Ontology-Based Data Access Using Rewriting, OWL 2 RL Systems and Repairing

Giorgos Stoilos
gstoil@image.ntua.gr

National Technical University of Athens (NTUA), Greece

European Semantic Web Conference 2014 (ESWC 2014)

Motivation

- Query Answering over OWL 2 DL ontologies is very hard:
  - ExpTime wrt schema (TBox $\mathcal{T}$).
  - coNP-hard wrt data (ABox $\mathcal{A}$).
- Partial solution: Restrict expressivity
  - Triple-stores with some reasoning support: OWLim, Jena
  - But can miss answers if schema is (too) expressive.

Motivation

- Query Answering over OWL 2 DL ontologies is very hard:
  - ExpTime wrt schema (TBox $\mathcal{T}$).
  - coNP-hard wrt data (ABox $\mathcal{A}$).
- Partial solution: Restrict expressivity
  - Triple-stores with some reasoning support: OWLim, Jena
  - But can miss answers if schema is (too) expressive.

1. Many applications that need more expressivity.
2. Scalable QA over OWL 2 DL is attractive/beneficial.

Related Work

A promising solution/approach:

- We already have scalable data-reasoning systems.
- add “modules” that would increase their completeness possibly without affecting performance.

Related Work

A promising solution/approach:

- We already have scalable data-reasoning systems.
- add "modules" that would increase their completeness possibly without affecting performance.

ABox independent: OWL reasoner to materialise "missed" inferences as TBox axioms.

- DLDB [Pan 04], Minerva [Zhou 06], DLEJena [Meditskos 08], Repair [Stoilos et al. 11].

ABox dependent: Lower/upper approx. of answer; "intermediate" answers checked using an OWL reasoner.

- Ontosearch2 [Pan 07], ?? [Zhou 13a, 13b, 14]

Previous Work [Stoilos et al. 11]

A formal framework for ABox-independent completeness improvement of incomplete systems.

A repair $\mathcal{R}$ of $\mathcal{T}$ for a system (triple-store) ans

A set of axioms $\mathcal{R}$ such that

1. $\mathcal{T} \models \mathcal{R}$
2. For all datasets $\mathcal{A}$ and all SPARQL CQs $\mathcal{Q}$:

$$\text{ans}(\mathcal{Q}, \mathcal{T} \cup \mathcal{A} \cup \mathcal{R}) \supseteq \text{cert}(\mathcal{Q}, \mathcal{T} \cup \mathcal{A})$$

Previous Work [Stoilos et al. 11]

A formal framework for ABox-independent completeness improvement of incomplete systems.

A repair $\mathcal{R}$ of $\mathcal{T}$ for a system (triple-store) ans

A set of axioms $\mathcal{R}$ such that

1. $\mathcal{T} \models \mathcal{R}$
2. For all datasets $\mathcal{A}$ and all SPARQL CQs $\mathcal{Q}$:

$$\text{ans}(\mathcal{Q}, \mathcal{T} \cup \mathcal{A} \cup \mathcal{R}) \supseteq \text{cert}(\mathcal{Q}, \mathcal{T} \cup \mathcal{A})$$

- $\mathcal{R}$ computed once and it is per ontology.
- It works only for SPARQL CQs.
- It can contain complex axioms: $\exists R.C \sqsubseteq A,\ C \sqcap D \sqsubseteq E$ i.e., classifying $\mathcal{T}$ won't work (most previous approaches).

Previous Work [Stoilos et al. 11]

$$\mathcal{T}: \quad \begin{array}{rcl} \text{Student} & \sqsupseteq & \exists \text{takes}.\text{Course}, \\ \text{GradStudent} & \sqsubseteq & \exists \text{takes}.\text{GradC}, \\ \text{GradC} & \sqsubseteq & \text{Course} \end{array}$$

$$\mathcal{Q}: \quad Q(x) \leftarrow \text{Student}(x)$$

$\mathcal{A} = \{\text{GradStudent}(tom)\}$ we have $tom \in \text{cert}(\mathcal{Q}, \mathcal{T} \cup \mathcal{A})$

Previous Work [Stoilos et al. 11]

$$\mathcal{T}: \quad \begin{aligned} \text{Student} &\sqsupseteq \exists \text{takes}.\text{Course}, \\ \text{GradStudent} &\sqsubseteq \exists \text{takes}.\text{GradC}, \\ \text{GradC} &\sqsubseteq \text{Course} \end{aligned}$$

$$\mathcal{Q}: \quad Q(x) \leftarrow \text{Student}(x)$$

$\mathcal{A} = \{\text{GradStudent}(tom)\}$ we have $tom \in \text{cert}(\mathcal{Q}, \mathcal{T} \cup \mathcal{A})$

But for ans: Jena/OWLim we have $tom \notin \text{ans}(\mathcal{Q}, \mathcal{T} \cup \mathcal{A})$.

Previous Work [Stoilos et al. 11]

$$\mathcal{T}: \quad \begin{array}{rcl} \text{Student} & \sqsupseteq & \exists \text{takes}.\text{Course}, \\ \text{GradStudent} & \sqsubseteq & \exists \text{takes}.\text{GradC}, \\ \text{GradC} & \sqsubseteq & \text{Course} \end{array}$$

$$\mathcal{Q}: \quad Q(x) \leftarrow \text{Student}(x)$$

$\mathcal{A} = \{\text{GradStudent}(tom)\}$ we have $tom \in \text{cert}(\mathcal{Q}, \mathcal{T} \cup \mathcal{A})$

But for ans: Jena/OWLim we have $tom \notin \text{ans}(\mathcal{Q}, \mathcal{T} \cup \mathcal{A})$.

$\mathcal{R} = \{\text{GradStudent} \sqsubseteq \text{Student}\}$ we have $tom \in \text{ans}(\mathcal{Q}, \mathcal{T} \cup \mathcal{R} \cup \mathcal{A})$.

$\mathcal{R}$ explicates the inference that ans would miss.

Previous Work [Stoilos et al. 11]

Existence: A repair exists when a Datalog-rewriting for $\mathcal{T}$ exists.

D-Rewriting: set of rules that capture from $\mathcal{T}$ all information relevant for answering every ground CQ [Motik 05a, 05b] (KAON2).

Previous Work [Stoilos et al. 11]

Existence: A repair exists when a Datalog-rewriting for $\mathcal{T}$ exists.

D-Rewriting: set of rules that capture from $\mathcal{T}$ all information relevant for answering every ground CQ [Motik 05a, 05b] (KAON2).

Running TBox	D-Rewriting
$\text{Student} \sqsupseteq \exists \text{takes}.\text{Course}$	$\text{takes}(x, y) \wedge \text{Course}(x) \rightarrow \text{Student}(x)$
$\text{GradStudent} \sqsubseteq \exists \text{takes}.\text{GradC}$	$\text{GradStudent}(x) \rightarrow \text{Student}(x)$
$\text{GradC} \sqsubseteq \text{Course}$	$\text{GradC}(x) \rightarrow \text{Course}(x)$

Previous Work [Stoilos et al. 11]

Existence: A repair exists when a Datalog-rewriting for $\mathcal{T}$ exists.

D-Rewriting: set of rules that capture from $\mathcal{T}$ all information relevant for answering every ground CQ [Motik 05a, 05b] (KAON2).

Running TBox	D-Rewriting
$\text{Student} \sqsupseteq \exists \text{takes}.\text{Course}$	$\text{takes}(x, y) \wedge \text{Course}(x) \rightarrow \text{Student}(x)$
$\text{GradStudent} \sqsubseteq \exists \text{takes}.\text{GradC}$	$\text{GradStudent}(x) \rightarrow \text{Student}(x)$
$\text{GradC} \sqsubseteq \text{Course}$	$\text{GradC}(x) \rightarrow \text{Course}(x)$

Previous Work [Stoilos et al. 11]

Existence: A repair exists when a Datalog-rewriting for $\mathcal{T}$ exists.

D-Rewriting: set of rules that capture from $\mathcal{T}$ all information relevant for answering every ground CQ [Motik 05a, 05b] (KAON2).

Running TBox	D-Rewriting
$\text{Student} \sqsupseteq \exists \text{takes}.\text{Course}$	$\text{takes}(x, y) \wedge \text{Course}(x) \rightarrow \text{Student}(x)$
$\text{GradStudent} \sqsubseteq \exists \text{takes}.\text{GradC}$	$\text{GradStudent}(x) \rightarrow \text{Student}(x)$
$\text{GradC} \sqsubseteq \text{Course}$	$\text{GradC}(x) \rightarrow \text{Course}(x)$

- D-rewritings exist if $\mathcal{T}$ is Horn-$\mathcal{SHIQ}$ [Eiter et al. 12].
- Can exist even if $\mathcal{T}$ is OWL 2 DL [Cuenca Grau 13].

  Weakly-linear/markeable TBoxes [Cuenca Grau 14a, 14b].

Previous Work [Stoilos et al. 11]

Algorithm: Repair($\mathcal{T}$, ans)

1. Compute a rewriting of $\mathcal{T}$ and transform it into an initial repair $\mathcal{R}^1$.

$$\text{takes}(x, y) \wedge \text{Course}(x) \rightarrow \text{Student}(x)$$
$$\text{GradStudent}(x) \rightarrow \text{Student}(x)$$
$$\text{GradC}(x) \rightarrow \text{Course}(x)$$

$$\Downarrow \text{Step 1}$$

$$\exists \text{takes}.\text{Course} \sqsubseteq \text{Student}$$
$$\text{GradStudent} \sqsubseteq \text{Student}$$
$$\text{GradC} \sqsubseteq \text{Course}$$

Previous Work [Stoilos et al. 11]

Algorithm: Repair($\mathcal{T}$, ans)

1. Compute a rewriting of $\mathcal{T}$ and transform it into an initial repair $\mathcal{R}^1$.
2. Minimise $\mathcal{R}^1$: remove redundant axioms using an OWL 2 DL reasoner.
   - (A) $\mathcal{R}^1 := \mathcal{R}^1 \setminus \{\alpha\}$ if $\mathcal{T}|_{\mathsf{rl}} \models \alpha$.
   - (B) $\mathcal{R}^1 := \mathcal{R}^1 \setminus \{\alpha_2\}$ if $\alpha_1 \in \mathcal{R}^1$ exists s.t. $\mathcal{T}|_{\mathsf{rl}} \cup \{\alpha_1\} \models \alpha_2$.

$$\mathsf{takes}(x, y) \wedge \mathsf{Course}(x) \rightarrow \mathsf{Student}(x)$$
$$\mathsf{GradStudent}(x) \rightarrow \mathsf{Student}(x)$$
$$\mathsf{GradC}(x) \rightarrow \mathsf{Course}(x)$$

$$\Downarrow \text{ Step 1}$$

~~$\exists$takes.Course $\sqsubseteq$ Student~~ captured by ans
$\mathsf{GradStudent} \sqsubseteq \mathsf{Student}$
~~$\mathsf{GradC} \sqsubseteq \mathsf{Course}$~~ captured by ans

Open Problems — Goals

Repairing is a promissing approach; but first approach had limitations:

- Evaluation too preliminary: LUBM, tiny version of Galen.
- Approach had no optimisations.

  Not clear whether we can handle large/complex TBoxes.
- No support for non-SPARQL CQs.

Goals/Contributions:

1. Investigate further on practicality of approach:
   - I Propose optimisations
   - II Perform extensive evaluation with large/complex ontos.
2. Provide support for non-SPARQL queries.
   - I Evaluate as well.

Improvements to First Approach – Step 1

1. Compute a rewriting of $\mathcal{T}$ and transform it into an initial repair $\mathcal{R}^1$.

Improvements to First Approach – Step 1

1. Compute a rewriting of $\mathcal{T}$ and transform it into an initial repair $\mathcal{R}^1$.

   - Condition $\mathcal{T} \models \mathcal{R}$ of definition too strict.
   - Prohibits the use of rewriting systems which *normalise* the input TBox: Rapid, Presto, Blackout, Ontop, ...

     $A \sqsubseteq \exists R.(C \sqcap D) \in \mathcal{T} \rightsquigarrow \{A \sqsubseteq \exists R.A_{new},\ A_{new} \sqsubseteq C \sqcap D\} \subseteq \text{norm}(T)$

     Rewriting: $A_{new}(x) \rightarrow C(x) \quad \rightsquigarrow \quad A_{new} \sqsubseteq C \in \text{REPAIR}(\mathcal{T}, \text{ans})$
   - Normalisation: performance & compactness: [Rosati 10, Kikot 12]

Improvements to First Approach – Step 1

1. Compute a rewriting of $\mathcal{T}$ and transform it into an initial repair $\mathcal{R}^1$.

   - Condition $\mathcal{T} \models \mathcal{R}$ of definition too strict.
   - Prohibits the use of rewriting systems which *normalise* the input TBox: Rapid, Presto, Blackout, Ontop, ...

     $A \sqsubseteq \exists R.(C \sqcap D) \in \mathcal{T} \rightsquigarrow \{A \sqsubseteq \exists R.A_{new},\ A_{new} \sqsubseteq C \sqcap D\} \subseteq \text{norm}(T)$

     Rewriting: $A_{new}(x) \rightarrow C(x) \quad \rightsquigarrow \quad A_{new} \sqsubseteq C \in \text{Repair}(\mathcal{T}, \text{ans})$
   - Normalisation: performance & compactness: [Rosati 10, Kikot 12]

Proposition

$\mathcal{T}' = \text{norm}(\mathcal{T}) \quad \mathcal{R}' = \text{Repair}(\mathcal{T}', \text{ans}) \quad \mathcal{T}' \models \mathcal{R}'$

$$\text{ans}(\mathcal{Q}, \mathcal{T}' \cup \mathcal{R}' \cup \mathcal{A}) \supseteq \text{cert}(\mathcal{Q}, \mathcal{T} \cup \mathcal{A}), \forall \mathcal{A} \text{ \& SPARQL } \mathcal{Q}$$

Improvements to First Approach – Step 2

2. Minimise $\mathcal{R}^1$:
   - (A) $\mathcal{R}^1 := \mathcal{R}^1 \setminus \{\alpha\}$ if $\mathcal{T}|_{\mathsf{rl}} \models \alpha$.
   - (B) $\mathcal{R}^1 := \mathcal{R}^1 \setminus \{\alpha_2\}$ if $\alpha_1 \in \mathcal{R}^1$ exists s.t. $\mathcal{T}|_{\mathsf{rl}} \cup \{\alpha_1\} \models \alpha_2$.

Problem:

- $\mathcal{R}^1$ can be exponentially larger than $\mathcal{T}$.
- Step 2(B): quadratic number of calls to an OWL 2 DL reasoner.

Improvements to First Approach – Step 2

2. Minimise $\mathcal{R}^1$:
   - (A) $\mathcal{R}^1 := \mathcal{R}^1 \setminus \{\alpha\}$ if $\mathcal{T}|_{\mathsf{rl}} \models \alpha$.
   - (B) $\mathcal{R}^1 := \mathcal{R}^1 \setminus \{\alpha_2\}$ if $\alpha_1 \in \mathcal{R}^1$ exists s.t. $\mathcal{T}|_{\mathsf{rl}} \cup \{\alpha_1\} \models \alpha_2$.

Problem:

- $\mathcal{R}^1$ can be exponentially larger than $\mathcal{T}$.
- Step 2(B): quadratic number of calls to an OWL 2 DL reasoner.

Solution:

- Most parameters are fixed ($\mathcal{T}|_{\mathsf{rl}}$) or rarely changing ($\{\alpha_1\}$).
- Apply calculus of the OWL 2 DL reasoner exhaustively on $\mathcal{T}|_{\mathsf{rl}}$ and store the computation.
- $\mathcal{T}|_{\mathsf{rl}} \models \alpha$: resume computation from previous point and backtrack.

SUPPORTING NON-SPARQL CQS

CQs with existential quantifiers: $\mathcal{Q} = Q(x) \leftarrow \exists y.\text{takesCourse}(x, y)$

SUPPORTING NON-SPARQL CQS

CQs with existential quantifiers: $\mathcal{Q} = Q(x) \leftarrow \exists y.\text{takesCourse}(x, y)$

- Repairing explicates gr-entailments using a rewriting for $\mathcal{T}$.
- Also explicate non-ground entailments related to $\mathcal{Q}$.

SUPPORTING NON-SPARQL CQS

CQs with existential quantifiers: $\mathcal{Q} = Q(x) \leftarrow \exists y.\text{takesCourse}(x, y)$

- Repairing explicates gr-entailments using a rewriting for $\mathcal{T}$.
- Also explicate non-ground entailments related to $\mathcal{Q}$.
- Use notion of a rewriting Rew for $\mathcal{T}, \mathcal{Q}$ [Calvanese 07, Artale 09].

  All information relevant for answering $\mathcal{Q}$ over $\mathcal{T}$ and any $\mathcal{A}$

  $\text{Rew} = \text{Rew}_D \uplus \text{Rew}_Q$

  $\text{Rew}_Q = \{Q(x) \leftarrow R(x, y), \ldots\}$ captures non-gr. entailments.

SUPPORTING NON-SPARQL CQS

CQs with existential quantifiers: $\mathcal{Q} = Q(x) \leftarrow \exists y.\text{takesCourse}(x, y)$

- Repairing explicates gr-entailments using a rewriting for $\mathcal{T}$.
- Also explicate non-ground entailments related to $\mathcal{Q}$.
- Use notion of a rewriting Rew for $\mathcal{T}, \mathcal{Q}$ [Calvanese 07, Artale 09].

  All information relevant for answering $\mathcal{Q}$ over $\mathcal{T}$ and any $\mathcal{A}$

  $\text{Rew} = \text{Rew}_D \uplus \text{Rew}_Q$

  $\text{Rew}_Q = \{Q(x) \leftarrow R(x, y), \ldots\}$ captures non-gr. entailments.

PROPOSITION

$\text{Rew} = \text{Rew}_D \uplus \text{Rew}_Q$ *rewriting for* $\mathcal{T}$ *and* $\mathcal{Q}$.

For every $\mathcal{A}$: $\text{ans}(\text{Rew}_Q, \mathcal{T} \cup \mathcal{R} \cup \mathcal{A}) \supseteq \text{cert}(\mathcal{Q}, \mathcal{T} \cup \mathcal{A})$

EXAMPLE

$$
\begin{array}{rrcl}
 & \text{Student} & \sqsupseteq & \exists \text{takes}.\text{Course}, \\
\mathcal{T}: & \text{GradStudent} & \sqsubseteq & \exists \text{takes}.\text{GradC}, \\
 & \text{GradC} & \sqsubseteq & \text{Course} \\
 & & & \\
\mathcal{Q}: & Q(x) \leftarrow \exists y.\text{takes}(x, y) & &
\end{array}
$$

$\mathcal{A} = \{\text{GradStudent}(\mathit{tom})\}$ we have $\mathit{tom} \in \text{cert}(\mathcal{Q}, \mathcal{T} \cup \mathcal{A})$

EXAMPLE

$$
\begin{array}{rrcl}
 & \text{Student} & \sqsupseteq & \exists \text{takes}.\text{Course}, \\
\mathcal{T}: & \text{GradStudent} & \sqsubseteq & \exists \text{takes}.\text{GradC}, \\
 & \text{GradC} & \sqsubseteq & \text{Course} \\
 & & & \\
\mathcal{Q}: & Q(x) \leftarrow \exists y.\text{takes}(x, y) & &
\end{array}
$$

$\mathcal{A} = \{\text{GradStudent}(\mathit{tom})\}$ we have $\mathit{tom} \in \text{cert}(\mathcal{Q}, \mathcal{T} \cup \mathcal{A})$

For most triple stores we have $\mathit{tom} \notin \text{ans}(\mathcal{Q}, \mathcal{T} \cup \mathcal{A})$

Example

$$\mathcal{T}: \quad \begin{array}{rcl} \text{Student} & \sqsupseteq & \exists \text{takes}.\text{Course}, \\ \text{GradStudent} & \sqsubseteq & \exists \text{takes}.\text{GradC}, \\ \text{GradC} & \sqsubseteq & \text{Course} \end{array}$$

$$\mathcal{Q}: \quad Q(x) \leftarrow \exists y.\text{takes}(x, y)$$

$\mathcal{A} = \{\text{GradStudent}(\mathit{tom})\}$ we have $\mathit{tom} \in \text{cert}(\mathcal{Q}, \mathcal{T} \cup \mathcal{A})$

For most triple stores we have $\mathit{tom} \notin \text{ans}(\mathcal{Q}, \mathcal{T} \cup \mathcal{A})$

$\text{Rew}_Q = \{Q(x) \leftarrow \text{takes}(x, y),\ Q(x) \leftarrow \text{GradStudent}(x)\ \}$

we have $\mathit{tom} \in \text{ans}(\text{Rew}_Q, \mathcal{T} \cup \mathcal{R} \cup \mathcal{A})$

Algorithm and Implementation

1. Compute a repair $\mathcal{R}$ of $\mathcal{T}$ for ans using procedure Repair.
2. Load the dataset $\mathcal{A}$, the input TBox $\mathcal{T}$, and the repair $\mathcal{R}$ to ans.
3. For a query $\mathcal{Q}$
   (A) If $\mathcal{Q}$ is SPARQL, then evaluate $\mathcal{Q}$ using ans; otherwise
   (B) compute a $(\mathcal{Q}, \mathcal{T})$-rewriting $\text{Rew}_D \uplus \text{Rew}_Q$ using any rewriting system and evaluate Rew_Q using ans.

Algorithm and Implementation

1. Compute a repair $\mathcal{R}$ of $\mathcal{T}$ for ans using procedure Repair.
2. Load the dataset $\mathcal{A}$, the input TBox $\mathcal{T}$, and the repair $\mathcal{R}$ to ans.
3. For a query $\mathcal{Q}$
   - (A) If $\mathcal{Q}$ is SPARQL, then evaluate $\mathcal{Q}$ using ans; otherwise
   - (B) compute a $(\mathcal{Q}, \mathcal{T})$-rewriting $\text{Rew}_D \uplus \text{Rew}_Q$ using any rewriting system and evaluate Rew_Q using ans.

Hydrowl: A Hybrid Query Answering System for OWL Ontologies

`http://www.image.ece.ntua.gr/~gstoil/hydrowl/`

- Rapid: to compute rewritings (Steps 1 & 3(B)).
- HermiT: to minimise repairs (Step 1).
- OWLim as a triple-store (Steps 2 & 3).

Evaluation – Setup

- We wanted to evaluate all 3 steps
  - How efficiently repairs can be computed (Step 1).
  - How much they affect loading (Step 2).
  - Query answering of non-SPARQL CQs (Step 3(b)).
- Ontologies (TBoxes): 152

  GO, 3 versions of Galen, Fly_Anatomy, UOBM.
- Dataset (ABoxes):

  UOBM has a data-generator (1, 2, 5, 10, 20 unis)

  Fly_Anatomy: $\sim$6000 assertions that we multiplied up to 5 times.
- non-SPARQL CQs:

  Fly_Anatomy comes with 4 realistic non-SPARQL CQs.

Evaluation – Computing Repairs

- Computability

  Computed repairs for 151 out of 152 ontologies.

  146 using old (non-norm. approach), but 6 required it.

Evaluation – Computing Repairs

- Computability

  Computed repairs for 151 out of 152 ontologies.

  146 using old (non-norm. approach), but 6 required it.

- Efficiency

  Vast majority computed within a few seconds.

  4 required a few minutes.

  4 out of the 6 “hard” required up to 1/2h

Evaluation – Computing Repairs

- Computability

  Computed repairs for 151 out of 152 ontologies.

  146 using old (non-norm. approach), but 6 required it.

- Efficiency

  Vast majority computed within a few seconds.

  4 required a few minutes.

  4 out of the 6 “hard” required up to 1/2h

- Size of Computed repairs

  Many ontologies had a repair of size 0 (nothing to encode)

  12 ontologies had a repair of size $<$15 axioms

  The 4 “hard” ones had sizes $>$3000 axioms

Evaluation – Computing Repairs

$\mathcal{T}$	$\lvert\mathcal{T}\rvert$	$\lvert\mathcal{R}\rvert$	t
Not-Galen	5471	3015 (4153)	298 (42)
Galen-doc	4229	6051 (6176)	1152 (28)
Galen	4229	3012 (3062)	257 (24)
Fly	19845	10361 (12368)	2884 (178)

- Repairs are large but not exponentially larger.
- Computation time is reasonable.

  Without opts for Step 2 it is several hours.
- Numbers in parenthesis are if we completely dispense with minimisation Step 2(B) (quadratic loop).

Evaluation – Loading Repairs

	1	2	5	10	20
UOBM	4.1	6.8	16.2	31.9	73.2
UOBM$\cup\mathcal{R}$	4.4	8.3	24.3	44.9	108.1

(a) UOBM

	$\mathcal{A}_1$	$\mathcal{A}_2$	$\mathcal{A}_3$	$\mathcal{A}_4$	$\mathcal{A}_5$
Fly	14.0	21.9	22.7	27.9	31.5
Fly$\cup\mathcal{R}$	31.9	55.1	68.5	93.0	119.3
Fly$\cup\mathcal{R}^-$	33.2	62.1	70.1	100.6	118.2

(b) Fly

- Effects of repairing are mostly noticeable in large and complex ontologies.

  But still overhead is not unmanageable.

  Recall that loading is usually done once.

- Non-minimal repairs ($\mathcal{R}^-$) not much worse than minimal ($\mathcal{R}$).

Evaluation – non-SPARQL CQs

t_{rew}: time for Rapid to compute the $(\mathcal{Q}, \mathcal{T})$-rewriting.

t_{ans}: time for OWLim to evaluate Rew_Q.

$\mathcal{Q}_1$			$\mathcal{Q}_2$		
$\lvert\text{Rew}_Q\rvert$	t_{rew}	t_{ans}	$\lvert\text{Rew}_Q\rvert$	t_{rew}	t_{ans}
64	0.31	0.31	2880	0.90	1.28
$\mathcal{Q}_3$			$\mathcal{Q}_4$		
$\lvert\text{Rew}_Q\rvert$	t_{rew}	t_{ans}	$\lvert\text{Rew}_Q\rvert$	t_{rew}	t_{ans}
91	0.07	0.04	6	0.05	0.02

- All CQs were answered almost instantaneously

  Size/structure of Rew_Q small/simple.

  Much of the reasoning work has been done during loading.

Summary & Additional Results

- Repairing is a highly promissing approach to QA

  Repairs can be computed for large ontos in reasonable time.

  Their size is usually small or manageable (not exp. larger).

  Loading is not affected significantly (no exp. increase).

  Evaluated non-SPARQL CQs for an expressive ontology instantaneously.

Summary & Additional Results

- Repairing is a highly promissing approach to QA

  Repairs can be computed for large ontos in reasonable time.

  Their size is usually small or manageable (not exp. larger).

  Loading is not affected significantly (no exp. increase).

  Evaluated non-SPARQL CQs for an expressive ontology instantaneously.

- What happens when a repair does not exist, it is very large, or very difficult to compute?

  Partial repair: captures some of the missed inferences.

  For a CQ $\mathcal{Q}$: decide if partially repaired triple-store is adequate or an OWL 2 DL reasoner is required.

  Implemented in Hydrowl presented at ECAI [Stoilos 14]