TECHNISCHE UNIVERSITÄT DRESDEN

Faculty of Computer Science ● Institute of Theoretical Computer Science ● Chair of Automata Theory

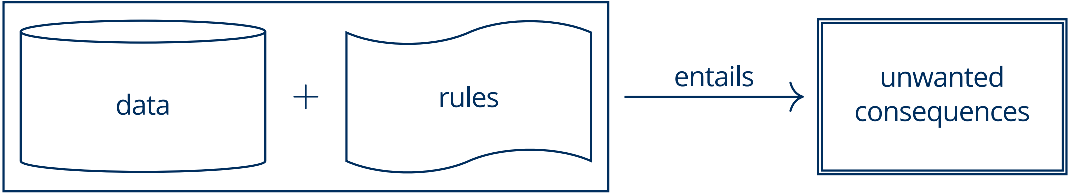
COMPUTING OPTIMAL REPAIRS OF QUANTIFIED ABOXES W.R.T. STATIC $\mathcal{EL}$ TBOXES

Franz Baader, Patrick Koopmann,
Francesco Kriegel, Adrian Nuradiansyah

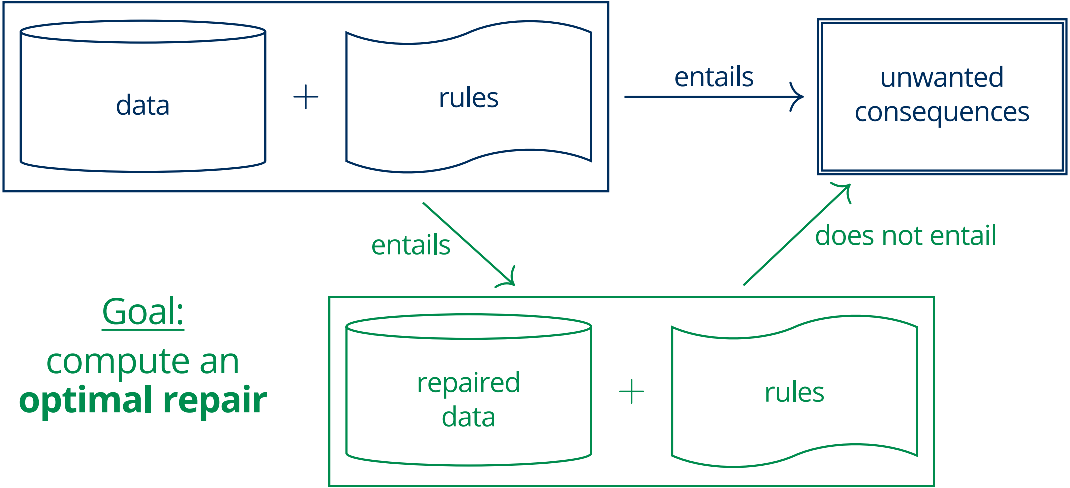
Funded by DFG in project number 430150274 and in TRR 248 (cpec, grant 389792660).

The 28th International Conference on Automated Deduction (CADE-28), 13 July 2021

Problem Setting



Problem Setting



Goal:

compute an **optimal repair**

A Pizza with Parmesan and Salami

- **Data:** quantified ABox (qABox) $\exists X.\mathcal{A}$
  example: $\exists\{x\}.\{\text{hasTopping}(\text{myPizza}, x),\ \text{Parmesan}(x),\ \text{Salami}(x)\}$

A Pizza with Parmesan and Salami

- **Data:** quantified ABox (qABox) $\exists X. \mathcal{A}$
  example: $\exists \{x\}. \{\mathsf{hasTopping}(\mathsf{myPizza}, x),\ \mathsf{Parmesan}(x),\ \mathsf{Salami}(x)\}$
- **Rules:** $\mathcal{EL}$ TBox $\mathcal{T}$
  example: $\{\mathsf{Parmesan} \sqsubseteq \mathsf{Cheese},\ \mathsf{Salami} \sqsubseteq \mathsf{Sausage}\}$

$\mathcal{EL}$ concepts:
$C ::= \top \mid A \mid C \sqcap C \mid \exists r.C$
where $A \in \Sigma_{\mathsf{C}}$ and $r \in \Sigma_{\mathsf{R}}$

$\mathcal{EL}$ concept inclusions:
$C_1 \sqsubseteq C_2$

A Pizza with Parmesan and Salami

- **Data:** quantified ABox (qABox) $\exists X.\mathcal{A}$
  example: $\exists\{x\}.\{\text{hasTopping}(\text{myPizza}, x),\ \text{Parmesan}(x),\ \text{Salami}(x)\}$
- **Rules:** $\mathcal{EL}$ TBox $\mathcal{T}$
  example: $\{\text{Parmesan} \sqsubseteq \text{Cheese},\ \text{Salami} \sqsubseteq \text{Sausage}\}$
- **Unwanted consequences:** $\mathcal{EL}$ repair request $\mathcal{R}$
  example: $\{(\exists\,\text{hasTopping}.(\text{Parmesan} \sqcap \text{Salami}))(\text{myPizza})\}$

$\mathcal{EL}$ concepts:
$C ::= \top \mid A \mid C \sqcap C \mid \exists r.C$
where $A \in \Sigma_\mathsf{C}$ and $r \in \Sigma_\mathsf{R}$

$\mathcal{EL}$ concept inclusions:
$C_1 \sqsubseteq C_2$

$\mathcal{EL}$ concept assertions:
$C(a)$ where $a \in \Sigma_\mathsf{I}$

A Pizza with Parmesan and Salami

- **Data:** quantified ABox (qABox) $\exists X.\mathcal{A}$
  example: $\exists\{x\}.\{\text{hasTopping}(\text{myPizza}, x),\ \text{Parmesan}(x),\ \text{Salami}(x)\}$
- **Rules:** $\mathcal{EL}$ TBox $\mathcal{T}$
  example: $\{\text{Parmesan} \sqsubseteq \text{Cheese},\ \text{Salami} \sqsubseteq \text{Sausage}\}$
- **Unwanted consequences:** $\mathcal{EL}$ repair request $\mathcal{R}$
  example: $\{(\exists\,\text{hasTopping}.(\text{Parmesan} \sqcap \text{Salami}))(\text{myPizza})\}$
- A **repair** of $\exists X.\mathcal{A}$ for $\mathcal{R}$ w.r.t. $\mathcal{T}$ is a qABox $\exists Y.\mathcal{B}$ such that
  - $\exists X.\mathcal{A}$ and $\mathcal{T}$ entail $\exists Y.\mathcal{B}$, and
  - $\exists Y.\mathcal{B}$ and $\mathcal{T}$ do not entail $C(a)$ for each $C(a) \in \mathcal{R}$.

$\mathcal{EL}$ concepts:
$C ::= \top \mid A \mid C \sqcap C \mid \exists r.C$
where $A \in \Sigma_{\mathsf{C}}$ and $r \in \Sigma_{\mathsf{R}}$

$\mathcal{EL}$ concept inclusions:
$C_1 \sqsubseteq C_2$

$\mathcal{EL}$ concept assertions:
$C(a)$ where $a \in \Sigma_{\mathsf{I}}$

A Pizza with Parmesan and Salami

- **Data:** quantified ABox (qABox) $\exists X.\mathcal{A}$
  example: $\exists\{x\}.\{\text{hasTopping}(\text{myPizza},x),\ \text{Parmesan}(x),\ \text{Salami}(x)\}$
- **Rules:** $\mathcal{EL}$ TBox $\mathcal{T}$
  example: $\{\text{Parmesan} \sqsubseteq \text{Cheese},\ \text{Salami} \sqsubseteq \text{Sausage}\}$
- **Unwanted consequences:** $\mathcal{EL}$ repair request $\mathcal{R}$
  example: $\{(\exists\,\text{hasTopping}.(\text{Parmesan} \sqcap \text{Salami}))(\text{myPizza})\}$
- A **repair** of $\exists X.\mathcal{A}$ for $\mathcal{R}$ w.r.t. $\mathcal{T}$ is a qABox $\exists Y.\mathcal{B}$ such that
  - $\exists X.\mathcal{A}$ and $\mathcal{T}$ entail $\exists Y.\mathcal{B}$, and
  - $\exists Y.\mathcal{B}$ and $\mathcal{T}$ do not entail $C(a)$ for each $C(a) \in \mathcal{R}$.
- A repair is **optimal** if it is not strictly entailed by another repair.

$\mathcal{EL}$ concepts:
$C ::= \top \mid A \mid C \sqcap C \mid \exists r.C$
where $A \in \Sigma_\mathsf{C}$ and $r \in \Sigma_\mathsf{R}$

$\mathcal{EL}$ concept inclusions:
$C_1 \sqsubseteq C_2$

$\mathcal{EL}$ concept assertions:
$C(a)$ where $a \in \Sigma_\mathsf{I}$

Related Work

Classical Repair Approach:

- Based on axiom pinpointing
- Removes axioms completely
- Syntax-dependent
- Unable to compute optimal repairs (in our sense)
- Applicable to every monotonic logic

Related Work

Classical Repair Approach:

- Based on axiom pinpointing
- Removes axioms completely
- Syntax-dependent
- Unable to compute optimal repairs (in our sense)
- Applicable to every monotonic logic

Novel, Gentle Repair Approach:

- Weakens axioms instead of removing them completely
- Produces better repairs than classical approach
- Syntax-dependent
- Unable to compute optimal repairs (in our sense)
- Applicable to every monotonic logic for which a weakening relation exists

Our Previous Work

Franz Baader, Francesco Kriegel, Adrian Nuradiansyah, Rafael Peñaloza:
Computing Compliant Anonymisations of Quantified ABoxes w.r.t. $\mathcal{EL}$ Policies.
19th International Semantic Web Conference (ISWC), Athens, Greece, November 2–6, 2020.

- Characterizes optimal repairs of qABoxes
- No TBox supported
- No optimized approach to computing repairs
- No implementation and evaluation

Our Previous Work

Franz Baader, Francesco Kriegel, Adrian Nuradiansyah, Rafael Peñaloza:
Computing Compliant Anonymisations of Quantified ABoxes w.r.t. $\mathcal{EL}$ Policies.
19th International Semantic Web Conference (ISWC), Athens, Greece, November 2–6, 2020.

- Characterizes optimal repairs of qABoxes
- ~~No~~ TBox support~~ed~~
- ~~No~~ optimized approach to computing repairs
- ~~No~~ implementation and evaluation

We expand on the above article and specifically address the last three points.

A Recipe for Removing an Unwanted Consequence

Two notions:

- An **atom** is either a concept name A or an existential restriction $\exists r.C$.
- Each $\mathcal{EL}$ concept is a conjunction of atoms (the **top-level conjunction**).
  $\mathsf{Conj}(C_1 \sqcap C_2 \sqcap \cdots \sqcap C_n) = \{C_1, C_2, \ldots, C_n\}$

A Recipe for Removing an Unwanted Consequence

Two notions:

- An **atom** is either a concept name A or an existential restriction $\exists r.C$.
- Each $\mathcal{EL}$ concept is a conjunction of atoms (the **top-level conjunction**).
  $\mathsf{Conj}(C_1 \sqcap C_2 \sqcap \cdots \sqcap C_n) = \{C_1, C_2, \ldots, C_n\}$

Repair Recipe: For each unwanted consequence $C(u)$:

- either choose a concept name $B \in \mathsf{Conj}(C)$ and remove $B(u)$ from $\mathcal{A}$,
- or choose an existential restriction $\exists r.D \in \mathsf{Conj}(C)$ and do the following for each $r(u, v) \in \mathcal{A}$:
  - if $D \neq \top$, then recursively modify $\mathcal{A}$ such that it does not entail $D(v)$,
  - otherwise, remove $r(u, v)$ from $\mathcal{A}$.

A Recipe for Removing an Unwanted Consequence

Two notions:

- An **atom** is either a concept name A or an existential restriction $\exists r.C$.
- Each $\mathcal{EL}$ concept is a conjunction of atoms (the **top-level conjunction**).
  $\mathsf{Conj}(C_1 \sqcap C_2 \sqcap \cdots \sqcap C_n) = \{C_1, C_2, \ldots, C_n\}$

Repair Recipe: For each unwanted consequence $C(u)$:

- either choose a concept name $B \in \mathsf{Conj}(C)$ and remove $B(u)$ from $\mathcal{A}$,
- or choose an existential restriction $\exists r.D \in \mathsf{Conj}(C)$ and do the following for each $r(u, v) \in \mathcal{A}$:
  - if $D \neq \top$, then recursively modify $\mathcal{A}$ such that it does not entail $D(v)$,
  - otherwise, remove $r(u, v)$ from $\mathcal{A}$.

For each processed object u, we collect the chosen atoms in a set $\mathcal{K}$ (a **repair type**).

Note: We do not yet take the TBox into account.

A Pizza with Parmesan and Salami

$$\exists\,\{x\}.\,\{\ \text{hasTopping}(\text{myPizza}, x),\ \ \text{Parmesan}(x),\ \ \text{Salami}(x)\ \}$$

entails

$$(\exists\,\text{hasTopping}.\,(\text{Parmesan} \sqcap \text{Salami}))(\text{myPizza})$$

A Pizza with Parmesan and Salami

Applying the repair recipe

$$\exists\{x\}.\{\,\mathsf{hasTopping}(\mathsf{myPizza},x),\ \mathsf{Parmesan}(x),\ \cancel{\mathsf{Salami}(x)}\,\}$$

does not entail

$$(\exists\,\mathsf{hasTopping}.(\mathsf{Parmesan}\sqcap\mathsf{Salami}))(\mathsf{myPizza})$$

Object	*Repair Type*
myPizza	$\{\exists\,\mathsf{hasTopping}.(\mathsf{Parmesan}\sqcap\mathsf{Salami})\}$
x	$\{\mathsf{Salami}\}$

A Pizza with Parmesan and Salami

We can do even better

$$\exists\{x,\overline{x}\}.\{\,\text{hasTopping}(\text{myPizza},x),\ \text{Parmesan}(x),\ \sout{\text{Salami}(x)},\ \text{hasTopping}(\text{myPizza},\overline{x}),\ \sout{\text{Parmesan}(\overline{x})},\ \text{Salami}(\overline{x})\,\}$$

does not entail

$$(\exists\,\text{hasTopping}.(\text{Parmesan}\sqcap\text{Salami}))(\text{myPizza})$$

Object	*Copy of*	*Repair Type*
myPizza		$\{\exists\,\text{hasTopping}.(\text{Parmesan}\sqcap\text{Salami})\}$
x		{Salami}
$\overline{x}$	x	{Parmesan}

A Pizza with Parmesan and Salami

$$\exists\,\{x,\overline{x}\}.\,\{\ \mathsf{hasTopping}(\mathsf{myPizza},x),\ \mathsf{Parmesan}(x),\ \cancel{\mathsf{Salami}(x)},\ \mathsf{hasTopping}(\mathsf{myPizza},\overline{x}),\ \cancel{\mathsf{Parmesan}(\overline{x})},\ \mathsf{Salami}(\overline{x})\ \}$$

does not entail

$$(\exists\,\mathsf{hasTopping}.\,(\mathsf{Parmesan}\sqcap\mathsf{Salami}))(\mathsf{myPizza})$$

Object	*Copy of*	*Repair Type*	*Canonical Name* $y_{u,\mathcal{K}}$
myPizza		$\{\exists\,\mathsf{hasTopping}.\,(\mathsf{Parmesan}\sqcap\mathsf{Salami})\}$	$y_{\mathsf{myPizza},\{\exists\mathsf{hasTopping}.(\mathsf{Parmesan}\sqcap\mathsf{Salami})\}}$
x		$\{\mathsf{Salami}\}$	$y_{x,\{\mathsf{Salami}\}}$
$\overline{x}$	x	$\{\mathsf{Parmesan}\}$	$y_{x,\{\mathsf{Parmesan}\}}$

Taking the TBox into account

Example:

- qABox $\exists X.\mathcal{A} \coloneqq \exists\emptyset.\{\mathsf{Salami}(\mathsf{someThing}),\ \mathsf{Sausage}(\mathsf{someThing})\}$
- TBox $\mathcal{T} \coloneqq \{\mathsf{Salami} \sqsubseteq \mathsf{Sausage}\}$
- Repair request $\mathcal{R} \coloneqq \{\mathsf{Sausage}(\mathsf{someThing})\}$

Taking the TBox into account

Example:

- qABox $\exists X.\mathcal{A} \coloneqq \exists\emptyset.\{\text{Salami}(\text{someThing}),\ \text{Sausage}(\text{someThing})\}$
- TBox $\mathcal{T} \coloneqq \{\text{Salami} \sqsubseteq \text{Sausage}\}$
- Repair request $\mathcal{R} \coloneqq \{\text{Sausage}(\text{someThing})\}$

Repairing the qABox without taking $\mathcal{T}$ into account:

$\exists\emptyset.\{\text{Salami}(\text{someThing}),\ \cancel{\text{Sausage}(\text{someThing})}\}$

Repair type of someThing: $\{\text{Sausage}\}$

Taking the TBox into account

Example:

- qABox $\exists X.\mathcal{A} \coloneqq \exists\emptyset.\{\mathsf{Salami}(\mathsf{someThing}),\ \mathsf{Sausage}(\mathsf{someThing})\}$
- TBox $\mathcal{T} \coloneqq \{\mathsf{Salami} \sqsubseteq \mathsf{Sausage}\}$
- Repair request $\mathcal{R} \coloneqq \{\mathsf{Sausage}(\mathsf{someThing})\}$

Repairing the qABox without taking $\mathcal{T}$ into account:

$\exists\emptyset.\{\mathsf{Salami}(\mathsf{someThing}),\ \sout{\mathsf{Sausage}(\mathsf{someThing})}\} \models^{\mathcal{T}} \mathsf{Sausage}(\mathsf{someThing})$

Repair type of someThing: $\{\mathsf{Sausage}\}$

Taking the TBox into account

Example:

- qABox $\exists X.\mathcal{A} \coloneqq \exists\emptyset.\{\mathsf{Salami}(\mathsf{someThing}),\ \mathsf{Sausage}(\mathsf{someThing})\}$
- TBox $\mathcal{T} \coloneqq \{\mathsf{Salami} \sqsubseteq \mathsf{Sausage}\}$
- Repair request $\mathcal{R} \coloneqq \{\mathsf{Sausage}(\mathsf{someThing})\}$

Repairing the qABox without taking $\mathcal{T}$ into account:

$\exists\emptyset.\{\mathsf{Salami}(\mathsf{someThing}),\ \mathsf{Sausage}(\mathsf{someThing})\} \models^{\mathcal{T}} \mathsf{Sausage}(\mathsf{someThing})$

Repair type of someThing: $\{\mathsf{Sausage}\}$

Thus, we need to **close the repair types under premises/implicants**:

$\exists\emptyset.\{\mathsf{Salami}(\mathsf{someThing}),\ \mathsf{Sausage}(\mathsf{someThing})\}$

Repair type of someThing: $\{\mathsf{Salami},\ \mathsf{Sausage}\}$

Taking the TBox into account

Example:

- qABox $\exists X.\mathcal{A} \coloneqq \exists\emptyset.\{\text{Salami}(\text{someThing}),\ \text{Sausage}(\text{someThing})\}$
- TBox $\mathcal{T} \coloneqq \{\text{Salami} \sqsubseteq \text{Sausage}\}$
- Repair request $\mathcal{R} \coloneqq \{\text{Sausage}(\text{someThing})\}$

Repairing the qABox without taking $\mathcal{T}$ into account:

$\exists\emptyset.\{\text{Salami}(\text{someThing}),\ \cancel{\text{Sausage}(\text{someThing})}\} \models^{\mathcal{T}} \text{Sausage}(\text{someThing})$

Repair type of someThing: $\{\text{Sausage}\}$

Thus, we need to **close the repair types under premises/implicants**:

$\exists\emptyset.\{\cancel{\text{Salami}(\text{someThing})},\ \cancel{\text{Sausage}(\text{someThing})}\} \not\models^{\mathcal{T}} \text{Sausage}(\text{someThing})$

Repair type of someThing: $\{\text{Salami},\ \text{Sausage}\}$

Taking the TBox into account

Example:

- qABox $\exists X.\mathcal{A} \coloneqq \exists\emptyset.\{\text{Parmesan}(\text{someThing})\}$
- TBox $\mathcal{T} \coloneqq \{\text{Parmesan} \sqsubseteq \text{Cheese}\}$
- Repair request $\mathcal{R} \coloneqq \{\text{Parmesan}(\text{someThing})\}$

Taking the TBox into account

Example:

- qABox $\exists X.\mathcal{A} \coloneqq \exists \emptyset.\{\text{Parmesan}(\text{someThing})\}$
- TBox $\mathcal{T} \coloneqq \{\text{Parmesan} \sqsubseteq \text{Cheese}\}$
- Repair request $\mathcal{R} \coloneqq \{\text{Parmesan}(\text{someThing})\}$

Repairing the qABox without taking the TBox into account:

$\exists \emptyset.\{$~~Parmesan(someThing)~~$\}$

Repair type of someThing: $\{\text{Parmesan}\}$

Taking the TBox into account

Example:

- qABox $\exists X.\mathcal{A} \coloneqq \exists\emptyset.\{\text{Parmesan}(\text{someThing})\}$
- TBox $\mathcal{T} \coloneqq \{\text{Parmesan} \sqsubseteq \text{Cheese}\}$
- Repair request $\mathcal{R} \coloneqq \{\text{Parmesan}(\text{someThing})\}$

Repairing the qABox without taking the TBox into account:

$\exists\emptyset.\{~~\text{Parmesan}(\text{someThing})~~\} \not\models^{\mathcal{T}} \text{Cheese}(\text{someThing})$

Repair type of someThing: $\{\text{Parmesan}\}$

Taking the TBox into account

Example:

- qABox $\exists X.\mathcal{A} \coloneqq \exists\emptyset.\{\text{Parmesan}(\text{someThing})\}$
- TBox $\mathcal{T} \coloneqq \{\text{Parmesan} \sqsubseteq \text{Cheese}\}$
- Repair request $\mathcal{R} \coloneqq \{\text{Parmesan}(\text{someThing})\}$

Repairing the qABox without taking the TBox into account:

$\exists\emptyset.\{\text{Parmesan}(\text{someThing})\} \not\models^{\mathcal{T}} \text{Cheese}(\text{someThing})$

Repair type of someThing: $\{\text{Parmesan}\}$

Thus, we need to first **saturate the qABox** before repairing it:

$\exists\emptyset.\{\text{Parmesan}(\text{someThing}),\ \text{Cheese}(\text{someThing})\}$

Repair type of someThing: $\{\text{Parmesan}\}$

Taking the TBox into account

Example:

- qABox $\exists X.\mathcal{A} \coloneqq \exists\emptyset.\{\text{Parmesan(someThing)}\}$
- TBox $\mathcal{T} \coloneqq \{\text{Parmesan} \sqsubseteq \text{Cheese}\}$
- Repair request $\mathcal{R} \coloneqq \{\text{Parmesan(someThing)}\}$

Repairing the qABox without taking the TBox into account:

$\exists\emptyset.\{\text{Parmesan(someThing)}\} \not\models^{\mathcal{T}} \text{Cheese(someThing)}$

Repair type of someThing: $\{\text{Parmesan}\}$

Thus, we need to first **saturate the qABox** before repairing it:

$\exists\emptyset.\{\text{Parmesan(someThing)}, \text{Cheese(someThing)}\}$

Repair type of someThing: $\{\text{Parmesan}\}$

Taking the TBox into account

Example:

- qABox $\exists X.\mathcal{A} \coloneqq \exists \emptyset.\{\text{Parmesan}(\text{someThing})\}$
- TBox $\mathcal{T} \coloneqq \{\text{Parmesan} \sqsubseteq \text{Cheese}\}$
- Repair request $\mathcal{R} \coloneqq \{\text{Parmesan}(\text{someThing})\}$

Repairing the qABox without taking the TBox into account:

$\exists \emptyset.\{~~\text{Parmesan}(\text{someThing})~~\} \not\models^{\mathcal{T}} \text{Cheese}(\text{someThing})$

Repair type of someThing: $\{\text{Parmesan}\}$

Thus, we need to first **saturate the qABox** before repairing it:

$\exists \emptyset.\{~~\text{Parmesan}(\text{someThing})~~, \text{Cheese}(\text{someThing})\} \models^{\mathcal{T}} \text{Cheese}(\text{someThing})$

Repair type of someThing: $\{\text{Parmesan}\}$

Saturations

Two entailment relations:

- **Classical entailment** compares qABoxes w.r.t. their models.
- **CQ-entailment** compares qABoxes w.r.t. which **C**onjunctive **Q**ueries they entail. It coincides with classical entailment and is NP-complete.
- **IQ-entailment** compares qABoxes w.r.t. which $\mathcal{EL}$ concept assertions (**I**nstance **Q**ueries) they entail. It can be decided in polynomial time.

Saturations

Two entailment relations:

- **Classical entailment** compares qABoxes w.r.t. their models.
- **CQ-entailment** compares qABoxes w.r.t. which **C**onjunctive **Q**ueries they entail. It coincides with classical entailment and is NP-complete.
- **IQ-entailment** compares qABoxes w.r.t. which $\mathcal{EL}$ concept assertions (**I**nstance **Q**ueries) they entail. It can be decided in polynomial time.

Two saturations:

- The saturation of a qABox w.r.t. a TBox depends on the chosen entailment relation.
- To guarantee termination of saturation w.r.t. CQ-entailment, we require that the TBox is cycle-restricted.
- For cycle-restricted TBoxes, **CQ-saturations** can be computed in exponential time.
- In contrast, saturation w.r.t. IQ-entailment always terminates.
- For all TBoxes, **IQ-saturations** can be computed in polynomial time.

Canonical Repairs

Canonical repairs are built from the saturation and consist of all objects $y_{u,\mathcal{K}}$.

Canonical Repairs

Canonical repairs are built from the saturation and consist of all objects $y_{u,\mathcal{K}}$.

CQ-Entailment:

1. **Up to equivalence, the set of all canonical repairs contains each optimal repair.**
2. Given a qABox $\exists X.\mathcal{A}$, a cycle-restricted TBox $\mathcal{T}$, and a repair request $\mathcal{R}$, **the set of all optimal repairs can be computed in exponential time using an NP-oracle.**

Canonical Repairs

Canonical repairs are built from the saturation and consist of all objects $y_{u,\mathcal{K}}$.

CQ-Entailment:

1. **Up to equivalence, the set of all canonical repairs contains each optimal repair.**
2. Given a qABox $\exists X.\mathcal{A}$, a cycle-restricted TBox $\mathcal{T}$, and a repair request $\mathcal{R}$, **the set of all optimal repairs can be computed in exponential time using an NP-oracle.**

IQ-Entailment:

1. **Up to equivalence, the set of all canonical repairs contains each optimal repair.**
2. Given a qABox $\exists X.\mathcal{A}$, a TBox $\mathcal{T}$, and a repair request $\mathcal{R}$, **the set of all optimal repairs can be computed in exponential time.**

Optimized Repairs

- **Problem:** Each canonical repair has exponential size.
- Thus, it is expensive or even impossible to compute canonical repairs of large ontologies.
- There are examples where an optimal repair need not be exponentially large.
- In these cases, the canonical repair is already equivalent to a small sub-qABox.

Optimized Repairs

- **Problem:** Each canonical repair has exponential size.
- Thus, it is expensive or even impossible to compute canonical repairs of large ontologies.
- There are examples where an optimal repair need not be exponentially large.
- In these cases, the canonical repair is already equivalent to a small sub-qABox.
- We propose a rule-based way to compute **optimized repairs**, which contain only relevant parts of the canonical repairs.
  - CQ-repairs only need objects reachable either from the individuals or from the unmodified copies of all objects.
  - IQ-repairs only need objects reachable from the individuals.
  - In both cases, not all successors of an object are needed, but only those with the fewest modifications.

Implementation and Evaluation

Prototypical implementation:

`https://github.com/de-tu-dresden-inf-lat/abox-repairs-wrt-static-tbox`

Dependencies: OWL-API, ELK, VLOG, Rulewerk

Implementation and Evaluation

Prototypical implementation:

`https://github.com/de-tu-dresden-inf-lat/abox-repairs-wrt-static-tbox`

Dependencies: OWL-API, ELK, VLOG, Rulewerk

Scenarios:

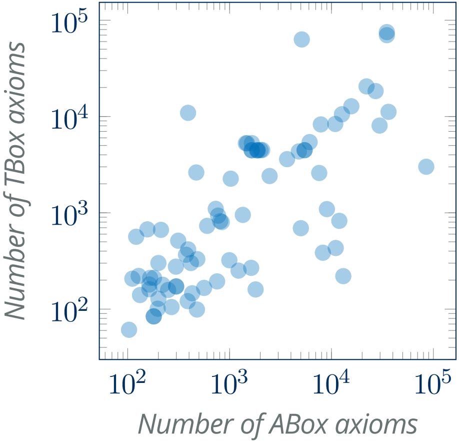
- **S1:** repairing a single unwanted consequence for a single individual
- **S2:** repairing a single unwanted consequence for $10\,\%$ of the individuals

Implementation and Evaluation

Evaluation corpus:

- 80 ontologies
- With up to 100,000 axioms
- Used in the 2015 OWL Reasoner Competition
- Track: OWL EL Realisation
- Cyclic ontologies ignored for CQ-experiments

Input Ontologies

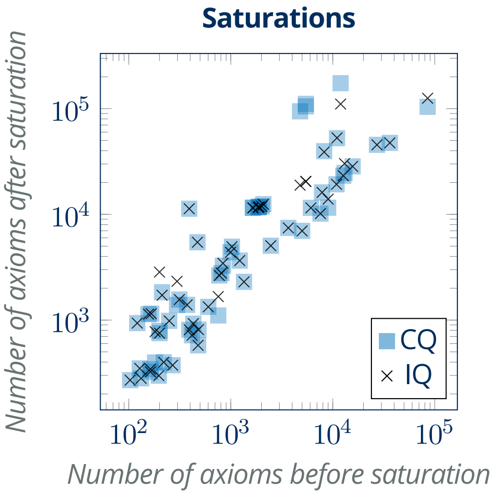


Implementation and Evaluation

Saturation timeout: 60 minutes

Success rates:

- CQ-saturations: $96.9\,\%$
- IQ-saturations: $100\,\%$



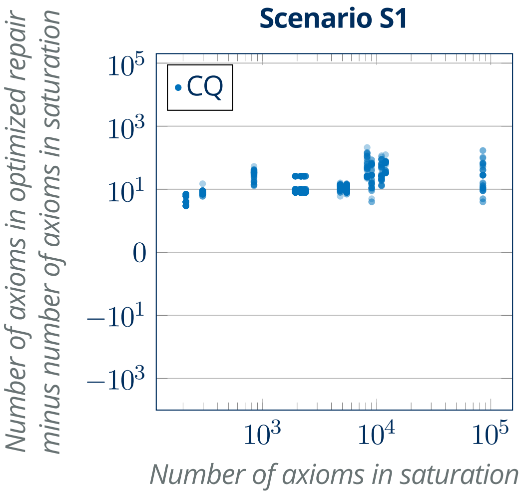
Implementation and Evaluation

Repair timeout: 10 minutes

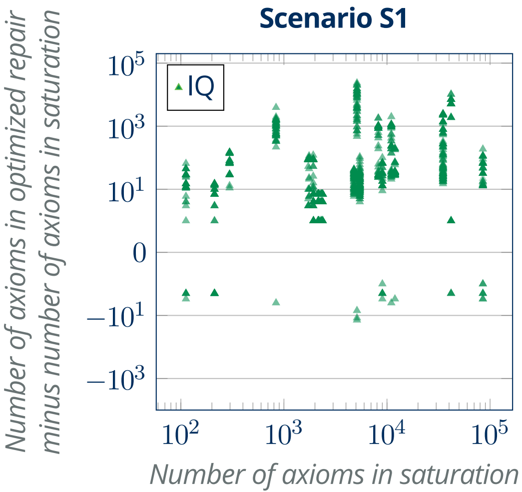
Success rates:

- Canonical CQ-repairs: 62.1 %
- Optimized CQ-repairs, S1: 100 %
- Optimized CQ-repairs, S2: 99.9 %
- Canonical IQ-repairs: 52.9 %
- Optimized IQ-repairs, S1: 99.9 %
- Optimized IQ-repairs, S2: 98.9 %

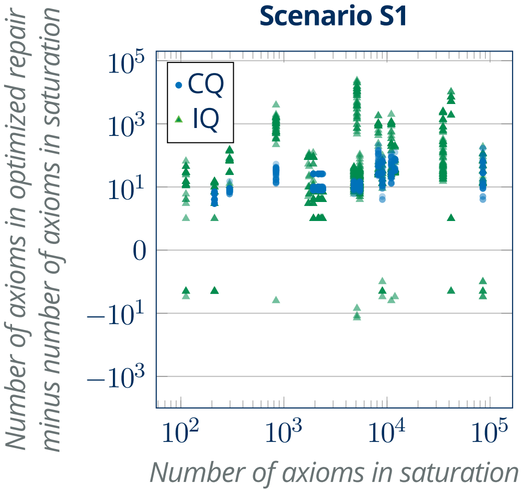
Implementation and Evaluation



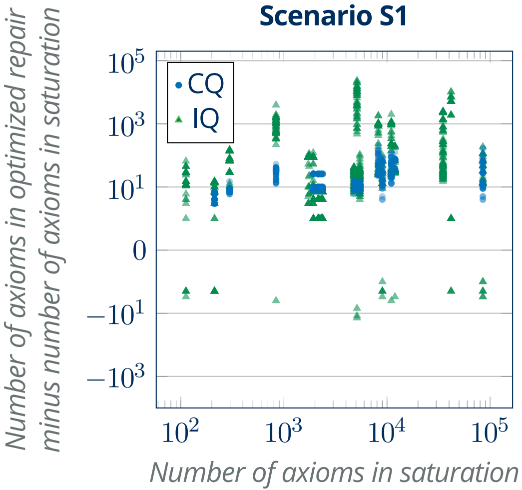
Implementation and Evaluation

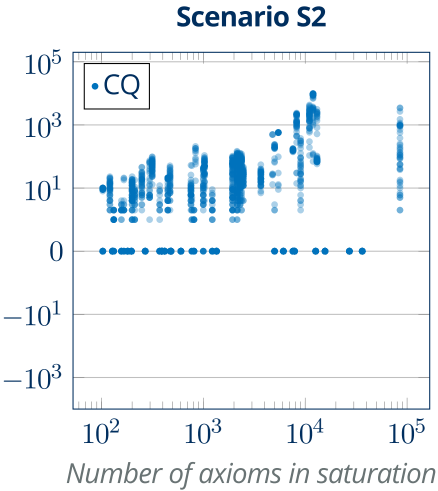


Implementation and Evaluation

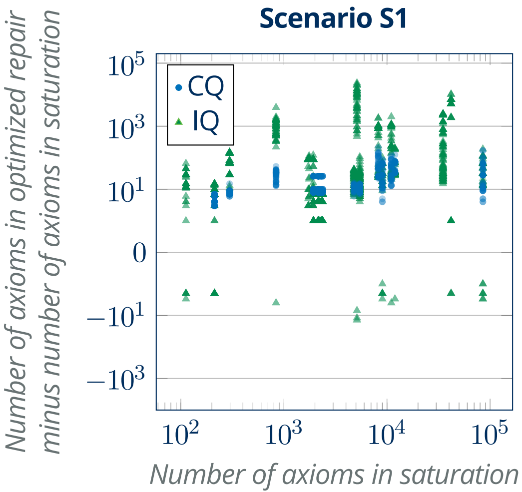


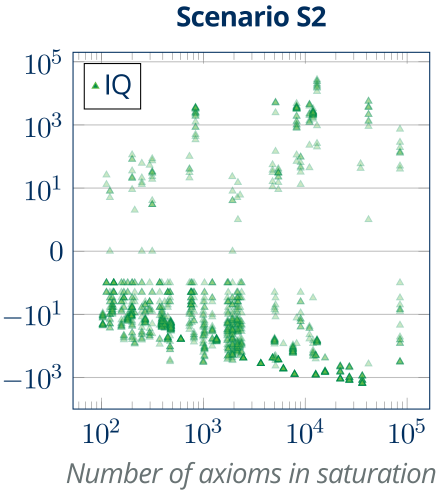
Implementation and Evaluation



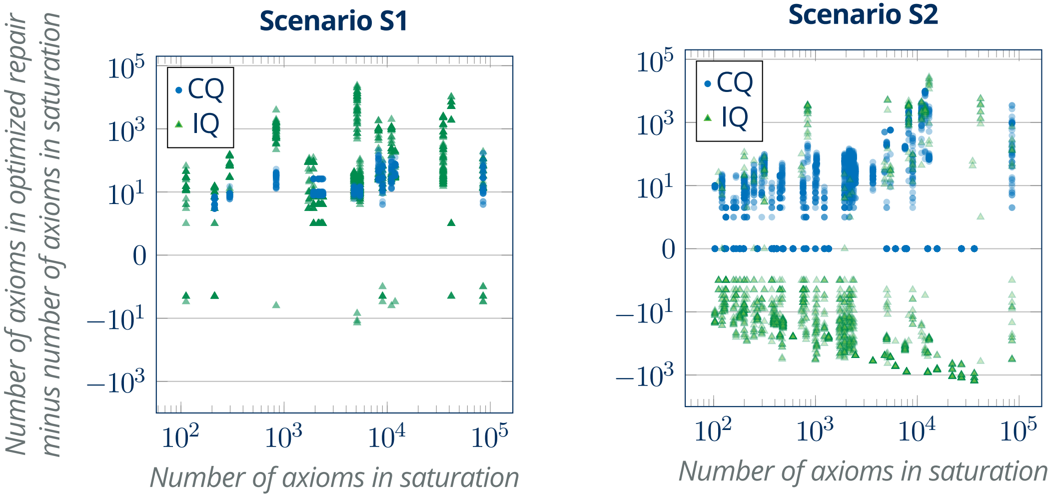


Implementation and Evaluation





Implementation and Evaluation



Conclusion and Outlook

Summary:

- We developed an approach to computing optimal repairs, i.e., which lose a minimal amount of consequences.
- As consequences to be preserved, we considered answers to conjunctive queries (CQ-entailment) as well as answers to $\mathcal{EL}$ instance queries (IQ-entailment).
- We support data in form of quantified ABoxes, sets of rules in form of $\mathcal{EL}$ TBoxes (that are assumed to be static), and $\mathcal{EL}$ concept assertions as unwanted consequences to be removed.
- We also devised an efficient construction of repairs that is only worst-case exponential.
- We implemented our methods and evaluated them on real-world ontologies.

Conclusion and Outlook

Summary:

- We developed an approach to computing optimal repairs, i.e., which lose a minimal amount of consequences.
- As consequences to be preserved, we considered answers to conjunctive queries (CQ-entailment) as well as answers to $\mathcal{EL}$ instance queries (IQ-entailment).
- We support data in form of quantified ABoxes, sets of rules in form of $\mathcal{EL}$ TBoxes (that are assumed to be static), and $\mathcal{EL}$ concept assertions as unwanted consequences to be removed.
- We also devised an efficient construction of repairs that is only worst-case exponential.
- We implemented our methods and evaluated them on real-world ontologies.

Future Work:

- Lifting the restriction to cycle-restricted TBoxes for CQ-entailment.
- Support for unwanted consequences in form of conjunctive queries.

That’s it for now!

Do you have questions or comments?