not affect satisfaction in $\mathcal{U}_{ir}$). Regarding deletions, at first notice that after the set-minus operation no other triple will be added apart from those in Δ^+. Since the pairs (Δ^-, Δ^+) produced by the differential functions are consistent, the addition of the triples in Δ^+ will not contain any triple of Δ^-. The rest of the proof shows that all the deletions will be satisfied and is similar to the proof of Prop. 8.
$\diamond$

PROP. 3 If $K \sim K'$ then $\Delta_d(K, K') = \Delta_c(K, K') = \Delta_{dc}(K, K') = \emptyset$.

Proof:
Recall that $\Delta_d(K, K') = \{Add(t) \mid t \in K' - C(K)\} \cup \{Del(t) \mid t \in K - C(K')\}$
Regarding additions, $K' - C(K) \overset{(Def1)}{=} \emptyset$, regarding deletions $K - C(K') \overset{(Def1)}{=} \emptyset$. This means that $\Delta_d(K, K') = \emptyset$. The proof for Δ_c and Δ_{dc} is similar. $\diamond$

PROP. 4 If $K \sim K'$, $\{K, K'\} \subseteq \Psi$ and $RF(K), RF(K')$ then $\Delta_e(K, K') = \emptyset$

Proof:

Suppose that $R(K) \neq K$. We know (from Prop. 1) that if $K \in \Psi$ then $R(K) \subseteq K$. This means that $\exists t \in K \quad \text{and} \quad t \notin R(K)$. In other words there exists a triple $t \in K$ that can be inferred from the other triples in K which is not true since we have $RF(K)$. So if $K \in \Psi$ and $RF(K)$ then $R(K) = K$. It follows that $K = K'$, therefore $\Delta_e(K, K') = \emptyset$. $\diamond$

PROP. 5 If $K \sim K'$, and (a) $K' \subseteq K$ or (b) $\{K, K'\} \subseteq \Psi$ and $RF(K), RF(K')$, then $\Delta_{ed}(K, K') = \emptyset$

Proof:
Recall that $\Delta_{ed}(K, K') = \{Add(t) \mid t \in K' - K\} \cup \{Del(t) \mid t \in K - C(K')\}$. We know that $K - C(K') \overset{(Def1)}{=} \emptyset$. Assuming the (a) premises of the proposition we

have $K' \subseteq K \Rightarrow K' - K = \emptyset$. This means that $\Delta_{ed}(K, K') = \emptyset$.

If $K \sim K'$ and the (b) premises of the proposition hold, then $K = K'$ (recall Prop. 4), so $\Delta_{ed}(K, K') = \emptyset$. $\diamond$

PROP. 6 For every pair of knowledge bases K, K' it holds:

$$Inv(\Delta_e(K, K')) = \Delta_e(K', K)$$
$$Inv(\Delta_c(K, K')) = \Delta_d(K', K)$$
$$Inv(\Delta_d(K, K')) = \Delta_c(K', K)$$

Proof:
$Inv(\Delta_e(K, K')) = \{Add(t) \mid t \in K - K'\} \cup \{Del(t) \mid t \in K' - K\} = \Delta_e(K', K)$. This means that $Inv(\Delta_e(K, K')) = \Delta_e(K', K)$. The proof for Δ_c and Δ_d is similar. $\diamond$

PROP. 7 For any knowledge bases $K_1, K_2, K_3, ..., K_n$ it holds:

$$\Delta_e(K_1, K_2) \circ \Delta_e(K_2, K_3) \circ ... \circ \Delta_e(K_{n-1}, K_n) \;=\; \Delta_e(K_1, K_n)$$
$$\Delta_c(K_1, K_2) \circ \Delta_c(K_2, K_3) \circ ... \circ \Delta_c(K_{n-1}, K_n) \;=\; \Delta_c(K_1, K_n)$$

Proof:
Let's first prove the composability for two deltas. In this case we have to prove that $\Delta_e(K_1, K_2) \circ \Delta_e(K_2, K_3)$ and $\Delta_e(K_1, K_3)$ produce the same add and del statements i.e.

$$\Delta^+ = (K_2 - K_1) \cup (K_3 - K_2) - (K_1 - K_2) \cup (K_2 - K_3) \;=\; K_3 - K_1$$
$$\Delta^- = (K_1 - K_2) \cup (K_2 - K_3) - (K_2 - K_1) \cup (K_3 - K_2) \;=\; K_1 - K_3$$

From the Venn diagram of Figure 17 we can see that the above equalities are correct.

We have proven that Δ_e satisfies composability for two deltas. The proof for n deltas follows easily by induction, i.e.:

$$\Delta_e(K_1, K_2) \circ \Delta_e(K_2, K_3) \circ ... \circ \Delta_e(K_{n-1}, K_n) \;=$$
$$\Delta_e(K_1, K_3) \circ \Delta_e(K_3, K_4) \circ ... \circ \Delta_e(K_{n-1}, K_n) \;=$$
$$\Delta_e(K_1, K_4) \circ \Delta_e(K_4, K_5) \circ ... \circ \Delta_e(K_{n-1}, K_n) \;=\; ... = \Delta_e(K_1, K_n)$$

The proof for Δ_c is similar to Δ_e with the difference that $C(K_1), C(K_2)$ and $C(K_3)$ are used instead of K_1, K_2 and K_3. $\diamond$

PROP. 8 For every set of change operations M produced by Δ_{dc}, Δ_c and Δ_d in Ψ, Alg. 1 terminates.

Proof

From the structure of the algorithm it is clear that the algorithm will not terminate unless all operations (both additions and deletions) in M are satisfied. The algorithm terminates because $UnSat$ keeps decreasing. It is trivial to see that this holds for the additions contained in $UnSat$, i.e. each one of them will be satisfied

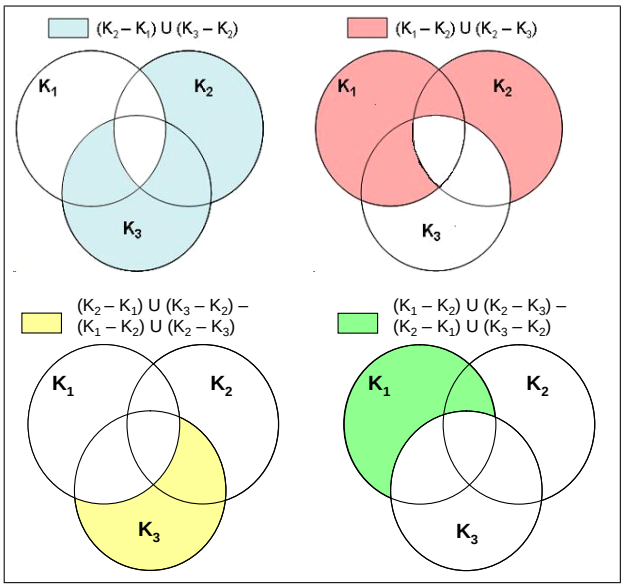


Fig. 17. On the Composability of Δ_e

the first time it is applied (at step 6). The deletions in $\mathcal{U}_p$ semantics will be satisfied the first time they are applied, since the deletion of a triple t is satisfied by a K if $t \notin K$, so those t that belong to K will be deleted and satisfied directly.

However it is not so trivial to see that the deletions will be satisfied in U_{ir}, due to their pre-conditions and the definition of satisfaction. Recall that according to the pre-conditions of deletions in U_{ir}, a $Del(t)$ cannot be satisfied if $t \notin K$ and $t \in C(K)$.

Notice that both Δ_c and Δ_{dc} produce the following set of deletions: $W = \{\ Del(t') \mid t' \in C(K) - C(K')\ \}$ which is shown shaded in Figure 18. In the upper case it is easy to see that all elements of W will be satisfied as the entire K (and the entire $R(K)$) is contained in W.

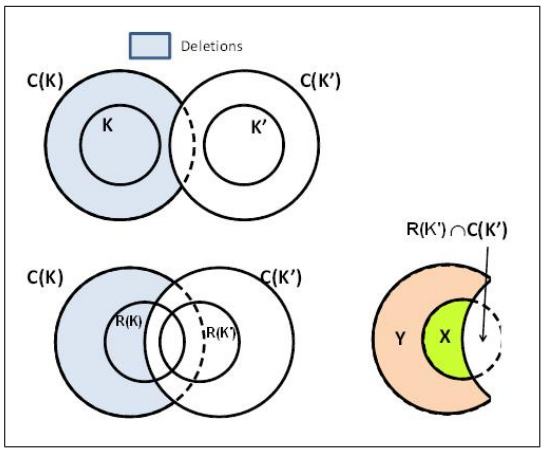


Fig. 18. On satisfaction of deletions in $\mathcal{U}_{ir}$ in streaming mode

Let's now focus on the lower case. Here W does not contain the entire $R(K)$ but a subset of it, call it X, i.e. $X = R(K) - C(K')$. Certainly the deletions in X will be satisfied. What we have to prove is that the rest deletions, i.e. the set Y (where $Y = C(K) - (C(K') \cup R(K))$), will be satisfied.

Assume that a relationship (ac) belongs to Y. Since it is an inferred relationship there are certainly at least two relationships, say (ab) and (bc) that belong to $R(K)$. We can distinguish the following cases:
1. If both belong to X, i.e. they belong to $R(K) - C(K')$, then $Del(ac)$ will be satisfied after the execution of one of $Del(ab)$ or $Del(bc)$, since the execution in $\mathcal{U}_{ir}$ applies the Reduction operation at the end.
2. If $(ab) \in X$ but $(bc) \in R(K) - X$, the deletion of (ab) will cause the deletion of (ac) from $C(K)$ so $Del(ac)$ will be satisfied (for the same reason)
3. If none belongs to X, i.e. if both belong to $R(K) - X = R(K) \cap C(K')$ then this means that $(ac) \in C(K')$ which contradicts our hypothesis that $(ac) \in Y$ because it holds $Y \cap C(K') = \emptyset$ since Y was defined as $Y = C(K) - (C(K') - R(K))$.

The proof for Δ_d is quite similar. It produces the following set of deletions: $Z = \{\ Del(t'') \mid t'' \in K - C(K')\}$. If K is redundancy free (i.e. $RF(K)$), then certainly all deletions are satisfied. If K is not redundancy free then it also contains some triples in Y, but in that case we have already proved that they will be also satisfied.

We conclude that all deletions will be satisfied in U_{ir} too.

PROP. 9. $\Delta_e^{\mathcal{U}_p}(K, K') \sim K'$

Proof:
$\Delta_e^{\mathcal{U}_p}(K, K') = (K - (K - K')) \cup (K' - K) \stackrel{(b)}{=} ((K' \cap K) \cup (K - K)) \cup (K' - K) =$
$(K' \cap K) \cup (K' - K) \stackrel{(b)}{=} K' - (K - K) = K'.\diamond$

PROP. 10. $\Delta_{ed}^{\mathcal{U}_p}(K, K') \sim K'$

Proof:
$\Delta_{ed}^{\mathcal{U}_p}(K, K') = (K-(K-C(K')))\cup(K'-K) \stackrel{(b)}{=} ((K-K)\cup(K\cap C(K'))\cup(K'-K)) =$
$(K \cap C(K')) \cup (K' - K) \stackrel{(c)}{=} K' \cup (K \cap C(K'))$

Let L=$K' \cup (K \cap C(K'))$ then

$K \cap C(K') \subseteq C(K') \Leftrightarrow K' \cup (K \cap C(K')) \subseteq K' \cup C(K') \Leftrightarrow K' \cup (K \cap C(K')) \subseteq C(K') \Leftrightarrow L \subseteq C(K')$

$\emptyset \subseteq K \cap C(K') \Leftrightarrow K' \subseteq K' \cup (K \cap C(K')) \Leftrightarrow K' \subseteq L$

From the above two inequations we get : $K' \subseteq L \subseteq C(K') \Leftrightarrow L \sim K'$. This means that $(\Delta_{ed}, \mathcal{U}_p)$ is correct.$\diamond$

PROP. 11. $\Delta_c^{\mathcal{U}_{ir}}(K, K') \sim K'$

Proof:
$\Delta_c^{\mathcal{U}_{ir}}(K, K') = R((C(K) - (C(K) - C(K'))) \cup (C(K') - C(K))) \stackrel{(b)}{=}$
$R(((C(K) \cap C(K')) \cup (C(K) - C(K))) \cup (C(K') - C(K))) = R((C(K) \cap C(K')) \cup$

$(C(K') - C(K))) \stackrel{(b)}{=} R(C(K') - (C(K) - C(K))) = R(C(K')) \sim K' \diamond$

PROP. 12. $\Delta_{dc}^{\mathcal{U}_{ir}}(K, K') \sim K'$

Proof:
$\Delta_{dc}^{\mathcal{U}_{ir}}(K, K') = R((C(K) - (C(K) - C(K'))) \cup (K' - C(K))) \stackrel{(b)}{=} R(((C(K) \cap C(K')) \cup (C(K) - C(K))) \cup (K' - C(K))) = R((C(K) \cap C(K')) \cup (K' - C(K))) \stackrel{(c)}{=} R((C(K) \cap C(K')) \cup K')$

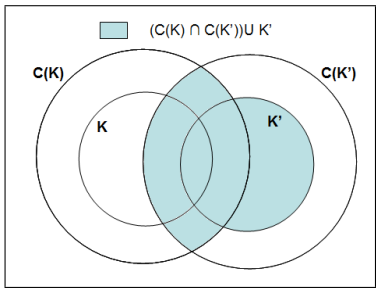


Fig. 19. On the correctness of $(\Delta_{dc}, \mathcal{U}_{ir})$

From Figure 19 we conclude that $K' \subseteq (C(K) \cap C(K')) \cup K' \subseteq C(K')$ this means that $R(K') \sim R((C(K) \cap C(K')) \cup K')$. From the definition of reduction we know that $R(K') \sim K'$ so $R((C(K) \cap C(K')) \cup K') \sim K' \diamond$

THEOREM 1 For any pair of knowledge bases K and K' it holds:

$$\Delta_{e}^{\mathcal{U}_{p}}(K, K') \sim \Delta_{ed}^{\mathcal{U}_{p}}(K, K') \sim$$
$$\Delta_{c}^{\mathcal{U}_{ir}}(K, K') \sim \Delta_{dc}^{\mathcal{U}_{ir}}(K, K') \sim K'$$

Proof:
The Propositions 9, 10, 11 and 12 prove the correctness of Theorem 1. $\diamond$

THEOREM 2 $\Delta_{d}^{\mathcal{U}_{ir}}(K, K') \sim K'$ iff $C(K) - K \subseteq C(K')$.

Proof:
$\Delta_{d}^{\mathcal{U}_{ir}}(K, K') = R((C(K) - (K - C(K'))) \cup (K' - C(K))) \stackrel{(b)}{=} R(((C(K) \cap C(K')) \cup (C(K) - K)) \cup (K' - C(K))) = R((C(K) \cap C(K')) \cup (K' - C(K))) \cup (C(K) - K)) \stackrel{(c)}{=} R(((C(K) \cap C(K')) \cup K') \cup (C(K) - K))$

It is obvious from the Figure 20 that $K' \subseteq ((C(K) \cap C(K')) \cup K') \cup (C(K) - K)$. In order to have $R(((C(K) \cap C(K')) \cup K') \cup (C(K) - K)) \sim K'$ then the following condition must hold $((C(K) \cap C(K')) \cup K') \cup (C(K) - K) \subseteq C(K')$. The last is true if $C(K) - K \subseteq C(K')$ (see Figure 20). $\diamond$

THEOREM 3 $\Delta_{c}^{\mathcal{U}_{p}}(K, K') \sim K'$ if K is closed.

Proof:

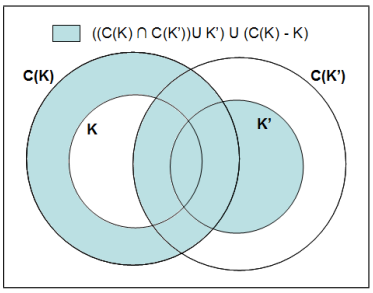


Fig. 20. On the correctness of $(\Delta_d, \mathcal{U}_{ir})$

$\Delta_c^{\mathcal{U}_p}(K, K') = (K - (C(K) - C(K'))) \cup (C(K') - C(K)) \stackrel{(b)}{=} ((K \cap C(K')) \cup (K - C(K))) \cup (C(K') - C(K)) = (K \cap C(K')) \cup (C(K') - C(K)) \stackrel{(b)}{=} C(K') - (C(K) - K)$

Obviously, if $C(K) - K = \emptyset$, then $\Delta_c^{\mathcal{U}_p}(K, K') = C(K')$ so $\Delta_c^{\mathcal{U}_p}(K, K') \sim K'$. The equality $C(K) - K = \emptyset$ can hold only if $C(K) = K$, i.e. if K is closed.

However we have to note that it can be $\Delta_c^{\mathcal{U}_p}(K, K') \sim K'$ even if K is not closed. For instance, Figure 21 shows the set $C(K') - (C(K) - K)$. If this set is equal to $C(K')$ then the pair $(\Delta_c, \mathcal{U}_p)$ is correct too. For instance, this is true if $C(K) \cap C(K') = \emptyset$.

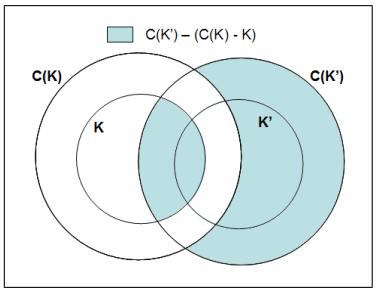


Fig. 21. On the correctness of Δ_c under $\mathcal{U}_p$

THEOREM 4 *If $\{K, K'\} \subseteq \Psi$ then for every delta produced by $\{\Delta_d, \Delta_{dc}, \Delta_c\}$ and $\{\Delta_{ed}, \Delta_e\}$ there exists at least one sequence of change operations that when executed under $\mathcal{U}_{ir}$ and $\mathcal{U}_p$ respectively the result is equivalent to the execution of the corresponding* set *of change operations. Alg. 1 produces such a sequence.*

Proof:

We have proved the correctness of Δ_{dc}, Δ_c and Δ_d in batch execution under $\mathcal{U}_{ir}$ and the correctness of Δ_e and Δ_{ed} under $\mathcal{U}_p$ (Th. 1 and Th. 2) and we know that all operations in Δ^+ and Δ^- will be satisfied and will produce the intended result. Obviously under $\mathcal{U}_p$ semantics the streaming execution will satisfy the same set of change operations. The proof of termination for the streaming execution mode (Prop 8) showed that the streaming execution under $\mathcal{U}_{ir}$ semantics will also satisfy the same set of operations. We have to show that only the operations in the delta will be satisfied. During the execution of the sequence of operations produced by Alg 1, only one operation (the one executed) will be satisfied at each step. This is

because the closure and reduction cannot satisfy more operations (i.e. they cannot add or remove anything from the closure). So only the operations in delta will be satisfied. This means that the streaming execution mode (of Δ_{dc}, Δ_c and Δ_d under $\mathcal{U}_{ir}$ and of Δ_e and Δ_{ed} under $\mathcal{U}_p$) is correct.

Recall that we have seen (e.g. at Figure 5) that we obtain an incorrect result only if the operations in the delta are either less or more than the right set of operations (for the corresponding semantics).