Optimization of Answer Set Programs for Consistent Query Answering by Means of First-Order Rewriting

Aziz Amezian El Khalfioui Jonathan Joertz Dorian Labeeuw
Gaëtan Staquet Jef Wijsen

Département d'Informatique
Faculté des Sciences
Université de Mons

29th ACM Int. Conf. on Information and Knowledge Management (CIKM 2020), October 19–23, 2020, Virtual Event, Ireland.

UMONS Université de Mons

Faculté des Sciences

Inconsistent databases and repairs

r	$\underline{Conf}$	$\underline{Year}$	$City$
	CIKM	2020	Galway
	CIKM	2021	Perth
	CIKM	2021	Sydney

s	$\underline{City}$	$Country$
	Perth	Australia
	Sydney	Australia
	Galway	Ireland

Inconsistent databases and repairs

r_1

Conf	*Year*	*City*
CIKM	2020	Galway
CIKM	2021	Perth

or

r_2

Conf	*Year*	*City*
CIKM	2020	Galway
CIKM	2021	Sydney

s

City	*Country*
Perth	Australia
Sydney	Australia
Galway	Ireland

Inconsistent databases and repairs

r_1

<u>*Conf*</u>	<u>*Year*</u>	*City*
CIKM	2020	Galway
CIKM	2021	Perth

or

r_2

<u>*Conf*</u>	<u>*Year*</u>	*City*
CIKM	2020	Galway
CIKM	2021	Sydney

s

<u>*City*</u>	*Country*
Perth	Australia
Sydney	Australia
Galway	Ireland

"CIKM 2021 will take place in Australia" is certain because it is true for both repairs (because Perth and Sydney are both certainly in Australia).

Consistent (or Certain) Query Answering (CQA)

- A database instance may violate its primary-key constraints.

Consistent (or Certain) Query Answering (CQA)

- A database instance may violate its primary-key constraints.
- A repair is any maximal consistent subinstance.
  A database instance with n tuples can have exponentially many repairs.

Consistent (or Certain) Query Answering (CQA)

- A database instance may violate its primary-key constraints.
- A repair is any maximal consistent subinstance.
  A database instance with n tuples can have exponentially many repairs.
- A Boolean query (a.k.a. a first-order sentence) is certain if it holds true in every repair.

Consistent (or Certain) Query Answering (CQA)

- A database instance may violate its primary-key constraints.
- A repair is any maximal consistent subinstance.
  A database instance with n tuples can have exponentially many repairs.
- A Boolean query (a.k.a. a first-order sentence) is certain if it holds true in every repair.
- For every fixed Boolean query q, we define CERTAINTY(q) as the following decision problem:

Decision problem CERTAINTY(q)

INPUT: A (possibly inconsistent) database instance $\mathbf{db}$.
QUESTION: Is q certain?

Two approaches for solving CERTAINTY(q)

1. Generate-and-test program Generate all (possibly exponentially many) repairs, and test whether there is one that falsifies q.

Two approaches for solving CERTAINTY(q)

1. Generate-and-test program Generate all (possibly exponentially many) repairs, and test whether there is one that falsifies q.
2. First-order rewriting Construct a new first-order query that says: "q is certain."

Two approaches for solving CERTAINTY(q)

1. Generate-and-test program Generate all (possibly exponentially many) repairs, and test whether there is one that falsifies q.
2. First-order rewriting Construct a new first-order query that says: "q is certain."

First-order rewriting: Example

$$q_0 = \exists X \left(r(\underline{\text{CIKM}, 2021}, X) \land s(\underline{X}, \text{Australia}) \right)$$

"q_0 is certain" = "every possible country Y of every possible city X for CIKM 2021 is equal to Australia":

$$\exists X \left(r(\underline{\text{CIKM}, 2021}, X) \land s(\underline{X}, \text{Australia}) \right) \land$$
$$\forall X \left(r(\underline{\text{CIKM}, 2021}, X) \rightarrow \left(\begin{array}{l} s(\underline{X}, \text{Australia}) \land \\ \forall Y \left(s(\underline{X}, Y) \rightarrow Y = \text{Australia} \right) \end{array} \right) \right)$$

Existence of first-order rewritings

We limit ourselves to sjfBCQ, i.e., the class of self-join-free Boolean conjunctive queries. These are of the form $\exists^* (R_1(\vec{x}_1) \land \cdots \land R_\ell(\vec{x}_\ell))$ such that $i \neq j$ implies $R_i \neq R_j$.

Existence of first-order rewritings

We limit ourselves to sjfBCQ, i.e., the class of self-join-free Boolean conjunctive queries. These are of the form $\exists^* \left(R_1(\vec{x}_1) \wedge \cdots \wedge R_\ell(\vec{x}_\ell)\right)$ such that $i \neq j$ implies $R_i \neq R_j$.

Not all queries in sjfBCQ have a first-order rewriting. The good news:

Theorem ([KW17; KW20])

Given $q \in$ sjfBCQ,

1. *it is decidable whether* CERTAINTY(q) *has a first-order rewriting; and*
2. *a first-order rewriting for* CERTAINTY(q) *can be constructed if it exists.*

Existence of first-order rewritings

We limit ourselves to sjfBCQ, i.e., the class of self-join-free Boolean conjunctive queries. These are of the form
$\exists^* (R_1(\vec{x}_1) \wedge \cdots \wedge R_\ell(\vec{x}_\ell))$ such that $i \neq j$ implies $R_i \neq R_j$.
Not all queries in sjfBCQ have a first-order rewriting. The good news:

Theorem ([KW17; KW20])

Given $q \in$ sjfBCQ,

1. *it is decidable whether* CERTAINTY(q) *has a first-order rewriting; and*
2. *a first-order rewriting for* CERTAINTY(q) *can be constructed if it exists.*

Research question: In Answer Set Programming (ASP), are first-order rewritings more efficient than generic generate-and-test programs?

NP search for a repair that falsifies the query

Let $q_0 := \exists X \left(r(\underline{\text{'CIKM'}, \text{'2021'}}, X) \wedge s(\underline{X}, \text{'Australia'}) \right)$.

```
% Generate a repair of relation r
{ r_repair(Conf, Year, V) : r(Conf, Year, V) } == 1
    :- r(Conf, Year, _).

% Generate a repair of relation s
{ s_repair(City, W) : s(City, W) } == 1
    :- s(City, _).

% Test that generated repair falsifies the query
    :- r_repair('CIKM', '2021', X),
                              s_repair(X,'Australia').
```

Listing 1: Generate-and-test program that searches for a repair that falsifies q_0.

FO algorithm in non-recursive datalog with negation

Let $q_0 := \exists X\,\big(r(\underline{\text{'CIKM', '2020'}}, X) \wedge s(\underline{X}, \text{'Australia'})\big)$.

```
yes :- r('CIKM', '2021', X), not wrongCity(X).

wrongCity(X) :- r(_, _, X), not inAustralia(X).

inAustralia(X) :- s(X, 'Australia'),
                  not outAustralia(X).

outAustralia(X) :- s(X, W), W != 'Australia'.
```

Listing 2: First-order rewriting of q_0 in non-recursive datalog with negation.

Experimental framework

- We fixed a database schema (the one of the running example).
- Our software Conquesto [JLS20] generates all (203 in total) non-equivalent queries on this schema.
- For each query q with a first-order rewriting (194 out of 203), Conquesto generates two ASP programs for solving CERTAINTY(q):
  1. a generate-and-test program that searches for a repair that falsifies q;
  2. a first-order rewriting of q in non-recursive datalog with negation.
- We measure and show runtimes on 'yes'- and 'no'-database instances for CERTAINTY(q), as well as on 'random' database instances [only shown in the paper].
- The ASP solver is clingo [Geb+14].

Results for 'yes'- and 'no'-database instances

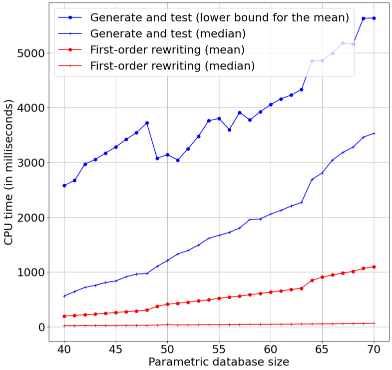

Figure 1: Results for 'yes'-instances (i.e., the query is true in every repair).

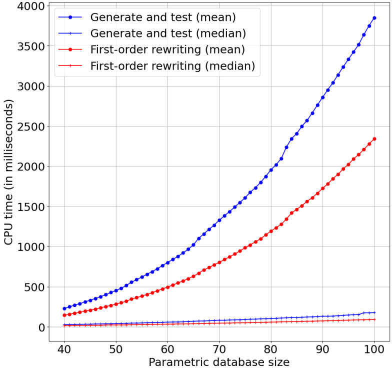

Figure 2: Results for 'no'-instances (i.e., the query is false in some repair).

Conclusion: First-order rewriting outperforms generate-and-test.

Conclusion

- For a Boolean query q, CERTAINTY(q) is the following problem:
  Given a database instance (possibly with primary-key violations), is q true in every repair?
- We asked the research question:
  Are there runtime differences between a straightforward generate-and-test program (in $\mathbf{NP}$*) and first-order rewritings (encoded in non-recursive datalog with negation)?*
- For clingo, our experiments show that the answer to this question is "yes."
- Similar findings were obtained with DLV [LPF11].

[Geb+14] Martin Gebser et al. 'Clingo = ASP + Control: Preliminary Report'. In: *CoRR* abs/1405.3694 (2014).

[JLS20] Jonathan Joertz, Dorian Labeeuw and Gaëtan Staquet. *Conquesto*. 2020. URL: https://github.com/DocSkellington/Conquesto/.

[KW17] Paraschos Koutris and Jef Wijsen. 'Consistent Query Answering for Self-Join-Free Conjunctive Queries Under Primary Key Constraints'. In: *ACM Trans. Database Syst.* 42.2 (2017), 9:1–9:45. DOI: 10.1145/3068334. URL: https://doi.org/10.1145/3068334.

[KW20] Paraschos Koutris and Jef Wijsen. 'Consistent Query Answering for Primary Keys in Datalog'. In: *Theory of Computing Systems* (2020), pp. 1–57. DOI: 10.1007/s00224-020-09985-6. URL: https://doi.org/10.1007/s00224-020-09985-6.

[LPF11] Nicola Leone, Gerald Pfeifer and Wolfgang Faber. *DLV*. 1996-2011. URL: http://www.dlvsystem.com/dlv/.