Semantic Acyclicity Under Constraints

Semantic Acyclicity Under Constraints*

Pablo Barceló[1], Georg Gottlob[2], and Andreas Pieris[3]

[1] Center for Semantic Web Research & DCC, University of Chile
pbarcelo@dcc.uchile.cl
[2] Department of Computer Science, University of Oxford georg.gottlob@cs.ox.ac.uk
[3] Institute of Information Systems, Vienna University of Technology
pieris@dbai.tuwien.ac.at

Query optimization is a fundamental database task that amounts to transform a query into one that is arguably more efficient to evaluate. The database theory community has developed several principled methods for optimization of conjunctive queries (CQs), many of which are based on *static-analysis* tasks such as containment [1]. In a nutshell, such methods compute a *minimal* equivalent version of a CQ, where minimality refers to number of atoms. As argued by Abiteboul, Hull, and Vianu [1], this provides a theoretical notion of "true optimality" for the reformulation of a CQ, as opposed to practical approaches based on heuristics. For each CQ, the minimal equivalent CQ is its *core* [16]. Although the static analysis tasks that support CQ minimization are NP-hard [9], this is not a problem for real-life applications, as the input CQ is small.

It is known, on the other hand, that semantic information about the data, in the form of integrity constraints, alleviates query optimization by reducing the space of possible reformulations. In the previous analysis, however, constraints play no role, as CQ equivalence is defined over *all* databases. Adding constraints yields a refined notion of CQ equivalence, which holds over those databases that satisfy a given set of constraints only. But finding a minimal equivalent CQ in this context is notoriously more difficult than before. This is because basic static analysis tasks such as containment become undecidable when considered in full generality. This motivated a long research program for finding larger "islands of decidability" of such containment problem, based on syntactical restrictions on constraints [2, 6–8, 17, 18].

An important shortcoming of the previous approach, however, is that there is no theoretical guarantee that the minimized version of a CQ is in fact easier to evaluate (recall that, in general, CQ evaluation is NP-complete [9]). We know, on the other hand, quite a bit about classes of CQs which can be evaluated efficiently. It is thus a natural problem to ask whether constraints can be used to reformulate a CQ as one in such tractable classes, and if so, what is the cost of computing such reformulation. Following Abiteboul et al. [1], this would provide us with a theoretical guarantee of "true efficiency" for those reformulations. Here we concentrate on one of the oldest and most studied tractability conditions for CQs; namely, *acyclicity*. It is known that acyclic CQs can be evaluated in linear time [21].

More formally, let us write $q \equiv_\Sigma q'$ whenever CQs q and q' are equivalent over all databases that satisfy Σ. In this work we study the following problem:

* This paper is a short version of [4].

PROBLEM : SEMANTIC ACYCLICITY	
INPUT :	A CQ q and a finite set Σ of constraints.
QUESTION :	Is there an acyclic CQ q' s.t. $q \equiv_\Sigma q'$?

We study this problem for the two most important classes of database constraints:

1. *Tuple-generating dependencies* (tgds), i.e., expressions of the form $\forall\bar{x}\forall\bar{y}(\phi(\bar{x},\bar{y}) \rightarrow \exists\bar{z}\psi(\bar{x},\bar{z}))$, where ϕ and ψ are conjuntions of atoms. Tgds subsume the important class of referential integrity constraints (or inclusion dependencies).
2. *Equality-generating dependencies* (egds), i.e., expressions of the form $\forall\bar{x}(\phi(\bar{x}) \rightarrow y = z)$, where ϕ is a conjunction of atoms and y, z are variables in $\bar{x}$. Egds subsume keys and functional dependencies (FDs).

A useful aspect of tgds and egds is that containment under them can be studied in terms of the *chase procedure* [19].

Coming back to semantic acyclicity, the main problem we study is, of course, decidability. Since basic reasoning with tgds and egds is, in general, undecidable, we cannot expect semantic acyclicity to be decidable for arbitrary such constraints. Thus, we concentrate on the following question:

Decidability: For which classes of tgds and egds is the problem of semantic acyclicity decidable? In such cases, what is the computational cost of the problem?

Since semantic acyclicity is defined in terms of CQ equivalence under constraints, and the latter has received a lot of attention, it is relevant also to study the following:

Relationship to CQ equivalence: What is the relationship between CQ equivalence under constraints and semantic acyclicity under constraints? Is the latter decidable for each class of tgds and egds for which the former is decidable?

Finally, we want to understand to what extent semantic acyclicity helps CQ evaluation. Although an acyclic reformulation of a CQ can be evaluated efficiently, computing such reformulation might be expensive. Thus, it is relevant to study the following:

Evaluation: What is the computational cost of evaluating semantically acyclic CQs under constraints?

Semantic acyclicity in the absence of constraints. The semantic acyclicity problem in the absence of dependencies (i.e., checking whether a CQ q is equivalent to an acyclic one over the set of all databases) is by now well-understood. Regarding decidability, it is easy to prove that a CQ q is semantically acyclic iff its core q' is acyclic. (Recall that such q' is the minimal equivalent CQ to q). It follows that checking semantic acyclicity in the absence of constraints is NP-complete (see, e.g., [5]). Regarding evaluation, semantically acyclic CQs can be evaluated efficiently [10, 11, 15].

The relevance of constraints. In the absence of constraints a CQ q is equivalent to an acyclic one iff its core q' is acyclic. Thus, the only reason why q is not acyclic in the first place is because it has not been minimized. This tells us that in this context semantic

acyclicity is not different from usual minimization. The presence of constraints, on the other hand, yields a more interesting notion of semantic acyclicity. This is because constraints can be applied on CQs to produce acyclic reformulations of them.

Example 1. This simple example helps understanding the role of tgds when reformulating CQs as acyclic ones. Consider a database that stores information about customers, records, and musical styles. The relation `Interest` contains pairs (c, s) such that customer c has declared interest in style s. The relation `Class` contains pairs (r, s) such that record r is of style s. Finally, the relation `Owns` contains a pair (c, r) when customer c owns record r. Consider now a CQ $q(x, y)$ defined as follows:

$$\exists z\big(\texttt{Interest}(x, z) \wedge \texttt{Class}(y, z) \wedge \texttt{Owns}(x, y)\big).$$

This query asks for pairs (c, r) such that customer c owns record r and has expressed interest in at least one of the styles with which r is associated. This CQ is a core but it is not acyclic. Thus, from our previous observations it is not equivalent to an acyclic CQ (in the absence of constraints).

Assume now that we are told that this database contains compulsive music collectors only. In particular, each customer owns every record that is classified with a style in which he/she has expressed interest. This means that the database satisfies the tgd:

$$\tau \;=\; \texttt{Interest}(x, z), \texttt{Class}(y, z) \;\rightarrow\; \texttt{Owns}(x, y).$$

We can now easily reformulate $q(x, y)$ as the following acyclic CQ $q'(x, y)$:

$$\exists z\big(\texttt{Interest}(x, z) \wedge \texttt{Class}(y, z)\big).$$

Notice that q and q' are in fact equivalent over every database that satisfies τ. ∎

Contributions. We observe that semantic acyclicity under constraints is not only more powerful, but also theoretically more challenging than in the absence of them. We start by studying decidability.

Results for tgds: Having a decidable CQ containment problem is a necessary condition for semantic acyclicity to be decidable under tgds.[4] Surprisingly enough, it is not a sufficient condition. This means that, contrary to what one might expect, there are natural classes of tgds for which CQ containment but not semantic acyclicity is decidable. In particular, this is the case for the well-known class of *full* tgds (i.e., tgds without existentially quantified variables in the head). In conclusion, we cannot directly export techniques from CQ containment to deal with semantic acyclicity.

In view of the previous results, we concentrate on classes of tgds that (a) have a decidable CQ containment problem, and (b) do not contain the class of full tgds. These restrictions are satisfied by several expressive languages considered in the literature. Such languages can be classified into three main families depending on the techniques used for studying their containment problem: (i) *guarded* tgds [6], which contain inclusion and linear dependencies, (ii) *non-recursive* [12], and (iii) *sticky* sets of tgds [7].

Instead of studying such languages one by one, we identify two semantic criteria that yield decidability for the semantic acyclicity problem, and then show that each one of the languages satisfies one such criteria.

[4] Modulo some mild technical assumptions; see the extended version for more details.

- The first criterion is *acyclicity-preserving chase*. This is satisfied by those tgds for which the application of the chase over an acyclic instance preserves acyclicity. Guarded tgds enjoy this property. From the analysis we conclude that semantic acyclicity under guarded tgds is decidable and has the same complexity as CQ containment: 2EXPTIME-complete, and NP-complete for a fixed schema.
- The second criterion is *rewritability by unions of CQs (UCQs)*. Intuitively, a class $\mathbb{C}$ of sets of tgds has this property if the CQ containment problem under a set in $\mathbb{C}$ can always be reduced to a UCQ containment problem without constraints. Non-recursive and sticky sets of tgds enjoy this property. In the first case the complexity matches that of CQ containment: NEXPTIME-complete, and NP-complete if the schema is fixed. In the second case, we get a NEXPTIME upper bound and an EXPTIME lower bound. For a fixed schema the problem is NP-complete.

The NP bounds for tgds in such decidable classes (under a fixed schema) can be seen as positive results: By spending exponential time in the size of the (small) query, we can not only minimize it, but also find an acyclic reformulation if one exists.

Results for egds: After showing that the techniques developed for tgds cannot be applied for showing the decidability of semantic acyclicity under egds, we focus on the class of keys over unary and binary predicates and we establish a positive result, namely semantic acyclicity is NP-complete. We prove this by showing that in such context keys have acyclicity-preserving chase. Interestingly, this positive result can be extended to unary functional dependencies (over unconstrained signatures); this result has been established independently by Figueira [13]. We leave open whether the problem of semantic acyclicity under arbitrary egds, or even keys over arbitrary schemas, is decidable.

Evaluation: For tgds for which semantic acyclicity is decidable (guarded, non-recursive, sticky), we can use the following algorithm to evaluate a semantically acyclic CQ q over a database D that satisfies the constraints Σ: (1) Convert q into an equivalent acyclic CQ q' under Σ, (2) evaluate q' on D, and (3) return $q(D) = q'(D)$. The running time is $O(|D| \cdot f(|q|, |\Sigma|))$, where f is a double-exponential function (since q' can be computed in double-exponential time for each one of the classes mentioned above and acyclic CQs can be evaluated in linear time). This constitutes a *fixed-parameter tractable algorithm* for evaluating q on D. No such algorithm is believed to exist for CQ evaluation [20]; thus, semantically acyclic CQs under these constraints behave better than the general case in terms of evaluation. However, in the absence of constraints one can do better: Evaluating semantically acyclic CQs in such context is feasible in polynomial time. It is natural to ask if this also holds in the presence of constraints. This is the case for guarded tgds and FDs. For the other classes the problem remains to be investigated.

Finite vs. infinite databases. The results mentioned above interpret the notion of CQ equivalence (and, thus, semantic acyclicity) over arbitrary (*finite* or *infinite*) databases. The reason is the wide application of the chase we make in our proofs, which characterizes CQ equivalence under arbitrary databases only. This is not a serious problem though, as all the particular classes of tgds for which we prove decidability in the paper (i.e., guarded, non-recursive, sticky) are *finitely controllable* [3, 14]. This means that CQ equivalence under arbitrary databases and under finite databases coincide. Thus, the results we obtain for such classes can be directly exported to the finite case.

Acknowledgements. Barceló would like to thank D. Figueira, M. Romero, S. Rudolph, and N. Schweikardt for insightful discussions about the nature of semantic acyclicity under constraints. Barceló is funded by the Millenium Nucleus Center for Semantic Web Research under grant NC120004. Gottlob is supported by the EPSRC Programme Grant EP/M025268/ "VADA: Value Added Data Systems – Principles and Architecture". Pieris is supported by the Austrian Science Fund (FWF), projects P25207-N23 and Y698, and Vienna Science and Technology Fund (WWTF), project ICT12-015.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley (1995)
2. Baget, J.F., Mugnier, M.L., Rudolph, S., Thomazo, M.: Walking the complexity lines for generalized guarded existential rules. In: IJCAI. pp. 712–717 (2011)
3. Bárány, V., Gottlob, G., Otto, M.: Querying the guarded fragment. Logical Methods in Computer Science 10(2) (2014)
4. Barceló, P., Gottlob, G., Pieris, A.: Semantic acyclicity under constraints. In: PODS (2016), to appear
5. Barceló, P., Romero, M., Vardi, M.Y.: Semantic acyclicity on graph databases. In: PODS. pp. 237–248 (2013)
6. Calì, A., Gottlob, G., Kifer, M.: Taming the infinite chase: Query answering under expressive relational constraints. J. Artif. Intell. Res. 48, 115–174 (2013)
7. Calì, A., Gottlob, G., Pieris, A.: Towards more expressive ontology languages: The query answering problem. Artif. Intell. 193, 87–128 (2012)
8. Calvanese, D., De Giacomo, G., Lenzerini, M.: Conjunctive query containment and answering under description logic constraints. ACM Trans. Comput. Log. 9(3) (2008)
9. Chandra, A.K., Merlin, P.M.: Optimal implementation of conjunctive queries in relational data bases. In: STOC. pp. 77–90 (1977)
10. Chen, H., Dalmau, V.: Beyond hypertree width: Decomposition methods without decompositions. In: CP. pp. 167–181 (2005)
11. Dalmau, V., Kolaitis, P.G., Vardi, M.Y.: Constraint satisfaction, bounded treewidth, and finite-variable logics. In: CP. pp. 310–326 (2002)
12. Fagin, R., Kolaitis, P.G., Miller, R.J., Popa, L.: Data exchange: Semantics and query answering. Theor. Comput. Sci. 336(1), 89–124 (2005)
13. Figueira, D.: Semantically acyclic conjunctive queries under functional dependencies. In: LICS (2016), to appear
14. Gogacz, T., Marcinkowski, J.: Converging to the chase - A tool for finite controllability. In: LICS. pp. 540–549 (2013)
15. Gottlob, G., Greco, G., Marnette, B.: Hyperconsistency width for constraint satisfaction: Algorithms and complexity results. In: Graph Theory, Computational Intelligence and Thought. pp. 87–99 (2009)
16. Hell, P., Nešetřil, J.: Graphs and Homomorphisms. Oxford University Press (2004)
17. Johnson, D.S., Klug, A.C.: Testing containment of conjunctive queries under functional and inclusion dependencies. J. Comput. Syst. Sci. 28(1), 167–189 (1984)
18. Krötzsch, M., Rudolph, S.: Extending decidable existential rules by joining acyclicity and guardedness. In: IJCAI. pp. 963–968 (2011)
19. Maier, D., Mendelzon, A.O., Sagiv, Y.: Testing implications of data dependencies. ACM Trans. Database Syst. 4(4), 455–469 (1979)
20. Papadimitriou, C.H., Yannakakis, M.: On the complexity of database queries. J. Comput. Syst. Sci. 58(3), 407–427 (1999)
21. Yannakakis, M.: Algorithms for acyclic database schemes. In: VLDB. pp. 82–94 (1981)