On Computing Deltas of RDF/S Knowledge Bases

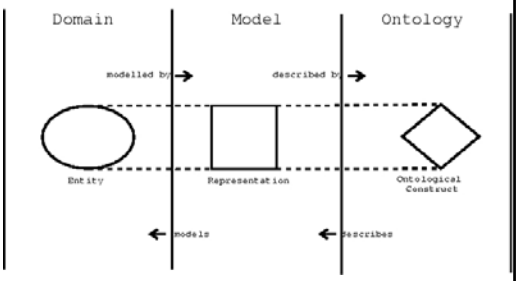
Dimitris Zeginis, Yannis Tzitzikas and Vassilis Christophides

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

FORTH - ICS

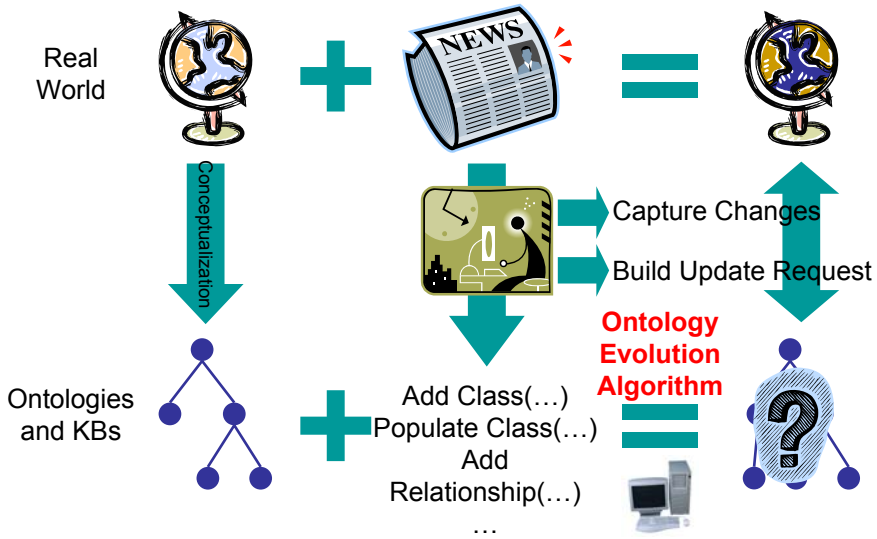
Ontology Evolution

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



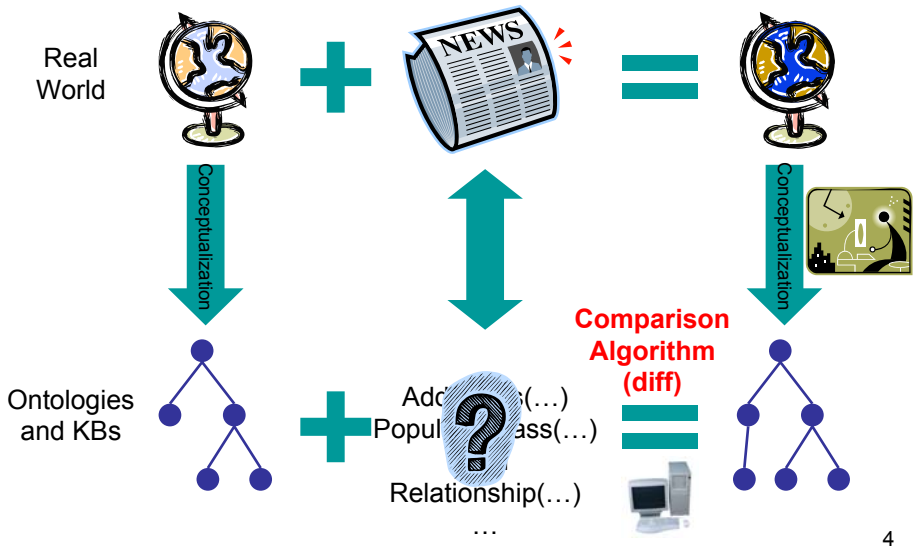
- Two setting in which changes occur:
  - **Controlled Evolution** (e.g. Ontology Editors)
  - **Open environments** (i.e. the Semantic Web)

What is Change?



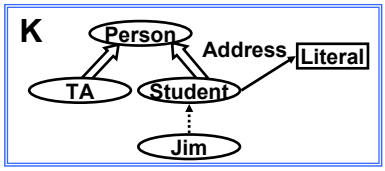
3

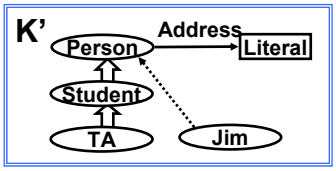
What is Diff?



4

RDF/S based Ontologies





RDF Knowledge Bases *K*, *K'*

{ (Person type Class),
(Student type Class),
(TA type Class),
(Student subClassOf Person),
(TA subClassOf Person),
(Address type Property)
(Address domain Student),
(Address range Literal),
(Jim type Student)
(Jim type Person)
}

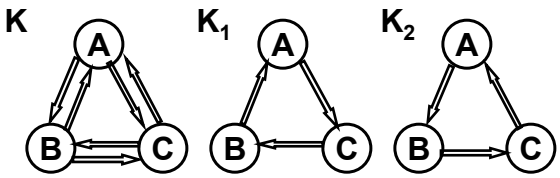
RDF KB Closures *C(K), C(K')*

{ (Person type Class),
(Student type Class),
(TA type Class),
(Student subClassOf Person),
(TA subClassOf Student),
(Address type Property),
(Address domain Person),
(Address range Literal),
(Jim type Person)
(TA subClassOf Person)
(Address domain Student),
(Address domain TA) }

- K and K' are **equivalent** (K ~ K') if C(K)=C(K')

5

Unique RDF/S KB Reductions



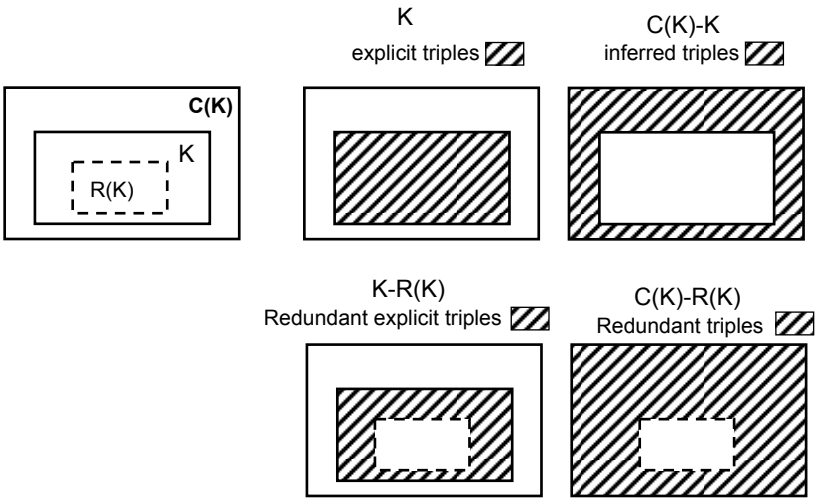
- K, K_1 and K_2 are equivalent
- K_1 and K_2 are redundancy free, but $K_1 \neq K_2$.

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

•*K* is **redundancy free KB RF(K))** if it does not contain explicit triples which can be inferred from *K*

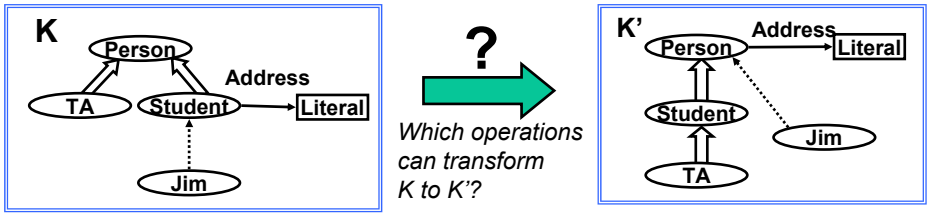
6

Distinctions of RDF/S Triple Sets



7

Comparing RDF/S KBs



•The output of a comparison function is a set of primitive change operations of the form ***Add**(t)*, ***Del**(t)* where *t* is a triple of K and K'

•We call such a set **Delta** Δ(K → K')

- Deltas can be exploited for providing **Versioning** and **Synchronization** services

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

8

Limits of Text-based Diffs

Why don't we use CVS as we do for text?

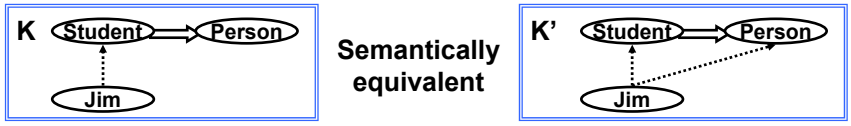
- RDF/S KBs have not a unique syntax

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

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

Syntactically equivalent

- RDF/S KBs (i.e. graphs) have not a unique serialization
- RDF/S KBs imply inferred facts w.r.t. semantics of RDFS



9

Outline

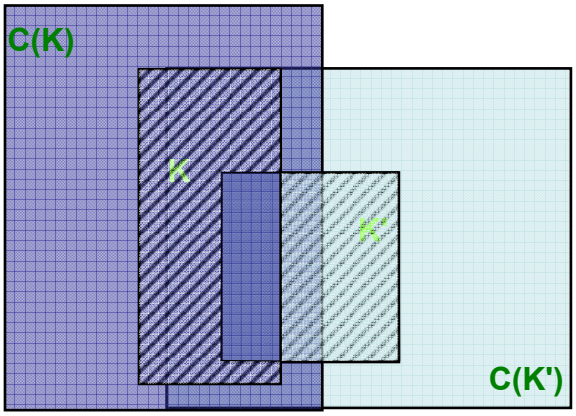
- Comparison functions
  - Size of Deltas
- Executing Deltas
  - Change Operation Semantics
  - Properties of RDF Deltas and Change Operation Semantics
- Synopsis of Results
- Related work
- Concluding remarks

10

ISWC '07

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

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

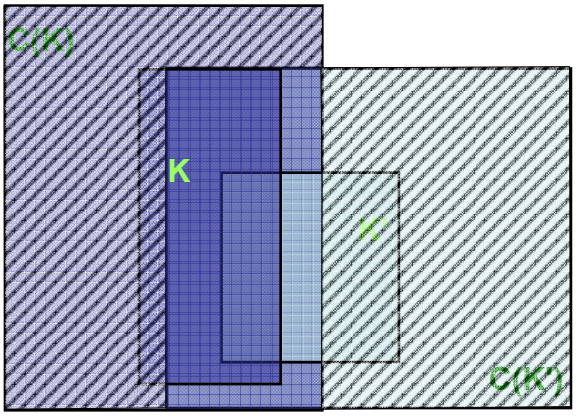


11

ISWC '07

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

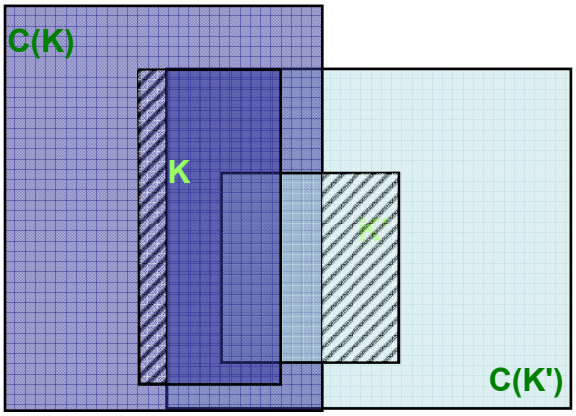
- **Delta Closure (Δ_c):** takes also into account inferred triples
  - $\Delta_c(K \rightarrow K') = \{Add(t) \mid t \in C(K') - C(K)\} \cup \{ Del(t) \mid t \in C(K) - C(K') \}$



12

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

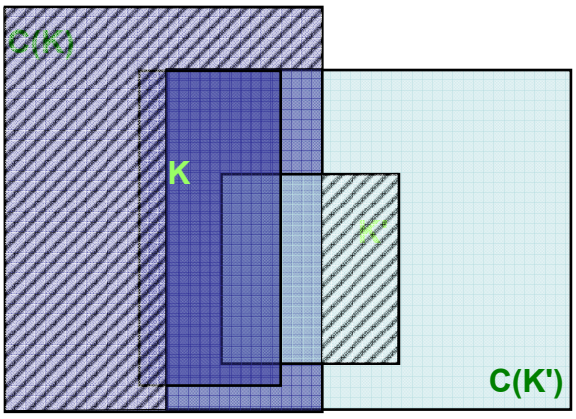
- **Delta Dense (Δ_d):** return the explicit triples of one KB that does not exist at the closure of the other KB
  - $\Delta_d(K \rightarrow K') = \{Add(t) \mid t \in K' - C(K)\} \cup \{ Del(t) \mid t \in K - C(K') \}$



13

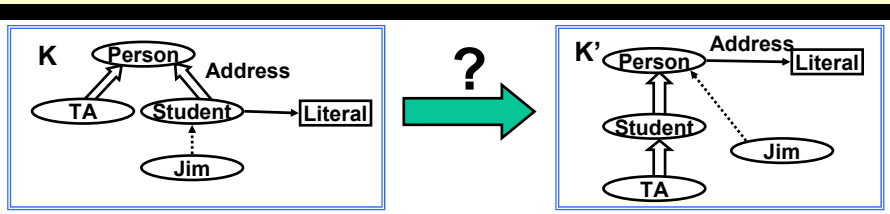
Comparison Function $\Delta_{dc}(K \rightarrow K')$

- **Delta Dense & Closure (Δ_{dc}):** resembles Δ_d regarding additions and Δ_c regarding deletions
  - $\Delta_{dc}(K \rightarrow K') = \{Add(t) \mid t \in K' - C(K)\} \cup \{ Del(t) \mid t \in C(K) - C(K') \}$



14

Comparison Functions: Example



$\Delta_e(K \rightarrow K')$ = { Del(TA subClassOf Person) ,Del(Address domain Student), Del(Jim type Student), Add(TA subClassOf Student), Add(Address domain Person), Add(Jim type Person) } — Size=6

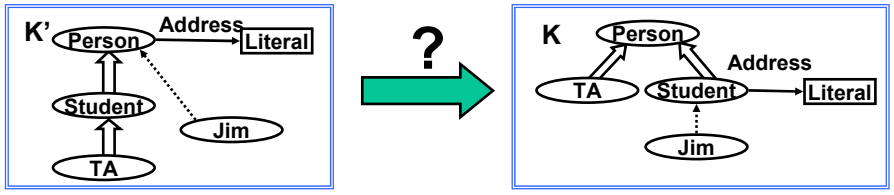
$\Delta_c(K \rightarrow K')$ ={ Del(Jim type Student), Add(TA subClassOf Student), Add(Address domain Person), Add(Address domain TA) } — Size=4

$\Delta_d(K \rightarrow K')$ = { Del(Jim type Student), Add(TA subClassOf Student), Add(Address domain Person) } — Size=3

$\Delta_{dc}(K \rightarrow K')$ = { Del(Jim type Student), Add(TA subClassOf Student), Add(Address domain Person) } — Size=3

15

Comparison Functions: Example



$\Delta_e(K' \rightarrow K)$ = {Del(TA subClassOf Student), Del(Address domain Person), Del(Jim type Person) Add(TA subClassOf Person), Add(Address domain Student), Add(Jim type Student)} — Size=6

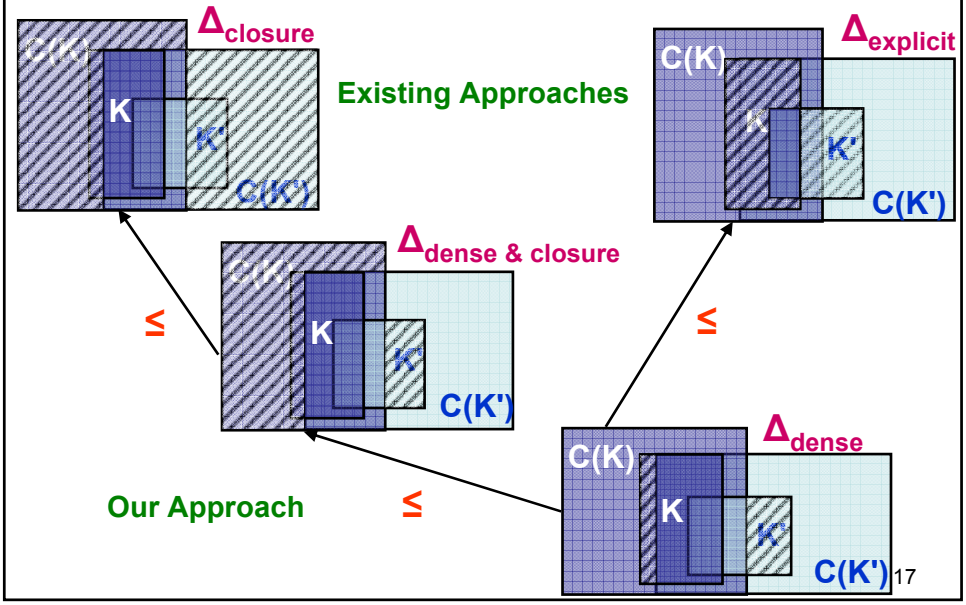
$\Delta_c(K' \rightarrow K)$ ={Del(TA subClassOf Student), Del(Address domain Person), Del(Address domain TA), Add(Jim type Student)} — Size=4

$\Delta_d(K' \rightarrow K)$ = {Del(TA subClassOf Student), Del(Address domain Person) Add(Jim type Student)} — Size=3

$\Delta_{dc}(K' \rightarrow K)$ = {Del(TA subClassOf Student), Del(Address domain Person), Del(Address domain TA), Add(Jim type Student)} — Size=4

16

Size of Deltas



17

On Executing Deltas

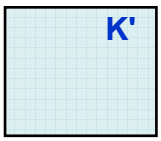
- ***We need to execute the Change operations returned by the Delta functions in order to transform one KB to another***
  - $\Delta_x(K \rightarrow K')(K)$?
  - $\Delta_x(K' \rightarrow K)(K')$?

  where x ∈ {e,c,d,dc}
- A KB *K* **satisfies** a **set** of `Add(t)` and `Del(t)` operations Δ_x iff:
  - Δ_x = { *`Add(t)`* } and $t \in C(K)$
  - Δ_x = { *`Del(t)`* } and $t \notin C(K)$
  - *K* satisfies every element of Δ_x
- **Two Important Questions**:
  - *Does the resulting, KB $\Delta_x(K \rightarrow K')(K)$, satisfy the Delta $\Delta_x(K \rightarrow K')$ for every x ∈ {e,c,d,dc}?*
  - *Does the order of execution of change operations matters?*

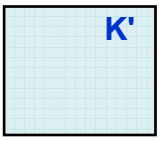
18

Change Operation Semantics

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



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

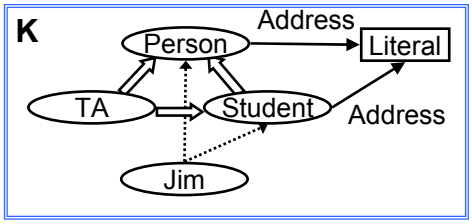


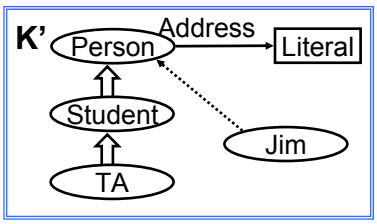
$\mathcal{V}_p$ – semantics

Oper.	Precond.	Postcond.
Add(t)	t ∈ K	K' = K
	t ∈ C(K) – K	K' = K U {t}
	t ∉ C(K)	K' = K U {t}

Oper.	Precond.	Postcond.
Del(t)	t ∈ K	K' = K – {t}
	t ∈ C(K) – K	K' = K
	t ∉ C(K)	K' = K

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



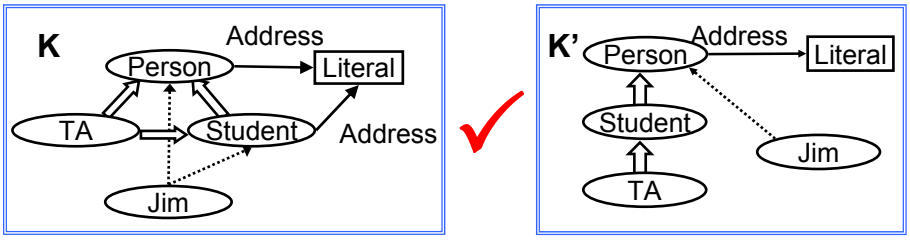


$\mathcal{V}_{ir}$ – semantics

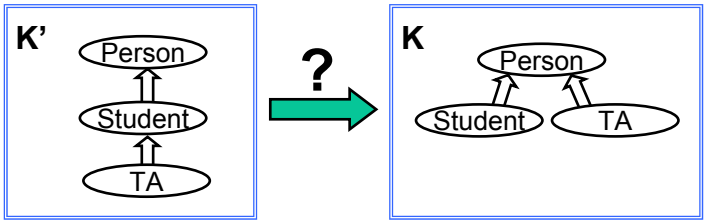
Oper.	Precond.	Postcond.
Add(t)	t∈ K	K' = K
	t∈ C(K) – K	K' = K
	t∉ C(K)	K' = R(K + {t})

Oper	Precond.	Postcond.
Del(t)	t∈ K	K' = R(C(K) – {t})
	t∈ C(K) – K	K' = K
	t∉ C(K)	K' = K

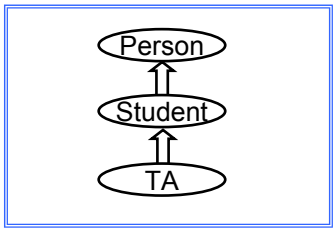
$\Delta_d(K \rightarrow K')$ ={ Del(Jim type Student), Add(TA subClassOf Student), Add(Address domain Person) }



Correctness of (Δ_c, $\mathcal{V}_p$)



$\Delta_c(K' \rightarrow K)$ ={ Del(TA subClassOf Student) }



The execution of Δ_c w.r.t. $\mathcal{V}_p$ is not always correct

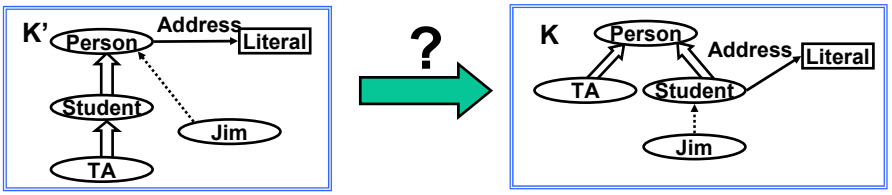
Properties of (Δ_x, $\mathcal{U}_y$) Pairs

- ***Which combinations of Change op semantics and Delta functions satisfy the following properties?***
  - **Correctness:** A pair (Δ_x, $\mathcal{U}_y$) is correct if for any K and K', the KB satisfying $\Delta_x(K \rightarrow K')$ (or $\Delta_x(K \rightarrow K')$) under $\mathcal{U}_y$ semantics is equivalent to K' (or K)
    - $\Delta_x(K \rightarrow K')^{\mathcal{U}_y}(K) \sim K'$
    - $\Delta_x(K' \rightarrow K)^{\mathcal{U}_y}(K') \sim K$

    where x ∈ {e,c,d,dc} and y ∈ {p, ir}
  - **Non Redundancy**: the execution of $\Delta_x(K \rightarrow K')$ upon K assuming $\mathcal{U}_y$ semantics results to a redundancy free KB
  - **Semantic Identity**: Delta is empty when its operands KBs are equivalent
    - If K ~ K' then $\Delta_x(K \rightarrow K') = \emptyset$

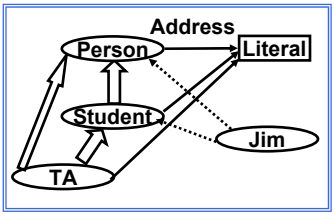
23

Correctness of (Δ_d, $\mathcal{U}_{ir}$)



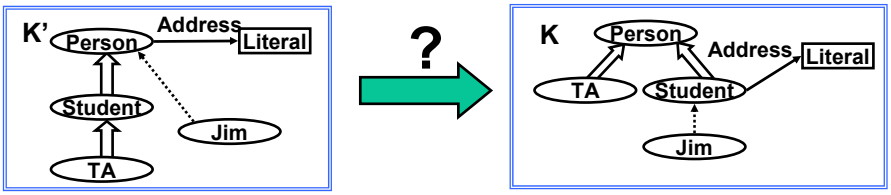
$\Delta_d(K' \rightarrow K)$ = {Del(TA subClassOf Student), Del(Address domain Person)
Add(Jim type Student)}

Precond.	Postcond.
t∈ K'	K = R(C(K') – {t})
t∈ C(K')–K'	K = K'
t∉ C(K')	K = K'



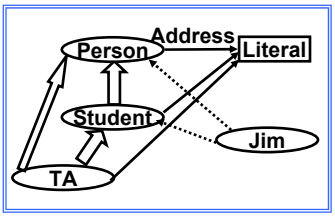
The execution of Δ_d w.r.t. $\mathcal{U}_{ir}$ is not always correct

Correctness of (Δ_{dc}, $\mathcal{U}_{ir}$)



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

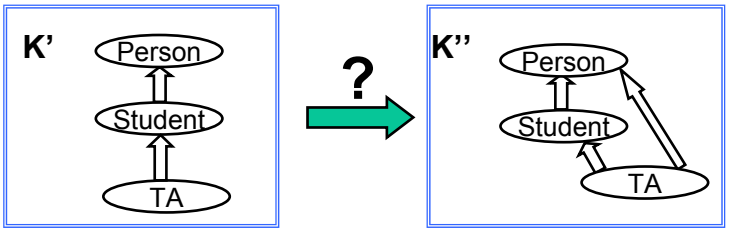
Precond.	Postcond.
$t \in K'$	$K = R(C(K') - \{t\})$
$t \in C(K') - K'$	$K = K'$
$t \notin C(K')$	$K = K'$



The execution of Δ_{dc} w.r.t. $\mathcal{U}_{ir}$ is correct

25

Semantic Identity for Δ_e



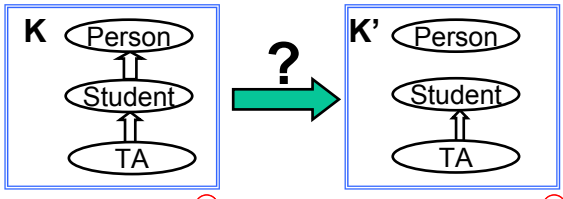
$\Delta_e(K' \rightarrow K'')$ = { Add(TA subClassOf Person) }

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

- **So Δ_e does not satisfy semantic identity**
  (although used by most existing systems)
- **However Δ_c, Δ_d and Δ_{dc} satisfy semantic identity**

26

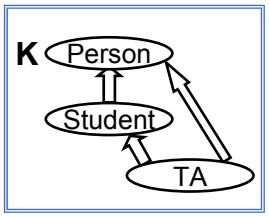
Order of Change Op Execution



② ②

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

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



$\mathcal{U}_{ir}$ – semantics

Order of execution affects the result for $\mathcal{U}_{ir}$

Multi-pass Execution Mode

- To avoid nondeterminism and to ensure correctness, a multi-pass execution mode is needed for:
  - Δ_c, Δ_d, Δ_{dc} under $\mathcal{U}_{ir}$ – semantics
- The algorithm repeats a loop until every change operation is satisfied

```
Execute (K, Δx) where x ∈ {c,d,dc}
Repeat
    Pick an element δ ∈ Δx such that is not satisfied by K
    K = δ^Uir (K) i.e. apply on K the appropriate post-
      conditions of δ w.r.t Uir-semantics
    // Note that K may still not satisfy δ
Until {δ | δ ∈ Δx and not satisfied by K} = Empty
```

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

Synopsis of Results

Pair	Correct	Exec Mode	Semantic ID	Non redund.
$(\Delta_e, \mathcal{U}_p)$	Y	S	Y if RF(K) & RF(K')	Y if RF(K')
$(\Delta_c, \mathcal{U}_p)$	Y if K compl.	S	Y	N
$(\Delta_d, \mathcal{U}_p)$	N	-	Y	N
$(\Delta_{dc}, \mathcal{U}_p)$	N	-	Y	N
$(\Delta_e, \mathcal{U}_{ir})$	N	-	Y if RF(K) & RF(K')	Y
$(\Delta_c, \mathcal{U}_{ir})$	Y in Ψ[1]	M	Y	Y
$(\Delta_d, \mathcal{U}_{ir})$	Y [2]	M	Y	Y
$(\Delta_{dc}, \mathcal{U}_{ir})$	Y in Ψ	M	Y	Y

[1] Knowledge Base with unique reduction

[2] If **(a)** K is complete or **(b)** $C(K) - K \subseteq C(K')$

Comparison With Related Work

System	Comparison Function	Semantics
CWM (w3c)	$\Delta_e(K \rightarrow K')$	$\mathcal{U}_p$
RDF-Utils	$\Delta_e(K \rightarrow K')$	$\mathcal{U}_p$
SemVersion	$\Delta_e(K \rightarrow K')$	$\mathcal{U}_p$
	$\Delta_c(K \rightarrow K')$	$\mathcal{U}_p$
PromptDiff	Heuristic matchers	-
Ontoview	Rules for changes	-
SWKM	$\Delta_e(K \rightarrow K')$, $\Delta_c(K \rightarrow K')$ $\Delta_d(K \rightarrow K')$, $\Delta_{dc}(K \rightarrow K')$	$\mathcal{U}_p$, $\mathcal{U}_{ir}$

Existing systems do not use Δ_d, Δ_{dc} and $\mathcal{U}_{ir}$

Delta Algorithms

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

Complexity of Delta Algorithms

$K_1 \rightarrow K_2$	(i) all stored	(ii) only K's stored	(iii) only K's stored and labeling
Δ_e	$O(N)$	$O(N)$	$O(N)$
Δ_c	$O(L)$	$O(N^2)$	$O(N^2)$
Δ_d	$O(N)$	$O(N^2)$	$O(N)$
Δ_{dc}	$O(\max(N_2, L_1))$	$O(N^2)$	$O(\max(N_2, N_1^2))$

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

Inference Strength and Delta Sizes

- The **inference strength** of a knowledge base K denoted by *is(K)* is defined as `is(K) = (|C(K)|-|K|)/|K|`
  - If `K=C(K)` then `is(K)=0`
- The less this factor is, the less the execution times and delta sizes of the four comparison functions differ
  - For Δ_e all kinds of changes affect in the same way the delta size
  - For Δ_c additions or deletions that occur highly at the subsumption hierarchy affect more the delta size
  - For Δ_d all kinds of changes affect in the same way the delta size
  - For Δ_{dc} all additions have the same impact on the result. But deletions that occur highly at the subsumption hierarchy affect more the size of the produced delta

33

Experiments Datasets

- Biological Data
  - Schema of the Gene ontology is very simple. It contains **only one metaclass** and all classes are instances of this metaclass
  - Uses many blank nodes (~ 50%)
- Synthetic Data set
  - We created a sequence of four KBs: K1, K2, K3, K4 with
    - 100, 200, 300, and 400 classes respectively
    - 300, 600, 900, 1200 properties respectively
  - For each class, 10 instances were created,
  - For each property, 10 instances were created
  - All classes, property and their instances in K_i are also present in K_{i+1}; their structuring may be different
  - The depth of the subclassOf hierarchy in each schema is 7

34

Real Dataset: GO Ontology

is(K) is very small in this data set

Time to compute deltas in seconds

Delta Size (in number of operations)

K_1			K_2			(T1) of $(\Delta(K_1 \rightarrow K_2))$				Size of $\Delta(K_1 \rightarrow K_2)$			
KBytes	Triples	$is(K)$	KBytes	Triples	$is(K)$	Δ_e	Δ_c	Δ_d	Δ_{dc}	Δ_e	Δ_c	Δ_d	Δ_{dc}
197	2898	0.229	201	2964	0.229	0.281	0.329	0.296	0.312	92	107	92	92
306	4493	0.225	331	4816	0.223	0.594	0.765	0.625	0.648	527	590	527	528
413	5994	0.218	418	6068	0.218	0.967	1.046	0.975	0.998	110	125	110	110
507	7340	0.205	512	7399	0.205	1.119	1.354	1.133	1.149	81	90	81	81
624	9023	0.196	638	9217	0.196	1.665	1.763	1.698	1.718	311	349	311	313
753	10806	0.182	757	12772	0.182	1.973	2.090	1.979	2.042	71	83	71	71
815	11680	0.178	822	11779	0.179	2.181	2.300	2.187	2.249	135	152	135	135

Table 6. Experimental Results for Biological Data

Inference strength

Inference strength

Small variations in delta sizes

Synthetic Dataset: RDFS Schemas

is(K) is higher in this data set

K_1			K_2			(T1) of $(\Delta(K_1 \rightarrow K_2))$				Size of $\Delta(K_1 \rightarrow K_2)$			
KBytes	Triples	$is(K)$	KBytes	Triples	$is(K)$	Δ_e	Δ_c	Δ_d	Δ_{dc}	Δ_e	Δ_c	Δ_d	Δ_{dc}
408	5162	0.821	919	10267	0.789	0.570	0.952	0.594	0.718	13999	22837	13864	16362
919	10267	0.789	1232	15389	0.806	0.889	1.499	1.187	1.388	23185	36449	23063	27553
1232	15389	0.806	1642	20460	0.826	1.296	2.217	1.623	1.702	32214	50415	32067	38941
408	5162	0.821	1642	20460	0.826	1.077	1.624	1.109	1.092	24004	41951	23937	26433

Table 7. Experimental Results for Synthetic Data Set

- Δ_c is 63% bigger than Δ_e
- Δ_{dc} is 16% bigger that Δe
  - **satisfies semantic identify**
- Δ_d is 1% smaller than Δe

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

Summary

- Four comparison functions were introduced and analyzed
  - Delta explicit (**Δ_e**), – Delta closure (**Δ_c**),
  - Delta dense (**Δ_d**) – Delta dense & closure (**Δ_{dc}**)
- Two change operation semantics were introduced and analyzed
  - One plain set theoretic $\mathcal{V}_p$
  - One that involves inference and redundancy elimination $\mathcal{V}_{ir}$
- We identified the always **correct** pairs: $(\Delta_e, \mathcal{V}_p)$, $(\Delta_{dc}, \mathcal{V}_{ir})$ $(\Delta_c, \mathcal{V}_{ir})$
- $(\Delta_{dc}; \mathcal{V}_{ir})$ is most promising:
  - Returns **empty result** if K, K' are semantically equivalent
  - Resulting KB is **redundancy free** when applying $\Delta_{dc}(K \rightarrow K')$ on K
  - Gives the **smallest result** if K' is an extension of K

37

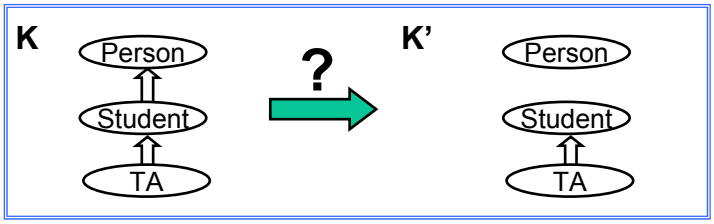
Thanks for your attention !



Any questions?

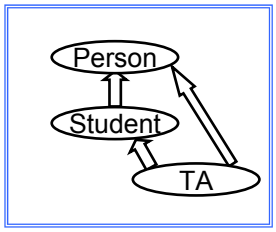
38

Correctness of (Δ_d, $\mathcal{U}_{ir}$)



$\Delta_d(K \rightarrow K') = \{$ Del(Student subClassOf Person) $\}$

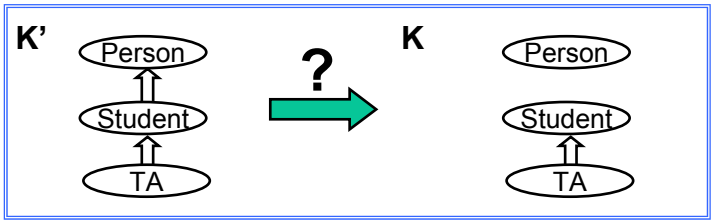
Precond.	Postcond.
$t \in K$	$K' = R(C(K) - \{t\})$
$t \in C(K) - K$	$K' = K$
$t \notin C(K)$	$K' = K$



The execution of Δ_d w.r.t. $\mathcal{U}_{ir}$ is not always correct

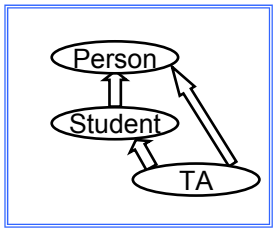
39

Correctness of (Δ_{dc}, $\mathcal{U}_{ir}$)



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

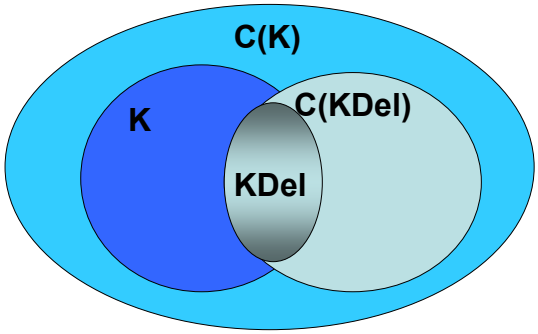
Precond.	Postcond.
$t \in K$	$K' = R(C(K) - \{t\})$
$t \in C(K) - K$	$K' = K$
$t \notin C(K)$	$K' = K$



The execution of Δ_{dc} w.r.t. $\mathcal{U}_{ir}$ is correct

40

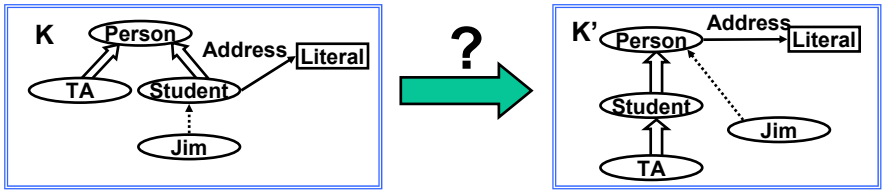
Note on Closures



- ∀K, K' K' ⊆ K => C(K') ⊆ C(K) and K ⊆ C(K) (monotonicity)
  - KDel ⊆ K, K-KDel ⊆ K, C(KDel) ⊆ C(K), KDel ⊆ C(KDel), KDel ⊆ C(K)
  - K – KDel ⊆ C(K) – C(KDel)
  - C(K) – C(KDel) ⊆ C(K) – KDel
  - K – KDel ⊆ C(K) – KDel
  - C(K – KDel) ⊆ C(K), C(K – KDel) ⊆ C(K) – KDel
  - C(K – KDel) ⊆ C(K) – C(KDel)
  - ⇒ K – KDel ⊆ C(K – KDel) ⊆ C(K) – C(KDel) ⊆ C(K) – KDel ⊆ C(K)

41

Comparison Functions: Example RDFS Standard



$\Delta_e(K \rightarrow K')$ = { Del(TA subClassOf Person) ,Del(Address domain Student), Del(Jim type Student), Add(TA subClassOf Student), Add(Address domain Person), Add(Jim type Person) } Size=6

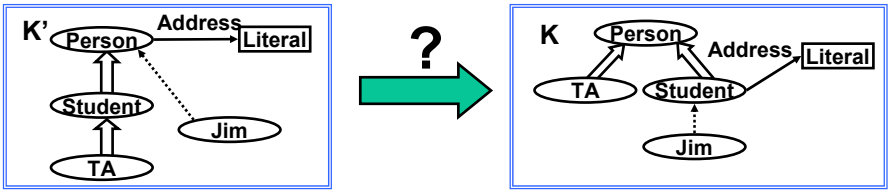
$\Delta_c(K \rightarrow K')$ ={ Del(Jim type Student), Del(Address domain Student), Add(TA subClassOf Student), Add(Address domain Person)} Size=4

$\Delta_d(K \rightarrow K')$ = { Del(Jim type Student), Del(Address domain Student), Add(TA subClassOf Student), Add(Address domain Person) } Size=4

$\Delta_{dc}(K \rightarrow K')$ = { Del(Jim type Student), Del(Address domain Student), Add(TA subClassOf Student), Add(Address domain Person) } Size=4

42

Comparison Functions: Example RDFS Standard



$\Delta_e(K' \rightarrow K)$ = {Del(TA subClassOf Student), Del(Address domain Person), Del(Jim type Person) Add(TA subClassOf Person), Add(Address domain Student), Add(Jim type Student)} — Size=6

$\Delta_c(K' \rightarrow K)$ ={Del(TA subClassOf Student), Del(Address domain Person), Add(Address domain Student), Add(Jim type Student)} — Size=4

$\Delta_d(K' \rightarrow K)$ = {Del(TA subClassOf Student), Del(Address domain Person), Add(Address domain Student), Add(Jim type Student)} — Size=4

$\Delta_{dc}(K' \rightarrow K)$ = {Del(TA subClassOf Student), Del(Address domain Person), Add(Address domain Student), Add(Jim type Student)} — Size=4

43