Computing Compliant Anonymisations of Quantified ABoxes w.r.t. $\mathcal{EL}$ Policies

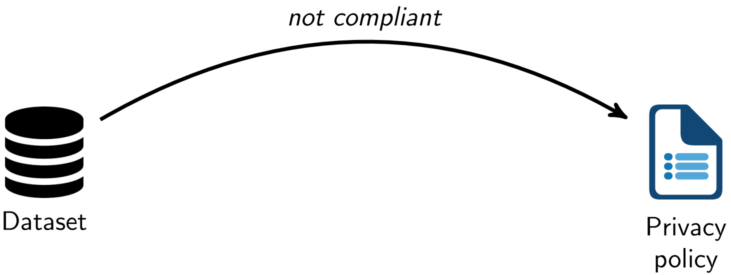
Franz Baader[1] Francesco Kriegel[1] **Adrian Nuradiansyah**[1]
Rafael Peñaloza[2]

[1]Technische Universität Dresden
[2]University of Milano-Bicocca

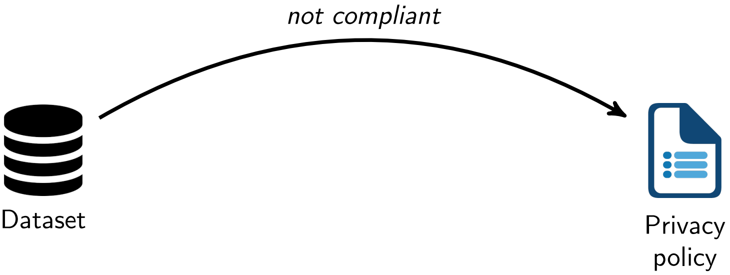
November 4th, 2020

Funded by DFG Deutsche Forschungsgemeinschaft German Research Foundation

An Illustration of Non-Compliance



An Illustration of Non-Compliance



Dataset:

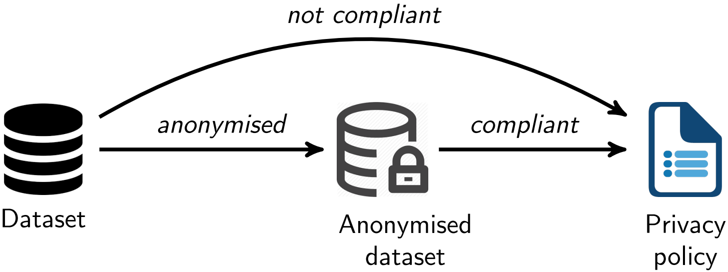
$\exists\{x\}.\{Politician(d), Businessman(d), related(d, x), Politician(x), Businessman(x)\}$

Policy:

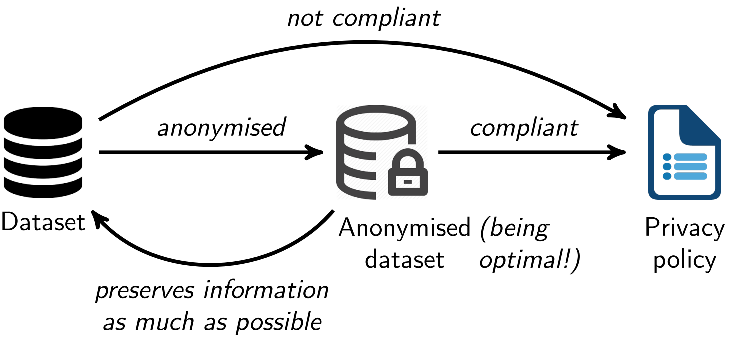
$\{Politician \sqcap Businessman, \exists r.(Politician \sqcap Businessman)\}$

The individual d is an instance of both concepts w.r.t. the dataset $\Rightarrow$ not compliant!

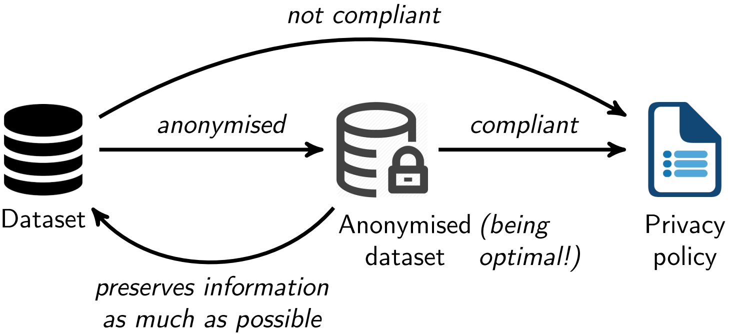
An Illustration of Non-Compliance



An Illustration of Non-Compliance



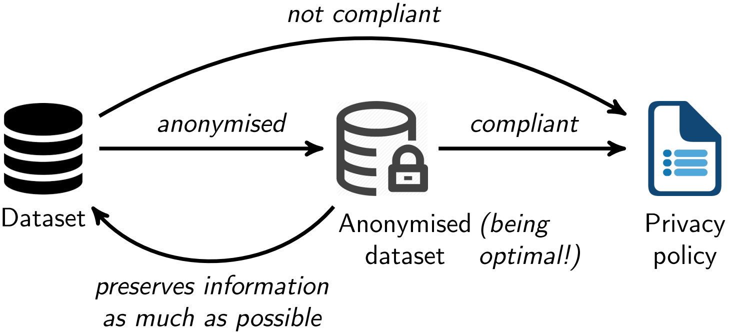
An Illustration of Non-Compliance



Question:

*How to **anonymise** a dataset **in a minimal way** s.t. all the published information **follows from the original one**, but **privacy** constraints **are satisfied**?*

An Illustration of Non-Compliance



Question:

*How to **anonymise** a dataset **in a minimal way** s.t. all the published information **follows from the original one**, but **privacy** constraints **are satisfied**?*

Assumption: Our problem will be considered in the context of Description Logic (DL) ontologies

How Our Dataset Looks Like

A **quantified ABox** $\exists X.\mathcal{A}$

$\exists\{x\}.\{Politician(d), Businessman(d), related(d, x), Politician(x), Businessman(x)\}$

is built over

- a set X of **variables**, e.g., $x, x_1, x_2, \ldots$
- a set of **individual** names, e.g., $d, d_1, d_2, \ldots$
- a set of **concept names**, e.g., $Politician, Businessman, P, B, \ldots$
- a set of **role names**, e.g., $related, r, s$

How Our Dataset Looks Like

A **quantified ABox** $\exists X.\mathcal{A}$

$\exists\{x\}.\{Politician(d), Businessman(d), related(d, x), Politician(x), Businessman(x)\}$

is built over

- a set X of **variables**, e.g., $x, x_1, x_2, \ldots$
- a set of **individual** names, e.g., $d, d_1, d_2, \ldots$
- a set of **concept names**, e.g., $Politician, Businessman, P, B, \ldots$
- a set of **role names**, e.g., $related, r, s$

and $\mathcal{A}$, in general, consists of:

- **concept assertions**, e.g., $Politician(d), Businessman(x), \ldots$
- **role assertions**, e.g., $related(d, x), \ldots$

Note: A *traditional DL ABox* is a quantified ABox where X is empty.

How Our Dataset Looks Like

A **quantified ABox** $\exists X.\mathcal{A}$

$\exists\{x\}.\{Politician(d), Businessman(d), related(d, x), Politician(x), Businessman(x)\}$

Entailment between Quantified ABoxes

- $\exists X.\mathcal{A} \models \exists Y.\mathcal{B}$ denotes that $\exists X.\mathcal{A}$ **entails** $\exists Y.\mathcal{B}$
- The entailment problem between quantified ABoxes is **NP-complete**

How the Policy Looks Like

A **policy** $\mathcal{P}$ is a **finite set of $\mathcal{EL}$ concepts**

$\{Politician \sqcap Businessman, \exists r.(Politician \sqcap Businessman)\}$

It has the following components:

- $\mathsf{Atoms}(\mathcal{P}) = \{Politician, Businessman, \exists r.(Politician \sqcap Businessman)\}$
- Let P_1 be the first concept in $\mathcal{P}$

  $\mathsf{Conj}(P_1) = \{Politician, Businessman\}$ occurs in the **top-level conjunction** of P_1

How the Policy Looks Like

A **policy** $\mathcal{P}$ is a **finite set of $\mathcal{EL}$ concepts**

$\{Politician \sqcap Businessman, \exists r.(Politician \sqcap Businessman)\}$

It has the following components:

- Atoms$(\mathcal{P}) = \{Politician, Businessman, \exists r.(Politician \sqcap Businessman)\}$
- Let P_1 be the first concept in $\mathcal{P}$
  Conj$(P_1) = \{Politician, Businessman\}$ occurs in the **top-level conjunction** of P_1

Reasoning Problems in $\mathcal{EL}$

- $C \sqsubseteq_\emptyset D$ means that the $\mathcal{EL}$ concept C is **subsumed by** the $\mathcal{EL}$ concept D
- $\exists X.\mathcal{A} \models C(a)$ means that the individual a is an **instance** of the $\mathcal{EL}$ concept C w.r.t. $\exists X.\mathcal{A}$
- Both **subsumption** and **instance relationships** can be checked in polynomial time for $\mathcal{EL}$

Optimal Compliant Anonymisations

A **quantified ABox** $\exists Y.\mathcal{B}$ is an **optimal $\mathcal{P}$-compliant anonymisation** of $\exists X.\mathcal{A}$ iff

- $\exists Y.\mathcal{B} \not\models P(a)$ for all $P \in \mathcal{P}$ and all individuals a (**compliance**)
- $\exists X.\mathcal{A} \models \exists Y.\mathcal{B}$ (**anonymisation**)
- there is no $\mathcal{P}$-compliant anonymisation $\exists Z.\mathcal{C}$ of $\exists X.\mathcal{A}$ that stricly entails $\exists Y.\mathcal{B}$ (**optimal**)

How to Make an ABox Compliant

Non-compliance means that there exist an individual a and $P \in \mathcal{P}$ s.t.

$$a \text{ is an instance of all atoms in } \mathsf{Conj}(P) \text{ w.r.t. } \exists X.\mathcal{A}.$$

How to Make an ABox Compliant

Non-compliance means that there exist an individual a and $P \in \mathcal{P}$ s.t.

a is an instance of all atoms in $\mathrm{Conj}(P)$ w.r.t. $\exists X.\mathcal{A}$.

$\Rightarrow$ To make the ABox compliant, choose one atom C from $\mathrm{Conj}(P)$ such that a will not be an instance of C in the resulting anonymisation

This idea is represented by the use of a **compliance seed function**

How to Make an ABox Compliant

Non-compliance means that there exist an individual a and $P \in \mathcal{P}$ s.t.

a is an instance of all atoms in $\mathrm{Conj}(P)$ w.r.t. $\exists X.\mathcal{A}$.

$\Rightarrow$ To make the ABox compliant, choose one atom C from $\mathrm{Conj}(P)$ such that a will not be an instance of C in the resulting anonymisation

This idea is represented by the use of a **compliance seed function**

A **compliance seed function (csf) s** on $\exists X.\mathcal{A}$ for $\mathcal{P}$ maps each individual name a to a subset of $\mathrm{Atoms}(\mathcal{P})$ such that

for each $P \in \mathcal{P}$, there is $C \in s(a)$ such that $C \in \mathrm{Conj}(P)$

$\exists X.\mathcal{A} = \exists\{x\}.\{P(d), B(d), r(d,x), P(x), B(x)\}$ $\qquad$ $\mathcal{P} = \{P \sqcap B, \exists r.(P \sqcap B)\}$

Mapping d to $s(d) = \{B, \exists r.(P \sqcap B)\}$ yields a csf

Computing a Compliant Anonymisation

From a given csf s, we can compute a compliant anonymisation with the following idea:

$\exists X.\mathcal{A} = \exists\{x\}.\{P(d), B(d), r(d,x), P(x), B(x)\}$ $\qquad$ $\mathcal{P} = \{P \sqcap B, \exists r.(P \sqcap B)\}$

Computing a Compliant Anonymisation

From a given csf s, we can compute a compliant anonymisation with the following idea:

$\exists X.\mathcal{A} = \exists\{x\}.\{P(d), B(d), r(d,x), P(x), B(x)\}$ $\qquad \mathcal{P} = \{P \sqcap B, \exists r.(P \sqcap B)\}$

1. **Copy operation**: select a variable/an individual, copy this object, and duplicate assertions involving it

Computing a Compliant Anonymisation

From a given csf s, we can compute a compliant anonymisation with the following idea:

$\exists X.\mathcal{A} = \exists\{x\}.\{P(d), B(d), r(d,x), P(x), B(x)\}$ $\qquad \mathcal{P} = \{P \sqcap B, \exists r.(P \sqcap B)\}$

1. **Copy operation**: select a variable/an individual, copy this object, and duplicate assertions involving it e.g., (**select** d **and make the copy** y_d)

$$\exists\{x, y_d\}.\{P(d), B(d), r(d,x), P(x), B(x), \\ P(y_d), B(y_d), r(y_d, x)\}$$

Computing a Compliant Anonymisation

From a given csf s, we can compute a compliant anonymisation with the following idea:

$\exists X.\mathcal{A} = \exists\{x\}.\{P(d), B(d), r(d, x), P(x), B(x)\}$ $\quad\quad$ $\mathcal{P} = \{P \sqcap B, \exists r.(P \sqcap B)\}$

1. **Copy operation**: select a variable/an individual, copy this object, and duplicate assertions involving it e.g., (**select** x **and make the copy** y_x)

$$\exists\{x, y_d, y_x\}.\{P(d), B(d), r(d, x), P(x), B(x),$$
$$P(y_d), B(y_d), r(y_d, x), r(d, y_x), r(y_d, y_x), P(y_x), B(y_x)\}$$

Computing a Compliant Anonymisation

From a given csf s, we can compute a compliant anonymisation with the following idea:

$\exists X.\mathcal{A} = \exists\{x\}.\{P(d), B(d), r(d,x), P(x), B(x)\}$ $\qquad$ $\mathcal{P} = \{P \sqcap B, \exists r.(P \sqcap B)\}$

1. **Copy operation**: select a variable/an individual, copy this object, and duplicate assertions involving it

$$\exists\{x, y_d, y_x\}.\{P(d), B(d), r(d,x), P(x), B(x),$$
$$P(y_d), B(y_d), r(y_d, x), r(d, y_x), r(y_d, y_x), P(y_x), B(y_x)\}$$

Note: *It suffices to create at most exponentially many copies of each object!*

Computing a Compliant Anonymisation

From a given csf s, we can compute a compliant anonymisation with the following idea:

$\exists X.\mathcal{A} = \exists\{x\}.\{P(d), B(d), r(d,x), P(x), B(x)\}$ $\qquad \mathcal{P} = \{P \sqcap B, \exists r.(P \sqcap B)\}$

1. **Copy operation**: select a variable/an individual, copy this object, and duplicate assertions involving it

$$\exists\{x, y_d, y_x\}.\{P(d), B(d), r(d,x), P(x), B(x),$$
$$P(y_d), B(y_d), r(y_d, x), r(d, y_x), r(y_d, y_x), P(y_x), B(y_x)\}$$

2. **Deletion operation**: The given csf s will guide which assertions should be removed from the current anonymisation

Computing a Compliant Anonymisation

From a given csf s, we can compute a compliant anonymisation with the following idea:

$\exists X.\mathcal{A} = \exists\{x\}.\{P(d), B(d), r(d,x), P(x), B(x)\}$ $\quad$ $\mathcal{P} = \{P \sqcap B, \exists r.(P \sqcap B)\}$

1. **Copy operation**: select a variable/an individual, copy this object, and duplicate assertions involving it

$$\exists\{x, y_d, y_x\}.\{P(d), B(d), r(d,x), P(x), B(x),\\ P(y_d), B(y_d), r(y_d, x), r(d, y_x), r(y_d, y_x), P(y_x), B(y_x)\}$$

2. **Deletion operation**: The given csf s will guide which assertions should be removed from the current anonymisation

   Since $s(d) = \{B, \exists r.(P \sqcap B)\} \Rightarrow d$ is not allowed to be an instance of B

$$\exists\{x, y_d, y_x\}.\{P(d), \cancel{B(d)}, r(d,x), P(x), B(x),\\ P(y_d), B(y_d), r(y_d, x), r(d, y_x), r(y_d, y_x), P(y_x), B(y_x)\}$$

Computing a Compliant Anonymisation

From a given csf s, we can compute a compliant anonymisation with the following idea:

$\exists X.\mathcal{A} = \exists\{x\}.\{P(d), B(d), r(d,x), P(x), B(x)\}$ $\qquad \mathcal{P} = \{P \sqcap B, \exists r.(P \sqcap B)\}$

1. **Copy operation**: select a variable/an individual, copy this object, and duplicate assertions involving it

$$\exists\{x, y_d, y_x\}.\{P(d), B(d), r(d,x), P(x), B(x),$$
$$P(y_d), B(y_d), r(y_d, x), r(d, y_x), r(y_d, y_x), P(y_x), B(y_x)\}$$

2. **Deletion operation**: The given csf s will guide which assertions should be removed from the current anonymisation

Since $s(d) = \{B, \exists r.(P \sqcap B)\} \Rightarrow$ r-successors of d are not allowed to be an instance of $P \sqcap B$

$$\exists\{x, y_d, y_x\}.\{P(d), \cancel{B(d)}, r(d,x), P(x), \cancel{B(x)},$$
$$P(y_d), B(y_d), r(y_d, x), r(d, y_x), r(y_d, y_x), \cancel{P(y_x)}, B(y_x)\}$$

Computing a Compliant Anonymisation

From a given csf s, we can compute a compliant anonymisation with the following idea:

$\exists X.\mathcal{A} = \exists\{x\}.\{P(d), B(d), r(d,x), P(x), B(x)\}$ $\qquad$ $\mathcal{P} = \{P \sqcap B, \exists r.(P \sqcap B)\}$

The following resulting anonymisation

$$\mathsf{ca}(\exists X.\mathcal{A}, s) = \exists Y.\mathcal{B}$$

is a $\mathcal{P}$-**compliant anonymisation** of $\exists X.\mathcal{A}$, where $\mathcal{B}$ is

$$\{P(d), r(d,x), P(x),$$
$$P(y_d), B(y_d), r(y_d, x), r(d, y_x), r(y_d, y_x), B(y_x)\}$$

and $Y = \{x, y_d, y_x\}$

Soundness, Completeness, Complexity

In general,

- For every csf s, the induced ABox

$$\mathsf{ca}(\exists X.\mathcal{A}, s) = \exists Y.\mathcal{B}$$

is **entailed by** $\exists X.\mathcal{A}$ and **complies with** $\mathcal{P}$

Soundness, Completeness, Complexity

In general,

- For every csf s, the induced ABox

$$\mathrm{ca}(\exists X.\mathcal{A}, s) = \exists Y.\mathcal{B}$$

  is **entailed by** $\exists X.\mathcal{A}$ and **complies with** $\mathcal{P}$

- The set

$$\mathrm{CA}(\exists X.\mathcal{A}, \mathcal{P}) = \{\mathrm{ca}(\exists X.\mathcal{A}, s) \mid s \text{ is a csf on } \exists X.\mathcal{A} \text{ for } \mathcal{P}\}$$

  - contains **all** optimal $\mathcal{P}$-compliant anonymisations of $\exists X.\mathcal{A}$
  - can be computed in **exponential time**

  (*exponentially many csfs!*)

Soundness, Completeness, Complexity

In general,

- For every csf s, the induced ABox

$$\mathrm{ca}(\exists X.\mathcal{A}, s) = \exists Y.\mathcal{B}$$

is **entailed by** $\exists X.\mathcal{A}$ and **complies with** $\mathcal{P}$

- The set

$$\mathrm{CA}(\exists X.\mathcal{A}, \mathcal{P}) = \{\mathrm{ca}(\exists X.\mathcal{A}, s) \mid s \text{ is a csf on } \exists X.\mathcal{A} \text{ for } \mathcal{P}\}$$

  - contains **all** optimal $\mathcal{P}$-compliant anonymisations of $\exists X.\mathcal{A}$
  - can be computed in **exponential time**

(*exponentially many csfs!*)

- To remove the ones that are not optimal, we use an **NP-oracle** to check entailment between compliant anonymisations

Soundness, Completeness, Complexity

In general,

- For every csf s, the induced ABox

$$\mathrm{ca}(\exists X.\mathcal{A}, s) = \exists Y.\mathcal{B}$$

  is **entailed by** $\exists X.\mathcal{A}$ and **complies with** $\mathcal{P}$

- The set

$$\mathrm{CA}(\exists X.\mathcal{A}, \mathcal{P}) = \{\mathrm{ca}(\exists X.\mathcal{A}, s) \mid s \text{ is a csf on } \exists X.\mathcal{A} \text{ for } \mathcal{P}\}$$

  - contains **all** optimal $\mathcal{P}$-compliant anonymisations of $\exists X.\mathcal{A}$
  - can be computed in **exponential time**

  (*exponentially many csfs!*)

- To remove the ones that are not optimal, we use an **NP-oracle** to check entailment between compliant anonymisations

Is it possible to get rid of the NP oracle?

Improving Complexity

1. Using a **partial order** $\leq$ on csfs

   We take only the $\leq$-**minimal csfs** for computing optimal compliant anonymisations

Improving Complexity

1. Using a **partial order** $\leq$ on csfs

   We take only the $\leq$-**minimal csfs** for computing optimal compliant anonymisations

2. Introducing **IQ-entailment**

   - $\mathcal{EL}$ concepts are **instance queries (IQ)**
   - Only compare ABoxes based on which instance queries entailed by them

   Deciding if $\exists X.\mathcal{A}$ IQ-entails $\exists Y.\mathcal{B}$ can be done in polynomial time

Table of Complexity Results

Settings	Completeness
standard entailment	all optimal compliant anonymisations
standard entailment and $\leq$ on csfs	only optimal compliant anonymisations, not all of them
IQ-entailment	all optimal compliant IQ-anonymisations

Table of Complexity Results

Settings	Completeness
standard entailment	all optimal compliant anonymisations
standard entailment and $\leq$ on csfs	only optimal compliant anonymisations, not all of them
IQ-entailment	all optimal compliant IQ-anonymisations

Settings	Combined Complexity	Data Complexity
standard entailment	exponential time with an NP-oracle	polynomial time with an NP-oracle
standard entailment and $\leq$ on csfs	exponential time	polynomial time
IQ-entailment	exponential time	polynomial time

Future Work and References

Future Work

- Safety for $\mathcal{EL}$ policies
  A quantified ABox is **safe** for $\mathcal{P}$ if its combination with other $\mathcal{P}$-compliant ABoxes is also compliant with $\mathcal{P}$
- Compliance w.r.t. **(general) TBoxes**
- Computing optimal compliant anonymisations w.r.t. **conjunctive queries**

Future Work and References

Future Work

- Safety for $\mathcal{EL}$ policies
  A quantified ABox is **safe** for $\mathcal{P}$ if its combination with other $\mathcal{P}$-compliant ABoxes is also compliant with $\mathcal{P}$
- Compliance w.r.t. **(general) TBoxes**
- Computing optimal compliant anonymisations w.r.t. **conjunctive queries**

Our work is based on the following **related work**:

- F. Baader, F. Kriegel, A. Nuradiansyah, *Privacy-Preserving Ontology Publishing for $\mathcal{EL}$ Instance Stores*, JELIA 2019
- B. Cuenca Grau and E. Kostylev, *Logical Foundations of Linked Data Anonymizations*, JAIR, 2019