ASP for Consistent Query Answering

YACINE SAHLI, University of Mons, belgium
JOACHIM SNEESSENS, University of Mons, belgium
MAXIME DANIELS, University of Mons, belgium

Fig. 1

Consistent query answering for inconsistent databases is a running problem. We realized an ASP implementation of the consistent query answering problem and ran some experiments comparing a generate and test method against a first-order rewriting.

CCS Concepts: • **Information systems → Database design and models**; **Database query processing**.

Additional Key Words and Phrases: Answer Set Programming, Consistent Query Answering

ACM Reference Format:
Yacine Sahli, Joachim Sneessens, and Maxime Daniels. 2020. ASP for Consistent Query Answering. In *Galway '20: ACM International Conference on Information and Knowledge Management, October 19–23, 2020, Galway, Ireland*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Galway '20, October 19–23, 2020, Galway, Ireland

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 RELATED WORK

Consistent query answering (CQA) started in 1999 with a seminal paper by Arenas, Bertossi and Chomicki [1]. Two decades of research in CQA have recently been surveyed in [3, 15].

The suitability of Answer Set Programming (ASP) and stable model semantics for CQA has been recognized since the early days in theoretical research [2, 10]. In [14], the authors present a prototype system for CQA that is theoretically founded in ASP.

In CQA, the existence of consistent first-order rewritings, for different classes of queries and integrity constraints, is a recurrent research problem. For self-join-free conjunctive queries and primary keys, the problem has been studied in depth since 2005 [6, 7], and was eventually solved in 2015 [12, 13]. Experiments of CQA with respect to primary keys have been conducted on several prototype systems, including ConQuer [5], EQUIP [11], and CAvSAT [4]. The latter study, in particular, has revealed that discrepancies may exists between the theoretical computational complexity of q and observed empirical performances. In particular, it was observed that a generic SAT-based approach to q may outperform solutions that use first-order rewriting. The main goal of the current paper is to investigate whether such observations also hold within the system of clingo ASP [8, 9].

2 INTRODUCTION

The aim of this article is to present a fair comparison between two methods for solving the problem of CERTAINTY(q). Considering an inconsistent database, a repair is a maximal set of tuples from this database that respects his constraints. The CERTAINTY(q) problem consists in answering the question of knowing if it exists a repair that falsifies the query. Depending on the query, the CERTAINTY(q) problem can have a first order complexity. For the queries that are in first order, we want to compare the efficiency of the generate-and-test method and of the first order rewriting method.

The comparison is realised here on few queries with ASP. For each query, we have measured the execution times of the two methods on databases of different sizes, while distinguishing the yes-instances (databases for which the CERTAINTY(q) problem is true) and the no-instances.

3 CHOSEN QUERIES

To make a one to one comparison with the results found by Akhil A. Dixit and Phokion G. Kolaitis in their "A SAT-Based System for Consistent Query Answering", we decided to reuse the same FO-rewritable queries they used to prove that the KW-fo rewriting can be more efficient by using ASP instead of SQL.

For a more simple implementation, we remove the free variables of the queries. At the end, here are the queries used for the tests we performed.

$$q_1 := \exists x, y, z, v, w(R_1(\underline{x}, y, z) \wedge R_2(\underline{y}, v, w))$$
$$q_2 := \exists x, y, z, v, u, p(R_1(\underline{x}, y, z) \wedge R_3(\underline{y}, v) \wedge R_2(\underline{v}, u, p))$$
$$q_3 := \exists x, y, z, v, u, (R_1(\underline{x}, y, z) \wedge R_2(\underline{y}, v, d))$$

Notice that d is a constant in q_3.

4 DATABASE GENERATION

To test the difference between the two methods, we have to get some databases to test them. We used a script in Python to generate random tuples that match with the query we want to test. We

add a parameter "inconsistency" that define the inconsistency of the db and a parameter that define if we want a no instance or a yes instance. A yes instance is a database that there is at least 1 consistant answer for the query, at the opposite, a no instance is a database that don't contains any tuple that is an answer (means no answer or no consistant answer). Example : for the first query, the generator will, for a yes instance, generate some r1(x,y,z) and r2(y,v,w) where x's have unique y and z response and y's that have unique v and w response and where the y in r1 is the same that the y in r2. A no instance would be a x's that define differents y and z for all the x's define.

Example of yes instance

$$r1(x1, y1, z1).r2(y1, v1, w1).$$
$$r1(x2, y1, z1).r2(y2, v1, w1).$$

Example of no instance

$$r1(x1, y1, z1).r2(y1, v1, w1).$$
$$r1(x1, y2, z2).r2(y2, v2, w2).$$

All the 6 databases are made that way with a certain ratio of inconsistency.

5 QUERIES IMPLEMENTATION IN ASP

5.1 First query

FO Rewriting:

```
q1:-r1(X,Y,Z),not p1(X).
p1(X):-r1(X,Y,Z),not p2(Y).
p2(Y):-r2(Y,V,W).

certainty:-q1.
certainty:-not certainty.

#show certainty/0.
```

Generate and Test:

```
1{rr1(X,Y,Z):r1(X,Y,Z)}1:-r1(X,_,_).
1{rr2(X,Y,Z):r2(X,Y,Z)}1:-r2(X,_,_).
:-rr1(X,Y,Z),rr2(Y,V,W).
```

5.2 Second query

FO Rewriting:

```
q1 :- r1(X,_,_), not p1(X).
p1(X) :- r1(X,Y,_), not q3(Y).
q3(Y) :- r3(Y,_), not q2(Y).
q2(Y) :- r3(Y,V), not q1(V).
q1(V) :- r2(V,_,_).
```

```
certainty:-q1.
certainty:-not certainty.

#show certainty/0.
```

Generate and Test:

```
1{rr1(X,Y,Z):r1(X,Y,Z)}1:-r1(X,_,_).
1{rr2(X,Y,Z):r2(X,Y,Z)}1:-r2(X,_,_).
1{rr3(X,Y):r3(X,Y)}1:-r3(X,_).
:-rr1(X,Y,Z),rr3(Y,V),rr2(V,U,D).
```

5.3 Third query

FO Rewriting:

```
q1:-r1(X,Y,Z),not p1(X).
p1(X):-r1(X,Y,Z),not p2(Y).
p2(Y):-r4(Y,V,W),W==w.

certainty:-q1.
certainty:-not certainty.

#show certainty/0.
```

Generate and Test:

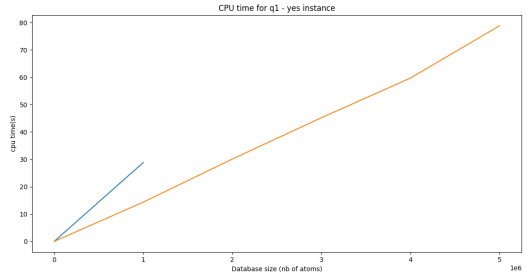
```
1{rr1(X,Y,Z):r1(X,Y,Z)}1:-r1(X,_,_).
1{rr4(X,Y,Z):r4(X,Y,Z)}1:-r4(X,_,_).
:-rr1(X,Y,Z),rr4(Y,V,w).
```

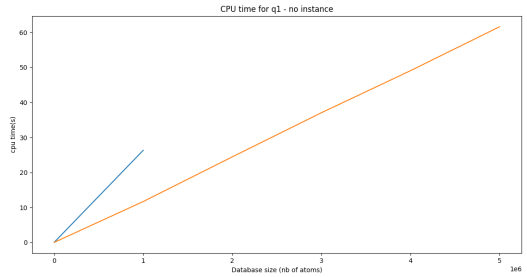
6 COMPARISON TOOLS

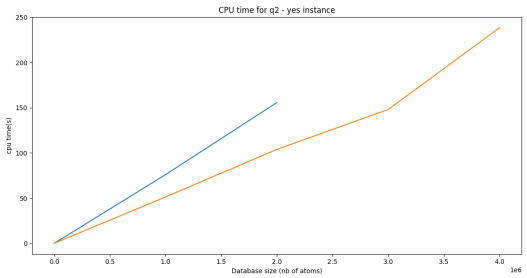
In order to compare the FO rewriting and the generate test ways we have to get a tool that computes the performance of the queries. We used the clingo tool that is a compiler and executor for logical programmation. With this tool, we are able to get some performance computings as cpu time.

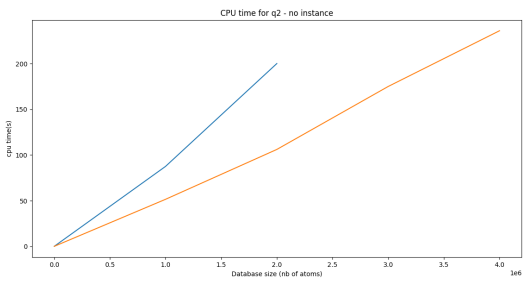
7 RESULTS

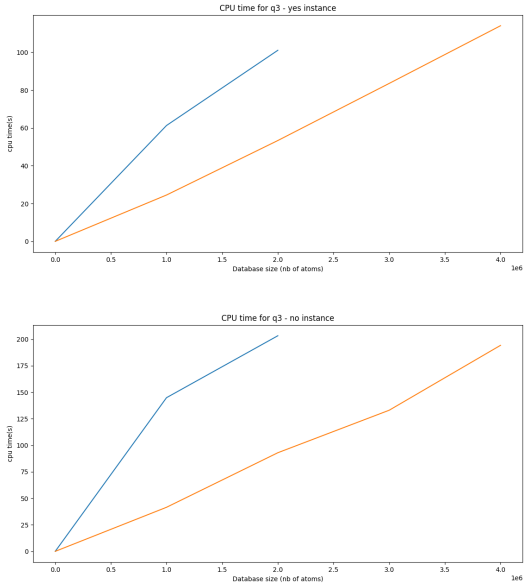
So we tried every query with databases with 20% inconsistency varying in size, from 1 million to 5 and plotted the results of the cpu time by numbers of entries in the query. For each graph, the blue line corresponds to the time taken by the generate-and-test method and the orange line to the time taken by the FO method. When a result is not in the graph, that means that the execution of the program was interrupted for insufficient memory. For example, the times for the generate-and-test for the yes-instance for $q1$ for the databases size greater than 1 million are absent.


CPU time for q1 - yes instance
cpu time(s)
Database size (nb of atoms)
1e6


CPU time for q1 - no instance
cpu time(s)
Database size (nb of atoms)
1e6


CPU time for q2 - yes instance
cpu time(s)
Database size (nb of atoms)
1e6


CPU time for q2 - no instance
cpu time(s)
Database size (nb of atoms)
1e6



We see that the fo rewriting leads to better results, in terms of cpu time, that the generate-and-test method. We tried with less than 1 millions entries and with less than 20% inconsistency but the results were the same. In general the growing of the cpu times are linear and the generate test computed cpu times grow faster the the FO rewriting. We can say that no matter this is a yes or a no instance, the FO rewriting is faster because the yes or no instance seems to not have any consequence on the cp time.

8 CONCLUSION

In order to prove that which one between the FO rewriting or the generate test method is the best to solve our problem (certainty(q)) we experiment on generated databases and see that with 20% of inconsistency the FO rewriting is the best, we don't have to going with larger inconsistency because the paper we were based on was experimenting on a 20% inconsistency generated database. We rewrited 3 first-order rewritable queries in ASP and showed that the fo rewriting are more efficient than a generate and test method. The first order rewriting consumes a lot less memory than the generate and test which cannot run on 8GB of RAM on a database containing 3 millions or more entries.

REFERENCES

[1] Marcelo Arenas, Leopoldo E. Bertossi, and Jan Chomicki. 1999. Consistent Query Answers in Inconsistent Databases. In *Proceedings of the Eighteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 31 - June 2, 1999, Philadelphia, Pennsylvania, USA*, Victor Vianu and Christos H. Papadimitriou (Eds.). ACM Press, 68–79. https://doi.org/10.1145/303976.303983

[2] Marcelo Arenas, Leopoldo E. Bertossi, and Jan Chomicki. 2003. Answer sets for consistent query answering in inconsistent databases. *Theory Pract. Log. Program.* 3, 4-5 (2003), 393–424. https://doi.org/10.1017/S1471068403001832

[3] Leopoldo E. Bertossi. 2019. Database Repairs and Consistent Query Answering: Origins and Further Developments. In *Proceedings of the 38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019*, Dan Suciu, Sebastian Skritek, and Christoph Koch (Eds.). ACM, 48–58. https://doi.org/10.1145/3294052.3322190

[4] Akhil A. Dixit and Phokion G. Kolaitis. 2019. A SAT-Based System for Consistent Query Answering. In *Theory and Applications of Satisfiability Testing - SAT 2019 - 22nd International Conference, SAT 2019, Lisbon, Portugal, July 9-12, 2019, Proceedings (Lecture Notes in Computer Science)*, Mikolás Janota and Inês Lynce (Eds.), Vol. 11628. Springer, 117–135. https://doi.org/10.1007/978-3-030-24258-9_8

[5] Ariel Fuxman, Elham Fazli, and Renée J. Miller. 2005. ConQuer: Efficient Management of Inconsistent Databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, Baltimore, Maryland, USA, June 14-16, 2005*, Fatma Özcan (Ed.). ACM, 155–166. https://doi.org/10.1145/1066157.1066176

[6] Ariel Fuxman and Renée J. Miller. 2005. First-Order Query Rewriting for Inconsistent Databases. In *Database Theory - ICDT 2005, 10th International Conference, Edinburgh, UK, January 5-7, 2005, Proceedings (Lecture Notes in Computer Science)*, Thomas Eiter and Leonid Libkin (Eds.), Vol. 3363. Springer, 337–351. https://doi.org/10.1007/978-3-540-30570-5_23

[7] Ariel Fuxman and Renée J. Miller. 2007. First-order query rewriting for inconsistent databases. *J. Comput. Syst. Sci.* 73, 4 (2007), 610–635. https://doi.org/10.1016/j.jcss.2006.10.013

[8] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. 2014. Clingo = ASP + Control: Preliminary Report. *CoRR* abs/1405.3694 (2014). arXiv:1405.3694 http://arxiv.org/abs/1405.3694

[9] Martin Gebser, Roland Kaminski, Arne König, and Torsten Schaub. 2011. Advances in *gringo* Series 3. In *Logic Programming and Nonmonotonic Reasoning - 11th International Conference, LPNMR 2011, Vancouver, Canada, May 16-19, 2011. Proceedings (Lecture Notes in Computer Science)*, James P. Delgrande and Wolfgang Faber (Eds.), Vol. 6645. Springer, 345–351. https://doi.org/10.1007/978-3-642-20895-9_39

[10] Gianluigi Greco, Sergio Greco, and Ester Zumpano. 2003. A Logical Framework for Querying and Repairing Inconsistent Databases. *IEEE Trans. Knowl. Data Eng.* 15, 6 (2003), 1389–1408. https://doi.org/10.1109/TKDE.2003.1245280

[11] Phokion G. Kolaitis, Enela Pema, and Wang-Chiew Tan. 2013. Efficient Querying of Inconsistent Databases with Binary Integer Programming. *PVLDB* 6, 6 (2013), 397–408. https://doi.org/10.14778/2536336.2536341

[12] Paraschos Koutris and Jef Wijsen. 2015. The Data Complexity of Consistent Query Answering for Self-Join-Free Conjunctive Queries Under Primary Key Constraints. In *Proceedings of the 34th ACM Symposium on Principles of Database Systems, PODS 2015, Melbourne, Victoria, Australia, May 31 - June 4, 2015*, Tova Milo and Diego Calvanese (Eds.). ACM, 17–29. https://doi.org/10.1145/2745754.2745769

[13] Paraschos Koutris and Jef Wijsen. 2017. Consistent Query Answering for Self-Join-Free Conjunctive Queries Under Primary Key Constraints. *ACM Trans. Database Syst.* 42, 2 (2017), 9:1–9:45. https://doi.org/10.1145/3068334

[14] Mónica Caniupán Marileo and Leopoldo E. Bertossi. 2010. The consistency extractor system: Answer set programs for consistent query answering in databases. *Data Knowl. Eng.* 69, 6 (2010), 545–572. https://doi.org/10.1016/j.datak.2010.01.005

[15] Jef Wijsen. 2019. Foundations of Query Answering on Inconsistent Databases. *SIGMOD Rec.* 48, 3 (2019), 6–16. https://doi.org/10.1145/3377391.3377393