Semantic Optimization of Conjunctive Queries

Semantic Optimization of Conjunctive Queries

PABLO BARCELÓ, Pontificia Universidad Católica de Chile, Chile and IMFD Chile
DIEGO FIGUEIRA, University of Bordeaux, CNRS, Bordeaux INP, LaBRI, France
GEORG GOTTLOB, University of Oxford, UK
ANDREAS PIERIS, University of Edinburgh, UK

This work deals with the problem of semantic optimization of the central class of conjunctive queries (CQs). Since CQ evaluation is NP-complete, a long line of research has focussed on identifying fragments of CQs that can be efficiently evaluated. One of the most general such restrictions corresponds to generalized hypetreewidth bounded by a fixed constant $k \geq 1$; the associated fragment is denoted GHW_k. A CQ is semantically in GHW_k if it is equivalent to a CQ in GHW_k. The problem of checking whether a CQ is semantically in GHW_k has been studied in the constraint-free case, and it has been shown to be NP-complete. However, in case the database is subject to constraints such as tuple-generating dependencies (TGDs) that can express, e.g., inclusion dependencies, or equality-generating dependencies (EGDs) that capture, e.g., key dependencies, a CQ may turn out to be semantically in GHW_k under the constraints, while not being semantically in GHW_k without the constraints. This opens avenues to new query optimization techniques. In this paper, we initiate and develop the theory of semantic optimization of CQs under constraints. More precisely, we study the following natural problem: Given a CQ and a set of constraints, is the query semantically in GHW_k, for a fixed $k \geq 1$, under the constraints, or, in other words, is the query equivalent to one that belongs to GHW_k over all those databases that satisfy the constraints? We show that, contrary to what one might expect, decidability of CQ containment is a necessary but not a sufficient condition for the decidability of the problem in question. In particular, we show that checking whether a CQ is semantically in GHW_1 is undecidable in the presence of full TGDs (i.e., Datalog rules) or EGDs. In view of the above negative results, we focus on the main classes of TGDs for which CQ containment is decidable, and that do not capture the class of full TGDs, i.e., guarded, non-recursive and sticky sets of TGDs, and show that the problem in question is decidable, while its complexity coincides with the complexity of CQ containment. We also consider key dependencies over unary and binary relations, and show that the problem in question is decidable in elementary time. Furthermore, we investigate whether being semantically in GHW_k alleviates the cost of query evaluation. Finally, in case a CQ is not semantically in GHW_k, we discuss how it can be approximated via a CQ that falls in GHW_k in an optimal way. Such approximations might help finding "quick" answers to the input query when exact evaluation is intractable.

ACM Reference Format:
Pablo Barceló, Diego Figueira, Georg Gottlob, and Andreas Pieris. 2020. Semantic Optimization of Conjunctive Queries. *J. ACM* 1, 1 (September 2020), 60 pages. https://doi.org/0000001.0000001

Authors' addresses: Pablo Barceló, Pontificia Universidad Católica de Chile, Institute for Mathematical and Computational Engineering, School of Engineering and Faculty of Mathematics, Avenida Vicuña Mackenna 4860, Macul, Santiago, 7820436, Chile, IMFD Chile, Millennium Institute for Foundational Research on Data, pbarcelo@ing.puc.cl; Diego Figueira, University of Bordeaux, CNRS, Bordeaux INP, LaBRI, UMR 5800, Talence, F-33400, France, diego.figueira@labri.fr; Georg Gottlob, University of Oxford, Wolfson Building, Parks Road, Oxford, OX1 3QD, UK, georg.gottlob@cs.ox.ac.uk; Andreas Pieris, University of Edinburgh, Informatics Forum, Crichton Street, Edinburgh, EH8 9AB, UK, apieris@inf.ed.ac.uk.

0004-5411/2020/9-ART $15.00
https://doi.org/0000001.0000001

1 INTRODUCTION

Query optimization is a fundamental database task that amounts to transforming 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, called the *core* [32], where minimality refers to the 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 considerations based on heuristics. Although the static analysis tasks that support CQ minimization are NP-complete [16], this is not a major problem for real-life applications as the input (i.e., the 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 above 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. This is because basic static analysis tasks such as containment become undecidable when considered in full generality. This has motivated a long line of research for finding larger "islands of decidability" of such containment problem based on syntactic restrictions on constraints; see, e.g., [2, 11, 13, 15, 33].

1.1 Semantic Generalized Hypertreewidth

An important shortcoming of CQ minimization is that there is no theoretical guarantee that the minimized CQ is actually easier to evaluate (recall that, in general, CQ evaluation is NP-complete [16]). We know, on the other hand, quite a bit about classes of CQs that can be evaluated efficiently. It is thus a natural question 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 a reformulation. Following Abiteboul et al., this would provide us with a theoretical guarantee of "true efficiency" for those reformulations.

We focus on one of the most general and widely studied tractability conditions for CQs, that is, *bounded generalized hypertreewidth* [29]; we fix an integer $k \geq 1$, and write GHW_k for the class of CQs of generalized hypertreewidth bounded by k. Notice that the notion of generalized hypertreewidth subsumes the well studied class of *acyclic* CQs [42]; in fact, acyclicity corresponds to the lowest level GHW_1 of the hierarchy. The main problem that we study is the following; as usual, we write $q \equiv_\Sigma q'$ whenever the CQs q and q' are equivalent over all databases that satisfy Σ:

PROBLEM :	Semantic Generalized HypetreeWidth k
INPUT :	A CQ q and a finite set Σ of constraints.
QUESTION :	Is there a CQ $q' \in \mathsf{GHW}_k$ such that $q \equiv_\Sigma q'$?

We study the above problem, from now on abbreviated as SemGHW_k, 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 conjunctions 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).

1.2 The Relevance of Constraints

The constraint-free version of SemGHW_k, i.e., checking whether a CQ q is equivalent to one that falls in GHW_k over the set of *all* databases, is rather well-understood. Regarding decidability, it is not difficult to prove that a CQ q is semantically in GHW_k if and only if its core q' is in GHW_k. (Recall that such q' is the minimal equivalent CQ to q). It is actually known that SemGHW_k in the absence of constraints is NP-complete (see, e.g., [8]). Regarding evaluation, CQs that are semantically in GHW_k can be evaluated efficiently [17, 19, 28]. From the above discussion, the only reason why q is not in GHW_k in the first place is because it has not been minimized. This simply tells us that in the constraint-free case being semantically in GHW_k is not really different from usual minimization. The presence of constraints, on the other hand, yields a more interesting notion of being semantically in GHW_k. The reason is that the constraints can be applied on CQs to produce reformulations of them that fall in GHW_k.

Example 1.1. This example illustrates how constraints can be used to reformulate a CQ as an acyclic one. Consider a database that stores information about customers, records, and musical styles. The relation Interest holds pairs (c, s) such that the customer c has declared interest in style s. The relation Class contains pairs (r, s) such that the record r is of style s. Finally, the relation Owns contains pairs (c, r) such that the customer c owns the record r. Consider now the CQ

$$q(x, y) \;=\; \exists z\big(\mathrm{Interest}(x, z) \wedge \mathrm{Class}(y, z) \wedge \mathrm{Owns}(x, y)\big),$$

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

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

With this information at hand, we can easily reformulate q as the acyclic CQ:

$$q'(x, y) \;=\; \exists z\big(\mathrm{Interest}(x, z) \wedge \mathrm{Class}(y, z)\big).$$

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

1.3 Research Challenges

Since basic reasoning with TGDs and EGDs is, in general, undecidable, we cannot expect SemGHW_k to be decidable for arbitrary such constraints. Thus, we ask the following:

Decidability: For which classes of TGDs and EGDs is SemGHW_k decidable? In such cases, what is the computational cost of the problem?

Since SemGHW_k is defined in terms of CQ equivalence under constraints, and the latter has received a lot of attention, another relevant question is the following:

CQ equivalence vs. SemGHW_k*:* What is the relationship between CQ equivalence and SemGHW_k? Is the latter decidable for each class of TGDs/EGDs for which the former is decidable? If this is the case, then one can transfer the mature theory of CQ equivalence to tackle SemGHW_k.

Another interesting issue is to what extent the notion of being semantically in GHW_k helps CQ evaluation. Although a GHW_k reformulation of a CQ can be evaluated efficiently, computing such a reformulation might be very expensive. Thus, it is relevant to study the following question:

Evaluation: What is the computational cost of evaluating CQs that are semantically in GHW_k under a given set of constraints?

Finally, in case a query is not semantically in GHW_k, it would be beneficial if it can be approximated via a CQ in GHW_k in an optimal way. Computing and evaluating such approximations might be useful for finding "quick" (i.e., fixed-parameter tractable) answers to the input query when exact evaluation is infeasible. Therefore, it is interesting to investigate the following questions:

Approximations: Is it possible to optimally approximate a CQ that is not semantically in GHW_k via a CQ that falls in GHW_k? If this is the case, what is the computational cost of computing such approximations? Does this help CQ evaluation?

1.4 Our Contributions

We first observe that the notion of being semantically in GHW_k under constraints is not only more powerful, but also technically more challenging than in the absence of constraints. We start by studying decidability, and in the process we also clarify the relationship between CQ equivalence and SemGHW_k. We then concentrate on evaluation and approximations.

Results for TGDs: Under TGDs, having a decidable CQ containment problem is a necessary condition for SemGHW_k to be decidable.[1] Surprisingly enough, it is not a sufficient condition. In particular, contrary to what one might expect, there are natural classes of TGDs for which CQ containment is decidable but SemGHW_1 is not. This is the case for the well-known class of *full* TGDs (i.e., TGDs without existentially quantified variables). In conclusion, we cannot directly export techniques from CQ containment to deal with semantic acyclicity.

In view of the above 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: (i) *guarded* [11], (ii) *non-recursive* [22], and (iii) *sticky* sets of TGDs [13]. Notice that guarded and sticky sets of TGDs generalize the well-known class of *inclusion dependencies*. Instead of studying such languages one by one, we identify two semantic criteria that yield decidability for SemGHW_k, and then show that each one of the above languages satisfies one such criterion.

- The first criterion is *generalized hypertreewidth preserving chase*. This is satisfied by those TGDs for which the application of the chase does not increase the generalized hypertreewidth of the input instance. Guarded TGDs, with only one atom in the right-hand side,[2] enjoy this property. We establish that in this case SemGHW_k is decidable and has the same complexity as CQ containment: 2ExpTime-complete, ExpTime-complete in case of bounded arity, and NP-complete for a fixed schema.
- The second criterion is *UCQ rewritability*; as usual, UCQ stands for union of CQs. Intuitively, a class of sets of TGDs has this property if the CQ containment problem under that class can be reduced to the UCQ containment problem without constraints. Non-recursive and sticky sets of TGDs enjoy this property. In the former case, the complexity matches that of the CQ containment problem: NExpTime-complete, even for bounded arity, and NP-complete if the

[1] Under some mild technical assumptions elaborated in the paper.
[2] This is a common assumption since every set of guarded TGDs can be easily transformed in this form.

schema is fixed. In the latter case, we get a NExpTime upper bound and an ExpTime lower bound. For a fixed schema, or even for bounded arity, the problem is NP-complete.

Let us stress that the NP bounds mentioned above should be seen as positive results: By spending exponential time in the size of the (small) query, we can not only minimize it using known techniques, but also find a GHW_k reformulation, if one exists.

Results for EGDs: By adapting the proof for showing that SemGHW_1 under full TGDs is undecidable, we show that SemGHW_1 under EGDs is also undecidable. In view of this fact, we focus on a restricted class of EGDs, i.e., keys. Assuming schemas with unary and binary predicates only, we show that keys enjoy the acyclicity preserving chase property, which in turn allows us to show that SemGHW_1 under keys is NP-complete. Unfortunately, this is not true once we go beyond acyclicity. In other words, keys do not enjoy the generalized hypertreewidth preserving chase for $k > 1$, even in the case of unary and binary relations. It is also easy to show that the UCQ rewritability property does not hold. Thus, the techniques developed for TGDs cannot be used for studying SemGHW_k under keys, for $k > 1$, even if we focus on unary and binary predicates. Nevertheless, we can show that the problem is decidable in elementary time. This is a rather involved result that employs arguments based on monadic second-order logic (MSO). Whether SemGHW_k under keys over arbitrary schemas is decidable is a highly non-trivial problem that remains open.

Query evaluation: Let $\mathbb{C}$ be a class of TGDs or EGDs under which SemGHW_k is decidable (i.e., guarded, non-recursive, and sticky sets of TGDs, as well as keys over unary and binary predicates). We can use the following algorithm to evaluate a CQ q that is semantically in GHW_k under a set of constraints Σ that falls in $\mathbb{C}$ over a database D that satisfies the constraints:

(1) Convert q into a CQ q' in GHW_k that is equivalent to q under Σ.
(2) Evaluate q' over D.
(3) Return $q'(D)$.

The running time is $O(||D||^{k+1} \cdot f(||q||, ||\Sigma||))$, where f is a computable function. This holds since q' can be computed in elementary time for each one of the classes mentioned above, while CQs in GHW_k can be evaluated in polynomial time. This constitutes a *fixed-parameter tractable algorithm* for evaluating q over D. No such algorithm is believed to exist for arbitrary CQs [37]. Therefore, CQs that are semantically in GHW_k under constraints that fall in $\mathbb{C}$ behave better than arbitrary CQs in terms of query evaluation. Interestingly, in the absence of constraints, one can do better: Evaluating CQs that are semantically in GHW_k is in polynomial time [17]. It is natural to ask whether this also holds in the presence of constraints. We show, by adopting a game-theoretic approach, that this is the case for guarded sets of TGDs and (arbitrary) FDs. For the other classes of constraints the problem remains to be investigated.

Query approximations: In case a CQ q is not equivalent to any CQ q' in GHW_k under a set Σ of constraints, we can exploit our proof techniques for TGD-based constraints in order to compute an *approximation of q under Σ* [6], that is, a CQ q' in GHW_k that is maximally contained in q under Σ. Computing and evaluating such approximation yields "quick" answers to q when exact evaluation is intractable. The problem of computing such query approximations in the presence of EGD-based constraints remains open.

Remark. The problems that we study in this work, or simplified versions thereof, have originally been studied in the conference papers [5, 23], which form the basis of the present journal paper. Here, we clarify which results are directly coming from the above conference papers, and which results are new and presented for the first time in this paper:

- The undecidability of SemGHW_1 under full TGDs has been shown in [5]. However, the decidability status of the same problem under EGDs has remained open, while it was not clear how the undecidability proof for full TGDs can be adapted to cover the case of EGDs. In this work, we provide a new proof for the fact that SemGHW_1 under full TGDs is undecidable. Although this new proof is more complex than the one in [5], it has the advantage that it can be easily adapted to establish that SemGHW_1 under EGDs is undecidable.
- The work [5] focussed on the TGD-based classes mentioned above for which CQ containment is decidable, and studied all the problems in question, i.e., SemGHW_k, query evaluation, and query approximations, but *only for* $k = 1$. Here, we extend the results of [5] to any $k \geq 1$.
- In [5], it has been also shown that evaluating a CQ that is semantically in GHW_1 under a set of (arbitrary) functional dependencies is tractable. We extend this result to any $k \geq 1$.
- Finally, the work [23] established that the problem of deciding whether a CQ is equivalent to one of bounded treewidth under keys over unary and binary predicates is decidable in elementary time. We extend this to bounded generalized hypertreewidth queries.

Finite vs. infinite databases. All the results mentioned above interpret the notion of CQ equivalence (and, thus, being semantically in GHW_k) over the set of both *finite* and *infinite* databases. The reason is that we heavily use the chase procedure in our proofs, which characterizes CQ equivalence under arbitrary databases only. This is not a serious problem though as all the classes of TGDs that we consider (i.e., guarded, non-recursive, sticky) are *finitely controllable* [3, 22, 25]. This means that CQ equivalence under arbitrary databases and under finite databases coincide. The same holds for EGDs since in this case the chase is always finite. In conclusion, our results can be directly exported to the finite case for all the classes of constraints in question.

1.5 Organization

Preliminaries are given in Section 2. In Section 3 we introduce our main problem, that is, SemGHW_k. In Section 4 we investigate the frontiers of decidability for SemGHW_k under TGDs. The property of generalized hypertreewidth preserving chase is studied in Section 5, while UCQ rewritable classes in Section 6. SemGHW_k under arbitrary EGDs is studied in Section 7, while the case of keys over unary and binary predicates in Section 8. Evaluation of CQs that are semantically in GHW_k is considered in Section 9. Finally, the problem of computing query approximations in the presence of TGDs is studied in Section 10, while conclusions and open problems are given in Section 11.

2 PRELIMINARIES

In this section, we recall the basics on relational databases, conjunctive queries and constraints. We assume disjoint countably infinite sets $\mathbf{C}$, $\mathbf{N}$ and $\mathbf{V}$ of *constants, (labeled) nulls* and *variables* (used in queries and constraints), respectively. We may refer to constants, nulls and variables as terms.

Relational databases. A *(relational) schema* σ is a finite set of relation symbols (or predicates) with associated arity. We write R/n to denote the fact that R has arity $n > 0$. An *atom* over σ is an expression of the form $R(\bar{t})$, where R/n is a relation symbol in σ and $\bar{t}$ is an n-tuple of terms. An *instance* over σ is a (possibly infinite) set of atoms over σ that contains constants and nulls, while a *database* over σ is simply a finite instance over σ. For an instance I we write $\mathsf{dom}(I)$ for the set of terms occurring in I. We also write $||I||$ for the size of I, i.e., the number of symbols occurring in I.

Generalized hypertreewidth. One of the central notions in our work is *generalized hypertreewidth* [29, 30], a.k.a. *coverwidth* [17], which measures the degree of acyclicity of an instance. The definition of generalized hypertreewidth relies on the notion of tree decomposition. A *tree*

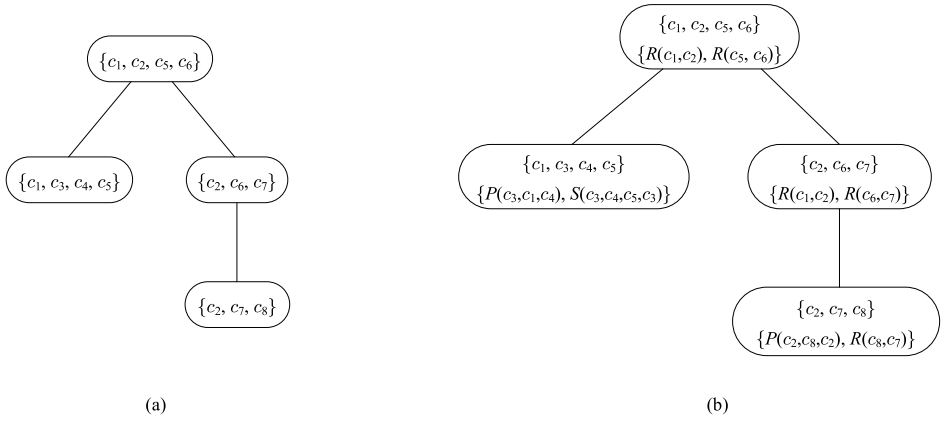


Fig. 1. Tree decomposition and generalized hypertree decomposition.

decomposition of an instance I is a pair (T, χ), where $T = (V, E)$ is a tree, and χ is a labeling function $V \longrightarrow 2^{\mathsf{dom}(I)}$, i.e., it assigns a subset of $\mathsf{dom}(I)$ to each node of T, such that:

(1) For each $R(\bar{t}) \in I$, there exists $v \in V$ such that $\chi(v)$ contains all the terms in $\bar{t}$.
(2) For each $t \in \mathsf{dom}(I)$, the set $\{v \in V \mid t \in \chi(v)\}$ induces a connected subtree of T.

The second condition above is generally known as the *connectedness condition*.

Example 2.1. Consider the database D over $\{R/2, P/3, S/4\}$ consisting of

$$R(c_1, c_2), P(c_3, c_1, c_4), S(c_3, c_4, c_5, c_3), R(c_5, c_6), R(c_6, c_7), P(c_2, c_8, c_2), R(c_8, c_7),$$

where $c_1, c_2, \ldots \in \mathbf{C}$. A tree decomposition of D is shown in Figure 1(a). ∎

A *generalized hypertree decomposition* of I is a triple (T, χ, λ), where (T, χ) is a tree decomposition of I, and, assuming $T = (V, E)$, λ is a labeling function $V \longrightarrow 2^I$, i.e., it assigns a subset of the atoms of I to each node of T, such that, for each $v \in V$, the terms in $\chi(v)$ are "covered" by the atoms in $\lambda(v)$, i.e., $\chi(v) \subseteq \bigcup_{R(t_1,\ldots,t_n)\in\lambda(v)}\{t_1, \ldots, t_n\}$. The *width* of (T, χ, λ) is the number $\max_{v\in V}\{|\lambda(v)|\}$, i.e., the maximal size of a set of the form $\lambda(v)$ over all nodes v of T. The *generalized hypertreewidth* of an instance I is the minimum width over all its generalized hypertree decompositions.

Example 2.2. As an example, consider again the database D given in Example 2.1. A generalized hypertree decomposition $H = (T, \chi, \lambda)$ of D is depicted in Figure 1(b); each node v consists of two sets, where the first one is the set of terms assigned to v by χ, while the second one is the set of atoms of D assigned to v by λ. Notice that, for each node v of H, $|\lambda(v)| = 2$, which means that the width of H is two, and it can be shown that this is optimal. In fact, every attempt to build a generalized hypertree decomposition of D such that, for each node v, $|\lambda(v)| = 1$, it will lead to a violation of the connectedness condition, i.e., (T, χ) will not be a valid tree decomposition anymore. Consequently, the generalized hypertreewidth of D is two. ∎

For $k \geq 1$, we denote by GHW_k the class of instances of generalized hypertreewidth at most k. The notion of generalized hypertreewidth subsumes the well studied class of *acyclic* instances [9, 26, 42]. In fact, the latter corresponds to the lowest level GHW_1 of the hierarchy. Correspondingly, low generalized hypertreewidth is associated with *mild acyclicity* [27]. For example, the database D given in Example 2.1 is not acyclic, but it is mildly acyclic since it belongs to GHW_2.

Conjunctive queries. A *conjunctive query* (CQ) over σ is a formula of the form:

$$q(\bar{x}) \ := \ \exists \bar{y}\big(R_1(\bar{t}_1) \wedge \cdots \wedge R_m(\bar{t}_m)\big),$$

where each $R_i(\bar{t}_i)$ $(1 \leq i \leq m)$ is an atom that contains only variables[3] of $\mathbf{V}$ over σ, each variable mentioned in the $\bar{t}_i$'s appears either in $\bar{x}$ or $\bar{y}$, and $\bar{x}$ contains all the *free variables* of q. A variable appearing in q that is not free is called *bound*. If $\bar{x}$ is empty, then q is a *Boolean CQ*. The evaluation of CQs is defined in terms of *homomorphisms*. A homomorphism from a CQ q to an instance I is a mapping h from the variables in q to the set of constants and nulls $\mathbf{C} \cup \mathbf{N}$ such that $R(h(\bar{t})) \in I$, for each atom $R(\bar{t})$ occurring in q; as usual, we write $h(t_1, \ldots, t_n)$ for $(h(t_1), \ldots, h(t_n))$. In the same way we can define the notion of homomorphism between CQs. The *evaluation of* $q(\bar{x})$ *over* I, denoted $q(I)$, is the set of all tuples $h(\bar{x})$ such that h is a homomorphism from q to I.

It is known that *CQ evaluation*, i.e., the problem of determining whether a tuple $\bar{t}$ belongs to the evaluation $q(D)$ of a CQ q over a database D, is NP-complete [16]. On the other hand, CQ evaluation becomes tractable by restricting the syntactic shape of CQs. One of the most general and widely studied such restrictions is bounded generalized hypertreewidth. Formally, a CQ q has generalized hypetreewidth $k \geq 1$ if the instance obtained from the atoms of q after replacing each variable x in q with a fresh null $\bot_x$, called the *canonical instance* of q and denoted $D[q]$, has generalized hypetreewidth k. We slightly abuse notation and we write GHW_k for the class of CQs of generalized hypertreewidth bounded by $k \geq 1$. It would be clear from the context when GHW_k refers to the class of instances or the class of CQs of generalized hypertreewidth bounded by k.

Example 2.3. Consider the schema $\sigma = \{\text{Friends}/2, \text{Likes}/2\}$, which describes a social network. Intuitively speaking, the atom $\text{Friends}(x, y)$ states that x is a friend of y, while $\text{Likes}(x, y)$ means that x likes the post y. The CQ

$$q(x, y) \ := \ \exists z(\text{Friends}(x, y) \wedge \text{Likes}(x, z) \wedge \text{Likes}(y, z))$$

asks for all the pairs of mutual friends that have some post they like in common. It is not difficult to verify that it belongs to GHW_2. Indeed, the canonical instance of q

$$\{\text{Friends}(\bot_x, \bot_y), \text{Likes}(\bot_x, \bot_z), \text{Likes}(\bot_y, \bot_z)\}$$

has generalized hypertreewidth two, which is witnessed by the generalized hypertree decomposition that has only one node labeled by

$$(\{\bot_x, \bot_y, \bot_z\}, \{\text{Likes}(\bot_x, \bot_z), \text{Likes}(\bot_y, \bot_z)\}).$$

Notice that $q \notin \mathsf{GHW}_1$, i.e., it is not acyclic. However, it is easy to verify that the CQ

$$q'(x, y) \ := \ \exists z \exists z'(\text{Friends}(x, y) \wedge \text{Likes}(x, z) \wedge \text{Likes}(y, z')),$$

obtained from q by "breaking" the join on the variable z, which asks for all the pairs of mutual friends that like at least one post, is in GHW_1. This is witnessed by the generalized hypertree decomposition whose root is labeled by $(\{\bot_x, \bot_y\}, \{\text{Friends}(x, y)\})$, while its two children are labeled by $(\{\bot_x, \bot_z\}, \{\text{Likes}(x, z)\})$ and $(\{\bot_x, \bot_{z'}\}, \{\text{Likes}(x, z')\})$. ∎

The problem of evaluating a CQ $q \in \mathsf{GHW}_k$ over a database D can be solved efficiently. As usual, we write $||q||$ for the size of q, i.e., the number of symbols occurring in q. The following holds:

Proposition 2.4. ([29], see also [27]) *Fix $k \geq 1$. The problem of deciding whether $\bar{t} \in q(D)$, given a database D, a CQ $q(\bar{x}) \in \mathsf{GHW}_k$, and a tuple $\bar{t} \in \text{dom}(D)^{|\bar{x}|}$, can be solved in time $O(||D||^{k+1} \cdot ||q||)$.*

[3]For technical clarity, we exclude constants from CQs. However, all of our results can be generalized to CQs with constants.

Integrity Constraints. The specification of semantic properties that should be satisfied by the input database can be achieved using integrity constraints, also known as dependencies. Two central classes of integrity constraints are the class of tuple-generating dependencies, and the class of equality-generating dependencies. The former allows us to specify properties of the form "if some tuples occur in the database, then some other tuples should also be in the database", while the latter allows us to express properties of the form "if some tuples occur in the database, then some values should be the same". The formal definitions follow.

A *tuple-generating dependency* (TGD) over a schema σ is an expression

$$\forall\bar{x}\forall\bar{y}\big(\phi(\bar{x},\bar{y}) \rightarrow \exists\bar{z}\,\psi(\bar{x},\bar{z})\big),$$

where ϕ and ψ are conjunctions of atoms over σ that contain only variables of $\mathbf{V}$. For simplicity, we write this TGD as $\phi(\bar{x},\bar{y}) \rightarrow \exists\bar{z}\,\psi(\bar{x},\bar{z})$, and use comma instead of $\wedge$ for conjoining atoms. Further, we assume that each variable in $\bar{x}$ is mentioned in some atom of ψ. We call ϕ and ψ the *body* and *head* of the TGD, respectively. The TGD above is logically equivalent to the expression $\forall\bar{x}(q_\phi(\bar{x}) \rightarrow q_\psi(\bar{x}))$, where $q_\phi(\bar{x})$ and $q_\psi(\bar{x})$ are the CQs $\exists\bar{y}\,\phi(\bar{x},\bar{y})$ and $\exists\bar{z}\,\psi(\bar{x},\bar{z})$, respectively. Therefore, an instance I over σ satisfies this TGD if and only if $q_\phi(I) \subseteq q_\psi(I)$. An instance I satisfies a set Σ of TGDs, denoted $I \models \Sigma$, if I satisfies every TGD in Σ. Let us clarify that we work with finite sets of TGDs. Henceforth, whenever we refer to a set Σ of TGDs, it is implicit that Σ is finite.

An *equality-generating dependency* (EGD) over σ is an expression

$$\forall\bar{x}\big(\phi(\bar{x}) \rightarrow x_i = x_j\big),$$

where ϕ is a conjunction of atoms over σ that contain only variables, and $x_i, x_j \in \bar{x}$. For clarity, we write this EGD as $\phi(\bar{x}) \rightarrow x_i = x_j$, and use comma for conjoining atoms. We call ϕ the *body* of the EGD. An instance I over σ satisfies this EGD if, for every homomorphism h such that $h(\phi(\bar{x})) \subseteq I$, it is the case that $h(x_i) = h(x_j)$. An instance I satisfies a set Σ of EGDs, denoted $I \models \Sigma$, if I satisfies every EGD in Σ. In the rest of the paper, sets of EGDs are always finite. Recall that EGDs subsume functional dependencies, which in turn subsume keys. A *functional dependency* (FD) over σ is an expression of the form $R[A{\rightarrow}B]$, where R/n is a relation symbol in σ, and $A, B \subseteq \{1,\ldots,n\}$, asserting that the values of the attributes of B are determined by the values of the attributes of A. For example, $R[\{1\}{\rightarrow}\{3\}]$, where R is a ternary relation, is actually the EGD $R(x,y,z), R(x,y',z') \rightarrow z = z'$. An FD $R[A{\rightarrow}B]$ is called *key* if $A \cup B = \{1,\ldots,n\}$.

TGDs and the chase procedure. The *chase* is a useful tool when reasoning with TGDs; see,e.g., [11, 22, 33, 36]. We start by defining a single chase step. Let I be an instance over a schema σ and τ a TGD of the form $\phi(\bar{x},\bar{y}) \rightarrow \exists\bar{z}\,\psi(\bar{x},\bar{z})$ over σ. We say that τ is *applicable* with respect to I if there exists a tuple $(\bar{t},\bar{t}')$ of terms in I such that $\phi(\bar{t},\bar{t}')$ holds in I. In this case, *the result of applying* τ *over* I *with* $(\bar{t},\bar{t}')$ is the instance J that extends I with every atom in $\psi(\bar{t},\bar{t}'')$, where $\bar{t}''$ is the tuple obtained by simultaneously replacing each variable $z \in \bar{z}$ with a fresh distinct null not occurring in I. For such a single chase step we write:

$$I \xrightarrow{\tau,(\bar{t},\bar{t}')} J.$$

Let I be an instance and Σ a set of TGDs. A *chase sequence for* I *under* Σ is a sequence:

$$I_0 \xrightarrow{\tau_0,(\bar{t}_0,\bar{t}'_0)} I_1 \xrightarrow{\tau_1,(\bar{t}_1,\bar{t}'_1)} I_2 \ldots$$

of chase steps such that:

(1) $I_0 = I$.
(2) For each $i \geq 0$ we have that τ_i is a TGD in Σ.
(3) $J \models \Sigma$, where $J = \bigcup_{i\geq 0} I_i$.

The instance J is the *result* of this chase sequence, which always exists. Note that, in general, different chase sequences lead to different results. However, it is well-known that those results are the same up to null renaming, and thus, we refer to *the* chase for I under Σ, denoted $\mathsf{chase}(I, \Sigma)$.[4] Moreover, for a CQ q we write $\mathsf{chase}(q, \Sigma)$ for the instance $\mathsf{chase}(D[q], \Sigma)$. The key property of the chase procedure is that the instance $\mathsf{chase}(I, \Sigma)$ is *universal*, i.e., for every instance J such that $J \supseteq I$ and $J \models \Sigma$, there is a homomorphism from $\mathsf{chase}(I, \Sigma)$ to J [20, 22].

EGDs and the chase procedure. As for TGDs, the chase is a useful tool when reasoning with EGDs. Let us first define a single chase step. Consider an instance I over schema σ and an EGD ϵ of the form $\phi(\bar{x}) \rightarrow x_i = x_j$ over σ. We say that ϵ is *applicable* with respect to I if there exists a homomorphism h such that $h(\phi(\bar{x})) \subseteq I$ and $h(x_i) \neq h(x_j)$. In this case, *the result of applying ϵ over I with h* is as follows: If $h(x_i), h(x_j)$ are constants, then the result is "failure"; otherwise, it is the instance J obtained from I by identifying $h(x_i)$ and $h(x_j)$ as follows: If one is a constant, then every occurrence of the null is replaced by the constant, and if both are nulls, the one is replaced everywhere by the other. We can define the notion of the chase sequence for an instance I under a set Σ of EGDs. Such a non-failing sequence is finite and unique (up to null renaming); thus, we refer to *the* chase for I under Σ, denoted $\mathsf{chase}(I, \Sigma)$. Observe that, if $\mathsf{chase}(I, \Sigma)$ exists, then the chase sequence that leads to it gives rise to a homomorphism $h_{I,\Sigma} : \mathsf{dom}(I) \rightarrow \mathsf{dom}(\mathsf{chase}(I, \Sigma))$, which is the identity on $\mathsf{dom}(\mathsf{chase}(I, \Sigma))$, such that $h_{I,\Sigma}(I) = \mathsf{chase}(I, \Sigma)$. For a CQ q we write $\mathsf{chase}(q, \Sigma)$ for $\mathsf{chase}(D[q], \Sigma)$, which always exists since q is constant-free, and we write $h_{q,\Sigma}$ for $h_{D[q],\Sigma}$.

Containment and equivalence. Let q and q' be CQs and Σ a set of TGDs or EGDs. Then q is *contained* in q' under Σ, denoted $q \subseteq_\Sigma q'$, if $q(I) \subseteq q'(I)$ for every instance I such that $I \models \Sigma$. Furthermore, q is *equivalent* to q' under Σ, denoted $q \equiv_\Sigma q'$, whenever $q \subseteq_\Sigma q'$ and $q' \subseteq_\Sigma q$ (or, equivalently, if $q(I) = q'(I)$ for every I such that $I \models \Sigma$). The following well-known characterization of CQ containment in terms of the chase will be widely used in our proofs. For a tuple of variables $\bar{x}$, we write $\perp(\bar{x})$ for the tuple of nulls obtained by replacing each variable $x \in \bar{x}$ with the null $\perp_x$.

Lemma 2.5. *Let $q(\bar{x}), q'(\bar{x}')$ be CQs, with $|\bar{x}| = |\bar{x}'|$, and Σ a set of TGDs (resp., EGDs). Then $q \subseteq_\Sigma q'$ if and only if $\perp(\bar{x})$ (resp., $h_{q,\Sigma}(\perp(\bar{x}))$) belongs to the evaluation of q' over* $\mathsf{chase}(q, \Sigma)$.

A problem that is quite important for our work is *CQ containment under constraints* (TGDs or EGDs), defined as follows: Given CQs q, q' and a set Σ of TGDs or EGDs, is it the case that $q \subseteq_\Sigma q'$? Whenever Σ is bound to belong to a certain class $\mathbb{C}$ of sets of TGDs or EGDs, we denote this problem as $\mathsf{Cont}(\mathbb{C})$. It is clear that Lemma 2.5 provides a decision procedure for the containment problem under EGDs. However, this is not the case for TGDs since the result of the chase can be infinite.

Decidable containment of CQs under TGDs. It is not surprising that Lemma 2.5 does not provide a decision procedure for solving CQ containment under TGDs since this problem is known to be undecidable [10]. This has led to an intensive research activity for identifying syntactic restrictions on sets of TGDs that lead to decidable CQ containment (even in the case when the chase does not terminate).[5] Such restrictions are often classified into three main paradigms:

Guardedness: A TGD is *guarded* if its body contains an atom, called *guard*, that contains all the body-variables. Although the chase under a set of guarded TGDs does not necessarily terminate, query containment is decidable. This follows from the fact that the result of the chase has finite (hyper)treewidth. More formally, let $\mathbb{G}$ be the class of sets of guarded TGDs. Then:

[4] This holds since we consider the oblivious version of the chase, where a TGD is blindly triggered as long as its body is satisfied, without checking whether the head is satisfied.

[5] In fact, these restrictions are designed to obtain decidable *query answering under TGDs*. However, this problem is equivalent to query containment under TGDs (Lemma 2.5).

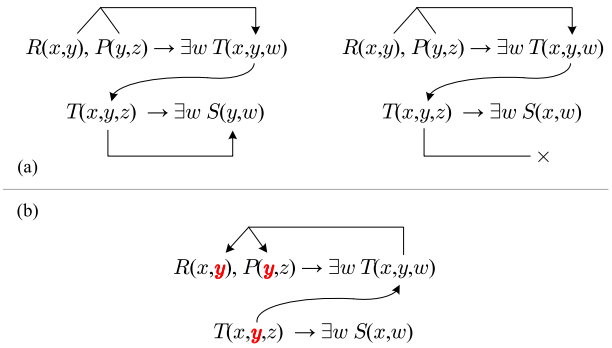


Fig. 2. Stickiness and marking.

Proposition 2.6. [11] Cont($\mathbb{G}$) *is 2ExpTime-complete, ExpTime-complete if the arity of the schema is fixed, and NP-complete if the schema is fixed.*

A key subclass of guarded TGDs is the class of *linear* TGDs, that is, TGDs whose body consists of a single atom [12], which in turn subsume the well-known class of *inclusion dependencies* (linear TGDs without repeated variables neither in the body nor in the head) [21]. Let $\mathbb{L}$ and $\mathbb{ID}$ be the classes of sets of linear TGDs and inclusions dependencies, respectively. Then:

Proposition 2.7. [33] Cont($\mathbb{C}$), *for* $\mathbb{C} \in \{\mathbb{L}, \mathbb{ID}\}$, *is PSpace-complete, and NP-complete if the arity of the schema is fixed.*

Non-recursiveness: A set Σ of TGDs is *non-recursive* if its *predicate graph* contains no directed cycles. Recall that the nodes of the predicate graph of Σ are the predicates occurring in Σ, and there is an edge from P to R if and only if there is a TGD $\tau \in \Sigma$ such that P occurs in the body and R occurs in the head of τ. Notice that non-recursive sets of TGDs are also known as *acyclic* [22, 35], but we reserve this term for CQs. Non-recursiveness ensures the termination of the chase, and thus decidability of CQ containment. Let $\mathbb{NR}$ be the class of non-recursive sets of TGDs. Then:

Proposition 2.8. [35] Cont($\mathbb{NR}$) *is complete for NExpTime, even if the arity of the schema is fixed. It becomes NP-complete if the schema is fixed.*

Stickiness: This condition ensures neither termination nor bounded (hyper)treewidth of the chase. Instead, the decidability of query containment is obtained by exploiting *query rewriting* techniques. The goal of stickiness is to capture joins among variables that are not expressible via guarded TGDs, but without forcing the chase to terminate. The key property underlying this condition can be described as follows: During the chase, terms that are associated (via a homomorphism) with variables that appear more than once in the body of a TGD (i.e., join variables) are always propagated (or "stick") to the inferred atoms. This is illustrated in Figure 2(a); the first set of TGDs is sticky, while the second is not. The formal definition, which is given below, is based on an inductive marking procedure that marks the variables that may violate the semantic property of the chase described above [13]. Roughly, during the base step of this procedure, a variable that appears in the body of a TGD τ but not in every head-atom of τ is marked. Then, the marking is inductively propagated from head to body as shown in Figure 2(b). Stickiness requires every marked variable to appear only once in the body of a TGD. The formal definition follows.

Consider a set Σ of TGDs; w.l.o.g., we assume that the TGDs in Σ do not share variables. For notational convenience, given an atom $R(\bar{t})$ and a variable $x \in \bar{t}$, $\text{pos}(R(\bar{t}), x)$ is the set of positions

in $R(\bar{t})$ at which x occurs; the i-th position in $R(\bar{t})$ identifies the i-th attribute of the predicate R. Let $\tau \in \Sigma$ and x a variable in the body of τ. We inductively define when x *is marked in* Σ as follows:

- If there exists an atom $R(\bar{t})$ in the head of τ such that $x \notin \bar{t}$, then x is marked in Σ.
- Assuming that there is an atom $R(\bar{t})$ in the head of τ such that $x \in \bar{t}$, if there is $\tau' \in \Sigma$ that has in its body an atom of the form $R(\bar{t}')$, and each variable in $R(\bar{t}')$ at a position of $\text{pos}(R(\bar{t}), x)$ is marked in Σ, then x is marked in Σ.

We say that Σ is *sticky* if there is no TGD that contains two occurrences of a variable that is marked in Σ. Let $\mathbb{S}$ be the class of sticky sets of TGDs. Then:

Proposition 2.9. [13] $\text{Cont}(\mathbb{S})$ *is* ExpTime*-complete, and NP-complete for fixed arity schemas.*

Weak versions: Each one of the previous classes has an associated weak version, called *weakly-guarded* [11], *weakly-acyclic* [22], and *weakly-sticky* [13], respectively, that guarantees the decidability of query containment. The underlying idea is always the same: Relax the conditions in the definition of the class, so that only those positions that receive null values during the chase procedure are taken into consideration. The formal definitions of those classes are not important for the present work, and thus are omitted. However, it is important to recall that all those classes extend the class of *full TGDs*, i.e., those without existentially quantified variables. This is not the case for the "unrelaxed" versions presented above.

3 SEMANTIC GENERALIZED HYPERTREEWIDTH

The main task of the present work is to study the problem of checking whether a CQ q is equivalent to a CQ of generalized hypertreewidth at most k, for some fixed $k \geq 1$, over those instances that satisfy a set Σ of TGDs or EGDs. Whenever this is the case, we say that q is *semantically in* GHW_k *under* Σ. In case $k = 1$, q is also called *semantically acyclic under* Σ since GHW_1 coincides with the class of acyclic CQs. The associated decision problem is defined below; $\mathbb{C}$ is a class of sets of TGDs (e.g., guarded, non-recursive, sticky, etc.) or sets of EGDs (e.g., arbitrary EGDs, FDs, etc.):

PROBLEM :	$\text{SemGHW}_k(\mathbb{C})$
INPUT :	A CQ q and a finite set $\Sigma \in \mathbb{C}$ of TGDs or EGDs, both over a schema σ.
QUESTION :	Is there a CQ $q' \in \text{GHW}_k$ over σ such that $q \equiv_\Sigma q'$?

One may ask whether there exist CQs that are in general not semantically in GHW_k, but are semantically in GHW_k under a set of constraints that falls in a class introduced in Section 2. We show via a couple of examples that this is the case. Notice that if this was not the case, then $\text{SemGHW}_k(\mathbb{C})$, where $\mathbb{C}$ is a class of TGDs or EGDs introduced in the previous section, is an artificial problem that coincides with the problem of checking whether the given CQ is semantically in GHW_k without taking into account the constraints. Our first example focusses on TGDs, and shows that even constraints of a very simple form can make a difference:

Example 3.1. Consider the CQ

$$q \;=\; \exists x \exists y \exists z \, (R(x, y) \wedge S(y, z) \wedge S(z, x)).$$

It is easy to verify that $q \notin \text{GHW}_1$, i.e., it is not acyclic. Moreover, we know that q, which is a core, is not semantically acyclic [8]. Consider now the set Σ of TGDs consisting of

$$R(x, y) \rightarrow \exists z \, P(x, y, z) \qquad P(x, y, z) \rightarrow S(y, z) \qquad P(x, y, z) \rightarrow S(z, x).$$

Notice that the above set of TGDs corresponds to the set of inclusion dependencies

$$R[1, 2] \subseteq P[1, 2] \qquad P[2, 3] \subseteq S[1, 2] \qquad P[3, 1] \subseteq S[1, 2].$$

Moreover, observe that Σ belongs to all the classes of TGDs mentioned in the previous section, i.e., $\mathbb{G}$, $\mathbb{NR}$ and $\mathbb{S}$. Interestingly, q is equivalent to the acyclic CQ

$$q' \;=\; \exists x \exists y \exists z \, (R(x,y) \wedge S(y,z) \wedge S(z,x) \wedge P(x,y,z))$$

if we focus on databases that satisfy Σ, i.e., $q \equiv_\Sigma q'$. Thus, although q is not semantically acyclic in general, it is semantically acyclic under Σ. ∎

Our second example shows that a similar effect may take place via a very simple EGD:

Example 3.2. Consider the CQ

$$q \;=\; \exists x \exists y \exists z \, (R(x,y) \wedge R(y,z) \wedge R(z,x) \wedge R(x,z)).$$

As above, it is easy to verify that q is not acyclic, and we also know that is not semantically acyclic. Consider now the set Σ consisting of the single EGD

$$R(x,y), R(x,z) \rightarrow y = z,$$

which simply states that the first attribute of the predicate R is the key. Thus, the above EGD is a simple key constraint. Interestingly, q is equivalent to the acyclic CQ

$$q' \;=\; \exists y \, R(y,y)$$

if we focus on databases that satisfy Σ, i.e., $q \equiv_\Sigma q'$. Thus, although q is not semantically acyclic in general, it is semantically acyclic under Σ. ∎

The above discussion shows that $\mathsf{SemGHW}_k(\mathbb{C})$, where $\mathbb{C}$ is a class of TGDs or EGDs from Section 2, deviates from the version of the problem without constraints. As said, one of our main tasks is to perform an in-depth investigation of $\mathsf{SemGHW}_k(\mathbb{C})$.

3.1 Infinite Instances vs. Finite Databases

It is important to clarify that $\mathsf{SemGHW}_k(\mathbb{C})$ asks for the existence of a CQ q' in GHW_k that is equivalent to q under Σ focussing on arbitrary (finite or infinite) instances. However, in practice we are concerned only with finite databases. Therefore, one may claim that the natural problem to investigate is $\mathsf{FinSemGHW}_k(\mathbb{C})$, which accepts as input a CQ q and a set $\Sigma \in \mathbb{C}$ of TGDs or EGDs, and asks whether there is a CQ $q' \in \mathsf{GHW}_k$ such that $q(D) = q'(D)$ for every (finite) database D that satisfies Σ. Interestingly, for all the classes of sets of TGDs discussed in the previous section, and the class of (arbitrary) EGDs, SemGHW_k and $\mathsf{FinSemGHW}_k$ coincide due to the fact that they ensure the so-called *finite controllability* of CQ containment. This means that query containment under arbitrary instances and query containment under finite databases are equivalent problems. For non-recursive and weakly-acyclic sets of TGDs, as well as for sets of EGDs, this immediately follows from the fact that the chase terminates. For inclusion dependencies this has been shown by Rosati [38], while for the more general setting of guarded TGDs it has been shown in [3]. For the sticky-based classes of sets of TGDs this has been shown in [25]. The reason why we focus on $\mathsf{SemGHW}_k(\mathbb{C})$, instead of $\mathsf{FinSemGHW}_k(\mathbb{C})$, is given by Lemma 2.5: CQ containment under arbitrary instances can be characterized in terms of the chase, which is not true for finite databases simply because the chase is, in general, infinite.

3.2 A Note on the Combination of TGDs and EGDs

Observe that our main problem $\mathsf{SemGHW}_k(\mathbb{C})$ is defined for TGDs or EGDs, i.e., $\mathbb{C}$ is a class of TGDs or EGDs. The reason why we do not consider classes of constraints that combine TGDs and EGDs is that CQ containment becomes undecidable very quickly. In particular, even if we consider the class that combines inclusion dependencies (the simplest class of TGDs), and key

dependencies (the simplest class of EGDs), CQ containment is undecidable [14]. Actually, the situation is worse than that. Even if we consider the well-known class of constraints that combines inclusion dependencies with key dependencies in a non-conflicting way [14], which essentially means that we can ignore the key dependencies as long as they are satisfied by the input database, CQ containment *under finite databases* is undecidable [38].[6]

4 SEMANTIC GENERALIZED HYPETREEWIDTH UNDER TGDS

It is clear that $\mathsf{SemGHW}_k(\mathbb{C})$ and $\mathsf{Cont}(\mathbb{C})$, where $\mathbb{C}$ is a class of constraints, are closely related problems. If we are not able to solve the latter, then it is unlikely that we can solve the former since we need to check whether two CQs are equivalent under a set $\Sigma \in \mathbb{C}$. In view of the fact that CQ containment under TGDs is undecidable [10], the question that comes up is whether the decidability of $\mathsf{Cont}(\mathbb{C})$ for a class $\mathbb{C}$ of TGDs is a necessary condition for the decidability of $\mathsf{SemGHW}_k(\mathbb{C})$. Of course, it is also interesting to ask the converse, i.e., whether the decidability of $\mathsf{Cont}(\mathbb{C})$ is a sufficient condition for the decidability of $\mathsf{SemGHW}_k(\mathbb{C})$. We proceed to give answers to the above fundamental questions, which help us to better understand SemGHW_k under TGDs.

4.1 Decidability of CQ Containment Under TGDs is Necessary

For each $k \geq 1$, there is a restricted version of CQ containment under sets of TGDs that can be reduced to SemGHW_k. To provide such a reduction, we first need to establish an auxiliary technical lemma. A Boolean CQ is *connected* if its *Gaifman graph* is connected – the nodes of the Gaifman graph of a CQ q are the variables occurring in q, and there is an edge between variables x and y if and only if they appear together in an atom of q. A component of a Boolean CQ q is essentially a maximally connected subquery of q. Formally, a *component* of $q = \exists \bar{y} \bigwedge_{1 \leq i \leq m} R_i(\bar{t}_i)$ is a Boolean connected CQ $q' = \exists \bar{y}' \bigwedge_{1 \leq j \leq k} R_{i_j}(\bar{t}_{i_j})$, where $\bar{y}' \subseteq \bar{y}$, $\{R_{i_j}(\bar{t}_{i_j})\}_{1 \leq j \leq k} \subseteq \{R_i(\bar{t}_i)\}_{1 \leq i \leq m}$, and the CQ obtained by adding an atom of $\{R_i(\bar{t}_i)\}_{1 \leq i \leq m} \setminus \{R_{i_j}(\bar{t}_{i_j})\}_{1 \leq j \leq k}$ to q' is not connected. Finally, a TGD τ is *body-connected* if its body, which can be seen as a CQ, is connected. The following holds:

Lemma 4.1. *Consider a set Σ of body-connected TGDs, and two Boolean CQs q, q', where q' is connected. If $q \subseteq_\Sigma q'$, then there exists a component q'' of q such that $q'' \subseteq_\Sigma q'$.*

Proof. By Lemma 2.5, it suffices to show that there exists a component q'' of q such that $q'(\mathsf{chase}(q'', \Sigma)) \neq \varnothing$. By hypothesis and Lemma 2.5, $q'(\mathsf{chase}(q, \Sigma)) \neq \varnothing$. Let $q_1, \ldots, q_k$, where $k \geq 1$, be the components of q. By exploiting the fact that Σ is body-connected, it is not difficult to show that $\mathsf{chase}(q, \Sigma)$ can be partitioned into $\{I_1, \ldots, I_k\}$ such that, for each $i \in \{1, \ldots, k\}$, the atoms of I_i depend only on q_i. In other words, for each atom $R(\bar{t}) \in I_i$, the chase derivation that derives $R(\bar{t})$ starts from atoms of q_i. Since q' is connected, $q'(\mathsf{chase}(q, \Sigma)) \neq \varnothing$ implies that there is $i \in \{1, \ldots, k\}$ such that $q'(I_i) \neq \varnothing$, i.e., $q'(\mathsf{chase}(q_i, \Sigma)) \neq \varnothing$, and the claim follows with $q'' = q_i$. □

We proceed to give the desired reduction from a restricted version of CQ containment under sets of TGDs to SemGHW_k, and show its correctness by exploiting the above lemma. Given two Boolean CQs $q = \exists \bar{y} \bigwedge_{1 \leq i \leq n} R_i(\bar{t}_i)$ and $q' = \exists \bar{z} \bigwedge_{1 \leq i \leq m} R'_i(\bar{v}_i)$, let $\mathsf{conj}(q, q')$ be the Boolean CQ

$$\exists \bar{y} \exists \bar{z}' \big(R_1(\bar{t}_1) \wedge \cdots \wedge R_n(\bar{t}_n) \wedge R'_1(\bar{v}'_1) \wedge \cdots \wedge R'_m(\bar{v}'_m) \big),$$

where $\bar{z}'$ is obtained from $\bar{z}$ by replacing each $z \in \bar{z}$ by a fresh variable z', and for each $i \in \{1, \ldots, m\}$, $\bar{v}'_i$ is obtained form v_i by replacing each $v \in \bar{v}_i$ by a fresh variable v'. In other words, $\mathsf{conj}(q, q')$ is the conjunction of q and q' after renaming the variables occurring in q' so that q and q' have no variables in common. We can then show the following:

[6] Note that unrestricted CQ containment under the non-conflicting combination of inclusion dependencies and key dependencies is decidable [14]. However, as discussed in Section 3.1, we are only concerned about finite databases.

Proposition 4.2. *Fix $k \geq 1$. Consider a set Σ of body-connected TGDs, and Boolean and connected CQs q, q' such that $q \in \mathsf{GHW}_k$ and q' is not semantically in GHW_k under Σ. It holds that $q \subseteq_\Sigma q'$ if and only if $\mathsf{conj}(q, q')$ is semantically in GHW_k under Σ.*

Proof. ($\Rightarrow$) It is clear that $q \subseteq_\Sigma \mathsf{conj}(q, q')$. Moreover, $\mathsf{conj}(q, q') \subseteq_\Sigma q$ holds trivially. Therefore, $\mathsf{conj}(q, q') \equiv_\Sigma q$, and the claim follows since, by hypothesis, $q \in \mathsf{GHW}_k$.

($\Leftarrow$) By hypothesis, there exists a Boolean CQ $q^\star \in \mathsf{GHW}_k$ such that $\mathsf{conj}(q, q') \equiv_\Sigma q^\star$. Let $q_1^\star, \ldots, q_k^\star$, where $k \geq 1$, be the components of $q^\star$. By definition, q and q' are the components of $\mathsf{conj}(q, q')$. Therefore, by Lemma 4.1, for each $i \in \{1, \ldots, k\}$ it is the case that $q \subseteq_\Sigma q_i^\star$ or $q' \subseteq_\Sigma q_i^\star$. We define the following two sets of indices:

$$S_q = \{i \in \{1, \ldots, k\} \mid q \subseteq_\Sigma q_i^\star\} \quad \text{and} \quad S_{q'} = \{i \in \{1, \ldots, k\} \mid q' \subseteq_\Sigma q_i^\star \text{ and } q \not\subseteq_\Sigma q_i^*\};$$

clearly, S_q and $S_{q'}$ form a partition of $\{1, \ldots, k\}$. We proceed to show that $q \subseteq_\Sigma q'$ by considering the following two cases:

- **Case 1.** Assume that $S_{q'} = \varnothing$. This implies that for each $i \in \{1, \ldots, k\}$ it is the case that $q \subseteq_\Sigma q_i^\star$; thus, $q \subseteq_\Sigma q^\star$. By hypothesis, $q^\star \subseteq_\Sigma \mathsf{conj}(q, q')$, which immediately implies that $q^\star \subseteq_\Sigma q'$. Therefore, $q \subseteq_\Sigma q'$, as needed.
- **Case 2.** Assume now that $S_{q'} \neq \varnothing$. Given that $q^\star \subseteq_\Sigma q'$, and it is the case that Σ is body-connected and q' is connected, Lemma 4.1 implies that there exists $j \in \{1, \ldots, k\}$ such that $q_j^\star \subseteq_\Sigma q'$. Observe that $j \notin S_{q'}$; otherwise, $q' \subseteq_\Sigma q_j^\star$, and thus $q' \equiv_\Sigma q_j^\star$, which contradicts the fact that q' is not semantically in GHW_k under Σ. Therefore $j \in S_q$, which implies that $q \subseteq_\Sigma q_j^\star$. We conclude that $q \subseteq_\Sigma q'$, as needed.

This completes the proof of Proposition 4.2. □

As an immediate corollary of Proposition 4.2, we obtain an initial boundary for the decidability of SemGHW_k: We can only obtain a positive result for those classes of sets of TGDs that ensure the decidability of the restricted containment problem presented above. Formally, for $k \geq 1$ we define $\mathsf{RestCont}_k(\mathbb{C})$ as the problem of deciding whether $q \subseteq_\Sigma q'$ given a set Σ of body-connected TGDs in $\mathbb{C}$, and two Boolean and connected CQs q and q' such that $q \in \mathsf{GHW}_k$ and q' is not semantically in GHW_k under Σ. By Proposition 4.2, we immediately conclude the following.

Corollary 4.3. *Consider a class $\mathbb{C}$ of TGDs. It holds that $\mathsf{SemGHW}_k(\mathbb{C})$ is decidable only if $\mathsf{RestCont}_k(\mathbb{C})$ is decidable.*

As we shall discuss later, $\mathsf{RestCont}_k$ is not easier than general CQ containment under TGDs. This means that the only classes of TGDs for which we know the former to be decidable are those for which we know the latter to be decidable (e.g., those introduced in Section 2).

4.2 Decidability of CQ Containment Under TGDs is Not Sufficient

At this point, one may be tempted to think that some version of the converse of Proposition 4.2 also holds, that is, for a class $\mathbb{C}$ of sets of TGDs, $\mathsf{SemGHW}_k(\mathbb{C})$ can be reduced to $\mathsf{Cont}(\mathbb{C})$. This would immediately imply the decidability of $\mathsf{SemGHW}_k(\mathbb{C})$ whenever $\mathsf{Cont}(\mathbb{C})$ is decidable. However, as the next result shows, the picture is more complicated than this as $\mathsf{SemGHW}_1(\mathbb{F})$ is undecidable, where $\mathbb{F}$ is the class of full TGDs, a class which ensures the decidability of CQ containment:[7]

Theorem 4.4. $\mathsf{SemGHW}_1(\mathbb{F})$ *is undecidable, even if we allow only unary and binary predicates.*

[7]Recall that full TGDs are TGDs without existentially quantified variables. This implies that the chase always terminates, and thus, by Lemma 2.5, we get that $\mathsf{Cont}(\mathbb{F})$ is decidable.

We provide a high-level description of the proof of the above result, while the rather long complete proof can be found in the appendix. We show Theorem 4.4 by a reduction from the *Post correspondence problem* (PCP) over the alphabet $\{a, b\}$. The input to this problem are two equally long lists $u_1, \ldots, u_n$ and $v_1, \ldots, v_n$ of non-empty words over the alphabet $\{a, b\}$, and we ask whether there is a *solution*, i.e., a sequence $i_1 \ldots i_m$, where $m \geq 1$, of indices in $\{1, \ldots, n\}$ such that $u_{i_1} \cdots u_{i_m} = v_{i_1} \cdots v_{i_m}$. Consider an instance of PCP over $\{a, b\}$ given by the lists $u_1, \ldots, u_n$ and $v_1, \ldots, v_n$. Our goal is to construct a Boolean CQ q, and a set Σ of full TGDs, both over the schema σ that consists of the binary predicates

$$\{a, b, (\diamond_{i,*})_{1 \leq i \leq n}, (\diamond_{*,i})_{1 \leq i \leq n}, (\diamond_{i,j})_{1 \leq i,j \leq n}, \text{start}, \text{end}, \eta_1, \eta_2, \eta\},$$

such that the PCP instance has a solution if and only if there exists a CQ $q' \in \text{GHW}_1$ (i.e., an acyclic CQ) such that $q \equiv_\Sigma q'$, or, equivalently, q is semantically in GHW_1 under Σ. Since σ consists only of binary predicates, we can naturally represent CQs over σ as directed graphs in which edges are labeled with symbols from σ. Intuitively speaking, if there exists a CQ $q' \in \text{GHW}_1$ such that $q \equiv_\Sigma q'$, then q' "encodes" a solution $i_1 \ldots i_m$ to the PCP instance, i.e., $u_{i_1} \ldots u_{i_m} = v_{i_1} \ldots v_{i_m}$. In particular, such a solution is "encoded" in q' in a directed path, which is labeled by a word w over the alphabet $\{a, b, (\diamond_{i,*})_{1 \leq i \leq n}, (\diamond_{*,i})_{1 \leq i \leq n}, (\diamond_{i,j})_{1 \leq i,j \leq n}, \text{start}, \text{end}\}$ such that:

- after removing from w the symbols $\diamond_{*,j}$, for $1 \leq j \leq n$, we obtain a word of the form:

  $$\text{start}\, u_{i_1} \diamond_{i_1,r_1} \ldots u_{i_m} \diamond_{i_m,r_m} \text{end},$$

  where $r_j \in \{*, 1, \ldots, n\}$ for each $1 \leq j \leq m$, and
- after removing from w the symbols $\diamond_{j,*}$, for $1 \leq j \leq n$, we obtain a word of the form:

  $$\text{start}\, v_{i_1} \diamond_{r_1,i_1} \ldots v_{i_m} \diamond_{r_m,i_m} \text{end},$$

  where $r_j \in \{*, 1, \ldots, n\}$ for each $1 \leq j \leq m$.

In other words, the symbols $\diamond_{i,*}$ and $\diamond_{*,i}$, for $1 \leq i \leq n$, mark how w is formed in terms of the u_i's and v_i's, respectively. The symbols $\diamond_{i,j}$, for $1 \leq i, j \leq n$, are used to mark positions that represent synchronous ocurrences of u_i and v_j.

The predicates η_1, η_2, and η in the set Σ of full TGDs are used to check that q' is of the desired form. In particular, this should take care of checking that the sequence of indices used to construct the word w by concatenating the u_i's coincides with the sequence used to construct w from the v_j's. This boils down to the following check. Let w_1 be the word that is obtained from w after removing the symbols $\diamond_{*,j}$, for $1 \leq j \leq n$, and, analogously, let w_2 be the word that is obtained from w after removing the symbols $\diamond_{j,*}$, for $1 \leq j \leq n$. Then the following statements hold:

(1) The word w_1 is of the form:

$$\text{start}\, u_{i_1} \diamond_{i_1,r_1} \ldots u_{i_m} \diamond_{i_m,r_m} \text{end},$$

for $1 \leq i_1, \ldots, i_m \leq n$, and $r_1, \ldots, r_m \in \{*, 1, \ldots, n\}$. That is, each word u_{i_j} is in fact followed by a symbol of the form $\diamond_{i_j,r_j}$ that marks occurrences of such word.

(2) The word w_2 is of the form:

$$\text{start}\, v_{j_1} \diamond_{r_1,j_1} \ldots v_{j_t} \diamond_{r_t,j_t} \text{end},$$

for $1 \leq j_1, \ldots, j_t \leq n$, and $r_1, \ldots, r_t \in \{*, 1, \ldots, n\}$. That is, each word v_{i_j} is in fact followed by a symbol of the form $\diamond_{r_j,i_j}$ that marks occurrences of such word.

(3) It is the case that $i_1 \ldots i_m = j_1 \ldots j_t$ (and, therefore, $m = t$).

In order to ensure that the above conditions hold, we define the predicates η_1, η_2, and η in the set Σ of full TGDs as a way to check that there is a "matching" between w_1 and w_2. This matching starts by "synchronizing" the pair (x_0, x_0') of positions immediately following the start symbol in w_1 and

w_2, respectively. For $j \geq 0$, let x_{j+1} be the position in w_1 that occurs immediately after the $(j+1)$-th occurrence of a symbol of the form $\diamond_{i,r}$. Analogously, we define x'_{j+1} over w_2, but this time counting occurrences of symbols of the form $\diamond_{r,i}$. The matching then recursively "synchronizes" pairs of the form (x_{j+1}, x'_{j+1}) as long as the following holds: (a) the pair (x_j, x'_j) is already synchronized, and (b) there exists an $1 \leq i \leq n$ such that, starting from x_j and finishing at x_{j+1}, it is possible to read a word of the form $u_i \diamond_{i,r}$ in w_1, while starting from x'_j and finishing at x'_{j+1}, it is possible to read a word of the form $v_i \diamond_{r,i}$ in w_2. Lastly, we apply a finalization rule checking that the last pair (x_m, x'_m) is synchronized, and there is an $1 \leq i \leq n$ such that starting from x_m it is possible to read a word of the form $u_i \diamond_{i,i}$ end in w_1, while starting from x'_m it is possible to read a word of the form $v_i \diamond_{i,i}$ end in w_2. Notice, however, that this synchronization process will have to be carried out over a single path in the acyclic CQ q', which is one of the technical issues that our proof has to deal with. As already said, the formal proof of Theorem 4.4 can be found in the appendix.

Theorem 4.4 rules out any class of TGDs that captures the class of full TGDs, e.g., weakly-guarded, weakly-acyclic and weakly-sticky sets of TGDs. The main question is whether the non-weak versions of the above classes, i.e., guarded, non-recursive and sticky sets of TGDs, ensure the decidability of SemGHW_k, and what is the exact complexity. This is the subject of Sections 5 and 6.

5 GENERALIZED HYPERTREEWIDTH PRESERVING CHASE

We propose a semantic criterion for a class $\mathbb{C}$ of sets of TGDs, the so-called *generalized hypertreewidth preserving chase* (or simply *GHW-preserving chase*), that ensures the decidability of $\mathsf{SemGHW}_k(\mathbb{C})$. This criterion guarantees that, starting from an instance in GHW_k, it is not possible to increase its generalized hypertreewidth during the construction of the chase. We then proceed to show that the class of guarded sets of TGDs has GHW-preserving chase, which implies the decidability of $\mathsf{SemGHW}_k(\mathbb{G})$, and we pinpoint the complexity of the latter problem. Notice that non-recursiveness and stickiness do not fulfil this criterion, even in the restrictive setting where only unary and binary predicates can be used; more details are given in the next section. The formal definition of our semantic criterion follows; in the rest of this section, we fix $k \geq 1$.

Definition 5.1. (**GHW-preserving chase**) We say that a class $\mathbb{C}$ of sets of TGDs has *generalized hypertreewidth preserving chase* (or simply *GHW-preserving chase*) if, for every CQ $q \in \mathsf{GHW}_k$, set $\Sigma \in \mathbb{C}$, and chase sequence for $D[q]$ under Σ of the form

$$D[q] = I_0 \xrightarrow{\tau_0,(\bar{t}_0,\bar{t}'_0)} I_1 \xrightarrow{\tau_1,(\bar{t}_1,\bar{t}'_1)} I_2 \ldots$$

it holds that the instance $\bigcup_{i \geq 0} I_i$ belongs to GHW_k. ∎

Recall that the purpose of the above property is to ensure the decidability of $\mathsf{SemGHW}_k(\mathbb{C})$, for a class $\mathbb{C}$ of TGDs. Let us first show that it ensures the decidability of $\mathsf{Cont}(\mathbb{C})$. To this end, we exploit the classical result that the satisfiability problem for a fragment $\mathbb{L}$ of first-order logic that enjoys the *finite treewidth model property* (FTMP) is decidable, that is, the problem of deciding whether a sentence that falls in $\mathbb{L}$ has a model is decidable [18]. Recall that the treewidth of an instance is the minimum width over all its tree decompositions, while the width of a tree decomposition (T, χ), with $T = (V, E)$, is the number $\max_{v \in V}\{|\chi(v)|\} - 1$. A fragment of first-order logic $\mathbb{L}$ enjoys the FTMP if, for every sentence Φ that falls in $\mathbb{L}$, if Φ has a model, then it has a model of finite treewidth – note that a model of Φ can be naturally seen as a relational instance. We can now show that:

Proposition 5.2. *For a class $\mathbb{C}$ of sets of TGDs that has GHW-preserving chase,* $\mathsf{Cont}(\mathbb{C})$ *is decidable.*

Proof. Consider an instance of $\mathsf{Cont}(\mathbb{C})$, that is, two CQs $q(\bar{x})$ and $q'(\bar{x}')$, and a set $\Sigma \in \mathbb{C}$ of TGDs. By Lemma 2.5, we need to show that the problem of deciding whether $\bot(\bar{x})$ belongs

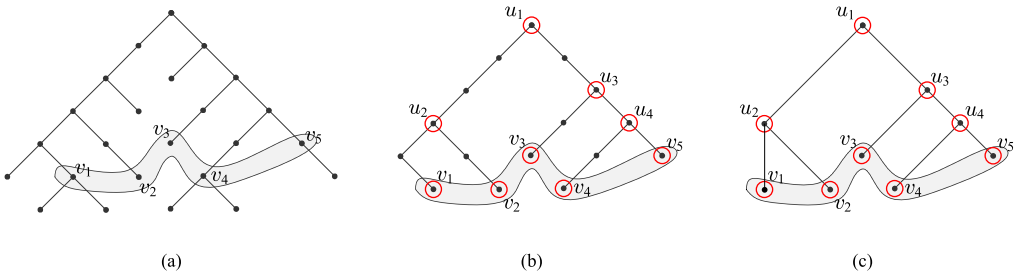


Fig. 3. The construction of the tree $F = (V, E)$ in the proof of Lemma 5.4.

to the evaluation of q' over chase(q, Σ), or, equivalently, the Boolean CQ $q'(\bot(\bar{x}))$, obtained by instantiating the free variables $\bar{x}'$ with $\bot(\bar{x})$, can be homomorphically mapped to chase(q, Σ), is decidable. The latter is equivalent to say that each model of the first-order sentence $\phi_{q,\Sigma}$ obtained by considering the conjunction of atoms in chase(q, Σ), with each null being interpreted as an existentially quantified variable, is a model of $q'(\bot(\bar{x}))$, i.e., the sentence $\Phi_{q,q',\Sigma} = \phi_{q,\Sigma} \wedge \neg q'(\bot(\bar{x}))$ is unsatisfiable. Consequently, to show that Cont$(\mathbb{C})$ is decidable, it suffices to show that the satisfiability problem for the fragment of first-order logic that allows for sentences $\Phi_{q,q',\Sigma}$, where q, q' are CQs, and $\Sigma \in \mathbb{C}$, is decidable. To this end, we show that this fragment enjoys the FTMP. Consider an arbitrary sentence $\Phi_{q,q',\Sigma}$ that falls in this fragment. If $\Phi_{q,q',\Sigma}$ is satisfiable, then it has a (possibly infinite) model M that has the same treewidth as the instance chase(q, Σ). Since Σ falls in a class that has GHW-preserving chase, the generalized hypertreewidth of M conincides with the generalized hypertreewidth of $D[q]$, and thus, is finite. Since, by definition, the treewidth of an instance is bounded by its generalized hypertreewidth times the maximum arity of the underlying schema, we conclude that the treewdith of M is finite. This implies that the fragment of first-order logic in question enjoys the FTMP, and the claim follows. □

Of course, Proposition 5.2 alone is not enough for showing that GHW-preserving chase ensures the decidability of SemGHW$_k$. We also need to show that, if the given query q is semantically in GHW$_k$ under the given set Σ of TGDs, then there exists a CQ $q' \in$ GHW$_k$ of "small" size such that $q \equiv_\Sigma q'$. We proceed to show such a small query property. By abuse of notation, for a CQ q, we write $|q|$ for the number of atoms occurring in q.

Proposition 5.3. *Consider a set Σ of TGDs over a schema σ that belongs to a class that has GHW-preserving chase, and a CQ q over σ. If q is semantically in* GHW$_k$ *under Σ, then there exists a CQ $q' \in$* GHW$_k$ *over σ, where $|q'| \leq |q| \cdot (2k + 1)$, such that $q \equiv_\Sigma q'$.*

Before giving the proof of the above result, we need to establish the main technical result of this section, which will also be used later in our investigation. At this point, we should say that a CQ may contain infinitely many atoms. The notions of evaluation, canonical instance, and generalized hypertreewidth, as well as Lemma 2.5, can be extended to infinite CQs. We show the following:

Lemma 5.4. *Consider a CQ q and a (possibly infinite) CQ $q' \in$* GHW$_k$*, both over a schema σ, such that $q' \subseteq q$. There exists a CQ $q'' \in$* GHW$_k$ *over σ such that $q' \subseteq q'' \subseteq q$ and $|q''| \leq |q| \cdot (2k + 1)$.*

Proof. Since, by hypothesis, $q' \in$ GHW$_k$, the canonical instance of q' admits a generalized hypertree decomposition (T, χ, λ) of width k. Our goal is, by exploiting the existence of (T, χ, λ) and the fact that $q' \subseteq q$, to first define a database $D \in$ GHW$_k$ over σ with at most $|q| \cdot (2k + 1)$ atoms, and then show that D can be converted into a CQ q'' over σ such that $q' \subseteq q'' \subseteq q$.

The Definition of the Database D

By hypothesis, $q'(\bar{x}') \subseteq q(\bar{x})$, and Lemma 2.5 implies the existence a homomorphism h from q to the canonical instance of q' such that $h(\bar{x}) = \bot(\bar{x}')$. We assume that q is of the form $\exists \bar{y} \bigwedge_{1 \leq i \leq m} R_i(\bar{t}_i)$. By definition, the atoms $R_1(h(\bar{t}_1)), \ldots, R_m(h(\bar{t}_m))$ are covered by some nodes $v_1, \ldots, v_n$, where $n \leq m$, of T. In other words, for each $1 \leq i \leq m$, there exists $1 \leq j \leq n$ such that $\chi(v_j)$ contains all the nulls occurring in $h(\bar{t}_i)$. Consider now the subtree T_q of T consisting of $v_1, \ldots, v_n$ and their ancestors in T. From T_q we extract the tree $F = (V, E)$ defined as follows:

- V consists of the root node of T_q, all the leaf nodes of T_q, and all the inner nodes of T_q with at least two children.
- For $v, u \in V$, $(v, u) \in E$ if and only if u is descendant of v in T_q, and the only nodes of V that occur on the unique shortest path from v to u in T_q is v and u.

The construction of F is graphically illustrated in Figure 3. The generalized hypertree decomposition (T, χ, λ) of q' is shown in Figure 3(a), where the shaded part are the nodes that cover the atoms in the image of the CQ q according to the homomorphism h. The subtree T_q of T is depicted in Figure 3(b), where the red nodes form the node set V of F. Finally, the tree $F = (V, E)$ is shown in Figure 3(c), which is essentially obtained from T_q by replacing the unique shortest path between two red nodes with a single edge. Let $D' = \bigcup_{v \in V} \lambda(v)$, i.e., D' is the database consisting of the atoms that label the nodes of F. At this point, one may be tempted to think that (F, χ', λ'), where χ' (resp., λ') is the restriction of χ (resp., λ) over the nodes of V, is a generalized hypertree decomposition of D'. Unfortunately, this is not the case; it is easy to verify that we may have an atom $R(\bar{t}) \in D'$, but no $v \in V$ such that $\chi'(v)$ contains all the nulls occurring in $\bar{t}$. However, from (F, χ', λ') we can construct a triple (F, χ'', λ'') that is a generalized hypertree decomposition of the database

$$D'' = \bigcup_{1 \leq i \leq m} R_i(h(\bar{t}_i)) \cup \bigcup_{v \in V} \lambda''(v)$$

with width k, which in turn implies that $D'' \in \mathrm{GHW}_k$. The construction of (F, χ'', λ'') from (F, χ', λ') follows. Consider an arbitrary node v of F such that $\chi'(v) = \{t_1, \ldots, t_\ell\}$, where $\ell \geq 1$, and the nulls in $\lambda'(v)$ are $\{t_1, \ldots, t_\ell\} \cup \{t_{\ell+1}, \ldots, t_{\ell+p}\}$, where $p \geq 0$; notice that $p = 0$ implies $\{t_{\ell+1}, \ldots, t_{\ell+p}\} = \varnothing$. Then $\chi''(v) = \{t_1, \ldots, t_\ell, \bot_{\ell+1}, \ldots, \bot_{\ell+p}\}$, where $\bot_{\ell+1}, \ldots, \bot_{\ell+p}$ are fresh nulls occurring only in $\chi''(v)$, i.e., there is no node $u \neq v$ such that $\chi''(u)$ contains any of those nulls, and $\lambda''(v)$ is obtained from $\lambda'(v)$ by replacing each null $t_{\ell+i}$, for $1 \leq i \leq p$, with the null $\bot_{\ell+i}$. For example, assume that $\chi'(v) = \{t_1, t_2\}$ and $\lambda'(v) = \{R(t_1, t_3), P(t_2, t_3, t_4)\}$. Then $\chi''(v) = \{t_1, t_2, \bot_3, \bot_4\}$ and $\lambda''(v) = \{R(t_1, \bot_3), P(t_2, \bot_3, \bot_4)\}$. It is easy to verify that (F, χ'', λ'') is indeed a generalized hypertree decomposition of D'' with width k, and thus $D'' \in \mathrm{GHW}_k$. It remains to show that D'' has at most $|q| \cdot (2k + 1)$ atoms, which then implies that the desired database D is precisely the database D''. By construction, F has $2 \cdot |q|$ nodes, while, for each node v, $|\lambda''(v)| \leq k$. Therefore, $|D''| \leq (2 \cdot |q| \cdot k + |q|) = |q| \cdot (2k + 1)$.

Converting D into a Query

We proceed to convert D into the desired CQ q''. We first observe that, by construction, for each $x \in \bar{x}'$ the null $\bot_x$ occurs in D. Let $q'(\bar{x}')$ be the CQ obtained by considering the conjunction of atoms in D, after renaming each null $\bot$ into a variable $v(\bot)$, with $v(\bot_x) = x$ for each $x \in \bar{x}'$. It is clear that $q'' \in \mathrm{GHW}_k$ since $D'' \in \mathrm{GHW}_k$, and $|q''| \leq |q| \cdot (2k + 1)$ since D'' contains at most $|q| \cdot (2k + 1)$ atoms. Moreover, $\bot(\bar{x}')$ belongs to the evaluation of q over $D[q'']$ due to the homomorphism h, and thus, by Lemma 2.5, $q'' \subseteq q$. Furthermore, there is a homomorphism μ from q'' to the canonical instance of q' such that $\mu(\bar{x}') = \bot(\bar{x}')$, and hence, by Lemma 2.5, $q' \subseteq q''$.

The homomorphism μ is obtained by reversing the renaming substitutions applied during the construction of (F, χ'', λ'') from (F, χ', λ'). Since $q' \subseteq q'' \subseteq q$ the claim follows. □

A result similar to Lemma 5.4 is implicit in [6] (see Claim 6.2), where the problem of approximating CQs is investigated. However, from the results of [6], we can only conclude the existence of an exponentially sized CQ $q'' \in \mathsf{GHW}_k$ such that $q' \subseteq q'' \subseteq q$, while the above lemma establishes the existence of a polynomially sized query. This is decisive for our later complexity analysis. Having the above lemma in place, it is not difficult to establish Proposition 5.3.

Proof. (of Proposition 5.3) Since, by hypothesis, q is semantically in GHW_k under Σ, there is a CQ $q''(\bar{x}) \in \mathsf{GHW}_k$ such that $q \equiv_\Sigma q''$. Let q''_Σ be the (possibly infinite) CQ obtained by applying the chase procedure over the atoms of q'' using the TGDs of Σ. More formally, $q''_\Sigma(\bar{x})$ is the CQ obtained by considering the (possibly infinite) conjunction of atoms in $\mathsf{chase}(q'', \Sigma)$, after renaming each null $\perp$ occurring in $\mathsf{chase}(q'', \Sigma)$ into a variable $v(\perp)$, with $v(\perp_x) = x$ for each $x \in \bar{x}$. By using Lemma 2.5, we can show that $q'' \subseteq_\Sigma q$ implies $q''_\Sigma \subseteq q$. Since $q'' \in \mathsf{GHW}_k$ and Σ belongs to a class that has GHW-preserving chase, we conclude that $q''_\Sigma \in \mathsf{GHW}_k$. Thus, by Lemma 5.4, there is a CQ $\hat{q} \in \mathsf{GHW}_k$ such that $q''_\Sigma \subseteq \hat{q} \subseteq q$ and $|\hat{q}| \leq |q| \cdot (2k+1)$. Since $q''_\Sigma \subseteq \hat{q}$ we get that $q'' \subseteq_\Sigma \hat{q}$; as above, this can be shown by using Lemma 2.5. By hypothesis, $q \subseteq_\Sigma q''$, and hence $q \subseteq_\Sigma q'' \subseteq_\Sigma \hat{q} \subseteq_\Sigma q$ since $\hat{q} \subseteq q$ immediately implies $\hat{q} \subseteq_\Sigma q$. Therefore, $\hat{q} \equiv_\Sigma q$ and the claim follows with $q' = \hat{q}$. □

The problem of determining whether a CQ q belongs to GHW_k is decidable. In fact, it is feasible in linear time for $k = 1$ [41], and NP-complete for $k > 1$ [24]. Therefore, Propositions 5.2 and 5.3 provide a decision procedure for $\mathsf{SemGHW}_k(\mathbb{C})$ whenever $\mathbb{C}$ has GHW-preserving chase. More precisely, given a CQ q and a set $\Sigma \in \mathbb{C}$, both over a schema σ:

(1) Guess a CQ q' over σ such that $|q'| \leq |q| \cdot (2k+1)$.
(2) If $q' \in \mathsf{GHW}_k$, $q \subseteq_\Sigma q'$ and $q' \subseteq_\Sigma q$, then accept; otherwise, reject.

The next decidability result follows:

Theorem 5.5. *For a class $\mathbb{C}$ of sets of TGDs that has GHW-preserving chase, $\mathsf{SemGHW}_k(\mathbb{C})$ is decidable.*

5.1 Guardedness

We proceed to show that $\mathsf{SemGHW}_k(\mathbb{G})$ is decidable and has the same complexity as $\mathsf{Cont}(\mathbb{G})$. As usual, for technical clarity, we assume that guarded TGDs have only one atom in the head. This is a common assumption since every set $\Sigma \in \mathbb{G}$ can be transformed in linear time into a set $\Sigma' \in \mathbb{G}$ with the desired property, while this transformation preserves containment; see, e.g., [11, 12].

Theorem 5.6. $\mathsf{SemGHW}_k(\mathbb{G})$ *is 2ExpTime-complete, and it becomes ExpTime-complete if the arity of the schema is fixed and NP-complete if the schema is fixed.*

Upper Bounds

The rest of this section is devoted to establishing Theorem 5.6. We first focus on the upper bounds, and start by showing that the class $\mathbb{G}$ has GHW-preserving chase. To this end, we need the auxiliary notion of the guarded chase forest, which is essentially a tree-like representation of the instance constructed by the chase. Consider an instance I, a set $\Sigma \in \mathbb{G}$, and an arbitrary chase sequence

$$I = I_0 \xrightarrow{\tau_0, (\bar{t}_0, \bar{t}'_0)} I_1 \xrightarrow{\tau_1, (\bar{t}_1, \bar{t}'_1)} I_2 \ldots$$

for I under Σ. The *guarded chase forest* for I and Σ (w.r.t. the above chase sequence) is a labeled forest $F = (V, E, \lambda)$, where $\lambda : V \to \mathsf{chase}(I, \Sigma)$, and

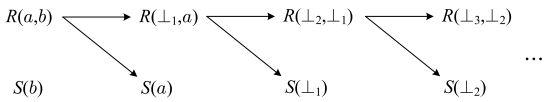


Fig. 4. Guarded chase forest.

- For each $R(\bar{t}) \in \text{chase}(I, \Sigma)$, there exists exactly one node $v \in V$ such that $\lambda(v) = R(\bar{t})$, and no other nodes occur in V, i.e., λ is a bijection.
- The edge (v, u) belongs to E if and only if, assuming that the atom $\lambda(u)$ is generated by the i-th chase step, i.e., $I_{i+1} \setminus I_i = \{\lambda(u)\}$, $\lambda(v) \in I_i$, and the guard of τ_i is satisfied by $\lambda(v)$.[8]

Example 5.7. Consider the database $D = \{R(a, b), S(b)\}$, the set Σ consisting of

$$\tau \;=\; R(x, y), S(y) \;\rightarrow\; \exists z\, R(z, x) \qquad \tau' \;=\; R(x, y) \;\rightarrow\; S(x),$$

and the chase sequence for D under Σ

$$D \xrightarrow{\tau,(a,b)} D \cup \{R(\bot_1, a)\} \xrightarrow{\tau',(a,b)} D \cup \{R(\bot_1, a), S(a)\} \xrightarrow{\tau,(\bot_1,a)}$$
$$D \cup \{R(\bot_1, a), S(a), R(\bot_2, \bot_1)\} \xrightarrow{\tau',(\bot_1,a)} D \cup \{R(\bot_1, a), S(a), R(\bot_2, \bot_1), S(\bot_1)\} \ldots,$$

where $\bot_1, \bot_2, \ldots \in \mathbf{N}$. The guarded chase forest for D and Σ is depicted in Figure 4. ∎

We are now ready to show that:

Proposition 5.8. *$\mathbb{G}$ has GHW-preserving chase.*

Proof. Consider a CQ $q \in \text{GHW}_k$, and a set of TGDs $\Sigma \in \mathbb{G}$. We need to show that $\text{chase}(q, \Sigma)$, that is, the result of an arbitrary chase sequence

$$D[q] = I_0 \xrightarrow{\tau_0,(\bar{t}_0,\bar{t}'_0)} I_1 \xrightarrow{\tau_1,(\bar{t}_1,\bar{t}'_1)} I_2 \ldots$$

for $D[q]$ under Σ, belongs to GHW_k, or, equivalently, it admits a generalized hypertree decomposition of width at most k. Let $F = (V, E, \lambda)$ be the guarded chase forest for $D[q]$ and Σ. We first show that each connected component of F, which is a tree with its root labeled by an atom α of $D[q]$, denoted T_α, enjoys the following property: For each term (constant or null) t occurring in T_α, the set $\{v \in V \mid t \text{ occurs in } \lambda(v)\}$ induces a connected subtree of T_α. Towards a contradiction, assume that the latter does not hold. This implies that there exists a path $vw_1 \ldots w_n u$ in T_α, where $n \geq 1$, and a term t that occurs in $\lambda(v)$ and $\lambda(u)$, but t does not occur in $\lambda(w_i)$, for each $i \in \{1, \ldots, n\}$. Assume that $\lambda(u)$ was generated during the i-th application of the chase step, i.e., $I_{i+1} \setminus I_i = \{\lambda(u)\}$. Since t does not occur in $\lambda(w_n)$, we conclude that τ_i is not guarded. But this contradicts our hypothesis that $\Sigma \in \mathbb{G}$; thus, T_α enjoys the desired property. This allows us to convert T_α into the generalized hypertree decomposition $(T_\alpha, \chi, \lambda)$ of width 1, where, for each $v \in V$, $\chi(v)$ consists of the nulls occurring in $\lambda(v)$. Since $q \in \text{GHW}_k$, it admits a generalized hypetree decomposition (T_q, χ', λ') of width at most k. Having $(T_\alpha, \chi, \lambda)$, for each atom $\alpha \in D[q]$, and (T_q, χ', λ') in place, we can now construct a generalized hypertree decomposition (T, χ'', λ'') of width at most k for $\text{chase}(q, \Sigma)$ as follows: For each $\alpha \in D[q]$, we attach $(T_\alpha, \chi, \lambda)$ to v_α, where v_α is a node in T_q such that $\chi'(v_\alpha)$ contains all the nulls in α. Note that v_α always exists since (T_q, χ') is a tree decomposition for q. Moreover,

[8]Here, we assume, w.l.o.g., that the chosen chase sequence is such that every chase step generates a different atom, i.e., for every $i > 0$, $I_{i+1} \setminus I_i \neq \emptyset$. This allows us to refer to *the* chase step that generates an atom of $\text{chase}(I, \Sigma) \setminus I$.

(T, χ'') is a tree decomposition for chase(q, Σ) since, for two distinct atoms $R_1(\bar{t}_1), R_2(\bar{t}_2) \in D[q]$, $T_{R_1(\bar{t}_1)}$ and $T_{R_2(\bar{t}_2)}$ share only the nulls occurring both in $\bar{t}_1$ and $\bar{t}_2$. This completes our proof. □

Proposition 5.8, together with Theorem 5.5, implies the decidability of SemGHW$_k(\mathbb{G})$. However, this does not say anything about the complexity of the problem. With the aim of pinpointing the exact complexity of SemGHW$_k(\mathbb{G})$, we proceed to analyze the complexity of the algorithm underlying Theorem 5.5. Recall that, given a CQ q and a set $\Sigma \in \mathbb{G}$, both over a schema σ, we simply need to guess a CQ q' over σ such that $|q'| \leq |q| \cdot (2k+1)$, and verify that $q' \in \text{GHW}_k$ and $q \equiv_\Sigma q'$. It is clear that this algorithm runs in non-deterministic polynomial time with a call to a C oracle, where C is a complexity class powerful enough for checking whether a CQ belongs to GHW$_k$ and solving Cont$(\mathbb{G})$. Recall that checking whether a CQ belongs to GHW$_k$ is in NP [24]. Thus, Proposition 2.6 implies that SemGHW$_k(\mathbb{G})$ is in 2ExpTime, in ExpTime if the arity of the schema is fixed, and in NP if the schema is fixed; recall that for $C \in \{\text{2ExpTime}, \text{ExpTime}\}$, $\text{NP}^C = C$. At this point, one may ask why for a fixed schema the obtained upper bound is NP and not $\text{NP}^{\text{NP}} = \Sigma_2^P$ since $C = \text{NP}$. Observe that the oracle is called only once in order to solve two problems that are already in NP, and thus it is not really needed in this case.

Lower Bounds

We proceed to show that the above complexity upper bounds are optimal. By Proposition 4.2, RestCont$_k(\mathbb{G})$ can be reduced in linear time to SemGHW$_k(\mathbb{G})$. Thus, to obtain the desired lower bounds, it suffices to reduce in polynomial time Cont$(\mathbb{G})$ to RestCont$_k(\mathbb{G})$. Interestingly, the lower bounds given in Section 2 for Cont$(\mathbb{G})$ hold even if we focus on Boolean CQs and the left-hand side query belongs to GHW$_1$. In fact, this is true, not only for guarded, but also for non-recursive and sticky sets of TGDs. Let BoolCont$_k(\mathbb{C})$ be the following problem: Given a Boolean CQ $q \in \text{GHW}_k$, a Boolean CQ q', and a set $\Sigma \in \mathbb{C}$ of TGDs, is it the case that $q \subseteq_\Sigma q'$?

From the above discussion, to establish the desired lower bounds for guarded sets of TGDs (and also for the other classes of TGDs considered in this work), it suffices to reduce in polynomial time BoolCont$_k$ to RestCont$_k$. To this end, we introduce the so-called connecting operator, which provides a generic reduction from BoolCont$_k$ to RestCont$_k$.

Connecting operator. Consider a Boolean CQ $q \in \text{GHW}_k$, a Boolean CQ q', and a set Σ of TGDs. Assume that q, q' are of the form $\exists \bar{y}\big(R_1(\bar{t}_1) \wedge \cdots \wedge R_m(\bar{t}_m)\big)$. The application of the *connecting operator* on (q, q', Σ) returns the triple $(\text{c}(q), \text{c}(q'), \text{c}(\Sigma))$, where

- c(q) is the query
$$\exists \bar{y} \exists w \big(R_1^\star(\bar{t}_1, w) \wedge \cdots \wedge R_m^\star(\bar{t}_m, w) \wedge \text{Aux}(w, w)\big),$$
where w is a new variable not in q, each $R_i^\star$ is a new predicate, and Aux is a new binary predicate not occurring in q, q' or Σ.
- c(q') is the query
$$\exists \bar{y} \exists w_1 \ldots \exists w_{2(k+1)} \big(R_1^\star(\bar{t}_1, w_1) \wedge \cdots \wedge R_m^\star(\bar{t}_m, w_1) \wedge \bigwedge_{1 \leq i < j \leq 2(k+1)} \text{Aux}(w_i, w_j)\big),$$
where $w_1, \ldots, w_{2(k+1)}$ are new variables not in q.
- Finally, $\text{c}(\Sigma) = \{\text{c}(\tau) \mid \tau \in \Sigma\}$, where for $\tau = \phi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z}\, \psi(\bar{x}, \bar{z})$, c$(\tau)$ is the TGD
$$\phi^\star(\bar{x}, \bar{y}, w) \rightarrow \exists \bar{z}\, \psi^\star(\bar{x}, \bar{z}, w),$$
where $\phi^\star(\bar{x}, \bar{y}, w)$ and $\psi^\star(\bar{x}, \bar{z}, w)$ are obtained from $\phi(\bar{x}, \bar{y})$ and $\psi(\bar{x}, \bar{z})$, respectively, by replacing each atom $R(x_1, \ldots, x_n)$ with $R^\star(x_1, \ldots, x_n, w)$, where w is a new variable.

This concludes the definition of the connecting operator. A class $\mathbb{C}$ of sets of TGDs is *closed under connecting* if, for every set $\Sigma \in \mathbb{C}$, $\mathrm{c}(\Sigma) \in \mathbb{C}$. It is not difficult to see that $q \subseteq_\Sigma q'$ if and only if $\mathrm{c}(q) \subseteq_{\mathrm{c}(\Sigma)} \mathrm{c}(q')$. It is easy to verify that $\mathrm{c}(q)$ belongs to GHW_k and is connected, $\mathrm{c}(\Sigma)$ is a set of body-connected TGDs, and $\mathrm{c}(q')$ is connected. However, we need to argue a bit more why $\mathrm{c}(q')$ is not semantically in GHW_k under $\mathrm{c}(\Sigma)$. To this end, we need an auxiliary result, which can be obtained from [8]. Recall that the core of a CQ q, which is unique up to variable renaming, is a CQ q' that is equivalent to q, i.e., $q(D) = q'(D)$, for every database D, and there is no CQ q'' with fewer atoms than q' that is equivalent to q. Recall also that a CQ q is semantically in GHW_k (without the presence of constraints) if there exists a CQ $q' \in \mathrm{GHW}_k$ that is equivalent to q. It is implicit in [8] that a CQ q is semantically in GHW_k if and only if the core of q belongs to GHW_k. Actually, the latter is shown only for $k = 1$, i.e., for acyclic CQs (see Proposition 8.4 in [8]), but it is an easy exercise to extend the proof to any $k \geq 1$. Now, observe that, by construction, the core of $\mathrm{c}(q')$ contains a clique-query over $2(k+1)$ variables in which only the binary predicate Aux is involved. This implies that $\mathrm{c}(q')$ is not semantically in GHW_k since the clique-query over n variables belongs to $\mathrm{GHW}_{\lceil \frac{n}{2} \rceil}$ but not to $\mathrm{GHW}_{\lceil \frac{n}{2} \rceil - 1}$. Since the predicate Aux does not occur in $\mathrm{c}(\Sigma)$, we get that $\mathrm{c}(q')$ is not semantically in GHW_k under $\mathrm{c}(\Sigma)$. From the above discussion, it is clear that the connecting operator provides a polynomial time reduction from $\mathsf{BoolCont}_k$ to $\mathsf{RestCont}_k$, for every class $\mathbb{C}$ of sets of TGDs that is closed under connecting. We can then easily show the following result:

Proposition 5.9. *For a class $\mathbb{C}$ of sets of TGDs that is closed under connecting, and $\mathsf{BoolCont}_k(\mathbb{C})$ is $\mathcal{C}$-hard, where $\mathcal{C}$ is a complexity class closed under polynomial time reductions, then $\mathsf{SemGHW}_k(\mathbb{C})$ is also $\mathcal{C}$-hard.*

Proof. It suffices to show that $\mathsf{BoolCont}_k(\mathbb{C})$ can be reduced in polynomial time to $\mathsf{SemGHW}_k(\mathbb{C})$. Since $\mathbb{C}$ is closed under connecting, the connecting operator provides a polynomial time reduction from $\mathsf{BoolCont}_k(\mathbb{C})$ to $\mathsf{RestCont}_k(\mathbb{C})$. Moreover, by Proposition 4.2, we know that $\mathsf{RestCont}_k(\mathbb{C})$ can be reduced in linear time to $\mathsf{SemGHW}_k(\mathbb{C})$. By composing these two reductions, we get a polynomial time reduction from $\mathsf{BoolCont}_k(\mathbb{C})$ to $\mathsf{SemGHW}_k(\mathbb{C})$, and the claim follows. □

Back to guardedness. It is clear that the class of guarded sets of TGDs is closed under connecting. Thus, the lower bounds for $\mathsf{SemGHW}_k(\mathbb{G})$ stated in Theorem 5.6 follow from Proposition 2.6 and Proposition 5.9. Note that, although Proposition 2.6 refers to $\mathsf{Cont}(\mathbb{G})$, the lower bounds hold for $\mathsf{BoolCont}_k(\mathbb{G})$; this can be observed after a careful inspection of the hardness proofs in [11].

As mentioned in Section 2, a key subclass of guarded sets of TGDs is the class of linear TGDs ($\mathbb{L}$), i.e., TGDs whose body consists of a single atom, which in turn subsumes the class of inclusion dependencies ($\mathbb{ID}$). By exploiting the non-deterministic procedure employed for $\mathsf{SemGHW}_k(\mathbb{G})$, and Proposition 2.7, we conclude that $\mathsf{SemGHW}_k(\mathbb{C})$, where $\mathbb{C} \in \{\mathbb{L}, \mathbb{ID}\}$, is in PSpace, and in NP if the arity of the schema is fixed. Moreover, since, by construction, $\mathbb{L}$ and $\mathbb{ID}$ are closed under connecting, we get matching lower bounds, and the next complexity result follows:

Theorem 5.10. $\mathsf{SemGHW}_k(\mathbb{C})$, *where $\mathbb{C} \in \{\mathbb{L}, \mathbb{ID}\}$, is PSpace-complete, and it becomes NP-complete if the arity of the schema is fixed.*

6 UCQ REWRITABILITY

Although the GHW-preserving chase criterion was very useful for solving $\mathsf{SemGHW}_k(\mathbb{G})$, it is of little use for non-recursive and sticky sets of TGDs. As illustrated in the following example, neither $\mathbb{NR}$ nor $\mathbb{S}$ has GHW-preserving chase:

Example 6.1. Consider the CQ and the TGD

$$q = P(x_1) \wedge \cdots \wedge P(x_n) \qquad \tau = P(x), P(y) \rightarrow R(x, y),$$

where $q \in \mathrm{GHW}_1$, and $\{\tau\}$ is non-recursive and sticky, but not guarded. In $\mathrm{chase}(q, \{\tau\})$ the predicate R holds all the pairs that can be formed using the nulls $\bot_{x_1}, \ldots, \bot_{x_n}$. More precisely,

$$\mathrm{chase}(q, \Sigma) \;=\; D[q] \;\cup \bigcup_{1 \le i,j \le n} \{R(\bot_{x_i}, \bot_{x_j})\}.$$

Therefore, even though q is acyclic, $\mathrm{chase}(q, \{\tau\})$ is highly cyclic; in fact, the generalized hypertreewidth of $\mathrm{chase}(q, \{\tau\})$ is $\lceil \frac{n}{2} \rceil$. ∎

Since the methods devised in Section 5 cannot be used for non-recursive and sticky sets of TGDs, new techniques must be developed. Interestingly, $\mathbb{NR}$ and $\mathbb{S}$ share an important property, which turned out to be very useful for our purposes, that is, UCQ rewritability. A *union of conjunctive queries (UCQ)* over a schema σ is an expression

$$Q(\bar{x}) \;=\; q_i(\bar{x}) \vee \cdots \vee q_n(\bar{x})$$

where each q_i is a CQ over σ. The evaluation of Q over an instance I, denoted $Q(I)$, is defined as $\bigcup_{1 \le i \le n} q_i(I)$. The formal definition of UCQ rewritability follows:

Definition 6.2. (**UCQ Rewritability**) We say that a class $\mathbb{C}$ of sets of TGDs is *UCQ rewritable* if, for every CQ q and $\Sigma \in \mathbb{C}$, both over a schema σ, we can construct a UCQ Q over σ such that, for every (finite or infinite) CQ q', $q' \subseteq_\Sigma q$ if and only if $q' \subseteq Q$. ∎

The above semantic property simply says that the problem of CQ containment under TGDs can be reduced to the problem of checking whether a CQ is contained into a UCQ. Since the latter is decidable (in fact, it is NP-complete) [39], we get that:

Proposition 6.3. *For a class $\mathbb{C}$ of sets of TGDs that is UCQ rewritable,* $\mathrm{Cont}(\mathbb{C})$ *is decidable.*

Let us stress that the reduction to the UCQ containment problem provided by UCQ rewritability depends only on the right-hand side CQ and the set of TGDs, but not on the left-hand side query. This is crucial for establishing the desirable small query property whenever we focus on sets of TGDs that belong to a UCQ rewritable class. At this point, let us clarify that the class of guarded sets of TGDs is not UCQ rewritable, which justifies our choice of a different semantic property, that is, GHW-preserving chase, for its study. This is shown via a simple example; for clarity, we use constants in the CQs, but the same can be shown even for constant-free queries:

Example 6.4. Consider the Boolean CQ and guarded TGD

$$q \;=\; P(b) \quad \text{and} \quad \tau \;=\; R(x, y), P(x) \rightarrow P(y)$$

where $b \in \mathbf{C}$. Assume that there exists a UCQ Q such that, for every Boolean CQ q', $q' \subseteq_{\{\tau\}} q$ if and only if $q' \subseteq Q$. Suppose that q' is of the form $P(a) \wedge \phi$, where ϕ is a conjunction of atoms of the form $R(c, d)$, where $c, d \in \mathbf{C}$. Recall that $q' \subseteq Q$ if and only if $Q(D[q']) \neq \varnothing$. This means that Q can check whether in $D[q']$ there exists a sequence of atoms $R(\bot_a, \bot_1), R(\bot_1, \bot_2), \ldots, R(\bot_{n-1}, \bot_b)$ of unbounded length. However, it is well-known that this is not possible via a finite (non-recursive) UCQ, which implies that $\mathbb{G}$ is not UCQ rewritable. ∎

Let us now show the desirable small query property. For each UCQ rewritable class $\mathbb{C}$ of sets of TGDs, there exists a computable function $f_{\mathbb{C}}(\cdot, \cdot)$ from the set of pairs (q, Σ), where q is a CQ and $\Sigma \in \mathbb{C}$, to the natural numbers, such that the following holds: For every CQ q, set $\Sigma \in \mathbb{C}$, and UCQ rewriting Q of q and Σ, the maximal size of its disjuncts is at most $f_{\mathbb{C}}(q, \Sigma)$. The existence of the function $f_{\mathbb{C}}$ follows by the definition of UCQ rewritability. We then show the following; in the rest of this section, we fix $k \ge 1$.

Proposition 6.5. *Consider a set Σ of TGDs over a schema σ that belongs to a UCQ rewritable class $\mathbb{C}$, and a CQ q over σ. If q is semantically in GHW_k under Σ, then there exists a CQ $q' \in \mathsf{GHW}_k$ over σ, where $|q'| \leq f_{\mathbb{C}}(q, \Sigma) \cdot (2k+1)$, such that $q \equiv_\Sigma q'$.*

Proof. Since, by hypothesis, q is semantically in GHW_k under Σ, there exists a CQ $q'' \in \mathsf{GHW}_k$ such that $q \equiv_\Sigma q''$. The fact that $\mathbb{C}$ is UCQ rewritable implies that there exists a UCQ Q over σ such that, for every (finite or infinite) CQ p, $p \subseteq_\Sigma q$ if and only if $p \subseteq Q$. Since $q'' \subseteq_\Sigma q$, we get that $q'' \subseteq Q$. This implies that there exists a disjunct $\hat{q}(\bar{x})$ of Q such that $q'' \subseteq \hat{q}$. By Lemma 5.4, there exists a CQ $q' \in \mathsf{GHW}_k$ such that $q'' \subseteq q' \subseteq \hat{q}$ and $|q'| \leq |\hat{q}| \cdot (2k+1) \leq f_{\mathbb{C}}(q, \Sigma) \cdot (2k+1)$. It remains to show that $q \equiv_\Sigma q'$. By hypothesis, $q \subseteq_\Sigma q''$, and thus, $q \subseteq_\Sigma q'$ since $q'' \subseteq q'$.

For the other direction, it suffices to show that $\hat{q} \subseteq_\Sigma q$, which immediately implies that $q' \subseteq_\Sigma q$ since $q' \subseteq \hat{q}$. Towards a contradiction, assume that $\hat{q} \not\subseteq_\Sigma q$. This means that there exists an instance I such that $I \models \Sigma$ and $\hat{q}(I) \not\subseteq q(I)$, i.e., there is a tuple $\bar{t} \in \mathsf{dom}(I)^{|\bar{x}|}$ such that $\bar{t} \in \hat{q}(I)$ and $\bar{t} \notin q(I)$. Without loss of generality, we assume that $\bar{t} = (\bot_{x_1}, \ldots, \bot_{x_n})$, where $\bar{x} = (x_1, \ldots, x_n)$ are the free variables of $\hat{q}$, with $\bot_{x_1}, \ldots, \bot_{x_n}$ being nulls. Since $I \models \Sigma$, $\mathsf{chase}(I, \Sigma)$ can be homomorphically mapped into I via a homomorphism that is the identity on $\mathsf{dom}(I)$, and thus, $\bar{t} \notin q(\mathsf{chase}(I, \Sigma))$; otherwise, $\bar{t} \in q(I)$ which contradicts the fact that $\bar{t} \notin q(I)$. Since $\hat{q}$ is a disjunct of Q, we get that $\bar{t} \in Q(I)$. Consider now the (possibly infinite) CQ q_I obtained by converting I into a CQ. More precisely, $q_I(\bar{x})$ is obtained by considering the conjunction of atoms in I, after renaming each null $\bot$ into a fresh variable $v(\bot)$, with $v(\bot_x) = x$ for each $x \in \bar{x}$. Clearly, $\bot(\bar{x}) \notin q(\mathsf{chase}(q_I, \Sigma))$ and $\bot(\bar{x}) \in Q(D[q_I])$. Hence, by Lemma 2.5, $q_I \not\subseteq_\Sigma q$ and $q_I \subseteq Q$. But this contradicts the fact that, for every (finite or infinite) CQ p, $p \subseteq_\Sigma q$ if and only if $p \subseteq Q$, and thus $\hat{q} \subseteq_\Sigma q$. □

It is clear that Propositions 6.3 and 6.5 provide a decision procedure for $\mathsf{SemGHW}_k(\mathbb{C})$ whenever $\mathbb{C}$ is UCQ rewritable. More precisely, given a CQ q and a set $\Sigma \in \mathbb{C}$, both over a schema σ, we can decide whether q is semantically in GHW_k under Σ as follows:

(1) Guess a CQ q' over σ such that $|q'| \leq f_{\mathbb{C}}(q, \Sigma) \cdot (2k+1)$.
(2) If $q' \in \mathsf{GHW}_k$, $q \subseteq_\Sigma q'$ and $q' \subseteq_\Sigma q$, then accept; otherwise, reject.

The next decidability result follows:

Theorem 6.6. *For a class $\mathbb{C}$ of sets of TGDs that is UCQ rewritable, $\mathsf{SemGHW}_k(\mathbb{C})$ is decidable.*

6.1 Non-Recursiveness

As already said, the crucial property of $\mathbb{NR}$ that we are going to exploit for solving $\mathsf{SemGHW}_k(\mathbb{NR})$ is UCQ rewritability. For a set Σ of TGDs, let b_Σ be the maximum number of atoms in the body of a TGD of Σ, and p_Σ be the number of predicates occurring in Σ. The next result is implicit in [31]:[9]

Proposition 6.7. *$\mathbb{NR}$ is UCQ rewritable with $f_{\mathbb{NR}}(q, \Sigma) = |q| \cdot (b_\Sigma)^{p_\Sigma}$.*

The above result, together with Theorem 6.6, implies that $\mathsf{SemGHW}_k(\mathbb{NR})$ is decidable. For the exact complexity of the problem, we need to analyze the complexity of the non-deterministic algorithm underlying Theorem 6.6. Observe that when the schema is fixed the function $f_{\mathbb{NR}}$ is polynomial, and therefore, Proposition 6.7 guarantees the existence of a polynomially sized CQ in GHW_k. In this case, the fact that checking whether a CQ falls in GHW_k is in NP together with the fact that $\mathsf{Cont}(\mathbb{NR})$ is in NP (by Proposition 2.8) immediately imply that $\mathsf{SemGHW}_k(\mathbb{NR})$ is in NP.

Things are a bit cryptic when the schema is not fixed. In this case, $f_{\mathbb{NR}}$ is exponential, and thus we have to guess a CQ q' of exponential size. It is clear that checking whether q' falls in GHW_k is

[9] Let us clarify that the work [31] does not explicitly consider the class $\mathbb{NR}$. However, the rewriting algorithm proposed in that paper works also for non-recursive sets of TGDs.

in NExpTime. However, the fact that Cont($\mathbb{NR}$) is in NExpTime (by Proposition 2.8) alone is not enough to conclude that SemGHW$_k(\mathbb{NR})$ is also in NExpTime. We need to understand better the complexity of the query containment algorithm for $\mathbb{NR}$. Recall that given two CQs $q(\bar{x})$, $q'(\bar{x})$, and a set $\Sigma \in \mathbb{NR}$, by Lemma 2.5, $q \subseteq_\Sigma q'$ if and only if $\bot(\bar{x}) \in q'(\text{chase}(q, \Sigma))$. We know from [35] that if $\bot(\bar{x}) \in q'(\text{chase}(q, \Sigma))$, then there exists a chase sequence

$$D[q] = I_0 \xrightarrow{\tau_0,(\bar{t}_0,\bar{t}'_0)} I_1 \xrightarrow{\tau_1,(\bar{t}_1,\bar{t}'_1)} I_2 \ldots I_{n-1} \xrightarrow{\tau_{n-1},(\bar{t}_{n-1},\bar{t}'_{n-1})} I_n$$

for $D[q]$ and Σ, where $n = |q'| \cdot (b_\Sigma)^{O(p_\Sigma)}$, such that $\bot(\bar{x}) \in q'(I_n)$. The query containment algorithm for $\mathbb{NR}$ simply guesses such a chase sequence for $D[q]$ and Σ, and checks whether $\bot(\bar{x}) \in q'(I_n)$. Since n is exponential, this algorithm runs in non-deterministic exponential time. Now, recall that for SemGHW$_k(\mathbb{NR})$ we need to perform two containment checks where either the left-hand side or the right-hand side query is of exponential size. But in both cases the containment algorithm for $\mathbb{NR}$ runs in non-deterministic exponential time, and hence SemGHW$_k(\mathbb{NR})$ is in NExpTime. The lower bounds are inherited from BoolCont$_k(\mathbb{NR})$ since $\mathbb{NR}$ is closed under connecting (see Proposition 5.9). From the above discussion we conclude that:

Theorem 6.8. SemGHW$_k(\mathbb{NR})$ *is NExpTime-complete, even if the arity of the schema is fixed, and it becomes NP-complete if the schema is fixed.*

6.2 Stickiness

We now focus on sticky sets of TGDs. As for $\mathbb{NR}$, the key property of $\mathbb{S}$ that we are going to use is UCQ rewritability. For a CQ q and a set Σ of TGDs, let $p_{q,\Sigma}$ be the number of predicates occurring in q and Σ, and $a_{q,\Sigma}$ be the maximum arity over all those predicates. The next result is from [31]:

Proposition 6.9. $\mathbb{S}$ *is UCQ rewritable with* $f_{\mathbb{S}}(q, \Sigma) = p_{q,\Sigma} \cdot (a_{q,\Sigma} \cdot |q| + 1)^{a_{q,\Sigma}}$.

The above result, together with Theorem 6.6, implies that SemGHW$_k(\mathbb{S})$ is decidable. Moreover, Proposition 6.9 allows us to establish an optimal upper bound when the arity of the schema is fixed since in this case the function $f_{\mathbb{S}}$ is polynomial. In fact, we show that SemGHW$_k(\mathbb{S})$ is NP-complete when the arity of the schema is fixed. The NP-hardness is inherited from BoolCont$_k(\mathbb{S})$ since $\mathbb{S}$ is closed under connecting (see Proposition 5.9). However, when the arity of the schema is not fixed the picture is still foggy. In this case, the function $f_{\mathbb{S}}$ is exponential, and thus, by following the usual guess and check approach we get that SemGHW$_k(\mathbb{S})$ is in NExpTime, while Proposition 5.9 implies ExpTime-hardness. Summing up, our machinery based on UCQ rewritability shows that:

Theorem 6.10. SemGHW$_k(\mathbb{S})$ *is in NExpTime and ExpTime-hard. It becomes NP-complete if the arity of the schema is fixed.*

An interesting question that comes up is whether for sticky sets of TGDs a stronger small query property than Proposition 6.5 can be established, which guarantees the existence of a polynomially sized equivalent CQ that falls in GHW$_k$. It is clear that such a result would allow us to establish an ExpTime upper bound for SemGHW$_k(\mathbb{S})$. At this point, one may think that this can be achieved by showing that the function $f_{\mathbb{S}}$ is actually polynomial, even if the arity of the schema is not fixed. Unfortunately, this is not the case. There exists a CQ q and a family $\{\Sigma_n\}_{n>0}$ of sticky sets of TGDs, where $a_{q,\Sigma_n} = n + 2$, such that $f_{\mathbb{S}}(q, \Sigma_n) \geq 2^n$. The set $\Sigma_n \in \mathbb{S}$ consists of the following TGDs; for brevity, we write $\bar{x}_i^j$ for the tuple of variables $x_i, x_{i+1}, \ldots, x_j$:

$$\begin{aligned} R_i(\bar{x}_1^{i-1}, z, \bar{x}_{i+1}^n, z, o), R_i(\bar{x}_1^{i-1}, o, \bar{x}_{i+1}^n, z, o) &\rightarrow R_{i-1}(\bar{x}_1^{i-1}, z, \bar{x}_{i+1}^n, z, o), \quad 1 \leq i \leq n, \\ R_0(\underbrace{z, \ldots, z}_{n}, z, o) &\rightarrow \text{Ans}(z, o), \end{aligned}$$

while $q = \text{Ans}(0, 1)$. Let us clarify that Σ_n is indeed a sticky set of TGDs since, for every TGD $\tau \in \Sigma_n$, all the variables in the body of τ appear also in the head of τ. This means that there are no variables that are marked in Σ_n, which implies that stickiness is trivially satisfied. Consider now an arbitrary (Boolean) UCQ Q such that, for every CQ q', $q' \subseteq_{\Sigma_n} q$ if and only if $q' \subseteq Q$ (or, equivalently, $Q(D[q']) \neq \varnothing$). It is not difficult to see that the disjunct of Q that mentions only the predicate R_n contains exactly 2^n atoms. This implies that the maximal size of the disjuncts of Q is at least 2^n, i.e., $f_{\mathbb{S}}(q, \Sigma_n) \geq 2^n$, as needed. Therefore, we need a more refined property for stickiness than UCQ rewritability, which will allow us to close the complexity of $\text{SemGHW}_k(\mathbb{S})$ when the arity is not fixed (unless the problem is NExpTime-hard). This is left as an interesting open problem.

7 SEMANTIC GENERALIZED HYPETREEWIDTH UNDER EGDS

In the previous sections, we were dealing with TGDs. Let us now concentrate on EGDs. From Lemma 2.5, and the fact that the chase under EGDs always terminates, we immediately get the decidability of CQ containment under sets of EGDs. One might think that this allows us to solve $\text{SemGHW}_k(\mathbb{EGD})$, where $\mathbb{EGD}$ is the class of sets of EGDs. Unfortunately, the problem is undecidable even for $k = 1$. The proof is along the lines of the proof of Theorem 4.4, which establishes the undecidability of SemGHW_1 under full TGDs, and can be found in the appendix.

At this point, we would like to clarify that the conference paper [5] provides a proof for the fact that $\text{SemGHW}_1(\mathbb{F})$ is undecidable that is simpler than the proof of Theorem 4.4. However, it is not clear at all how the proof in [5] can be adapted to the case of EGDs. Thus, although the proof of Theorem 4.4 is more complicated than the one given in [5], it has the advantage that it can be easily adapted to the case of EGDs, which leads to the following undecidability result:

Theorem 7.1. $\text{SemGHW}_1(\mathbb{EGD})$ *is undecidable, even if we allow only unary and binary predicates.*

7.1 Towards Positive Results

Theorem 7.1 brings us to the crucial question whether restricted classes of EGDs ensure the decidability of SemGHW_k. At this point, one may wonder whether the techniques developed in the previous sections for TGDs can be applied for EGD-based classes of constraints. Unfortunately, the situation changes dramatically even for simple EGD-based classes such as keys. Recall that a functional dependency (FD) over a schema σ is an expression of the form $R[A \rightarrow B]$, where R/n is a relation symbol in σ, and $A, B \subseteq \{1, \ldots, n\}$, asserting that the values of the attributes of B are determined by the values of the attributes of A. A FD $R[A \rightarrow B]$ is called *key* if $A \cup B = \{1, \ldots, n\}$.

It is not difficult to show that the techniques developed in the previous sections for TGDs cannot be used for studying SemGHW_k under EGDs. Indeed, although the notions of GHW-preserving chase (Definition 5.1) and UCQ rewritability (Definition 6.2) can be naturally defined for EGDs, are of little use even if we focus on keys. It is easy to show via a simple example that keys do not enjoy the GHW-preserving chase property:

Example 7.2. Let $q \in \text{GHW}_1$ be the CQ

$$R(x, y) \wedge S(x, y, z) \wedge S(x, z, w) \wedge S(x, w, v) \wedge R(x, v).$$

After applying on q the key $R[1 \rightarrow 2]$[10] we obtain the CQ

$$R(x, y) \wedge S(x, y, z) \wedge S(x, z, w) \wedge S(x, w, y),$$

which clearly falls in GHW_2 but not in GHW_1. ∎

It is also not hard to show that keys over unary and binary predicates are not UCQ rewritable. This is not surprising due to the transitive nature of equality. Here is a simple example:

[10]For brevity, we write $R[1 \rightarrow 2]$ instead of the more formal $R[\{1\} \rightarrow \{2\}]$.

Example 7.3. Consider the Boolean CQs and key

$$q \;=\; \exists x\,(R(x,x)\wedge P(x)) \quad \text{and} \quad \tau \;=\; R[1\rightarrow 2].$$

Assume there exists a UCQ Q such that $q' \subseteq_{\{\tau\}} q$ if and only if $q' \subseteq Q$. Thus, the latter holds for every Boolean CQ q' of the form $\exists \bar{x}\,(R(x,x)\wedge\phi\wedge P(y))$, where $x, y \in \bar{x}$ and ϕ is a conjunction of atoms that use only the predicate R and variables from $\bar{x}$. Recall that $q' \subseteq Q$ if and only if $Q(D[q']) \neq \varnothing$. Hence, Q checks whether in $D[q']$ there is a sequence of atoms $R(\bot_x, \bot_1), R(\bot_1, \bot_2), \ldots, R(\bot_{n-1}, \bot_y)$ of unbounded length. However, this can only be done via a recursive query, and thus, it is not possible via a finite UCQ, which implies that keys are not UCQ rewritable. ∎

From the above discussion, it is clear that we cannot reuse the techniques introduced in Sections 5 and 6 for EGD-based classes such as keys. However, Example 7.2 leaves some room for using the GHW-preserving chase property if we focus on schemas consisting of unary and binary predicates. Indeed, the example relies on the fact that both binary and ternary predicates are mentioned in the CQ. Interestingly, assuming schemas with unary and binary predicates only, we can show that the class of keys, denoted $\mathbb{K}$, enjoys the GHW-preserving chase property as long as we focus on CQs from GHW_1, i.e., acyclic CQs. In other words, $\mathbb{K}$ has *acyclicity-preserving chase*, which is defined in the obvious way; roughly, given $q \in \mathrm{GHW}_1$ and $\Sigma \in \mathbb{K}$, $\mathrm{chase}(q, \Sigma)$ is acyclic.

Proposition 7.4. *For schemas with unary and binary predicates, $\mathbb{K}$ has acyclicity-preserving chase.*

Proof. Consider a CQ $q \in \mathrm{GHW}_1$ over unary and binary predicates. It suffices to show that, after applying a key $R[1\rightarrow 2]$ on q, the obtained query q' is in GHW_1. Note that over unary and binary predicates, GHW_1 is equivalent to acyclicity of the underlying undirected graph of q, without self-loops nor parallel edges. Note that q' is obtained from q by collapsing two nodes at distance ≤ 2 of this graph. Since acyclicity is preserved by this operation on graphs the claim follows. □

Having the above result in place, we can then establish a small query property for $\mathbb{K}$, similar to the one given in Proposition 5.3 for classes of TGDs that have GHW-preserving chase. This, together with the fact that CQ containment under $\mathbb{K}$ is in NP, shows that checking whether a CQ is semantically acyclic under a set of keys (assuming unary and binary predicates) is in NP. A matching lower bound follows from [19], which shows that the problem of checking whether a Boolean CQ over a single binary relation is equivalent to an acyclic one is NP-hard. Then:

Theorem 7.5. *Assuming schemas with unary and binary predicates, $\mathrm{SemGHW}_1(\mathbb{K})$ is NP-complete.*

Can we generalize Proposition 7.4 to queries from GHW_k, for any $k \geq 1$? In other words, can we show that $\mathbb{K}$ has GHW-preserving chase, always assuming unary and binary predicates, without focussing on acyclic CQs? Such a generalization of Proposition 7.4 will immediately show that Theorem 7.5 extends to $\mathrm{SemGHW}_k(\mathbb{K})$, for every $k \geq 1$. Unfortunately, as shown by the following example, this is not the case. Actually, we can show that the chase can arbitrarily increase the generalized hypertreewidth by using one key only. This shows that keys behave in a fundamentally different way when we focus on acyclic CQs and when we consider CQs from GHW_k, for $k > 1$.

Example 7.6. Consider the schema $\sigma = \{R/2, S/2\}$, and let $\Sigma = \{R[1\rightarrow 2]\}$. For every $n \geq 2$, we can construct a CQ $q_n \in \mathrm{GHW}_2$, while the instance $\mathrm{chase}(q_n, \Sigma)$ belongs to $\mathrm{GHW}_{\lceil \frac{n}{2} \rceil}$. The CQ q_n is defined as the one in Figure 5. It is clear that $\mathrm{chase}(q_n, \Sigma)$ is of the form

$$\{S(\bot_{i,j}, \bot_{i+1,j}) \mid i \in \{1,\ldots,n-1\} \text{ and } j \in \{1,\ldots,n\}\}$$
$$\cup\; \{R(\bot_{i,j}, \bot_{i,j+1}) \mid i \in \{1,\ldots,n\} \text{ and } j \in \{1,\ldots,n-1\}\}.$$

It is easy to verify that $\mathrm{chase}(q_n, \Sigma)$ is of width at least $\lceil \frac{n}{2} \rceil$ since it corresponds to an $n \times n$ grid. ∎

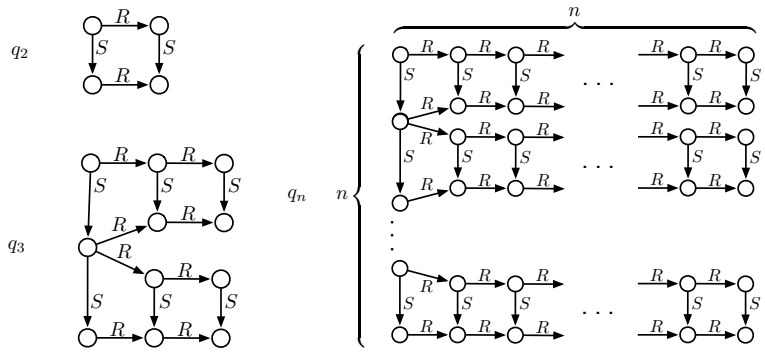


Fig. 5. Depiction of q_2, q_3, q_n.

Assuming unary and binary predicates, can we show that $\mathsf{SemGHW}_k(\mathbb{K})$, for $k > 1$, is decidable by exploiting different techniques than the ones employed so far? It turned out that this is a highly non-trivial question, which we affirmatively answer in the next section via a rather involved proof. Such a positive result may have interesting implications to graph databases and description logics, where only unary and binary predicates are employed. In fact, the main result of the next section has been already used in [4], which studies similar problems concerning description logics.

8 SEMANTIC GENERALIZED HYPETREEWIDTH UNDER KEYS

The goal of this section is to show that $\mathsf{SemGHW}_k(\mathbb{K})$, for $k > 1$, is decidable assuming schemas with unary and binary predicates only. Actually, for technical clarity, we exclude unary predicates, and we also assume that binary relations have at most one key, which in turn allows us to focus on keys of the form $R[1 \rightarrow 2]$, i.e., always the first attribute functionally determines the second one. We refer to this kind of schemas as *binary*. However, our decidability result can be easily extended to the general case, where unary predicates can be used, and binary predicates can have more than one key. The main result of this section follows; henceforth, we fix $k > 1$:

Theorem 8.1. *For binary schemas,* $\mathsf{SemGHW}_k(\mathbb{K})$ *is decidable in elementary time.*

Before we present the proof of this theorem, we need to introduce some terminology. For clarity, we are going to focus on Boolean CQs (BCQ), but the proof can be easily extended to non-Boolean CQs. Consider two BCQs q and p. We write $q \rightarrow p$ for the fact that there is a homomorphism from q to p, and we write $h : q \rightarrow p$ to say that h is a homomorphism that witnesses the fact $q \rightarrow p$. We call q and p *homomorphically equivalent* (or simply *hom-equivalent*) if $q \rightarrow p$ and $p \rightarrow q$. We say that q and p are *isomorphic*, denoted by $q \cong p$, if there are $h : q \rightarrow p$ and $g : p \rightarrow q$ such that h is the inverse of g, *i.e.*, if both $h \circ g$ and $g \circ h$ are the identity. The *core* of a BCQ q, denoted by $\mathsf{core}(q)$, is a subquery p of q of minimal size that is homomorphically equivalent to q, *i.e.*, $q \equiv p$.

Recall that $\mathsf{chase}(q, \Sigma)$, where q is a BCQ and Σ a set of keys, refers to the instance $\mathsf{chase}(D[q], \Sigma)$, which can be naturally seen as a BCQ: simply consider the conjunction of the atoms occurring in $\mathsf{chase}(D[q], \Sigma)$, and then rename each null $\perp$ into an existentially quantified variable $v(\perp)$. Notice that in this case $\mathsf{chase}(q, \Sigma)$ is always finite. We write $q \Rightarrow_\Sigma p$ to denote that $D[p]$ can be obtained from $D[q]$ via a single chase step, i.e., $D[p]$ is the result of one application of a key of Σ that is applicable to $D[q]$. We write $\Rightarrow_\Sigma^*$ for the reflexive-transitive closure of $\Rightarrow_\Sigma$. Note that each step of the chase $q \Rightarrow_\Sigma p$ induces a homomorphism h that maps q to p by identifying the variables affected by the chase. This extends to a sequence of chase steps by composition of homomorphisms. For any

$q \Rightarrow^*_\Sigma p$ we refer to its induced homomorphism h that maps q to p as the *provenance homomorphism*, since it tracks the set of variables that give rise to any variable of the chased query.

8.1 General Structure of the Proof

A bird's eye view of the proof of Theorem 8.1 is as follows:

- We first characterize the class of BCQs that are semantically in GHW_k in terms of the existence of certain BCQs. In fact, we show that a BCQ q is semantically in GHW_k if and only if there is a BCQ in GHW_k such that the core of its chase is isomorphic to the core of the chase of q (Proposition 8.2).
- We then check for the existence of such a BCQ via a Monadic Second-Order (MSO) sentence by relying on the fact that satisfiability of MSO sentences over databases of bounded generalized hypertreewidth is decidable (Proposition 8.3).

Let us give more details for each of the above steps. The first one is easy. Consider a BCQ q and a set Σ of keys. We first define the set of BCQs

$$\mathsf{W}_{q,\Sigma} \;=\; \{p \mid \mathsf{core}(\mathsf{chase}(p,\Sigma)) \cong \mathsf{core}(\mathsf{chase}(q,\Sigma))\},$$

which simply collects all the BCQs such that the core of their chase is isomorphic to the core of the chase of q. We can then show the following:

Proposition 8.2. *Consider a BCQ q and a set Σ of keys. The following are equivalent:*

(1) q is semantically in GHW_k under Σ.
(2) $\mathsf{W}_{q,\Sigma} \cap \mathsf{GHW}_k \neq \varnothing$.

Proof. For (1) $\Rightarrow$ (2), by hypothesis, there exists a BCQ $p \in \mathsf{GHW}_k$ such that $q \equiv_\Sigma p$, and thus, $\mathsf{chase}(q,\Sigma) \equiv \mathsf{chase}(p,\Sigma)$ from Lemma 2.5. It is well-known that if two BCQs are hom-equivalent, then their cores are isomorphic (see, e.g., [32]). Hence, $\mathsf{core}(\mathsf{chase}(p,\Sigma)) \cong \mathsf{core}(\mathsf{chase}(q,\Sigma))$, which implies that $p \in \mathsf{W}_{q,\Sigma} \cap \mathsf{GHW}_k \neq \varnothing$, as needed. For (2) $\Rightarrow$ (1), by hypothesis, there exists a BCQ $p \in \mathsf{W}_{q,\Sigma} \cap \mathsf{GHW}_k$. By Lemma 2.5, we conclude that $q \equiv_\Sigma p$, and the claim follows. □

The second main step of the proof of Theorem 8.1 is to show that the problem of deciding whether $\mathsf{W}_{q,\Sigma} \cap \mathsf{GHW}_k \neq \varnothing$ can be reduced to the satisfiability problem of MSO sentences over databases that belong to GHW_k. In particular, we want to show the following:

Proposition 8.3. *Consider a BCQ q and a set Σ of keys. There is an MSO-definable set $\mathsf{M}_{q,\Sigma}$ of BCQs such that the following are equivalent:*

(1) $\mathsf{W}_{q,\Sigma} \cap \mathsf{GHW}_k \neq \varnothing$;
(2) $\mathsf{M}_{q,\Sigma} \cap \mathsf{GHW}_k \neq \varnothing$.

Furthermore, $\mathsf{M}_{q,\Sigma}$ is definable via an MSO sentence $\xi_{q,\Sigma}$ that can be computed in single exponential time from q and Σ, and has a fixed number of quantifier alternations.

The combination of Propositions 8.2 and 8.3 imply Theorem 8.1. Indeed, since satisfiability of MSO is decidable on databases of bounded generalized hypertreewidth (by a simple adaptation of the same result on bounded treewidth databases [40]), we would therefore obtain a decision procedure for $\mathsf{SemGHW}_k(\mathbb{K})$. This procedure takes elementary time since the sentence $\xi_{q,\Sigma}$ provided by Proposition 8.3 can be computed in exponential time, and it has a fixed number $\ell \geq 1$ of quantifier alternations (*i.e.*, ℓ is independent of q and Σ), and hence the problem of checking whether $\xi_{q,\Sigma}$ has a model in GHW_k can be solved in ℓ-ExpTime [40].

At this point, one might wonder whether Proposition 8.3 could be established by simply showing that the set $\mathsf{W}_{q,\Sigma}$ is MSO-definable, that is, by showing that there exists an effective MSO sentence

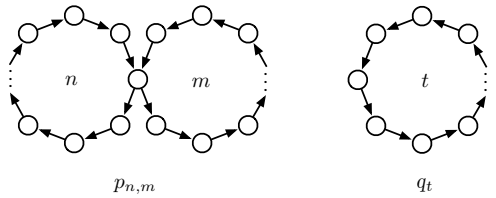


Fig. 6. Counterexample of Lemma 8.4.

φ such that $p \in \mathrm{GHW}_k$ belongs to $\mathsf{W}_{q,\Sigma}$ iff $D[p]$ is a model of φ. Unfortunately, as we show below, this is not possible since the pre-image of the chase is, in general, not MSO-definable.

Lemma 8.4. *For a BCQ q and a set of keys Σ, $\{p \mid \mathrm{chase}(p, \Sigma) \cong q\}$ is not MSO-definable.*

Proof. Let q_t and $p_{n,m}$, for $t, n, m \geq 1$, be BCQs as defined in Figure 6 over a schema consisting of a single binary relation R. That is, $p_{n,m}$ consists of two "nested" R-cycles of size n and m, where n and m refers to the number of edges, and q_t is a single R-cycle of size t. Let $\Sigma = \{R[1 \rightarrow 2]\}$. Notice that for $n > m$ we have that $p_{n,m} \Rightarrow_\Sigma^* p_{n-m,m}$ and $\mathrm{chase}(p_{n,n}, \Sigma) = q_n$. Thus, $\mathrm{chase}(p_{n,m}, \Sigma)$ essentially computes the greatest common divisor of n and m, $\mathrm{GCD}(n, m)$, through the Euclidean algorithm, i.e., $\mathrm{chase}(p_{n,m}, \Sigma) \cong q_{\mathrm{GCD}(n,m)}$. Suppose now, for the sake of contradiction, that there exists an MSO sentence ϕ of quantifier rank ℓ such that, for every BCQ q, it is the case that $D[q] \models \phi$ iff $\mathrm{chase}(q, \Sigma) = q_1$, where q_1 corresponds to a singleton node with a loop, that is, the query $\exists x\, R(x, x)$. For two BCQs q and q', we write $q \equiv_\ell q'$ to denote that q and q' are indistinguishable by MSO sentences of quantifier rank ℓ. It is well-known that the equivalence class $\equiv_\ell$ is of finite index (cf., [34]). It is not hard to prove, using a standard argument based on Ehrenfeucht-Fraïssé games, that if $q_n \equiv_\ell q_{n'}$ and $q_m \equiv_\ell q_{m'}$, where $n, n', m, m' \geq 1$, then also $p_{n,m} \equiv_\ell p_{n',m'}$. Let s_i be the i-th smallest positive prime number. Since $\equiv_\ell$ is of finite index, there must be i and j with $i < j$ such that $q_{s_i} \equiv_\ell q_{s_j}$. Therefore, $D[p_{s_j,(s_j-1)!}] \models \phi$ iff $D[p_{s_i,(s_j-1)!}] \models \phi$, which is in contradiction with our assumption since $\mathrm{GCD}(p_j, (p_j - 1)!) = 1$ but $\mathrm{GCD}(p_i, (p_j - 1)!) \neq 1$. □

Since the BCQs in the proof above are all cores, we also have that:

Corollary 8.5. *For a BCQ q and a set of keys Σ, $\mathsf{W}_{q,\Sigma}$ is not MSO-definable.*

In view of the above negative result, the statement of Proposition 8.3 makes now more sense. To prove it, we will define the set $\mathsf{M}_{q,\Sigma}$ as an MSO-definable superset of $\mathsf{W}_{q,\Sigma}$ in such a way that every BCQ $p \in \mathsf{M}_{q,\Sigma}$ can be modified into a BCQ $\hat{p}$, without increasing its width, such that $\hat{p} \in \mathsf{W}_{q,\Sigma}$. In this way, we obtain Proposition 8.3, as illustrated in Figure 7. We call $\hat{p}$ the "expansion" of p, since it consists of attaching to each variable of p a query of width 2 (at most).

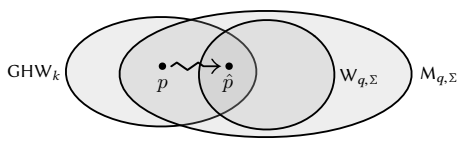


Fig. 7. Basic idea of the proof of Proposition 8.3.

Fix a BCQ q and a set Σ of keys. In the rest of the section, we define the expansion of a BCQ (Section 8.2) in a way that preserves the width, we define $\mathsf{M}_{q,\Sigma}$ (Section 8.3), we show that $\mathsf{W}_{q,\Sigma} \subseteq$

$\mathsf{M}_{q,\Sigma}$ (Section 8.4), and we show that the expansion of every query from $\mathsf{M}_{q,\Sigma}$ belongs to $\mathsf{W}_{q,\Sigma}$ (also in Section 8.4). We conclude, in Section 8.5, with a brief reminder of how all the pieces fit together to prove Theorem 8.1. Recall that whenever we focus on binary predicates, we can naturally represent CQs as directed edge-labeled graphs. Hence, a 'node' of a BCQ p is simply a variable in p, and an 'edge' $x \xrightarrow{R} y$ of p is an atom $R(x, y)$ in p. Moreover, we write $p|_\Sigma$ for the BCQ obtained by restricting p to the atoms that have a relation name occurring in Σ.

8.2 The Expansion of a CQ

A Σ-*cycle* of a BCQ p is a subquery $p' \subseteq p|_\Sigma$ consisting of a directed simple cycle over the relations of Σ. Similarly, a Σ-*path* of p is a subquery induced by a directed simple path in $p|_\Sigma$. Given a Σ-cycle C of p, a node x of p in the same strongly connected component (SCC) as C, and a (possibly empty) simple Σ-path π of p from x to a node of C, let $P_{x,\pi,C}$ be the query $\pi \cup C$, where all variables but x are renamed to fresh variables. We call $P_{x,\pi,C}$ a *petal*, and we say that a variable $y \neq x$ of $P_{x,\pi,C}$ is a "copy" of a variable y' of $\pi \cup C$ if y was the result of renaming y' to y. The *flower decomposition (w.r.t. Σ)* of (p, x) is the union of all $P_{x,\pi,C}$ for all Σ-paths π of q, and Σ-cycles C of q in the SCC of x; we illustrated this notion in Figure 8. We call x the *root* of the flower decomposition.

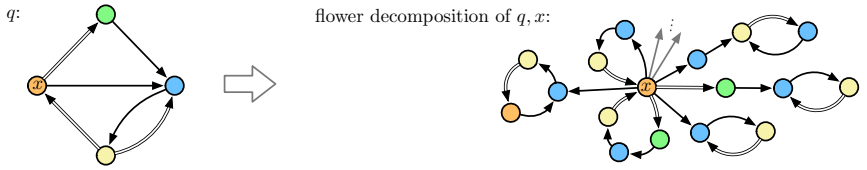


Fig. 8. Example of flower decomposition with root x (not all petals are depicted). Variables with the same color indicate copies of the same variable of p. Different styles of arrows correspond to different relations.

We establish a useful technical lemma concerning the notion of flower decomposition.

Lemma 8.6. *Consider a BCQ $p = p|_\Sigma$ that consists of one strongly connected component, let x be a node of p, and let p' be the flower decomposition of (p, x). Then:*

1. $p' \in \mathsf{GHW}_2$.
2. $p' \Rightarrow^*_\Sigma p$ *with a provenance homomorphism that maps the variable x of p' to the variable x of p, and every variable $y \neq x$ of p' to the variable y' of p, where y is the copy of y'.*

Proof. Point 1 is trivial from the simple shape of the query: a bunch of lassos with one common vertex (in particular, a pseudotree).

For point 2, let $C_1, \ldots, C_n$ be the set of all simple cycles of p, such that C_1 contains x, and for every $i > 1$ we have that C_i contains a variable from $C_1 \cup \cdots \cup C_{i-1}$. We prove by induction that, for every $i \in [n]$, $p'_i \Rightarrow^*_\Sigma p_i$, where p_i is the subquery $C_1 \cup \cdots \cup C_i$ of p, and p'_i is the flower decomposition of (p_i, x). The base case is straightforward. For the inductive case $i > 1$, let C_i be a simple cycle of p, and let $\{x_1, \cdots, x_m\}$ be the (non-empty) set of variables of C_i that appear also in $C_1 \cup \cdots \cup C_{i-1}$. For every $j \in [m]$, let π_j be a simple path of p_{i-1} from x to x_j. Observe that since, by inductive hypothesis, $p'_{i-1} \Rightarrow^*_\Sigma p_{i-1}$, we have $p'_{i-1} \cup \bigcup_{j\in[m]} P_{x,\pi_j,C_i} \Rightarrow^*_\Sigma p_{i-1} \cup \bigcup_{j\in[m]} P_{x,\pi_j,C_i}$. It is not hard to see that $p_{i-1} \cup \bigcup_{j\in[m]} P_{x,\pi_j,C_i} \Rightarrow^*_\Sigma p_i$, since every π_j assures that the copy of x_j in P_{x,π_j,C_i} is "glued" to the variable x_j of p_{i-1}. From this it follows that $p'_i \Rightarrow^*_\Sigma p_i$, as needed. □

Consider two BCQs p, p' such that $h : p' \to p$. The *expansion (w.r.t. h and Σ)* of p' is the BCQ $\hat{p}'$ obtained by attaching the flower decomposition (w.r.t. Σ) of $(p, h(x))$ to every variable x of p'. Note

that if p' has width 1, then $\hat{p}'$ has width at most 2, and if p' has width $k > 1$, then $\hat{p}'$ has width k. The homomorphism h can be canonically extended to a homomorphism $\hat{h} : \hat{p}' \to p$, as shown in Figure 9. We finally say that $(\hat{p}', \hat{h})$ is the Σ-*expansion* of (p', h).

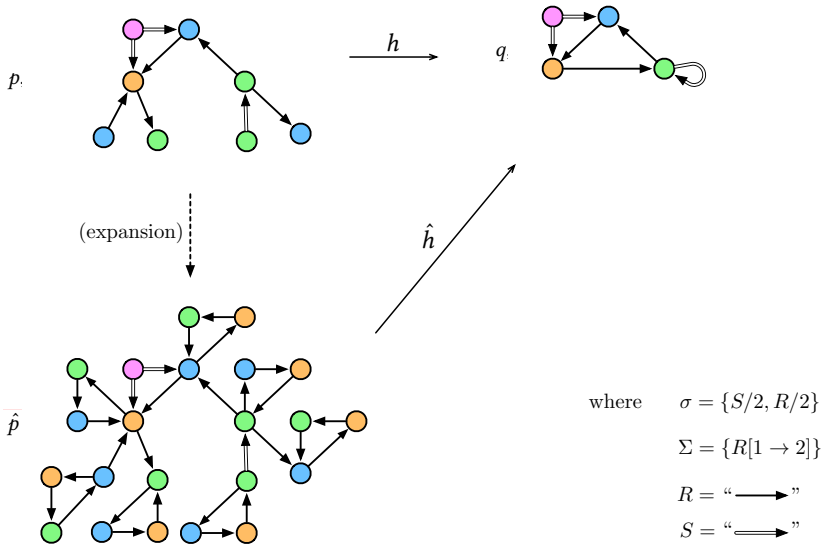


Fig. 9. Example of an expansion from p to $\hat{p}$ for a given $h : p \to q$, and the canonical extension $\hat{h} : \hat{p} \to q$ of h. Note that, for clarity, not all the petals of each flower decomposition are depicted in $\hat{p}$. Different styles of arrows correspond to different relations.

8.3 Definition of $\mathsf{M}_{q,\Sigma}$

For a BCQ p, we define the partial order relation $\preceq_p$ among the variables of p, where $x \preceq_p y$ iff there is a (possibly empty) directed path from x to y in $p|_\Sigma$. Note that $x \preceq_p x$ for every variable x in $p|_\Sigma$. If $x \preceq_p y$ and $y \preceq_p x$, then we write $x \equiv_p y$, which means that x and y belong to the same SCC of $p|_\Sigma$. If $x \preceq_p y$ and $y \not\preceq_p x$, then we write $x \prec_p y$. A *regular cycling path* of p (w.r.t. Σ) is either

(i) an empty path, starting and ending at the same node; or

(ii) a path of the form $(x \overset{R}{\leftarrow} y)\ \pi\ (y \overset{R}{\rightarrow} x)$, where $y \overset{R}{\rightarrow} x$ is an edge in $p|_\Sigma$, π is a regular cycling path from y to y, and $y \prec_p x$; or

(iii) a path of the form $(x_1 \overset{R_1}{\leftrightsquigarrow}_1 y_1)\ \pi_1 \cdots \pi_{n-1}\ (x_n \overset{R_n}{\leftrightsquigarrow}_n y_n)$, where,
- $x_1 = y_n$ and $x_i \equiv_p y_j$ for every $i, j \in [n]$,
- $x_i \overset{R_i}{\leftrightsquigarrow}_i y_i$ is either an edge $x_i \overset{R_i}{\longrightarrow} y_i$ or $x_i \overset{R_i}{\longleftarrow} y_i$ in p_Σ for every $i \in [n]$, and
- π_i is a regular cycling path from y_i to x_{i+1} for every $i \in [n-1]$; or

(iv) a path of the form $\pi\ \pi'$, where for some variable x, both π and π' are regular cycling paths of p from x to x.

Note that any regular cycling path induces a cycle in the underlying undirected graph (hence the name *cycling*). Furthermore, these paths form a regular language over the alphabet $\{x \overset{R}{\rightarrow} y, y \overset{R}{\leftarrow} x \mid x \overset{R}{\rightarrow} y \text{ in } p\}$ (hence the name *regular*). This is because: rules of type (iii) do not need any nesting (even if it is allowed) and they describe a regular language if seen on their own; rules of type (iv)

are actually regular, and they can be translated as a Kleene star; and rules of type (ii) can only be nested up to a linear depth, due to the requirement $y \prec_p x$.

Given an equivalence relation $\sim$ on the variables of p, let $p/\sim$ be the quotient query obtained by identifying the variables in the same equivalence class. We write $[x]_\sim$ to denote the variable of $p/\sim$ resulting from the identification of all the variables in the equivalence class of x, as shown in Figure 10. Given two BCQs p, p', a homomorphism $h : p' \to p$, and a directed path π between

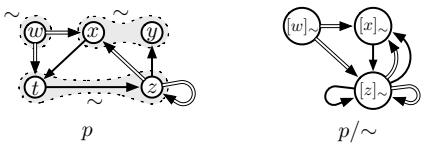


Fig. 10. The quotient $p/\sim$ of a BCQ p.

two variables x, y of p', we say that π is an *h-regular cycling path* if $h(\pi)$, that is, the path obtained after applying h to every node of π, is a regular cycling path of p. We write $x \sim_h y$ to denote the existence of an h-regular cycling path from x to y in p'. Observe that $\sim_h$ is an equivalence relation on the variables of p'. The interest of these paths will become evident below; the intuition is that any pair of $\sim_h$-related variables of p' are identified as *one* variable in $\mathsf{chase}(p', \Sigma)$, provided p' is an expansion.

We can now define $\mathsf{M}_{q,\Sigma}$ as the set of BCQs

$$\mathsf{M}_{q,\Sigma} = \{p \in \mathrm{BCQ} \mid \text{there exists } h : p \to \mathsf{core}(\mathsf{chase}(q,\Sigma)) \text{ such that } \mathsf{core}(\mathsf{chase}(q,\Sigma)) \to \hat{p}/\sim_{\hat{h}},$$
$$\text{where } (\hat{p}, \hat{h}) \text{ is the } \Sigma\text{-expansion of } (p, h)\}$$

We proceed to show that $\mathsf{M}_{q,\Sigma}$ is indeed MSO-definable.

Lemma 8.7. *There exists an MSO sentence $\xi_{q,\Sigma}$ defining $\mathsf{M}_{q,\Sigma}$. Further, $\xi_{q,\Sigma}$ is of polynomial size, has a fixed number of quantifier alternations, and it can be computed in exponential time.*

Proof. Recall that $\mathsf{chase}(q,\Sigma)$ can be computed in polynomial time, while the core of a CQ can be computed in exponential time. Thus, $\mathsf{core}(\mathsf{chase}(q,\Sigma))$ is of polynomial size, while it can be computed in exponential time. Hence, we can assume that $q \cong \mathsf{core}(\mathsf{chase}(q,\Sigma))$. With an existential MSO formula we can guess a function $h : p \to q$ and verify that it is a homomorphism; this is standard, we can use one monadic variable X_y containing $h^{-1}(y)$ for each variable y of q.

Using this guessing $\{X_y\}_y$, one can define the relation $\sim_{\hat{h}}$ in MSO. Remember that $\hat{p}$ has one distinct copy of the flower decomposition of $(q, h(x))$ attached to each node x, and remember also that $\sim_{\hat{h}}$ is an equivalence relation on the variables of $\hat{p}$, and thus, we have to encode the new elements of $\hat{p}$. It is possible to define an MSO formula $\psi_{y,y'}(x, x')$ which holds true at a pair of variables x, x' of p iff: y is a node of the flower decomposition of $(q, h(x))$; y' is a node of the flower decomposition of $(q, h(x'))$; and the y node of the copy of the flower decomposition of $(q, h(x))$ attached to x in $\hat{p}$ is $\sim_{\hat{h}}$-related to the the y' node of the copy of the flower decomposition of $(q, h(x'))$ attached to x'. Indeed, this boils down to checking: (i) $h(y) = h(y')$, and (ii) there is an h-regular cycling path between x and x'. Observe that this last condition asks whether there is a path between x and x' belonging to some regular language (where both the relation names and the X_y-labels of the nodes are part of the path alphabet).

Finally, with such a family of formulas $\{\psi_{y,y'}\}_{y,y'}$ at hand, we can verify the existence of $g : q \to \hat{p}/\sim_{\hat{h}}$: We guess (using an existentially quantified first-order variable) one element z_y from p for each variable y of q, and for each such guessing z_y we also guess a variable $f(z_y)$ of the flower

decomposition of $(q, h(z_y))$. This is to take into account the case where $g(y)$ is in $\hat{p}$ but not in p. Finally, we check that, for every $y \xrightarrow{R} y'$ in q, there exist z'_y, $z'_{y'}$ and variables $f(z'_y)$, $f(z'_{y'})$ of the flower decompositions of (q, y) and (q, y') such that $\psi_{f(z_y),f(z'_y)}(z_y, z'_y) \wedge \psi_{f(z_{y'}),f(z'_{y'})}(z_{y'}, z'_{y'})$ and

- $z'_y = z'_{y'}$ and $f(z'_y) \xrightarrow{R} f(z'_{y'})$ in the flower decomposition of $(q, h(z'_y))$; or
- $z'_y \neq z'_{y'}$, and both $f(z'_y)$ and $f(z'_{y'})$ are the root of the respective flower decompositions, and $z'_y \xrightarrow{R} z'_{y'}$ is in p. □

8.4 Suitability of $\mathsf{M}_{q,\Sigma}$

In this section we prove the following two statements:

(a) $\mathsf{W}_{q,\Sigma} \subseteq \mathsf{M}_{q,\Sigma}$ (Lemma 8.9), and
(b) for every $p \in \mathsf{M}_{q,\Sigma} \cap \mathsf{GHW}_k$ such that $h : p \to \mathsf{core}(\mathsf{chase}(q, \Sigma))$, it holds that $\hat{p} \in \mathsf{W}_{q,\Sigma} \cap \mathsf{GHW}_k$, where $(\hat{p}, \hat{h})$ is the Σ-expansion of (p, h) (Lemma 8.10).

Both statements follow from the following lemma, which is the main focus of this section:

Lemma 8.8. *Consider a BCQ p. If $h : p \to \mathsf{core}(\mathsf{chase}(q, \Sigma))$, then $\mathsf{chase}(\hat{p}, \Sigma) \cong \hat{p}/{\sim_{\hat{h}}}$, where $(\hat{p}, \hat{h})$ is the Σ-expansion of (p, h).*

Let us first show that indeed the above statements (a) and (b) are consequences of Lemma 8.8.

Lemma 8.9. *It holds that $\mathsf{W}_{q,\Sigma} \subseteq \mathsf{M}_{q,\Sigma}$.*

Proof. Consider a BCQ $p \in \mathsf{W}_{q,\Sigma}$. We prove the following chain of homomorphic mappings

$$p \to \mathsf{chase}(p, \Sigma) \to \mathsf{chase}(q, \Sigma) \to \mathsf{core}(\mathsf{chase}(q, \Sigma)) \to \mathsf{core}(\mathsf{chase}(p, \Sigma)) \to \mathsf{chase}(\hat{p}, \Sigma) \to \hat{p}/{\sim_{\hat{h}}},$$

where h is the homomorphic mapping that embeds p into $\mathsf{chase}(q, \Sigma)$. The first mapping $p \to \mathsf{chase}(p, \Sigma)$ is the provenance homomorphism which holds by the properties of the chase, the second homomorphism $\mathsf{chase}(p, \Sigma) \to \mathsf{core}(\mathsf{chase}(q, \Sigma))$ holds by definition of $\mathsf{W}_{q,\Sigma}$. $\mathsf{chase}(q, \Sigma) \to \mathsf{core}(\mathsf{chase}(q, \Sigma))$ holds by definition of core. By definition of $\mathsf{W}_{q,\Sigma}$, $\mathsf{chase}(p, \Sigma)$ and $\mathsf{chase}(q, \Sigma)$ are hom-equivalent (*i.e.*, $\mathsf{chase}(p, \Sigma) \to \mathsf{chase}(q, \Sigma)$ and $\mathsf{chase}(q, \Sigma) \to \mathsf{chase}(p, \Sigma)$), and thus the fourth homomorphism holds. Now let us define $h : p \to \mathsf{core}(\mathsf{chase}(q, \Sigma))$ as the composition of the first three homomorphisms. Since $\mathsf{core}(\mathsf{chase}(q, \Sigma)) \to \mathsf{chase}(p, \Sigma)$ (by definition of core) and $\mathsf{chase}(p, \Sigma) \to \mathsf{chase}(\hat{p}, \Sigma)$ (since p is a subquery of $\hat{p}$), we have that $\mathsf{core}(\mathsf{chase}(q, \Sigma)) \to \mathsf{chase}(\hat{p}, \Sigma)$ by composition of homomorphisms. Finally, by Lemma 8.8, $\mathsf{chase}(\hat{p}, \Sigma) \cong \hat{p}/{\sim_{\hat{h}}}$, where $(\hat{p}, \hat{h})$ is the Σ-expansion of (p, h). Thus, $\mathsf{core}(\mathsf{chase}(q, \Sigma)) \to \hat{p}/{\sim_{\hat{h}}}$, which in turn implies that $p \in \mathsf{M}_{q,\Sigma}$. □

Lemma 8.10. *For every BCQ $p \in \mathsf{M}_{q,\Sigma} \cap \mathsf{GHW}_k$ with $h : p \to \mathsf{core}(\mathsf{chase}(q, \Sigma))$, it holds that $\hat{p} \in \mathsf{W}_{q,\Sigma} \cap \mathsf{GHW}_k$, where $(\hat{p}, \hat{h})$ is the Σ-expansion of (p, h).*

Proof. Let $q' = \mathsf{core}(\mathsf{chase}(q, \Sigma))$. Since $p \in \mathsf{M}_{q,\Sigma}$, there is $h : p \to q'$ such that $q' \to \hat{p}/{\sim_{\hat{h}}}$, where $\hat{p}, \hat{h}$ is the expansion of p, h. Remember that the width of $\hat{p}$ is bounded by k as a consequence of Lemma 8.6-1. By Lemma 8.8 we have $\hat{p}/{\sim_{\hat{h}}} \cong \mathsf{chase}(\hat{p}, \Sigma)$ and thus

$$q' \to \mathsf{chase}(\hat{p}, \Sigma). \tag{1}$$

Since we also have $\hat{h} : \hat{p} \to q'$, and since $\sim_{\hat{h}}$ relates only nodes with equal $\hat{h}$-image, we have $\hat{p}/{\sim_{\hat{h}}} \to q'$ and thus by Lemma 8.8

$$\mathsf{chase}(\hat{p}, \Sigma) \to q'. \tag{2}$$

From (1) and (2) we obtain $\mathsf{core}(\mathsf{chase}(\hat{p}, \Sigma)) \cong q'$, which shows $\hat{p} \in \mathsf{W}_{q,\Sigma}$. □

Towards proving Lemma 8.8, we define some notation related to the stepwise computation of the chase. Consider a BCQ p. For any partition S of the nodes of p, let $\approx_S$ be the equivalence relation on the nodes of p associated to S, and, for every node x of p, let $[x]_S$ be the set of S containing x. (Think of $\approx_S$ as the equivalence classes of all variables that have been "glued together" after some steps of the chase.) Let S_0^p be the finest partition, that is, the set of all singletons $\{x\}$ such that x is a variable of p. We now define a binary relation $\overset{p}{\Rightarrow}_\Sigma$ over the set of all partitions of the variables of p in the following way: $S \overset{p}{\Rightarrow}_\Sigma S'$ if, and only if, there are $R[1 \to 2] \in \Sigma$ and $x \xrightarrow{R} y$, $x' \xrightarrow{R} z$ in p such that $x \approx_S x'$, $y \not\approx_S z$ and $S' = S \setminus \{[y]_S, [z]_S\} \cup \{[y]_S \cup [z]_S\}$. The intuition is that $S \overset{p}{\Rightarrow}_\Sigma S'$ if glueing together variables according to S allows the chase to glue, in one step, $[y]_S$ and $[z]_S$. The next lemma follows directly from the definitions of $\Rightarrow_\Sigma$ and $\overset{p}{\Rightarrow}_\Sigma$.

Lemma 8.11. *For every $n \geq 0$, and BCQs p, p', the following are equivalent:*

(1) $p \Rightarrow_\Sigma^n p'$.

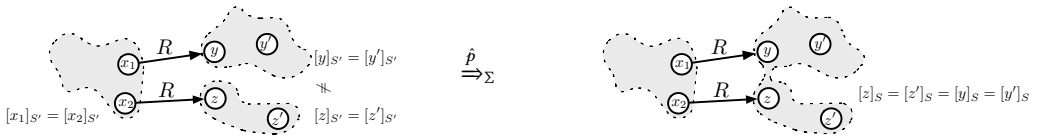
(2) $p' \cong p/\approx_S$ for some S such that $S_0^p \overset{p}{\Rightarrow}_\Sigma^n S$.

We now establish the last technical lemma of this section that, together with Lemma 8.11, immediately shows Lemma 8.8.

Lemma 8.12. *Consider a BCQ p, with $h : p \to \mathsf{core}(\mathsf{chase}(q, \Sigma))$, and let $(\hat{p}, \hat{h})$ be the Σ- expansion of (p, h). The following are equivalent:*

(1) $x \approx_S x'$ for some S such that $S_0^{\hat{p}} \overset{\hat{p}}{\Rightarrow}_\Sigma^* S$.
(2) $x \sim_{\hat{h}} x'$.

Proof. We first show that (1) implies (2). We proceed by induction on the number n of $\overset{\hat{p}}{\Rightarrow}_\Sigma$ steps. The base case $n = 0$ is trivial since $x \sim_{\hat{h}} x$ for every x. Suppose now that $z' \approx_S y'$ for $S_0^{\hat{p}} \overset{\hat{p}}{\Rightarrow}_\Sigma^{n+1} S$. We need to show that $z' \sim_{\hat{h}} y'$. Without loss of generality, assume that $S_0^{\hat{p}} \overset{\hat{p}}{\Rightarrow}_\Sigma^n S' \overset{\hat{p}}{\Rightarrow}_\Sigma S$, where, for some key $R[1 \to 2] \in \Sigma$ and variables x_1, x_2, y, z, we have $x_1 \xrightarrow{R} y$, $x_2 \xrightarrow{R} z$, $x_1 \approx_{S'} x_2$, $y \approx_{S'} y'$, $z \approx_{S'} z'$ and $y \not\approx_{S'} z$. That is, we are in this situation:



Note that this implies that $\hat{h}(y) = \hat{h}(z)$ since there is $\hat{h}(x_1) \xrightarrow{R} \hat{h}(y)$ and $\hat{h}(x_2) = \hat{h}(x_1) \xrightarrow{R} \hat{h}(z)$ in $\mathsf{core}(\mathsf{chase}(q, \Sigma))$, while $\mathsf{core}(\mathsf{chase}(q, \Sigma))$ satisfies Σ. By induction hypothesis, there is an $\hat{h}$-regular cycling path between any two variables in $[x_1]_{S'}$, any two variables in $[y]_{S'}$ and any two variables in $[z]_{S'}$ (and, in particular, they have the same $\hat{h}$-image). Consider the following $\hat{h}$-regular cycling paths of $\hat{p}$: π_{x_2,x_1} between x_2 and x_1, $\pi_{y,y'}$ between y and y', and $\pi_{z',z}$ between z' and z, which they exist by induction hypothesis. Now, observe that the path $\pi = \pi_{z',z} \cdot (z \xleftarrow{R} x_2) \cdot \pi_{x_2,x_1} \cdot (x_1 \xrightarrow{R} y) \cdot \pi_{y,y'}$ between z' and y' is an $\hat{h}$-regular cycling path in $\hat{p}$. Indeed, there are two cases to consider:

1. $\hat{h}(x_1) = \hat{h}(x_2) \equiv_{\mathsf{core}(\mathsf{chase}(q,\Sigma))} \hat{h}(z) = \hat{h}(z') = \hat{h}(y) = \hat{h}(y')$,
2. $\hat{h}(x_1) = \hat{h}(x_2) \prec_{\mathsf{core}(\mathsf{chase}(q,\Sigma))} \hat{h}(z) = \hat{h}(z') = \hat{h}(y) = \hat{h}(y')$.

These cases are covered, respectively, by items (ii) and (iii) in the definition of regular cycling path, and therefore, π is a valid $\hat{h}$-regular cycling path. Thus, $y' \sim_{\hat{h}} z'$.

We now show that (2) implies (1). For brevity, let $q' = \mathsf{core}(\mathsf{chase}(q, \Sigma))$. Since $\overset{\hat{p}}{\Rightarrow}_\Sigma$ is monotone in the sense that it coarsens partitions, it is sufficient to prove the statement for the coarsest partition S such that $S_0^{\hat{p}} \overset{\hat{p}}{\Rightarrow}_\Sigma^* S$ (*i.e.*, the partition S so that $\mathsf{chase}(\hat{p}, \Sigma) \cong \hat{p}/{\approx_S}$ in view of Lemma 8.11). We proceed by structural induction on the $\hat{h}$-regular cycling path π of p between the variables x, x'. The base case (i) is the empty path ε where $x = x'$, which trivially verifies $x \approx_S x'$. The case (iv) of concatenation $\hat{h}(\pi) = \pi_1 \cdot \pi_2$ goes by induction plus transitivity of $\approx_S$. The case (ii) where $\hat{h}(\pi) = (y \overset{R}{\leftarrow} y')\pi(y' \overset{R}{\rightarrow} y)$ goes by induction hypothesis and the definition of $\overset{\hat{p}}{\Rightarrow}_\Sigma$. The only interesting case is (iii), where $\hat{h}(\pi)$ is of the form $(x_1 \overset{R_1}{\rightsquigarrow}_1 y_1)\ \pi_1 \cdots \pi_{n-1}\ (x_n \overset{R_n}{\rightsquigarrow}_n y_n)$, where, $x_1 = y_n$, $x_i \equiv_{q'} y_j$ for every $i, j \in [n]$, $x_i \overset{R_i}{\rightsquigarrow}_i y_i$ is either an edge $x_i \overset{R_i}{\rightarrow} y_i$ or $x_i \overset{R_i}{\leftarrow} y_i$ in $q'|_\Sigma$ for every $i \in [n]$, and π_i is a regular cycling path from y_i to x_{i+1} for every $i \in [n-1]$. Let $\pi = (x_1' \overset{R_1}{\rightsquigarrow}_1 y_1')\ \pi_1' \cdots \pi_{n-1}'\ (x_n' \overset{R_n}{\rightsquigarrow}_n y_n')$ where $\hat{h}(x_i') = x_i$, $\hat{h}(y_i') = y_i$ and $\hat{h}(\pi_i') = \pi_i$. By induction hypothesis, $y_i' \approx_S x_{i+1}'$ for every i. For a variable z of q', let $\mathsf{scc}(q', z)$ be the subquery of $q'|_\Sigma$ consisting of the strongly connected component of $q'|_\Sigma$ containing z. In view of Lemma 8.11 and Lemma 8.6-2, note that for every node x of π, $\hat{p}/{\approx_S}$ has attached a homomorphic copy of $\mathsf{scc}(q', \hat{h}(x))$ —that is, there is a homomorphism $\mathsf{scc}(q', \hat{h}(x)) \to \hat{p}/{\approx_S}$ mapping $\hat{h}(x)$ to $[x]_{\approx_S}$. Therefore, for every $x \overset{R}{\rightarrow} y$ of $\hat{p}$, where $\hat{h}(x), \hat{h}(y)$ are in the same strongly connected component of $q'|_\Sigma$ and $R[1 \to 2] \in \Sigma$, we have that the homomorphic copies of $\mathsf{scc}(q', \hat{h}(x))$ and $\mathsf{scc}(q', \hat{h}(y))$ attached to x and y respectively are actually "merged" into one in $\hat{p}/{\approx_S}$ (since S is the coarsest partition). In particular, $y \approx_S y'$ where y' is the element corresponding to $\hat{h}(y)$ in the homomorphic copy of $\mathsf{scc}(q', \hat{h}(x))$. This means that, since $\hat{h}(x_1) = \hat{h}(x_n)$, since $y_i' \approx_S x_{i+1}'$ for every i, and since every x_i, y_i is in the selfsame strongly connected component of $q'|_\Sigma$, we have that $x_1 \approx_S x_n$, as needed. □

Lemma 8.8 follows as a direct corollary of Lemmas 8.11 and 8.12.

8.5 Finalizing the Decidability Proof

We finally refresh here the arguments scattered throughout the previous sections to prove decidability of $\mathsf{SemGHW}_k(\mathbb{K})$, for $k > 1$. Let us first recall and properly show the crucial Proposition 8.3.

Proposition 8.3 (restatement). *Consider a BCQ q and a set Σ of keys. There is an MSO-definable set $\mathsf{M}_{q,\Sigma}$ of BCQs such that the following are equivalent:*

(1) $\mathsf{W}_{q,\Sigma} \cap \mathsf{GHW}_k \neq \varnothing$;
(2) $\mathsf{M}_{q,\Sigma} \cap \mathsf{GHW}_k \neq \varnothing$.

Furthermore, $\mathsf{M}_{q,\Sigma}$ is definable via an MSO sentence $\xi_{q,\Sigma}$ that can be computed in single exponential time from q and Σ, and has a fixed number of quantifier alternations.

Proof. The fact that (1) implies (2) is an immediate consequence of the fact that $\mathsf{W}_{q,\Sigma} \subseteq \mathsf{M}_{q,\Sigma}$ by Lemma 8.9. For showing that (2) implies (1), consider an arbitrary BCQ $p \in \mathsf{M}_{q,\Sigma} \cap \mathsf{GHW}_k$, where $k > 1$. Assuming that $h : p \to \mathsf{core}(\mathsf{chase}(q, \Sigma))$, Lemma 8.10 implies that $\hat{p} \in \mathsf{W}_{q,\Sigma} \cap \mathsf{GHW}_k$, where $(\hat{p}, \hat{h})$ is the Σ-expansion of (p, h). Therefore, $\mathsf{W}_{q,\Sigma} \cap \mathsf{GHW}_k \neq \varnothing$, and the claim follows. □

We are now ready to provide a proof for the main decidability result of this section.

Theorem 8.1 (restatement). *For binary schemas, $\mathsf{SemGHW}_k(\mathbb{K})$ is decidable in elementary time.*

Proof. Consider a BCQ q, and a set Σ of keys. By Proposition 8.2, the problem of deciding whether q is semantically in GHW_k under Σ is equivalent to checking whether $\mathsf{W}_{q,\Sigma} \cap \mathsf{GHW}_k \neq \varnothing$, which in turn, by Proposition 8.3, is equivalent to checking whether $\mathsf{M}_{q,\Sigma} \cap \mathsf{GHW}_k \neq \varnothing$. The latter, again by Proposition 8.3, is equivalent to deciding whether the MSO sentence $\xi_{q,\Sigma}$ has a model in GHW_k, which is feasible in ℓ-ExpTime, assuming that ℓ is the number of quantifier alternations of $\xi_{q,\Sigma}$ [40]. Since $\xi_{q,\Sigma}$ can be computed in single exponential time, and has a fixed number of quantifies alternations, the claim follows. □

9 QUERY EVALUATION

As it has been noted in different scenarios in the absence of constraints, the property of being semantically in GHW_k, for $k \geq 1$, has a positive impact on query evaluation [7, 8, 17, 19]. The crucial question that comes up is whether such a good behavior extends to the notion of being semantically in GHW_k, for $k \geq 1$, in the presence of constraints. Interestingly, the answer to the above question is affirmative. We first discuss that, for each class $\mathbb{C}$ of constraints considered in the previous sections, the fact that we have a decision procedure for solving $\mathsf{SemGHW}_k(\mathbb{C})$ allows us to show that the problem of evaluating a CQ q that is semantically in GHW_k under a set $\Sigma \in \mathbb{C}$ is fixed-parameter tractable. The crucial fact here is that our techniques for solving $\mathsf{SemGHW}_k(\mathbb{C})$ yields an equivalent CQ $q' \in \mathsf{GHW}_k$ in case q is semantically in GHW_k under Σ. We then ask ourselves whether a pure tractability result can be obtained. We show that this is the case for the class of guarded sets of TGDs, and the class of (arbitrary) functional dependencies. Crucially, for those classes of constraints, computing an equivalent CQ that falls in GHW_k is not needed for query evaluation purposes.

Before we proceed further, let us introduce the query evaluation problem. In the rest of this section, we fix $k \geq 1$. Let $\mathbb{C}$ be a class of sets of TGDs or EGDs. We define:

PROBLEM :	$\mathsf{EvalSemGHW}_k(\mathbb{C})$
INPUT :	A set $\Sigma \in \mathbb{C}$, a CQ $q(\bar{x})$ that is semantically in GHW_k under Σ, a database D such that $D \models \Sigma$, and a tuple $\bar{t} \in \mathsf{dom}(D)^{\lvert\bar{x}\rvert}$.
QUESTION :	Is $\bar{t} \in q(D)$?

9.1 Fixed-parameter Tractable Evaluation

We show that $\mathsf{EvalSemGHW}_k(\mathbb{C})$, for $\mathbb{C} \in \{\mathbb{G}, \mathbb{NR}, \mathbb{S}\}$, as well as $\mathsf{EvalSemGHW}_k(\mathbb{K})$ assuming unary and binary predicates, is *fixed-parameter tractable* (fpt) with the parameter being $(||q|| + ||\Sigma||)$; as usual, $||q||$ and $||\Sigma||$ represent the size of reasonable encodings of q and Σ, respectively. Recall that fpt means that the problem can be solved in time

$$O\left(||D||^c \cdot f(||q|| + ||\Sigma||)\right)$$

where $c \geq 1$ and $f : \mathbb{N} \to \mathbb{N}$ is a computable function. Then:

Theorem 9.1. $\mathsf{EvalSemGHW}_k(\mathbb{C})$, *where* $\mathbb{C} \in \{\mathbb{G}, \mathbb{NR}, \mathbb{S}\}$, *can be solved in time*

$$O\left(||D||^{k+1} \cdot 2^{2^{O(||q||+||\Sigma||)}}\right),$$

and thus is fixed-parameter tractable.

Proof. Consider an arbitrary instance of $\mathsf{EvalSemGHW}_k(\mathbb{C})$, i.e, a set $\Sigma \in \mathbb{C}$, a CQ $q(\bar{x})$ that is semantically in GHW_k under Σ, a database D such that $D \models \Sigma$, and a tuple $\bar{t} \in \mathsf{dom}(D)^{\lvert\bar{t}\rvert}$. We can decide whether $\bar{t} \in q(D)$ as follows:

(1) Compute a CQ q' such that $q' \in \mathsf{GHW}_k$ and $q \equiv_\Sigma q'$.

(2) Accept if $\bar{t} \in q'(D)$; otherwise, reject.

The decision procedures underlying Theorems 5.6, 6.8 and 6.10 allow us to compute the CQ q' in double-exponential time in $(||q|| + ||\Sigma||)$. The size of q' is at most exponential in $(||q|| + ||\Sigma||)$; for $\mathbb{G}$ is polynomial, while for $\mathbb{NR}$ and $\mathbb{S}$ is exponential. Proposition 2.4 implies that the cost of computing q' and checking if $\bar{t} \in q'(D)$ is:

$$\underbrace{O\left(2^{2^{O(||q||+||\Sigma||)}}\right)}_{\text{Step 1}} + \underbrace{O\left(||D||^{k+1} \cdot 2^{O(||q||+||\Sigma||)}\right)}_{\text{Step 2}}.$$

Clearly, the running time of the algorithm is dominated by

$$O\left(||D||^{k+1} \cdot 2^{2^{O(||q||+||\Sigma||)}}\right)$$

and the claim follows. □

Analogously, by exploiting the decision procedure underlying Theorem 8.1, we can show the same for keys over binary signatures:

Theorem 9.2. *Assuming binary signatures,* $\mathsf{EvalSemGHW}_k(\mathbb{K})$ *can be solved in time*

$$O\left(||D||^{k+1} \cdot f(O(||q|| + ||\Sigma||))\right),$$

where f is an elementary function, and thus is fixed-parameter tractable.

Indeed, Theorems 9.1 and 9.2 provide an improvement over general CQ evaluation for which no fpt algorithm is believed to exist [37]. It is worth remarking, nonetheless, that $\mathsf{EvalSemGHW}_k(\mathbb{C})$ corresponds to a *promise version* of the evaluation problem, where the property that defines the class is at least NP-hard for each $k \geq 1$, and class $\mathbb{C}$ in question.

9.2 Tractable Evaluation

We proceed to establish a pure tractability result if we focus on sets of guarded TGDs and sets of (arbitrary) FDs. The class of FDs is denoted $\mathbb{FD}$.

Theorem 9.3. $\mathsf{EvalSemGHW}_k(\mathbb{C})$, *where* $\mathbb{C} \in \{\mathbb{G}, \mathbb{FD}\}$, *is in* PTime.

The case of FDs is very interesting since, although $\mathsf{EvalSemGHW}_k(\mathbb{FD})$ is tractable, the decidability of checking whether a CQ is semantically in GHW_k under a set of FDs is a challenging open problem – recall that we have decidability only for keys assuming schemas with unary and binary predicates only. The rest of this section is devoted to establishing Theorem 9.3.

Recall that computing a CQ q' in GHW_k that is equivalent to the given CQ q under the given set Σ of constraints might take double-exponential time. Thus, in order to achieve tractability of evaluation, we should avoid the explicit computation of q'. Indeed, computing such a CQ q' is not needed at the moment of evaluating CQs that are semantically in GHW_k under a set Σ of constraints that falls in $\mathbb{G}$ or $\mathbb{FD}$. The high-level idea behind the proof of Theorem 9.3, which avoids the explicit computation of q', can be described as follows. Evaluating a CQ q that is semantically in GHW_k in the absence of constraints over a database D, amounts to checking the existence of a winning strategy for the duplicator in a particular version of the pebble game, known as the *existential k-cover game*, on q and D [17]. The existence of such a winning strategy, denoted $q \rightarrow_k D$, can be checked in polynomial time. Now, when q is semantically in GHW_k under a set Σ of TGDs or EGDs, and in addition $D \models \Sigma$, evaluating q over D amounts to checking whether $\mathsf{chase}(q, \Sigma) \rightarrow_k D$ (see Proposition 9.5 below). Having this technical result in place, Theorem 9.3 is obtained as follows:

- If $\Sigma \in \mathbb{FD}$, then $\mathsf{chase}(q, \Sigma)$ can be computed in polynomial time. This implies that checking whether $\mathsf{chase}(q, \Sigma) \rightarrow_k D$ is feasible in polynomial time, and thus, $\mathsf{EvalSemGHW}_k(\mathbb{FD})$ is in PTime. In fact, this shows that $\mathsf{EvalSemGHW}_k(\mathbb{C})$ is in PTime for any class $\mathbb{C}$ of sets of TGDs or EGDs for which the chase can be computed in polynomial time.
- If $\Sigma \in \mathbb{G}$, then, although $\mathsf{chase}(q, \Sigma)$ may be infinite, we can prove that:

$$\mathsf{chase}(q, \Sigma) \rightarrow_k D \quad \Longleftrightarrow \quad q \rightarrow_k D.$$

  That is, the problem boils down to checking $q \rightarrow_k D$, i.e., evaluating q as if it was semantically in GHW_k without constraints, and thus, $\mathsf{EvalSemGHW}_k(\mathbb{G})$ is in PTime.

We proceed to formalize the above high-level description.

Existential k-cover game. Let us first recall the *existential k-cover game* from [17], and show our technical result (Proposition 9.5), which establishes the desired connection between query evaluation and the existence of a winning strategy for the duplicator. The existential k-cover game is played by *spoiler* and *duplicator* on pairs $(I, \bar{t})$ and $(I', \bar{t}')$, where I and I' are instances and $\bar{t}$ and $\bar{t}'$ are two equally long tuples of terms of $\mathsf{dom}(I)$ and $\mathsf{dom}(I')$, respectively. The game proceeds in rounds. In each round, the spoiler places (resp., removes) a pebble on (resp., from) a term occurring in I, while the duplicator responds by placing (resp., removing) its corresponding pebble on (resp., from) a term in I'. The number of pebbles is not bounded, but the spoiler is constrained as follows: At any round r of the game, if $c_1, \ldots, c_\ell$, where $\ell \leq r$, are the terms marked by the spoiler's pebbles in I, then there must be at most k atoms in I that contain all those terms (this is why the game is called k-cover, as pebbled terms are *covered* by such k atoms). Duplicator wins if she can indefinitely continue playing the game in such way that after each round, if $c_1, \ldots, c_\ell$ are the terms marked by the spoiler's pebbles in I and $d_1, \ldots, d_\ell$ are the terms marked by the corresponding pebbles of the duplicator in I', then, assuming that $\bar{t} = (t_1, \ldots, t_n)$ and $\bar{t}' = (t'_1, \ldots, t'_n)$

$$h \;=\; \{c_i \mapsto d_i\}_{1 \leq i \leq \ell} \;\cup\; \{t_i \mapsto t'_i\}_{1 \leq i \leq n}$$

is a *partial homomorphism* from I to I'. In other words, for every atom $R(e_1, \ldots, e_m) \in I$, with $\{e_1, \ldots, e_m\} \subseteq \{c_1, \ldots, c_\ell, t_1, \ldots, t_n\}$, $R(h(e_1, \ldots, e_m)) \in I'$. The fact that duplicator wins the existential k-cover game on $(I, \bar{t})$ and $(I', \bar{t}')$ is denoted $(I, \bar{t}) \rightarrow_k (I', \bar{t}')$.

The following proposition, which can be established by exploiting results from [17], collects a couple of important facts. The first one relates the existential k-cover game with the evaluation of CQs in GHW_k, while the second one states that the existence of a winning strategy for the duplicator can be checked efficiently:

Proposition 9.4. *The following statements hold:*

(1) Given instances I and I', and tuples $\bar{t} \in \mathsf{dom}(I)^n$ and $\bar{t}' \in \mathsf{dom}(I')^n$, where $n \geq 0$, if $(I, \bar{t}) \rightarrow_k (I', \bar{t}')$, then for every CQ $q \in \mathsf{GHW}_k$, $\bar{t} \in q(I)$ implies $t' \in q(I')$.

(2) Given (finite) databases D and D', and tuples $\bar{t} \in \mathsf{dom}(D)^n$ and $\bar{t}' \in \mathsf{dom}(D')^n$, where $n \geq 0$, $(D, \bar{t}) \rightarrow_k (D', \bar{t}')$ can be checked in polynomial time.

We are now ready to show that evaluating a CQ q that is semantically in GHW_k under a set Σ of TGDs or EGDs over an instance I that satisfies Σ amounts to checking whether $\mathsf{chase}(q, \Sigma) \rightarrow_k I$.

Proposition 9.5. *Consider a set Σ of TGDs or EGDs, a CQ $q(\bar{x})$ that is semantically in GHW_k under Σ, an instance I that satisfies Σ, and a tuple $\bar{t} \in \mathsf{dom}(I)^{|\bar{x}|}$. Then:*

$$\bar{t} \in q(I) \quad \Longleftrightarrow \quad (\mathsf{chase}(q, \Sigma), \bot(\bar{x})) \rightarrow_k (I, \bar{t}).$$

Proof. ($\Rightarrow$) By hypothesis, there exists a homomorphism h from $D[q]$ to I such that $h(\bot(\bar{x})) = \bar{t}$. Since $I \models \Sigma$, by using the universality of the chase, we can show that h can be extended into a homomorphism h' from $\mathsf{chase}(q, \Sigma)$ to I such that $h'(\bot(\bar{x})) = \bar{t}$. This implies that $(\mathsf{chase}(q, \Sigma), \bot(\bar{x})) \rightarrow_k (I, \bar{t})$ since the duplicator can always respond during the game by exploiting the homomorphism h'.

($\Leftarrow$) Conversely, assume that $(\mathsf{chase}(q, \Sigma), \bot(\bar{x})) \rightarrow_k (I, \bar{t})$. Since q is semantically in GHW_k under Σ, there exists a CQ $q' \in \mathsf{GHW}_k$ that is equivalent to q under Σ. Thus, by Lemma 2.5, $\bot(\bar{x}) \in q'(\mathsf{chase}(q, \Sigma))$. By Proposition 9.4, we get that $\bar{t} \in q'(I)$. Since $q \equiv_\Sigma q'$ and $I \models \Sigma$, we conclude that $\bar{t} \in q(I)$, and the claim follows. □

Based on Proposition 9.5, we identify two conditions that lead to the tractability of the problem $\mathsf{EvalSemGHW}_k(\mathbb{C})$ for a class $\mathbb{C}$ of sets of TGDs or EGDs:

- The first condition corresponds to *tractability of chase computation* for sets of constraints in $\mathbb{C}$. This is satisfied, in particular, by the class $\mathbb{FD}$ of sets of FDs.
- The second condition corresponds to *chase redundancy*, which establishes that the problem of checking $\mathsf{chase}(q, \Sigma) \rightarrow_k I$, for a set $\Sigma \in \mathbb{C}$ and an instance I that satisfies Σ, boils down to simply checking $q \rightarrow_k I$. This condition is satisfied, in particular, by the class $\mathbb{G}$.

Efficiently computable chase. A class $\mathbb{C}$ of sets of TGDs or EGDs enjoys *tractability of chase computation*, if the instance $\mathsf{chase}(q, \Sigma)$ can be computed in polynomial time, for every CQ q and set $\Sigma \in \mathbb{C}$. As an immediate corollary to Propositions 9.4 and 9.5, we obtain that if $\mathbb{C}$ enjoys tractability of chase computation, then $\mathsf{EvalSemGHW}_k(\mathbb{C})$ can be solved in polynomial time:

Proposition 9.6. *For every class $\mathbb{C}$ of TGDs or EGDs that enjoys tractability of chase computation,* $\mathsf{EvalSemGHW}_k(\mathbb{C})$ *is in PTime.*

Importantly, the class $\mathbb{FD}$ enjoys tractability of chase computation. This follows from two known facts. Given a CQ q and a set $\Sigma \in \mathbb{FD}$: (1) the length of an arbitrary chase sequence for $D[q]$ under Σ is polynomial (in fact, this holds even for arbitrary EGDs), and (2) a single chase step in such a chase sequence takes polynomial time – for a FD ϵ, determining all the pairs of atoms in a database that violate ϵ, and then applying ϵ over every such pair, takes polynomial time. Then:

Proposition 9.7. $\mathbb{FD}$ *enjoys tractability of chase computation.*

It is clear that Propositions 9.6 and 9.7 imply that $\mathsf{EvalSemGHW}_k(\mathbb{FD})$ is in PTime. We proceed to show that the same holds for $\mathsf{EvalSemGHW}_k(\mathbb{G})$, which will conclude the proof of Theorem 9.3.

Redundancy of constraints. We say that a class $\mathbb{C}$ enjoys *chase redundancy*, if, for each $k \geq 1$, CQ $q(\bar{x})$, set $\Sigma \in \mathbb{C}$, instance I that satisfies Σ, and tuple $\bar{t} \in \mathsf{dom}(I)^{|\bar{x}|}$:

$$(\mathsf{chase}(q, \Sigma), \bot(\bar{x})) \rightarrow_k (I, \bar{t}) \quad \Longleftrightarrow \quad (D[q], \bot(\bar{x})) \rightarrow_k (I, \bar{t}).$$

From Propositions 9.4 and 9.5 we immediately get that:

Proposition 9.8. *For every class $\mathbb{C}$ of TGDs or EGDs that enjoys chase redundancy, it holds that* $\mathsf{EvalSemGHW}_k(\mathbb{C})$ *is in PTime.*

Consequently, in order to show that $\mathsf{EvalSemGHW}_k(\mathbb{G})$ is feasible in polynomial time, and complete the proof of Theorem 9.3, it suffices to show that:

Proposition 9.9. $\mathbb{G}$ *enjoys chase redundancy.*

We proceed to sketch the proof of the above result; the full proof can be found in the appendix. Let us first provide a useful characterization of the fact $(I, \bar{t}) \rightarrow_k (I', \bar{t}')$, where I and I' are instances, $\bar{t} = (t_1, \ldots, t_n) \in \mathsf{dom}(I)^n$, and $\bar{t}' = (t'_1, \ldots, t'_n) \in \mathsf{dom}(I')^n$, for $n \geq 0$, that can be easily obtained

from a characterization in [17]. An instance $J \subseteq I$ is a *k-union* of I if $1 \leq |J| \leq k$, i.e., is a non-empty subinstance of I with at most k atoms. For brevity, let $U_{I,k}$ be the set of all k-unions of I, and $H_{I \to I'} = \bigcup_{J \subseteq I} \{h \mid h \text{ is a homomorphism from } J \text{ to } I'\}$. Two homomorphisms h and h' are *consistent* if, for each term t that occurs in the domain of both h and h', i.e., both h and h' are defined over t, it holds that $h(t) = h'(t)$. A function $\mu : U_{I,k} \to H_{I \to I'}$ is called *winning strategy* for the duplicator in the existential k-cover game on $(I, \bar{t})$ and $(I', \bar{t}')$ if, for each $J \in U_{I,k}$:

(1) $\mu(J) = \{h \in H_{I \to I'} \mid h(J) \subseteq I' \text{ and } t_i \in \mathrm{dom}(J) \implies h(t_i) = t'_i\}$, and
(2) for each $h \in \mu(J)$ and $J' \in U_{I,k}$, there is $h' \in \mu(J')$ such that h, h' are consistent.

It is possible to show that:

Lemma 9.10. *$(I, \bar{t}) \to_k (I', \bar{t}')$ if and only if there exists $\mu : U_{I,k} \to H_{I \to I'}$ that is a winning strategy for the duplicator in the existential k-cover game on $(I, \bar{t})$ and $(I', \bar{t}')$.*

Consider now a CQ $q(\bar{x})$, a set $\Sigma \in \mathbb{G}$ of TGDs, an instance I that satisfies Σ, and a tuple $\bar{t} \in \mathrm{dom}(I)^{|\bar{x}|}$. We need to show that, for each $k \geq 1$:

$$(\mathsf{chase}(q, \Sigma), \bot(\bar{x})) \to_k (I, \bar{t}) \iff (D[q], \bot(\bar{x})) \to_k (I, \bar{t}).$$

($\Rightarrow$) This direction holds trivially since $D[q] \subseteq \mathsf{chase}(q, \Sigma)$; recall that, by convention, we write $\mathsf{chase}(q, \Sigma)$ for the instance $\mathsf{chase}(D[q], \Sigma)$.

($\Leftarrow$) Assume now that $(D[q], \bot(\bar{x})) \to_k (I, \bar{t})$. By Lemma 9.10, there is a winning strategy μ for the duplicator in the existential k-cover game on $(D[q], \bot(\bar{x}))$ and $(I, \bar{t})$. Our goal is, by exploiting μ, to show that there exists a winning strategy μ' for the duplicator in the existential k-cover game on $(\mathsf{chase}(q, \Sigma), \bot(\bar{x}))$ and $(I, \bar{t})$, which will immediately imply, by Lemma 9.10, that $(\mathsf{chase}(q, \Sigma), \bot(\bar{x})) \to_k (I, \bar{t})$. In other words, with $\bot(\bar{x}) = (\bot_{x_1}, \ldots, \bot_{x_n})$ and $\bar{t} = (t_1, \ldots, t_n)$, we need to show that there is $\mu' : U_{\mathsf{chase}(q,\Sigma),k} \to H_{\mathsf{chase}(q,\Sigma) \to I}$ such that, for each $J \in U_{\mathsf{chase}(q,\Sigma),k}$:

(1) $\mu'(J) = \{h \in H_{\mathsf{chase}(q,\Sigma) \to I} \mid h(J) \subseteq I \text{ and } \bot_{x_i} \in \mathrm{dom}(J) \implies h(\bot_{x_i}) = t_i\}$, and
(2) for each $h \in \mu'(J)$ and $J' \in U_{\mathsf{chase}(q,\Sigma),k}$, there exists $h' \in \mu'(J')$ such that h, h' are consistent.

Suppose that $\mathsf{chase}(q, \Sigma)$ is obtained by the chase sequence for $D[q]$ under Σ

$$D[q] = I_0 \xrightarrow{\tau_0, (\bar{u}_0, \bar{u}'_0)} I_1 \xrightarrow{\tau_1, (\bar{u}_1, \bar{u}'_1)} I_2 \ldots$$

We show that there are functions $(\mu_j : U_{I_j,k} \to H_{I_j \to I})_{j \geq 0}$ such that, for each $j \geq 0$:

(1) the function μ_j is a winning strategy for the duplicator in the existential k-cover game on $(I_j, \bot(\bar{x}))$ and $(I, \bar{t})$, and
(2) for each $J \in U_{I_j,k}$, $\mu_j(J) = \mu_{j+1}(J)$.

Having the above sequence of functions in place, the claim follows with $\mu' = \bigcup_{j \geq 0} \mu_j$. The proof that $(\mu_j)_{j \geq 0}$ exists is by induction on $j \geq 0$, and can be found in the appendix.

Let us conclude this section by saying that for the other classes $\mathbb{C}$ of TGDs considered above, i.e., $\mathbb{NR}$ and $\mathbb{S}$, the question whether $\mathsf{EvalSemGHW}_k(\mathbb{C})$ is feasible in polynomial time is still unanswered. This is left as an interesting open problem.

10 QUERY APPROXIMATIONS WITH TGDS

In this section, we study the more liberal notion of GHW_k-approximation of a CQ under a set of constraints. Given a CQ q, and a set Σ of TGDs that falls in $\mathbb{G}$ or $\mathbb{NR}$ or $\mathbb{S}$, we can exploit the techniques developed in Sections 5 and 6 in order to compute the CQs in GHW_k, for $k \geq 1$, that are maximally contained in q and Σ. Following the recent database literature, such CQs correspond to the GHW_k-approximations of q under Σ; see, e.g., [6–8]. Computing and evaluating the GHW_k-approximations of q under Σ might help finding "quick" (i.e., fixed-parameter tractable) answers to

it when exact evaluation is infeasible. We proceed to formalize the notion of GHW_k-approximation of q under Σ; in the rest of the section, we fix $k \geq 1$:

Definition 10.1. Consider a CQ q and a set Σ of TGDs, both over a schema σ. A GHW_k-*approximation* of q under Σ is a CQ $q' \in \mathsf{GHW}_k$ over σ such that:

(1) $q' \subseteq_\Sigma q$, and
(2) for every $q'' \in \mathsf{GHW}_k$, $q' \subseteq_\Sigma q'' \subseteq_\Sigma q \implies q' \equiv_\Sigma q''$.

We write $\mathsf{approx}_{\mathsf{GHW}_k}(q, \Sigma)$ for all the GHW_k-approximations of q under Σ (up to $\equiv_\Sigma$). ∎

Intuitively, condition (1) corresponds to *soundness*, i.e., q' only returns sound answers w.r.t. q and Σ, while condition (2) corresponds to *maximality*, i.e., there is no CQ $q'' \in \mathsf{GHW}_k$ that approximates q better than q' in terms of containment under Σ. Notice that whenever q is semantically in GHW_k under Σ, i.e., there is a CQ $q' \in \mathsf{GHW}_k$ such that $q \equiv_\Sigma q'$, then the unique GHW_k-approximation of q under Σ is q' itself. Therefore, the notion of GHW_k-approximation under TGDs provides a suitable extension of the notion of being semantically in GHW_k under TGDs.

We proceed to present the main result of this section, which establishes some fundamental properties of GHW_k-approximations of a CQ q under a set Σ of TGDs that falls in one of the decidable classes of TGDs considered in the previous sections. In particular, we show that they always exist, they consist of exponentially many atoms (actually, in the case of guarded TGDs, they consist of polynomially many atoms), and they can be computed in double-exponential time. In what follows, for notational convenience, given a CQ q and a set Σ of TGDs, let $g_{\mathbb{G}}(q, \Sigma) = |q| \cdot (2k + 1)$, and $g_{\mathbb{C}}(q, \Sigma) = f_{\mathbb{C}}(q, \Sigma) \cdot (2k + 1)$ if $\mathbb{C} \in \{\mathbb{NR}, \mathbb{S}\}$; recall that the functions $f_{\mathbb{NR}}$ and $f_{\mathbb{S}}$ are given by Propositions 6.7 and 6.9, respectively.

Theorem 10.2. *Consider a CQ q and a set $\Sigma \in \mathbb{C}$ of TGDs, where $\mathbb{C} \in \{\mathbb{G}, \mathbb{NR}, \mathbb{S}\}$. Then:*

(1) $\mathsf{approx}_{\mathsf{GHW}_k}(q, \Sigma) \neq \varnothing$.
(2) $|q'| \leq g_{\mathbb{C}}(q, \Sigma)$, *for each* $q' \in \mathsf{approx}_{\mathsf{GHW}_k}(q, \Sigma)$.
(3) $\mathsf{approx}_{\mathsf{GHW}_k}(q, \Sigma)$ *can be computed in double-exponential time in* $||q|| + ||\Sigma||$.

Before giving the proof of the above result, we need to establish an auxiliary lemma, which is reminiscent of Lemma 5.4 presented in Section 5. In fact, this auxiliary result is implicit in the proof of Propositions 5.3 and 6.5. Nevertheless, for the sake of clarity and completeness, we would like to explicitly state this lemma; its proof can be found in the appendix:

Lemma 10.3. *Let $q, q' \in \mathsf{GHW}_k$ over a schema σ, and $\Sigma \in \mathbb{C}$ over σ, where $\mathbb{C} \in \{\mathbb{G}, \mathbb{NR}, \mathbb{S}\}$, such that $q' \subseteq_\Sigma q$. There is a CQ $q'' \in \mathsf{GHW}_k$ over σ such that $q' \subseteq_\Sigma q'' \subseteq_\Sigma q$ and $|q''| \leq g_{\mathbb{C}}(q, \Sigma)$.*

Having the above lemma in place, we can now give the proof of Theorem 10.2. In the sequel, given a CQ q and a set $\Sigma \in \mathbb{C}$ of TGDs, we write $\mathsf{cont}_{\mathsf{GHW}_k}(q, \Sigma)$ for the set of CQs $q' \in \mathsf{GHW}_k$ such that $q' \subseteq_\Sigma q$ and $|q'| \leq g_{\mathbb{C}}(q, \Sigma)$. Moreover, we write $\mathsf{maximal}_{\mathsf{GHW}_k}(q, \Sigma)$ for the $\subseteq_\Sigma$-maximal elements of $\mathsf{cont}_{\mathsf{GHW}_k}(q, \Sigma)$. In other words, $\mathsf{maximal}_{\mathsf{GHW}_k}(q, \Sigma)$ consists of the CQs $q' \in \mathsf{cont}_{\mathsf{GHW}_k}(q, \Sigma)$ for which there is no $q'' \in \mathsf{cont}_{\mathsf{GHW}_k}(q, \Sigma)$ such that $q' \subset_\Sigma q''$, where the latter notation means that $q' \subseteq_\Sigma q''$ and, in addition, there exists an instance I that satisfies Σ such that $q'(I) \subset q''(I)$.

Proof. (of Theorem 10.2) We first observe that there exists a CQ $q' \in \mathsf{GHW}_k$ such that $q' \subseteq_\Sigma q$ and $|q'| \leq |q|$. Assuming that $(x_1, \ldots, x_n)$ are the free variables of q, and $R_1/\ell_1, \ldots, R_m/\ell_m$ are the predicates occurring in q, then q' is defined as

$$q'(\underbrace{x, \ldots, x}_{n}) \;:=\; R_1(\underbrace{x, \ldots, x}_{\ell_1}) \wedge \cdots \wedge R_m(\underbrace{x, \ldots, x}_{\ell_m}).$$

It is clear that q' is acyclic, i.e., $q' \in \mathsf{GHW}_1$, which implies that $q' \in \mathsf{GHW}_k$. Moreover, $q' \subseteq_\Sigma q$ since $(\bot_x, \ldots, \bot_x)$ belongs to the evaluation of q over $\mathsf{chase}(q', \Sigma)$, and $|q'| \leq |q|$ since $m \leq |q|$. This immediately implies that the set $\mathsf{cont}_{\mathsf{GHW}_k}(q, \Sigma)$ is non-empty, and thus, $\mathsf{maximal}_{\mathsf{GHW}_k}(q, \Sigma)$ is non-empty. We now show that the set $\mathsf{maximal}_{\mathsf{GHW}_k}(q, \Sigma)$ consists of all the GHW_k-approximations of q under Σ (up to $\equiv_\Sigma$). Formally:

(†) For each $q' \in \mathsf{approx}_{\mathsf{GHW}_k}(q, \Sigma)$, there is $q'' \in \mathsf{maximal}_{\mathsf{GHW}_k}(q, \Sigma)$ such that $q' \equiv_\Sigma q''$.
(††) For each $q' \in \mathsf{maximal}_{\mathsf{GHW}_k}(q, \Sigma)$, there is $q'' \in \mathsf{approx}_{\mathsf{GHW}_k}(q, \Sigma)$ such that $q' \equiv_\Sigma q''$.

We first show that (†) holds. Consider an arbitrary $q' \in \mathsf{approx}_{\mathsf{GHW}_k}(q, \Sigma)$. By definition, $q' \in \mathsf{GHW}_k$ and $q' \subseteq_\Sigma q$. Therefore, by Lemma 10.3, there exists a CQ $\hat{q} \in \mathsf{GHW}_k$ such that $q' \subseteq_\Sigma \hat{q} \subseteq_\Sigma q$, and $|\hat{q}| \leq g_{\mathbb{C}}(q, \Sigma)$. Clearly, $\hat{q} \in \mathsf{cont}_{\mathsf{GHW}_k}(q, \Sigma)$, and thus, there exists a CQ $q'' \in \mathsf{maximal}_{\mathsf{GHW}_k}(q, \Sigma)$ such that $q' \subseteq_\Sigma \hat{q} \subseteq_\Sigma q'' \subseteq_\Sigma q$. By definition, $q'' \in \mathsf{GHW}_k$, which in turn implies that $q' \equiv_\Sigma q''$ since q' is a GHW_k-approximation of q under Σ, and the claim follows.

We now show that (††) holds. Consider a CQ $q' \in \mathsf{maximal}_{\mathsf{GHW}_k}(q, \Sigma)$. We are going to show that $q' \in \mathsf{approx}_{\mathsf{GHW}_k}(q, \Sigma)$. Fix an arbitrary CQ $q'' \in \mathsf{GHW}_k$. We need to show that $q' \subseteq_\Sigma q'' \subseteq_\Sigma q$ implies $q' \equiv_\Sigma q''$. Since, by hypothesis, $q'' \subseteq_\Sigma q$, Lemma 10.3 implies the existence of a CQ $\hat{q} \in \mathsf{GHW}_k$ such that $q' \subseteq_\Sigma q'' \subseteq_\Sigma \hat{q} \subseteq_\Sigma q$, and $|\hat{q}| \leq g_{\mathbb{C}}(q, \Sigma)$. Clearly, $\hat{q} \in \mathsf{cont}_{\mathsf{GHW}_k}(q, \Sigma)$. Since $q' \in \mathsf{maximal}_{\mathsf{GHW}_k}(q, \Sigma)$, either $\hat{q} \subseteq_\Sigma q'$, or $q' \not\subseteq_\Sigma \hat{q}$ and $\hat{q} \not\subseteq_\Sigma q'$. Notice that the latter case contradicts the fact that $q' \subseteq_\Sigma \hat{q}$. Thus, the only valid case is $\hat{q} \subseteq_\Sigma q'$. Therefore, $q' \equiv_\Sigma \hat{q}$, which in turn implies that $q' \equiv_\Sigma q''$, and the claim follows.

It is now easy to establish items (1), (2) and (3). Recall that $\mathsf{maximal}_{\mathsf{GHW}_k}(q, \Sigma)$ is non-empty since $\mathsf{cont}_{\mathsf{GHW}_k}(q, \Sigma)$ is non-empty. Thus, $\mathsf{approx}_{\mathsf{GHW}_k}(q, \Sigma) \neq \varnothing$ since $\mathsf{maximal}_{\mathsf{GHW}_k}(q, \Sigma)$ consists of all the GHW_k-approximations of q under Σ (up to $\equiv_\Sigma$), and (1) follows. Item (2) follows by definition, since each CQ $q' \in \mathsf{maximal}_{\mathsf{GHW}_k}(q, \Sigma)$ consists of at most $g_{\mathbb{C}}(q, \Sigma)$ atoms. Finally, for item (3) it suffices to show that the set $\mathsf{maximal}_{\mathsf{GHW}_k}(q, \Sigma)$ can be computed in double-exponential time in $||q|| + ||\Sigma||$. This is done by simply enumerating all CQs q' such that $|q'| \leq g_{\mathbb{C}}(q, \Sigma)$, and for each one we check that (a) $q' \in \mathsf{GHW}_k$, (b) $q' \subseteq_\Sigma q$, and (c) there is no CQ $q'' \in \mathsf{GHW}_k$ such that $q' \subset_\Sigma q'' \subseteq_\Sigma q$ and $|q''| \leq g_{\mathbb{C}}(q, \Sigma)$. In general, we have to consider double-exponentially many queries (in fact, exponentially many when $\mathbb{C} = \mathbb{G}$), and for every query, each one of the above steps can be carried out in double-exponential time in $||q|| + ||\Sigma||$. □

Let us conclude this section by saying a few words about that problem of evaluating the GHW_k-approximations of a CQ under a set of TGDs, that is, given a CQ $q(\bar{x})$, a set $\Sigma \in \mathbb{C}$ of TGDs, where $\mathbb{C} \in \{\mathbb{G}, \mathbb{NR}, \mathbb{S}\}$, a database D such that $D \models \Sigma$, and a tuple $\bar{t} \in \mathsf{dom}(D)^{|\bar{x}|}$, decide whether there exists a GHW_k-approximation q' of q under Σ such that $\bar{t} \in q'(D)$. Since each such q' is contained in q under Σ, we can then be sure that $\bar{t}$ is a sound answer to q over D. By Theorem 10.2, the set $\mathsf{approx}_{\mathsf{GHW}_k}(q, \Sigma)$ can be computed in double-exponential time in $||q|| + ||\Sigma||$, while each GHW_k-approximation of q under Σ is of single-exponential size. This allows us to show that checking whether $\bar{t} \in q'(D)$ for some GHW_k-approximation q' of q under Σ takes time:

$$O\left(||D||^{k+1} \cdot 2^{2^{p(||q||+||\Sigma||)}}\right),$$

for a suitable polynomial $p : \mathbb{N} \to \mathbb{N}$. This shows that the problem of evaluating the GHW_k-approximations of a CQ q under a set Σ of TGDs that falls in $\mathbb{G}$ or $\mathbb{NR}$ or $\mathbb{S}$ is fixed-parameter tractable. Thus, as said above, computing and evaluating the GHW_k-approximations of q under Σ might help finding "quick" answers to it when exact evaluation is infeasible.

The crucial question that comes up is whether the above results can be established in the presence of functional dependencies, or, more generally, in the presence of equality-generating dependencies. This is left as an interesting open problem.

11 CONCLUSIONS AND OPEN PROBLEMS

We have concentrated on the static analysis task of checking whether a CQ is semantically in the class of CQs of generalized hypertreewidth bounded by a fixed constant $k \geq 1$, denoted GHW_k, under a set of database constraints; in fact, TGDs or EGDs. In other words, we check whether the given CQ is equivalent to one that belongs to GHW_k over all those databases that satisfy the given set of constraints. This problem has been dubbed SemGHW_k.

Surprisingly, we have shown that there are classes of constraints for which containment is decidable, while SemGHW_1 is undecidable. In particular, this holds for full TGDs, i.e., TGDs without existentially quantified variables in the head, and EGDs; in both cases, the undecidability holds even if we focus on unary and binary predicates. We have then focussed on the main classes of TGDs for which CQ containment is decidable, and do not subsume full TGDs, i.e., guarded, non-recursive and sticky sets of TGDs. For these classes we have shown that SemGHW_k is decidable, and obtained several complexity results. We have also shown that SemGHW_k is decidable in elementary time if we focus on keys assuming schemas with unary and binary predicates only. Moreover, we have considered the problem of evaluating a CQ that is semantically in GHW_k under a set Σ of constraints over a database that satisfies Σ. We have shown that whenever Σ falls in one of the classes of constraints mentioned above, the problem is fixed-parameter tractable. Moreover, in the case of guarded TGDs and (arbitrary) functional dependencies, we obtain a pure tractability result. Finally, we have considered the problem of optimally approximating a CQ, which is not semantically in GHW_k, via a CQ in GHW_k, and show that for our TGD-based classes of constraints such an approximation always exists. Computing and evaluating such approximations might help finding "quick" answers to the input query when exact evaluation is infeasible.

11.1 Open Problems

Our technical results provide a comprehensive picture of SemGHW_k in the presence of constraints, as well as of the problems of query evaluation and approximation. Nevertheless, there are still interesting, and some of them highly non-trivial, open problems that we are planning to tackle:

(1) The precise complexity of SemGHW_k under sticky sets of TGDs is still unknown. Recall that we have a NExpTime upper bound and an ExpTime lower bound (Theorem 6.10). As discussed in Section 6, our goal is to identify a more refined property than UCQ rewritability, which will allow us to show that SemGHW_k under sticky sets of TGDs is in ExpTime.
(2) The exact complexity of SemGHW_k under keys over unary and binary predicates is still unknown. We have shown via an involved proof that this problem is decidable in elementary time (Theorem 8.1). In fact, as shown in the conference paper [23], this can be reduced to 2ExpTime by exploiting tree-walking automata.
(3) The decidability status of SemGHW_k under arbitrary key or functional dependencies, without any assumption on the underlying schema, is still open. We conjecture that in this case SemGHW_1 is undecidable. However, it seems that this is a highly non-trivial result. Let us stress that the EGDs employed in the proof that SemGHW_1 under EGDs is undecidable (Theorem 7.1) are far from being keys or FDs.
(4) The question whether the evaluation of CQs that are semantically in GHW_k under non-recursive or sticky sets of TGDs is tractable remains open. Recall that the above problem is fixed-parameter tractable (Theorem 9.1), while for guarded TGDs and functional dependencies is even tractable (Theorem 9.3). Can we establish such a pure tractability result for non-recursive and sticky sets of TGDs?
(5) Recall that for the classes of TGDs considered in this work we can always approximate a CQ, which is not semantically in GHW_k, via a CQ that falls in GHW_k in an optimal way

(Theorem 10.2). The question whether this result can be extended to keys, or, more generally, to EGDs, remains unanswered.

ACKNOWLEDGMENTS

The authors would like to thank Thomas Schwentick and Luc Segoufin for reading a previous version of this manuscript and spotting two major bugs in the proofs. Barceló is supported by the Millennium Institute for Foundational Research on Data (IMFD Chile) and Fondecyt grant 1170109. Figueira is partially supported by ANR project DéLTA (grant ANR-16-CE40-0007) and ANR project QUID (grant ANR-18-CE40-0031). Gottlob is a Royal Society Research Professor and acknowledges support by the Royal Society in the context of the project "RAISON DATA" (Project reference: RP/R1/201074). Gottlob is also supported by the EPSRC Programme Grant EP/M025268/ VADA. Pieris is supported by the EPSRC grant EP/S003800/1 EQUID.

REFERENCES

[1] Serge Abiteboul, Richard Hull, and Victor Vianu. 1995. *Foundations of Databases*. Addison-Wesley.
[2] Jean-François Baget, Marie-Laure Mugnier, Sebastian Rudolph, and Michaël Thomazo. 2011. Walking the Complexity Lines for Generalized Guarded Existential Rules. In *IJCAI*. 712–717.
[3] Vince Bárány, Georg Gottlob, and Martin Otto. 2014. Querying the Guarded Fragment. *Logical Methods in Computer Science* 10, 2 (2014).
[4] Pablo Barceló, Cristina Feier, Carsten Lutz, and Andreas Pieris. 2019. When is Ontology-Mediated Querying Efficient?. In *LICS*. 1–13.
[5] Pablo Barceló, Georg Gottlob, and Andreas Pieris. 2016. Semantic Acyclicity Under Constraints. In *PODS*. 343–354.
[6] Pablo Barceló, Leonid Libkin, and Miguel Romero. 2014. Efficient Approximations of Conjunctive Queries. *SIAM J. Comput.* 43, 3 (2014), 1085–1130.
[7] Pablo Barceló, Reinhard Pichler, and Sebastian Skritek. 2015. Efficient Evaluation and Approximation of Well-designed Pattern Trees. In *PODS*. 131–144.
[8] Pablo Barceló, Miguel Romero, and Moshe Y. Vardi. 2016. Semantic Acyclicity on Graph Databases. *SIAM J. Comput.* 45, 4 (2016), 1339–1376.
[9] Catriel Beeri, Ronald Fagin, David Maier, Alberto O. Mendelzon, Jeffrey D. Ullman, and Mihalis Yannakakis. 1981. Properties of Acyclic Database Schemes. In *STOC*. 355–362.
[10] Catriel Beeri and Moshe Y. Vardi. 1981. The Implication Problem for Data Dependencies. In *ICALP*. 73–85.
[11] Andrea Calì, Georg Gottlob, and Michael Kifer. 2013. Taming the Infinite Chase: Query Answering under Expressive Relational Constraints. *J. Artif. Intell. Res.* 48 (2013), 115–174.
[12] Andrea Calì, Georg Gottlob, and Thomas Lukasiewicz. 2012. A general Datalog-based framework for tractable query answering over ontologies. *J. Web Sem.* 14 (2012), 57–83.
[13] Andrea Calì, Georg Gottlob, and Andreas Pieris. 2012. Towards more expressive ontology languages: The query answering problem. *Artif. Intell.* 193 (2012), 87–128.
[14] Andrea Calì, Domenico Lembo, and Riccardo Rosati. 2003. On the Decidability and Complexity of Query Answering Over Inconsistent and Incomplete Databases. In *PODS*. 260–271.
[15] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. 2008. Conjunctive query containment and answering under description logic constraints. *ACM Trans. Comput. Log.* 9, 3 (2008).
[16] Ashok K. Chandra and Philip M. Merlin. 1977. Optimal Implementation of Conjunctive Queries in Relational Data Bases. In *STOC*. 77–90.
[17] Hubie Chen and Víctor Dalmau. 2005. Beyond Hypertree Width: Decomposition Methods Without Decompositions. In *CP*. 167–181.
[18] Bruno Courcelle. 1989. The Monadic Second-Order Logic of Graphs, II: Infinite Graphs of Bounded Width. *Mathematical Systems Theory* 21, 4 (1989), 187–221.
[19] Víctor Dalmau, Phokion G. Kolaitis, and Moshe Y. Vardi. 2002. Constraint Satisfaction, Bounded Treewidth, and Finite-Variable Logics. In *CP*. 310–326.
[20] Alin Deutsch, Alan Nash, and Jeff B. Remmel. 2008. The Chase Revisisted. In *PODS*. 149–158.
[21] Ronald Fagin. 1981. A Normal Form for Relational Databases That Is Based on Domians and Keys. *ACM Trans. Database Syst.* 6, 3 (1981), 387–415.
[22] Ronald Fagin, Phokion G. Kolaitis, Renée J. Miller, and Lucian Popa. 2005. Data exchange: Semantics and query answering. *Theor. Comput. Sci.* 336, 1 (2005), 89–124.

[23] Diego Figueira. 2016. Semantically Acyclic Conjunctive Queries under Functional Dependencies. In *LICS*. 847–856.
[24] Wolfgang Fischl, Georg Gottlob, and Reinhard Pichler. 2018. General and Fractional Hypertree Decompositions: Hard and Easy Cases. In *PODS*. 17–32.
[25] Tomasz Gogacz and Jerzy Marcinkowski. 2017. Converging to the chase - A tool for finite controllability. *J. Comput. Syst. Sci.* 83, 1 (2017), 180–206.
[26] Nathan Goodman and Oded Shmueli. 1982. Tree Queries: A Simple Class of Relational Queries. *ACM Trans. Database Syst.* 7, 4 (1982), 653–677.
[27] Georg Gottlob, Gianluigi Greco, Nicola Leone, and Francesco Scarcello. 2016. Hypertree Decompositions: Questions and Answers. In *PODS*. 57–74.
[28] Georg Gottlob, Gianluigi Greco, and Bruno Marnette. 2009. HyperConsistency Width for Constraint Satisfaction: Algorithms and Complexity Results. In *Graph Theory, Computational Intelligence and Thought*. 87–99.
[29] Georg Gottlob, Nicola Leone, and Francesco Scarcello. 2002. Hypertree Decompositions and Tractable Queries. *J. Comput. Syst. Sci.* 64, 3 (2002), 579–627.
[30] Georg Gottlob, Zoltán Miklós, and Thomas Schwentick. 2009. Generalized hypertree decompositions: NP-hardness and tractable variants. *J. ACM* 56, 6 (2009).
[31] Georg Gottlob, Giorgio Orsi, and Andreas Pieris. 2014. Query Rewriting and Optimization for Ontological Databases. *ACM Trans. Database Syst.* (2014).
[32] Pavol Hell and Jaroslav Nešetřil. 2004. *Graphs and Homomorphisms*. Oxford University Press.
[33] David S. Johnson and Anthony C. Klug. 1984. Testing Containment of Conjunctive Queries under Functional and Inclusion Dependencies. *J. Comput. Syst. Sci.* 28, 1 (1984), 167–189.
[34] Leonid Libkin. 2004. *Elements of Finite Model Theory*. Springer.
[35] Thomas Lukasiewicz, Maria Vanina Martinez, Andreas Pieris, and Gerardo I. Simari. 2015. From Classical to Consistent Query Answering under Existential Rules. In *AAAI*. 1546–1552.
[36] David Maier, Alberto O. Mendelzon, and Yehoshua Sagiv. 1979. Testing Implications of Data Dependencies. *ACM Trans. Database Syst.* 4, 4 (1979), 455–469.
[37] Christos H. Papadimitriou and Mihalis Yannakakis. 1999. On the Complexity of Database Queries. *J. Comput. Syst. Sci.* 58, 3 (1999), 407–427.
[38] Riccardo Rosati. 2011. On the finite controllability of conjunctive query answering in databases under open-world assumption. *J. Comput. Syst. Sci.* 77, 3 (2011), 572–594.
[39] Yehoshua Sagiv and Mihalis Yannakakis. 1980. Equivalences Among Relational Expressions with the Union and Difference Operators. *J. ACM* 27, 4 (1980), 633–655.
[40] Detlef Seese. 1991. The structure of the models of decidable monadic theories of graphs. *Annals of pure and applied logic* 53, 2 (1991), 169–195.
[41] Robert Endre Tarjan and Mihalis Yannakakis. 1984. Simple Linear-Time Algorithms to Test Chordality of Graphs, Test Acyclicity of Hypergraphs, and Selectively Reduce Acyclic Hypergraphs. *SIAM J. Comput.* 13, 3 (1984), 566–579.
[42] Mihalis Yannakakis. 1981. Algorithms for Acyclic Database Schemes. In *VLDB*. 82–94.

A ADDITIONAL PROOFS FROM SECTION 4

THEOREM 4.4. $\mathsf{SemGHW}_1(\mathbb{F})$ *is undecidable, even if we allow only unary and binary predicates.*

For an instance of PCP over the alphabet $\{a, b\}$ given by the lists $u_1, \ldots, u_n$ and $v_1, \ldots, v_n$, we are going to construct a Boolean CQ q, and a set $\Sigma \in \mathbb{F}$ of TGDs, both over the schema σ described above, such that the PCP instance has a solution if and only if there exists a CQ $q' \in \mathsf{GHW}_1$ (i.e., an acyclic CQ) such that $q \equiv_\Sigma q'$. The proof is structured as follows:

- We first define the CQ q, and summarize some crucial syntactic properties of it (Claim A.1).
- We then define the set $\Sigma \in \mathbb{F}$ of TGDs, and show that q is closed under the applications of the TGDs in Σ, i.e., $D[q] = \mathsf{chase}(q, \Sigma)$ (Claim A.2).
- We finally show that our construction is a reduction, i.e., the PCP instance has a solution if and only if q is semantically in GHW_1 under Σ. The proof exploits the syntactic properties of q summarized in Claim A.1, and the fact that $D[q] = \mathsf{chase}(q, \Sigma)$ shown in Claim A.2.

The Boolean Conjunctive Query q

The query q mentions the variables y, z, and $x_1, \ldots, x_{2(n^2+2n+2)}$, and is defined as follows; notice that all the variables in q are existentially quantified, and whenever we use addition (+) we mean addition modulo $2(n^2 + 2n + 2)$:

$$\bigwedge_{1 \leq i \leq 2(n^2+2n+2)} \mathsf{start}(y, x_i) \wedge \mathsf{end}(x_i, z) \wedge \bigwedge_{1 \leq i,j \leq 2(n^2+2n+2)} \eta_1(x_i, x_j) \wedge \eta_2(x_i, x_j) \wedge \eta(x_i, x_j) \wedge$$

$$\bigwedge_{1 \leq i \leq 2(n^2+2n+2)} a(x_i, x_{i+1}) \wedge a(x_{i+2}, x_i) \wedge b(x_i, x_{i+3}) \wedge b(x_{i+4}, x_i) \wedge$$

$$\bigwedge_{1 \leq i \leq 2(n^2+2n+2)} \bigwedge_{1 \leq j \leq n} \left(\diamond_{j,*}(x_i, x_{i+2j+3}) \wedge \diamond_{j,*}(x_{i+2j+4}, x_i) \wedge \diamond_{*,j}(x_i, x_{i+2j+2n+3}) \wedge \diamond_{*,j}(x_{i+2j+2n+4}, x_i) \right) \wedge$$

$$\bigwedge_{1 \leq i \leq 2(n^2+2n+2)} \bigwedge_{1 \leq j,\ell \leq n} \diamond_{j,\ell}(x_i, x_{i+2(j-1)n+2\ell+4n+3}) \wedge \diamond_{j,\ell}(x_{i+2(j-1)n+2\ell+4n+4}, x_i)$$

Essentially, the variable y can be seen as a source that is linked via an edge labeled start to each variable x_i, for $i \in \{1, \ldots, 2(n^2 + 2n + 2)\}$. Correspondingly, the variable z can be seen as a sink that has an incoming edge labeled end from each such variable x_i. Moreover, all the pairs of variables (x_i, x_j), for $1 \leq i, j \leq 2(n^2 + 2n + 2)$, are linked via edges labeled η_1, η_2, and η. Finally, each variable x_i, for $i \in \{1, \ldots, 2(n^2 + 2n + 2)\}$, has outgoing edges labeled a, b, $\diamond_{j,*}$, $\diamond_{*,j}$, and $\diamond_{j,\ell}$, for each $1 \leq j, \ell \leq 2(n^2 + 2n + 2)$, as well as incoming edges with the same labels. In Figure 11, we depict part of q comprised by variables y, z, and how variable x_i is connected to all variables x_j, with $i < j$.

The following claim, which holds by definition, summarizes some important syntactic properties of the CQ q. As shown in Figure 11, q can be naturally seen as a directed graph in which edges are labeled with binary predicates occurring in q. In the following claim, we see q as a directed graph.

CLAIM A.1. *The following statements hold for q:*

(1) For each word w over $\{a, b, (\diamond_{k,*})_{1 \leq k \leq n}, (\diamond_{*,k})_{1 \leq k \leq n}, (\diamond_{k,\ell})_{1 \leq k,\ell \leq n}\}$, and $i \in \{1, \ldots, 2(n^2+2n+2)\}$, there is a directed path that starts at x_i, contains only variables from $\{x_1, \ldots, x_{2(n^2+2n+2)}\}$, and is labeled by w.

(2) For each $1 \leq i < j \leq 2(n^2 + 2n + 2)$, there is exactly one undirected edge between x_i and x_j labeled with a symbol from $\{a, b, (\diamond_{k,*})_{1 \leq k \leq n}, (\diamond_{*,k})_{1 \leq k \leq n}, (\diamond_{k,\ell})_{1 \leq k,\ell \leq n}\}$. In other words, q

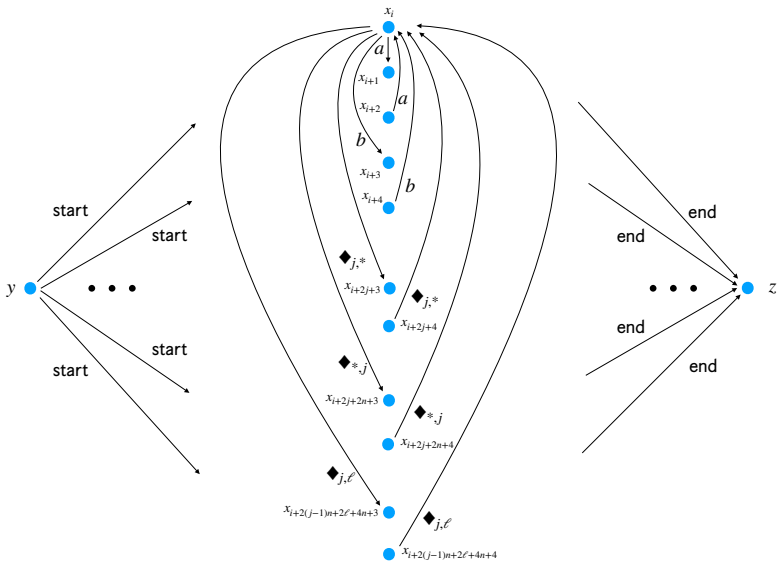


Fig. 11. A part of CQ q showing variables y, z, and how variable x_i is connected to all variables x_j, with $i < j$. We omit relation symbols η, η_1, and η_2 from the figure, as all pairs (x_i, x_j) are connected with an edge labeled with all such symbols.

contains exactly one atom from the set

$$\bigcup_{1\leq i<j\leq 2(n^2+2n+2)} \left\{ R(x_i,x_j), R(x_j,x_i) \mid R \in \{a, b, (\diamond_{k,*})_{1\leq k\leq n}, (\diamond_{*,k})_{1\leq k\leq n}, (\diamond_{k,\ell})_{1\leq k,\ell\leq n}\} \right\}.$$

(3) Each self-loop is labeled only with η_1, η_2 and η. In other words, the only atoms of q that contain exactly one variable are

$$\bigcup_{1\leq i\leq 2(n^2+2n+2)} \{\eta_2(x_i,x_i), \eta_1(x_i,x_i), \eta(x_i,x_i)\}.$$

The above claim will be exploited in the correctness proof given below. This concludes our discussion on the query q. We proceed with the definition of Σ.

The Set Σ of Full TGDs

We start by introducing some terminology. Recall that our PCP instance is defined by two lists $u_1, \ldots, u_n$ and $v_1, \ldots, v_n$ of words over $\{a, b\}$. Consider the first one of the words u_i, for $1 \leq i \leq n$. We denote by Extended(u_i) the set of words that extend u_i by adding arbitrarily many occurrences of symbols from $\{\diamond_{*,j} \mid 1 \leq j \leq n\}$ at non-consecutive positions (apart from the last one). Formally, we define Extended(u_i) as the set of all words u_i' over the alphabet $\{a, b, \diamond_{*,j} \mid 1 \leq j \leq n\}$ such that: (a) the restriction of u_i' to the alphabet $\{a, b\}$ is precisely u_i, (b) no two symbols from $\{\diamond_{*,j} \mid 1 \leq j \leq n\}$ appear consecutively in u_i', and (c) the last symbol of u_i' is either a or b. Notice that each word in Extended(u_i) is of size at most $2 \cdot |u_i|$, and, thus, Extended(u_i) is a finite set. Analogously, we define Extended(v_i), for $1 \leq i \leq n$. The only difference is that now words in Extended(v_i) are

obtained from v_i by adding arbitrarily many occurrences of symbols from $\{\diamond_{j,*} \mid 1 \leq j \leq n\}$ at non-consecutive positions.

Let w be a word of the form $t_1 \dots t_m$, where each t_j, for $1 \leq j \leq m$, is a symbol in

$$\{a, b, \diamond_{*,k}, \diamond_{k,*}, \diamond_{k,l}, \text{start}, \text{end} \mid 1 \leq k, l \leq n\}.$$

In our TGDs, we write $w(x, y)$ as an abbreviation for $t_1(x, z_1), \dots, t_m(z_{m-1}, y)$, where the z_i's are fresh variables. Our set $\Sigma \in \mathbb{F}$ consists of the following TGDs:

(1) An *initialization rule* of the form:

$$\text{start}(x, y), \eta_1(y, z), \eta_2(y, z') \quad \rightarrow \quad \eta(z, z'), \eta(z', z).$$

That is, if a variable y has an incoming edge labeled start and outgoing edges labeled η_1 and η_2 pointing out to variables z and z', respectively, then z and z' are linked to each other via edges labeled η. The recursive rules that define the predicate η, presented next, start iterating from this initialization rule.

(2) For each $1 \leq i \leq n$, words $u_i' \in \text{Extended}(u_i)$ and $v_i' \in \text{Extended}(v_i)$, and symbols $r, r' \in \{*, 1, \dots, n\}$, we have a recursive *matching rule* of the form:

$$\eta_1(x_1, x), \eta_2(y_1, y), \eta(x, y), u_i'(x_1, x^*), \diamond_{i,r}(x^*, x_2), \eta_1(x_2, x'), \\ v_i'(y_1, y^*), \diamond_{r',i}(y^*, y_2), \eta_2(y_2, y') \ \rightarrow \ \eta(x', y').$$

Intuitively, if in a certain path, a pair of nodes (x_1, y_1) is a "match" (represented by the presence of the atoms $\eta_1(x_1, x), \eta_2(y_1, y), \eta(x, y)$), and x_2 and y_2 are the nodes reached from x_1 and y_1 after reading $u_i\diamond_{i,r}$ and $v_i\diamond_{r',i}$, respectively, then (x_2, y_2) is also a "match" (represented by the presence of the atoms $\eta_1(x_2, x'), \eta_2(y_2, y')$ and the creation of the atom $\eta(x, y)$).
The reason why we use a word u_i' in $\text{Extended}(u_i)$ instead of u_i itself (resp., a word v_i' in $\text{Extended}(v_i)$ instead of v_i), is because the reading of u_i from x_1 (resp., v_i from y_1) might need to "jump" over several occurrences of symbols of the form $\diamond_{*,j}$ (resp., $\diamond_{j,*}$), for $1 \leq j \leq n$. These symbols occur non-consecutively, since the words in the PCP instance are non-empty, and thus the word we read belongs to $\text{Extended}(u_i)$ (resp., to $\text{Extended}(v_i)$).

(3) For each $1 \leq i \leq n$, and words $u_i' \in \text{Extended}(u_i)$ and $v_i' \in \text{Extended}(v_i)$, we have a *finalization rule* of the form:

$$\text{start}(y, x), \eta_1(z_1, z_1'), \eta_2(z_2, z_2'), \eta(z_1', z_2'), u_i'(z_1, x_1'), v_i'(z_2, x_1'), \diamond_{i,i}(x_1', x_1), \\ a(x_1, x_2), \dots, a(x_{2(n^2+2n+2)-1}, x_{2(n^2+2n+2)}), \text{end}(x_{2(n^2+2n+2)}, z) \ \rightarrow \ q,$$

where q is an abbreviation for the set of atoms in q (which are defined over the variables $y, z, x_1, \dots, x_{2(n^2+2n+2)}$, as mentioned in the body of the TGDs).
Intuitively, this states that if a path encodes a solution to PCP (a condition that is stated in the rules by the atoms $\eta_1(z_1, z_1'), \eta_2(z_2, z_2'), \eta(z_1', z_2')$ and the synchronization via words $u_i\diamond_{i,i}$ and $v_i\diamond_{i,i}$ on the node x_1), then by appending a tail of $2(n^2 + 2n + 2)$ edges labeled a, and a final edge labeled end, we get a copy of q by applying the finalization rule. This allows $D[q]$ to be mapped to the chase of q', in case the latter represents a solution to the PCP instance.

This concludes the definition of the set $\Sigma \in \mathbb{F}$. Let us stress that Σ can be effectively constructed since, as explained above, for each $i \in [n]$, the sets $\text{Extended}(u_i)$ and $\text{Extended}(v_i)$ are finite. We now establish a simple claim, which states that q is closed under the applications of the TGDs in Σ, that will be used in the correctness proof.

Claim A.2. $D[q] = \text{chase}(q, \Sigma)$.

Proof. It suffices to show that $D[q]$ satisfies Σ. It is clear that $D[q]$ satisfies the initialization and matching rules since, for each $1 \leq i, j \leq 2(n^2+2n+2)$, the atom $\eta(x_i, x_j)$ occurs in q. Consider now the finalization rule τ, and assume that there is a homomorphism h that maps the body of τ to $D[q]$. We need to show that h also maps q to $D[q]$. It is clear that $h(y) = \bot_y$ and $h(z) = \bot_z$, while the variables $z_1, z_2, z_1', z_2', x_1', x_1, \ldots, x_{2(n^2+2n+2)}$ are mapped to nulls in $\{\bot_{x_1}, \ldots, \bot_{x_{2(n^2+2n+2)}}\}$. (Notice that h is not forced to map a variable $x \in \{x_1, \ldots, x_{2(n^2+2n+2)}\}$ to $\bot_x$). Thus, for each $1 \leq i \leq 2(n^2 + 2n + 2) - 1$, it is the case that $a(h(x_i), h(x_j)) \in D[q]$. In other words, $h(x_1)h(x_2)\ldots h(x_{2(n^2+2n+2)})$ is a directed path in $D[q]$ whose edges are labeled a. It is not difficult to verify by inspection of q that this path does not repeat nodes. This implies that there exists $1 \leq j \leq 2(n^2 + 2n + 2)$ such that, for each $1 \leq i \leq 2(n^2+2n+2)$, $h(x_i) = \bot_{x_{j+i-1}}$. It is also not difficult to see that, for each $1 \leq j \leq 2(n^2+2n+2)$, there is an isomorphism from q to $D[q]$ that maps y and z to $\bot_y$ and $\bot_z$, respectively, and, for each $1 \leq i \leq 2(n^2 + 2n + 2)$, it maps x_i to $\bot_{x_{i+j-1}}$. Thus, h maps q to $D[q]$, as needed. □

We now proceed to the last part of the proof of Theorem 4.4, which aims to show that the instance of PCP has a solution if and only if q is semantically in GHW_1 under Σ.

The Correctness Proof

We first explain how to represent solutions to the PCP instance with a word over the alphabet $\{a, b, (\diamond_{i,*})_{1\leq i\leq n}, (\diamond_{*,i})_{1\leq i\leq n}, (\diamond_{i,j})_{1\leq i,j\leq n}\}$. Assume that the solution s is given by the sequence $1 \leq i_1, \ldots, i_m \leq n$ of indices. Consider then the words:

$$w_1^s := u_{i_1} \diamond_{i_1,*} \ldots u_{i_m} \diamond_{i_m,*} \qquad w_2^s := v_{i_1} \diamond_{*,i_1} \ldots v_{i_m} \diamond_{*,i_m}.$$

We define a new word $w_1^s \otimes w_2^s$ that combines the information contained in w_1^s and w_2^s as follows. We write w_1^s and w_2^s in two different tapes (each letter being in a different cell), and start reading them in parallel from left-to-right using one head in each tape. Depending on the symbols read by the heads, we will write a new symbol to an output tape. If the symbols read by the two heads are the same, then we write such symbol to the output tape and move all heads one cell to the right (including the one on the output tape). If the symbols are different, then it must be the case that the symbol c_1 read on the first tape is of the form $\diamond_{i,*}$, or the symbol c_2 read on the second tape is of the form $\diamond_{*,i}$, for $1 \leq i \leq n$. This follows from the fact that $u_{i_1} \ldots u_{i_m} = v_{i_1} \ldots v_{i_m}$, and the way the procedure is defined. Then, we consider three cases:

- If $c_1 = \diamond_{i,*}$ and $c_2 = \diamond_{*,j}$, for $1 \leq i, j \leq n$, then we write the symbol $\diamond_{i,j}$ to the output tape and move all heads one cell to the right.
- If $c_1 = \diamond_{i,*}$, for $1 \leq i \leq n$, while c_2 is either a or b, then we write the symbol $\diamond_{i,*}$ to the output tape, and move the head of the first tape and the output tape one cell to the right; the head of the second tape remains at the same position.
- If $c_2 = \diamond_{*,i}$, for $1 \leq i \leq n$, while c_1 is either a or b, then we write the symbol $\diamond_{*,i}$ to the output tape, and move the head of the second tape and the output tape one cell to the right; the head of the first tape remains at the same position.

As an example, consider the instance of PCP consisting of the lists u_1, u_2, u_3, u_4 and v_1, v_2, v_3, v_4 of words over the alphabet $\{a, b\}$ such that:

$$u_1 = bb, u_2 = ab, u_3 = bb, u_4 = ab \qquad \text{and} \qquad v_1 = bbab, v_2 = b, v_3 = ba, v_4 = b.$$

Then the solution s given by the sequence $(1, 2, 3, 4)$ yields the following words:

$$w_1^s = bb \diamond_{1,*} ab \diamond_{2,*} bb \diamond_{3,*} ab \diamond_{4,*} \qquad \text{and} \qquad w_2^s = bbab \diamond_{*,1} b \diamond_{*,2} ba \diamond_{*,3} b \diamond_{*,4}.$$

The combined word $w_1^s \otimes w_2^s$ is then defined as:

$$bb \diamond_{1,*} ab \diamond_{2,1} b \diamond_{*,2} b \diamond_{3,*} a \diamond_{*,3} b \diamond_{4,4}.$$

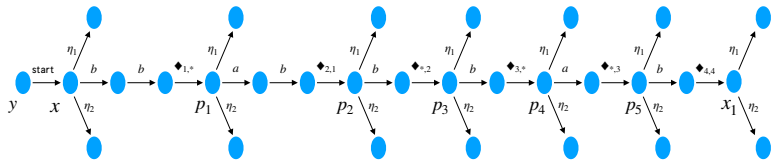


Fig. 12. The initial part of CQ $q[w]$, from y to x_1, for $w = bb \diamond_{1,*} ab \diamond_{2,1} b \diamond_{*,2} b \diamond_{3,*} a \diamond_{*,3} b \diamond_{4,4}$.

The key ingredient of the correctness proof is Lemma A.3 below. We first need some definitions. Let w be a word over $\{a, b, (\diamond_{i,*})_{1\le i\le n}, (\diamond_{*,i})_{1\le i\le n}, (\diamond_{i,j})_{1\le i,j\le n}\}$ and consider a directed path

$$y \xrightarrow{\text{start}} x \xrightarrow{w} x_1 \xrightarrow{a} x_2 \xrightarrow{a} x_3 \cdots x_{2(n^2+2n+2)-1} \xrightarrow{a} x_{2(n^2+2n+2)} \xrightarrow{\text{end}} z$$

Here we use $x \xrightarrow{\ell} x'$ as a visual representation of the expression $\ell(x, x')$. In particular, $x \xrightarrow{w} x_1$ means that there is a directed path from x to x_1, composed of fresh internal nodes, that is labeled w. We extend this path with the following edges:

- $x \xrightarrow{\eta_1} x^{\eta_1}$ and $x \xrightarrow{\eta_2} x^{\eta_2}$, where x^{η_1} and x^{η_2} are fresh variables.
- For each variable p in the path from x to x_1 with an incoming edge labeled $\diamond_{i,*}$, $\diamond_{*,i}$, or $\diamond_{i,j}$, for $1 \le i, j \le n$, we add the edges $p \xrightarrow{\eta_1} p^{\eta_1}$ and $p \xrightarrow{\eta_2} p^{\eta_2}$, where p^{η_1} and p^{η_2} are fresh variables.

We denote the obtained CQ $q[w]$. Clearly, $q[w]$ is a directed tree, which in turn implies that it belongs to GHW_1, i.e., is acyclic. Figure 12 shows a graphical depiction of the initial part of $q[w]$, from y to x_1, for $w = bb \diamond_{1,*} ab \diamond_{2,1} b \diamond_{*,2} b \diamond_{3,*} a \diamond_{*,3} b \diamond_{4,4}$.

We can then prove the following crucial lemma.

Lemma A.3. *The following statements hold:*

(1) If the PCP instance has a solution s, then there is $S \subseteq \mathsf{chase}(q[w_1^s \otimes w_2^s], \Sigma)$, where $\mathsf{dom}(S) = \{\bot_y, \bot_z, \bot_{x_1}, \ldots, \bot_{x_{2(n^2+2n+2)}}\}$, and a bijection $\iota : \mathsf{dom}(S) \to \mathsf{dom}(D[q])$, such that $\iota(S) = D[q]$.

(2) Let w be a word the alphabet $\{a, b, (\diamond_{i,*})_{1\le i\le n}, (\diamond_{*,i})_{1\le i\le n}, (\diamond_{i,j})_{1\le i,j\le n}\}$. If there exists $S \subseteq \mathsf{chase}(q[w], \Sigma)$, where $\mathsf{dom}(S) = \{\bot_y, \bot_z, \bot_{x_1}, \ldots, \bot_{x_{2(n^2+2n+2)}}\}$, and a bijection $\iota : \mathsf{dom}(S) \to \mathsf{dom}(D[q])$ such that $\iota(S) = D[q]$, then the PCP instance has a solution.

Proof. We first prove (1). Assume that the solution s is given by the sequence $1 \le i_1 \ldots i_m \le n$. It is clear that $\eta(\bot_{x^{\eta_1}}, \bot_{x^{\eta_2}}) \in \mathsf{chase}(q[w_1^s \otimes w_2^s], \Sigma)$, due to the initialization rule, since the atoms $\mathsf{start}(\bot_y, \bot_x)$, $\eta_1(\bot_x, \bot_{x^{\eta_1}})$ and $\eta_2(\bot_x, \bot_{x^{\eta_2}})$ occur in the canonical instance of $q[w_1^s \otimes w_2^s]$. From such an atom, the matching rules start iterating over the path (in the canonical instance of $q[w_1^s \otimes w_2^s]$) from $\bot_x$ to $\bot_{x_1}$ labeled $w_1^s \otimes w_2^s$. We claim the following: Over such a path, the matching rules generate all the atoms of the form $\eta(\bot_{y_j^{\eta_1}}, \bot_{z_j^{\eta_2}})$, for $1 \le j \le m$, where the nulls $\bot_{y_j}$ and $\bot_{z_j}$ are the ones that satisfy the following conditions. Suppose that the path from $\bot_x$ to $\bot_{y_j}$ is labeled λ, and the path from $\bot_x$ to $\bot_{z_j}$ is labeled λ', then:

- The restriction of λ to the alphabet $\{a, b\}$ is precisely $u_{i_1} \ldots u_{i_j}$, and the last symbol of λ is of the form $\diamond_{i_j,r}$, for $r \in \{*, 1, \ldots, n\}$.
- The restriction of λ' to the alphabet $\{a, b\}$ is precisely $v_{i_1} \ldots v_{i_j}$, and the last symbol of λ' is of the form $\diamond_{r',i_j}$, for $r' \in \{*, 1, \ldots, n\}$.

We prove this claim by induction on $j \in \{1, \ldots, m\}$. (We sometimes abuse notation, and write y_j and z_j for the positions in $w_1^s \otimes w_2^s$ that are naturally associated with $\bot_{y_j}$ and $\bot_{z_j}$, respectively).

Base case. We assume that u_{i_1} is shorter than v_{i_1} – the other case can be treated in an analogous way. By construction, $w_1^s \otimes w_2^s$ has a prefix of the form $u_{i_1} \diamond_{i_1,*} u'_{i_1} \diamond_{r,i_1}$, where $u_{i_1} \diamond_{i_1,*} u'_{i_1} \in \mathsf{Extended}(v_{i_1})$ and $r \in \{*, 1, \ldots, n\}$. Therefore, the path from $\bot_x$ to $\bot_{x_1}$, in the canonical instance of $q[w_1^s \otimes w_2^s]$, contains an initial path of the form:

$$\bot_x \xrightarrow{u_{i_1} \diamond_{i_1,*}} \bot_{y_1} \xrightarrow{u'_{i_1} \diamond_{r,i_1}} \bot_{z_1}.$$

Observe that the the nulls $\bot_{y_1}$ and $\bot_{z_1}$ satisfy the conditions stated above. Moreover, since $u_{i_1} \diamond_{i_1,*} u'_{i_1}$ belongs to $\mathsf{Extended}(v_{i_1})$, the matching rule:

$$\eta_1(x, x^{\eta_1}), \eta_2(x, x^{\eta_2}), \eta(x^{\eta_1}, x^{\eta_2}), u_{i_1}(x, y^*), \diamond_{i_1,*}(y^*, y_1), \eta_1(y_1, y_1^{\eta_1}), \\ [u_{i_1} \diamond_{i_1,*} u'_{i_1}](x, z^*), \diamond_{r,i_1}(z^*, z_1), \eta_2(z_1, z_1^{\eta_2}) \;\rightarrow\; \eta(y_1^{\eta_1}, z_1^{\eta_2})$$

is triggered, and the atom $\eta(\bot_{y_1^{\eta_1}}, \bot_{z_1^{\eta_2}})$ is generated.

Inductive step. By induction hypothesis, the atom $\eta(\bot_{y_j^{\eta_1}}, \bot_{z_j^{\eta_2}})$ has been generated during the chase. By the construction of $w_1^s \otimes w_2^s$, and the definition of $\bot_{y_j}$ and $\bot_{z_j}$, the subword of $w_1^s \otimes w_2^s$ from position y_j to y_{j+1} is the word $u'_{i_{j+1}} \diamond_{i_{j+1},r}$ such that $u'_{i_{j+1}} \in \mathsf{Extended}(u_{i_{j+1}})$ and $r \in \{*, 1, \ldots, n\}$. Analogously, the subword of $w_1^s \otimes w_2^s$ from z_j to z_{j+1} is the word $v'_{i_{j+1}} \diamond_{r',i_{j+1}}$ such that $v'_{i_{j+1}} \in \mathsf{Extended}(v_{i_{j+1}})$ and $r' \in \{*, 1, \ldots, n\}$. Therefore, the matching rule:

$$\eta_1(y_j, y_j^{\eta_1}), \eta_2(z_j, z_j^{\eta_2}), \eta(y_j^{\eta_1}, z_j^{\eta_2}), u'_{i_{j+1}}(y_j, y^*), \diamond_{i_{j+1},r}(y^*, y_{j+1}), \eta_1(y_{j+1}, y_{j+1}^{\eta_1}), \\ v'_{i_{j+1}}(z_j, z^*), \diamond_{r',i_{j+1}}(z^*, z_{j+1}), \eta_2(z_{j+1}, z_{j+1}^{\eta_2}) \;\rightarrow\; \eta(y_{j+1}^{\eta_1}, z_{j+1}^{\eta_2})$$

is triggered, and the atom $\eta(\bot_{y_{j+1}^{\eta_1}}, \bot_{z_{j+1}^{\eta_2}})$ is generated. This completes the proof of the claim.

From the above discussion, we conclude that $\eta(\bot_{y_m^{\eta_1}}, \bot_{z_m^{\eta_2}}) \in \mathsf{chase}(q[w_1^s \otimes w_2^s], \Sigma)$. We proceed to show that the finalization rule will be triggered and generate an isomorphic copy S of $D[q]$ with $\mathsf{dom}(S) = \{\bot_y, \bot_z, \bot_{x_1}, \ldots, \bot_{x_{2(n^2+2n+2)}}\}$. We assume that v_{i_m} is shorter than u_{i_m} (the other case is analogous). By the construction of $w_1^s \otimes w_2^s$, and the definition of $\bot_{y_m}$ and $\bot_{z_m}$, the subword of $w_1^s \otimes w_2^s$ from y_m to the last position of $w_1^s \otimes w_2^s$ is the word $u'_{i_m} \diamond_{i_m,r}$ such that $u'_{i_m} \in \mathsf{Extended}(u_{i_m})$ and $r \in \{*, 1, \ldots, n\}$. Analogously, the subword of $w_1^s \otimes w_2^s$ from z_m to the last position of $w_1^s \otimes w_2^s$ is the word $v_{i_m} \diamond_{r',i_m}$ such that $r' \in \{*, 1, \ldots, n\}$. We conclude that $r = r' = i_m$. Therefore, the finalization rule:

$$\mathsf{start}(y, x), \eta_1(y_m, y_m^{\mu_1}), \eta_2(z_m, z_m^{\mu_2}), \eta(y_m^{\mu_1}, z_m^{\mu_2}), u'_{i_m}(y_m, x'_1), v_{i_m}(z_m, x'_1), \diamond_{i_m,i_m}(x'_1, x_1), \\ a(x_1, x_2), \ldots, a(x_{2(n^2+2n+2)-1}, x_{2(n^2+2n+2)}), \mathsf{end}(x_{2(n^2+2n+2)}, z) \;\rightarrow\; q,$$

is triggered, and generates in $\mathsf{chase}(q[w_1^s \otimes w_2^s], \Sigma)$ an isomorphic copy S of $D[q]$, where $\mathsf{dom}(S) = \{\bot_y, \bot_z, \bot_{x_1}, \ldots, \bot_{x_{2(n^2+2n+2)}}\}$, as needed.

We now prove (2). Assume there is an isomorphic copy S of $D[q]$, where $\mathsf{dom}(S) = \{\bot_y, \bot_z, \bot_{x_1}, \ldots, \bot_{x_{2(n^2+2n+2)}}\}$, in $\mathsf{chase}(q[w], \Sigma)$. This implies that the finalization rule was triggered during the construction of $\mathsf{chase}(q[w], \Sigma)$. Let w_1 be the word that is obtained from w by (i) removing each occurrence of a symbol of the form $\diamond_{*,i}$, and (ii) changing each occurrence of a symbol of the form $\diamond_{i,j}$ by $\diamond_{i,*}$. Analogously, we define w_2 to be the word that is obtained from w by (i) removing each

occurrence of a symbol of the form $\diamond_{i,*}$, and (ii) changing each occurrence of a symbol of the form $\diamond_{i,j}$ by $\diamond_{*,j}$. We claim that w_1 and w_2 are of the following form:

$$w_1 \;=\; u_{i_1}\diamond_{i_1,*}\ldots u_{i_m}\diamond_{i_m,*} \qquad \text{and} \qquad w_2 \;=\; v_{i_1}\diamond_{*,i_1}\ldots v_{i_m}\diamond_{*,i_m},$$

for some sequence of indices $1 \leq i_1, \ldots, i_m \leq n$. Towards a contradiction, assume that this is not the case. Then, starting from the initialization rule, there would be no way for the matching rules to go through the path from $\bot_x$ to $\bot_{x_1}$, in the canonical instance of $q[w]$, in order to force the application of a finalization rule. Therefore, $i_1, \ldots, i_m$ represents a solution to the PCP instance. □

Having Lemma A.3 in place, we are now ready to conclude the correctness proof, i.e., show that the PCP instance has a solution iff q is semantically in GHW_1 under Σ.

($\Rightarrow$) Assume the PCP instance has a solution s. We claim that $q \equiv_\Sigma q[w_1^s \otimes w_2^s]$, which in turn implies that q is semantically in GHW_1 under Σ since $q[w_1^s \otimes w_2^s] \in \mathsf{GHW}_1$. By Lemma 2.5, we need to show that (i) $D[q[w_1^s \otimes w_2^s]]$ can be homomorphically mapped to $\mathsf{chase}(q, \Sigma)$, and (ii) $D[q]$ can be homomorphically mapped to $\mathsf{chase}(q[w_1^s \otimes w_2^s], \Sigma)$. The first item of Claim A.1 implies that there exists a path in q labeled

$$\mathsf{start}\; w_1 \otimes w_2 \quad \underbrace{aaa\ldots a}_{2(n^2+2n+2)\text{ times}} \quad \mathsf{end}.$$

In particular, this path starts at y, moves via an edge labeled start to an arbitrary node x_i, for $1 \leq i \leq 2(n^2 + 2n + 2)$, then follows a path labeled

$$w_1 \otimes w_2 \quad \underbrace{aaa\ldots a}_{2(n^2+2n+2)\text{ times}}$$

through the nodes $\{x_1, \ldots, x_{2(n^2+2n+2)}\}$, and then finishes by moving through an edge labeled end to the node z. Since each node of the form x_i, for $1 \leq i \leq 2(n^2 + 2n + 2)$, has outgoing edges labeled η_1 and η_2 in q, we conclude that the canonical instance of $q[w_1^s \otimes w_2^s]$ can be homomorphically mapped to $D[q]$, and thus, to $\mathsf{chase}(q, \Sigma)$. On the other hand, by Lemma A.3, $\mathsf{chase}(q[w_1^s \otimes w_2^s], \Sigma)$ contains an isomorphic copy of $D[q]$, which immediately implies that $D[q]$ can be homomorphically mapped to $\mathsf{chase}(q[w_1^s \otimes w_2^s], \Sigma)$.

($\Leftarrow$) Assume that there is an acyclic CQ q' such that $q \equiv_\Sigma q'$. To show that the PCP instance has a solution, it suffices to show that q' contains a subquery of the form $q[w]$, where w is a word over the alphabet $\{a, b, (\diamond_{i,*})_{1\leq i\leq n}, (\diamond_{*,i})_{1\leq i\leq n}, (\diamond_{i,j})_{1\leq i,j\leq n}\}$, such that $\mathsf{chase}(q[w], \Sigma)$ contains an isomorphic copy S of $D[q]$ with $\mathsf{dom}(S) = \{\bot_y, \bot_z, \bot_{x_1}, \ldots, \bot_{x_{2(n^2+2n+2)}}\}$. This allows us to apply the second item of Lemma A.3, and conclude that the PCP instance has a solution.

By Lemma 2.5, we get that $D[q]$ can be homomorphically mapped to $\mathsf{chase}(q', \Sigma)$, and $D[q']$ can be homomorphically mapped to $\mathsf{chase}(q, \Sigma)$. Since, by Claim A.2, $D[q] = \mathsf{chase}(q, \Sigma)$, we get that $D[q']$ can be homomorphically mapped to $D[q]$. We proceed to show the existence of $q[w]$ by case analysis on the shape of q'.

Case 1. We first assume that q' is a directed rooted tree. Since $D[q]$ can be mapped to $\mathsf{chase}(q', \Sigma)$, we get that $\mathsf{chase}(q', \Sigma)$ contains at least one edge labeled start and another one labeled end (because $D[q]$ contains such edges). Moreover, at least one edge of each such kind must have already been present in $D[q']$; otherwise, by definition of Σ, it is the case that $\mathsf{chase}(q', \Sigma)$ contains no such an edge, which is a contradiction. Moreover, an edge in $D[q]$ is labeled start if and only if it leaves the source node $\bot_y$, while an edge in $D[q]$ is labeled end if and only if it reaches the sink node $\bot_z$. Therefore, since there is a homomorphism from $D[q']$ to $D[q]$, an edge in $D[q']$ is labeled start if

and only it leaves the root element $\perp_r$. Moreover, the only nodes that can have incoming edges labeled end in $D[q']$ are the leaves.

We now perform a "backchase" analysis as follows. Let h be a homomorphism from $D[q]$ to chase(q', Σ). Then, $h(\perp_z)$ must correspond to a leaf $\perp_{z'}$ of $D[q']$ with an incoming edge labeled end. Correspondingly, for each $1 \leq i \leq 2(n^2 + 2n + 2)$, it must be the case that $h(\perp_{x_i})$ is a term of dom$(D[q'])$ that has (i) an outgoing edge labeled end pointing out to $\perp_{z'}$, and (ii) at least another outgoing edge labeled a. No such a node exists in $D[q']$ (since in $D[q']$ every node has at most one outgoing edge), and thus such edges must have been created during the chase. But the only way this could have happened is by triggering a finalization rule. Let us assume that a finalization rule

$$\text{start}(y, x), \eta_1(z_1, z_1'), \eta_2(z_2, z_2'), \eta(z_1', z_2'), u_i'(z_1, x_1'), v_i'(z_2, x_1'), \diamond_{i,i}(x_1', x_1), \\ a(x_1, x_2), \ldots, a(x_{2(n^2+2n+2)-1}, x_{2(n^2+2n+2)}), \text{end}(x_{2(n^2+2n+2)}, z) \rightarrow q,$$

where $1 \leq i \leq n$, u_i' in Extended(u_i), and v_i' in Extended(v_i), is triggered for the first time before any other finalization rule. This means that the unique path in $D[q']$ from the root $\perp_r$ to the leaf $\perp_{z'}$ is of the form:

$$\perp_r \xrightarrow{w'} \perp_{z_2} \xrightarrow{v_i''} \perp_{z_1} \xrightarrow{u_i \diamond_{i,i}} \perp_{x_1} \xrightarrow{a} \perp_{x_2} \ldots \perp_{x_{2(n^2+2n+2)-1}} \xrightarrow{a} \perp_{x_{2(n^2+2n+2)}} \xrightarrow{\text{end}} \perp_{z'},$$

assuming without loss of generality that u_i is shorter than v_i and that $v_i' = v_i'' u_i$. But then, all the atoms in such a path must have been already in $D[q']$. This is because the initialization and recursive matching rules can only generate atoms labeled η, and hence, none of these atoms could have been generated by the chase before a finalization rule is applied. For the same reasons, $D[q']$ also contains the atoms $\eta_1(\perp_{z_1}, \perp_{z_1'})$ and $\eta_2(\perp_{z_2}, \perp_{z_2'})$. But the atom $\eta(\perp_{z_1'}, \perp_{z_2'})$ also belongs to chase(q', Σ), and this atom could have only been generated during the chase. Assume otherwise, i.e., $\eta(\perp_{z_1'}, \perp_{z_2'}) \in D[q']$. Then $D[q']$ contains a path from $\perp_{z_2}$ to $\perp_{z_1}$ labeled v_i'', and another one labeled $\eta_1 \eta(\eta_2)^-$, where $(\eta_2)^-$ represents a backward traversal of an edge labeled η_2. These paths are disjoint, apart from the external nodes. This contradicts the fact that $D[q']$ is acyclic. Therefore, $\eta(\perp_{z_1'}, \perp_{z_2'})$ was either generated by the initialization rule, or the recursive matching rules. By recursively applying this reasoning, we get that the unique path from $\perp_r$ to $\perp_{z'}$ in $D[q']$ is

$$\perp_r \xrightarrow{\text{start}} \perp_y \xrightarrow{w} \perp_{x_1} \xrightarrow{a} \perp_{x_2} \ldots \perp_{x_{2(n^2+2n+2)-1}} \xrightarrow{a} \perp_{x_{2(n^2+2n+2)}} \xrightarrow{\text{end}} \perp_{z'},$$

where w is a word over the alphabet $\{a, b, (\diamond_{i,*})_{1 \leq i \leq n}, (\diamond_{*,i})_{1 \leq i \leq n}, (\diamond_{i,j})_{1 \leq i,j \leq n}\}$. Moreover, we can assume, without loss of generality, that each term in the path from $\perp_x$ to $\perp_{x_1}$ with an incoming edge labeled over $\{(\diamond_{i,*})_{1 \leq i \leq n}, (\diamond_{*,i})_{1 \leq i \leq n}, (\diamond_{i,j})_{1 \leq i,j \leq n}\}$ has outgoing edges labeled η_1 and η_2 in $D[q']$. Therefore, such a path, with all such outgoing edges, forms a subquery of q' of the form $q[w]$, where w is a word over the alphabet $\{a, b, (\diamond_{i,*})_{1 \leq i \leq n}, (\diamond_{*,i})_{1 \leq i \leq n}, (\diamond_{i,j})_{1 \leq i,j \leq n}\}$, as needed.

Case 2. Consider now the case where q' contains parallel edges (in both directions). Notice first that $D[q']$ cannot contain parallel edges labeled with two different symbols from $\{a, b, (\diamond_{i,*})_{1 \leq i \leq n}, (\diamond_{*,i})_{1 \leq i \leq n}, (\diamond_{i,j})_{1 \leq i,j \leq n}\}$. This is because $D[q']$ maps to $D[q]$, and, by the second item of Claim A.1, $D[q]$ contains no such edges. Therefore, if $D[q']$ contains any parallel edges, it must be the case that all of them, apart from at most one, are labeled η_1, η_2 or η.

The problem with our "backchase" analysis now is that $D[q']$ might contain "artificial" matching edges labeled η, which are not generated by the chase, but are used to "cheat" such chase. For instance, such edges might force an "artificial" triggering of the termination rules, thus not allowing us to conclude (as in the previous case) that q' contains a subquery $q[w]$ of the required form.

We can however solve this technical problem by assuming, without loss of generality, that the input to the PCP instance consists of words of even length only (if not, we simply replace each occurrence of a letter a or b in one of the words of the PCP instance by aa or bb, respectively). In fact,

if one of these edges is used to trigger a matching rule, then all pairs of "matched" positions from then on occur at odd distance from each other, thus forbidding the possibility of synchronization at the end of the path.

Case 3. Let us assume now that q' also admits self-loops. Since $D[q']$ homomorphically maps to $D[q]$, these loops can only be labeled η_1, η_2 or η (see item (3) in Claim A.1). This is not dangerous for our "backchase" analysis since, if one of those loops is used as a starting point for a chase sequence, it can only mean that the synchronization of the words in the PCP instance occurs earlier than expected. Therefore, there is still a solution to the PCP instance.

Case 4. Finally, the case when the query q' has, in addition, disconnected components is analogous, since we can still carry out the previous analysis over one of the connected components of q'. This finishes the proof.

B ADDITIONAL PROOFS FROM SECTION 7

Theorem 7.1. $\mathsf{SemGHW}_1(\mathbb{EGD})$ *is undecidable, even if we allow only unary and binary predicates.*

We adapt the proof of Theorem 4.4. In particular, we replace the initialization rule:

$$\mathrm{start}(x, y), \eta_1(y, z), \eta_2(y, w) \rightarrow \eta(z, w), \eta(w, z).$$

by the initialization EGD:

$$\mathrm{start}(x, y), \eta_1(y, z), \eta_2(y, w) \rightarrow z = w,$$

and each matching rule of the form:

$$\eta_1(x_1, x), \eta_2(y_1, y), \eta(x, y), u_i'(x_1, x^*), \diamond_{i,r}(x^*, x_2), \eta_1(x_2, x'), \\ v_i'(y_1, y^*), \diamond_{r',i}(y^*, y_2), \eta_2(y_2, y') \rightarrow \eta(x', y').$$

by a matching EGD of the form:

$$\eta_1(x_1, x), \eta_2(y_1, x), u_i'(x_1, x^*), \diamond_{i,r}(x^*, x_2), \eta_1(x_2, x'), v_i'(y_1, y^*), \diamond_{r',i}(y^*, y_2), \eta_2(y_2, y') \rightarrow x' = y'.$$

That is, we simply mimic the "matching" atom $\eta(x, y)$ by equating x and y. For the finalization rules the modifications are a bit more cumbersome to describe. In fact, in order to emulate with a set of EGDs a finalization rule of the form:

$$\mathrm{start}(y, x), \eta_1(z_1, z_1'), \eta_2(z_2, z_2'), \eta(z_1', z_2'), u_i'(z_1, x_1'), v_i'(z_2, x_1'), \diamond_{i,i}(x_1', x_1), \\ a(x_1, x_2), \ldots, a(x_{2(n^2+2n+2)-1}, x_{2(n^2+2n+2)}), \mathrm{end}(x_{2(n^2+2n+2)}, z) \rightarrow q,$$

we have to add "dangling" edges to the body of the rule. In case such a body is satisfied, the finalization EGDs will force several of the variables in such dangling edges to be equated, thus generating as before a copy of q among the corresponding variables. Since the idea is easy to grasp, yet quite tedious to write the actual EGDs, we keep the explanation at the intuitive level.

Let $\Sigma^=$ be the resulting set of EGDs. It is easy to see that if the PCP instance has a solution s, then $q \equiv_\Sigma q[w_1^s \otimes w_2^s]^=$, where $q[w_1^s \otimes w_2^s]^=$ is a suitable modification of $q[w_1^s \otimes w_2^s]$ with the dangling edges. It follows that q is semantically in GHW_1 under $\Sigma^=$ since $q[w_1^s \otimes w_2^s]^= \in \mathsf{GHW}_1$.

Assume, on the other hand, that there is an acyclic CQ q' such that $q \equiv_{\Sigma^=} q'$. As before, we show that q' must contain a subquery of the form $q[w]$, where w is a word over the alphabet $\{a, b, (\diamond_{i,*})_{1 \le i \le n}, (\diamond_{*,i})_{1 \le i \le n}, (\diamond_{i,j})_{1 \le i,j \le n}\}$, such that $\mathrm{chase}(q[w], \Sigma)$ contains a copy of q. This allows us to apply the second item of Lemma A.3 and obtain that the PCP instance has a solution. The key ingredient to our proof is, as before, a "backchase" analysis. In particular, it is necessary to show that every time that the sink nodes of two dangling edges labeled η_1 and η_2, respectively, are

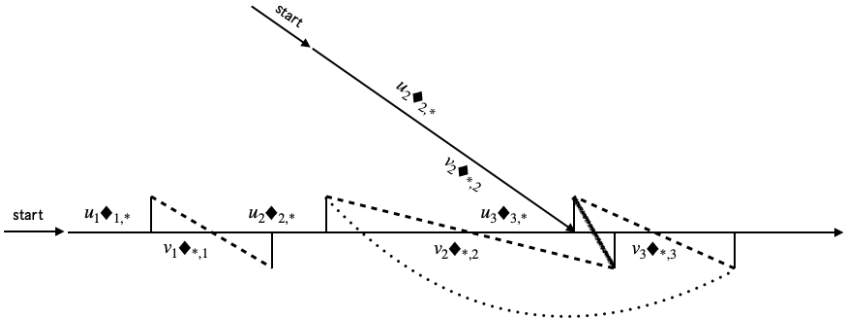


Fig. 13. A part of CQ q showing variables y, z, and how variable x_i is connected to all variables x_j, with $i < j$. We omit relation symbols η, η_1, and η_2 from the figure, as all pairs (x_i, x_j) are connected with an edge labeled with all such symbols.

equated, it is due to an application of a matching rule. Notice, however, that this is not straightforward to achieve since the equality of variables generated by EGDs is transitive, and thus, such variables could have been equated for transitivity reasons. An example of this phenomenon is depicted in Figure 13, where we have a "blurred" equality being generated from the upper path and three "dashed" equalities generated from the lower one (vertical strokes are dangling edges). The transitivity of the equalities generated by the EGDs forces two variables to be equated with no justification due to the matching rules (this is the "dotted" equality). This might cause an artificial triggering of the matching rules ending in the firing of the finalization rule even when no solution to the PCP instance exists.

This problem, however, only appears when two paths converge as shown in Figure 13. Let us then examine this phenomenon in more detail. So, let us assume that q' contains two such paths that converge in variable z via edges from different variables x and y. If the labels of these two edges are the same, then the issue can easily be solved by extending $\Sigma^=$ with each EGD of the form

$$R(x,z), R(y,z) \;\rightarrow\; x = y, \quad \text{for } R \in \{a, b, (\diamond_{k,*})_{1\le k\le n}, (\diamond_{*,k})_{1\le k\le n}, (\diamond_{k,\ell})_{1\le k,\ell\le n}\}.$$

In fact, each one of these EGDs is satisfied by the query q, and, in addition, if q' satisfies the extended set $\Sigma^=$ then it means that no two paths can converge via two edges with the same label in the set $\{a, b, (\diamond_{k,*})_{1\le k\le n}, (\diamond_{*,k})_{1\le k\le n}, (\diamond_{k,\ell})_{1\le k,\ell\le n}\}$.

Let us consider then the case when the two edges have different labels. It is easy to see that this leads to problems with transitivity only if the labels of these edges are of the form $\diamond_{k,r}$ and $\diamond_{k',r'}$, or of the form $\diamond_{r,k}$ and $\diamond_{r',k'}$, for $1 \le k < k' \le n$ and $r, r' \in \{*, 1, \ldots, n\}$. In fact, consider for instance that one of the edges is labeled $\diamond_{r,k}$ and the other one is labeled a. Then the variable z is a synchronization variable only for one of the two paths (namely, the one which enters z via the edge labeled $\diamond_{k,r}$), and thus no interaction with the other path can cause undesired equalities of variables. Consider now the case when one of the labels is of the form $\diamond_{k,r}$ and the other one is of the form $\diamond_{r',k'}$, for $1 \le k < k' \le n$ and $r, r' \in \{*, 1, \ldots, n\}$. Then the path that enters z via an edge labeled $\diamond_{k,r}$ uses an edge labeled η_1 for synchronization, while the other path uses an edge labeled η_2. Hence, no interaction between these two paths can cause undesired inequalities. All other cases are similar.

Consider then the case when the two edges that converge in z are of the form $\diamond_{k,r}$ and $\diamond_{k',r'}$, for $1 \le k < k' \le n$ and $r, r' \in \{*, 1, \ldots, n\}$, as the other case is analogous. An easy way to solve this problem is by using different dangling edges η_1^k and η_2^k, for each $1 \le k \le n$. Hence, now the element z will have two different dangling edges η_1^k and $\eta_1^{k'}$: the first one corresponding to the

incoming edge labeled $\diamond_{k,r}$ and the second one to the incoming edge labeled $\diamond_{k',r'}$. The dangling edge labeled η_1^k can only "match" with another dangling edge labeled η_2^k, while the one labeled $\eta_1^{k'}$ can only match with another dangling edge labeled $\eta_2^{k'}$. This prevents undesired inequalities to arise, as we can now ensure that each edge labeled η_1^k is "matched" with at most one dangling edge labeled η_2^k. The cost to be paid is that we now have to extend our query q with extra symbols η_1^k and η_2^k, for each $1 \leq k \leq n$, interpreted exactly as η_1 and η_2 before. We also need to adapt the matching and finalization rules in $\Sigma^=$; e.g., now matching rules have to use η_1^k and η_2^k as opposed to simply η_1 and η_2 in the following way:

$$\eta_1^{k'}(x_1, x), \eta_2^{k'}(y_1, x), u'_k(x_1, x^*), \diamond_{k,r}(x^*, x_2), \eta_1^k(x_2, x'), v'_k(y_1, y^*), \diamond_{r',k}(y^*, y_2), \eta_2^k(y_2, y') \rightarrow x' = y'.$$

C ADDITIONAL PROOFS FROM SECTION 9

Proposition 9.9. *$\mathbb{G}$ enjoys chase redundancy.*

The first part of the proof has been already given in the main body of the paper. It only remains to show that there are functions $(\mu_j : U_{I_j,k} \rightarrow H_{I_j \rightarrow I})_{j \geq 0}$ such that, for each $j \geq 0$:

(†) the function μ_j is a winning strategy for the duplicator in the existential k-cover game on $(I_j, \bot(\bar{x}))$ and $(I, \bar{t})$, and
(††) for each $J \in U_{I_j,k}$, $\mu_j(J) = \mu_{j+1}(J)$.

We proceed by induction on $j \geq 0$.

Base case. For $j = 0$, we have that $D[q] = I_0$. Thus, we can define μ_0 to be μ.

Inductive step. By induction hypothesis, there exists $\mu_j : U_{I_j,k} \rightarrow H_{I_j \rightarrow I}$ that is a winning strategy for the duplicator in the existential k-cover game on $(I_j, \bot(\bar{x}))$ and $(I, \bar{t})$. For each $J \in (U_{I_j,k} \cap U_{I_{j+1},k})$, let $\mu_{j+1}(J) = \mu_j(J)$. We explain next how to define $\mu_{j+1}(J)$, for each $J \in U_{I_{j+1},k} \setminus U_{I_j,k}$. Recall that I_{j+1} is the result of applying the TGD τ_j over I_j with $(\bar{u}_j, \bar{u}'_j)$. Assume that τ_j is of the form $\phi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z}\, \psi(\bar{x}, \bar{z})$. This implies that $I_{j+1} = I_j \cup \psi(\bar{u}_j, \bot(\bar{z}))$, where $\psi(\bar{u}_j, \bot(\bar{z}))$ is the set of atoms obtained after replacing $\bar{x}$ with $\bar{u}_j$, and each variable $z \in \bar{z}$ with a fresh null $\bot_z$.

Suppose that the guard of τ_j is $R(\bar{x}, \bar{y})$. Since $\phi(\bar{u}_j, \bar{u}'_j) \subseteq I_j$, we have that $R(\bar{u}_j, \bar{u}'_j) \in I_j$, and, therefore, μ_j is defined over the 1-union $\{R(\bar{u}_j, \bar{u}'_j)\}$ of I_j. Consider an arbitrary homomorphism $h \in \mu_j(\{R(\bar{u}_j, \bar{u}'_j)\})$. We claim that $\phi(h(\bar{u}_j), h(\bar{u}'_j)) \subseteq I$. It is clear that $R(h(\bar{u}_j), h(\bar{u}'_j)) \in I$. Consider any other atom $S(\bar{x}')$ in the body of τ_j. Since τ_j is guarded, all the variables in $S(\bar{x}')$ occur also in the guard atom $R(\bar{x}, \bar{y})$ of τ_j, i.e., $\bar{x}' \subseteq \bar{x} \cup \bar{y}$. Let $\bar{v}_j$ be the reduction of the tuple $(\bar{u}_j, \bar{u}'_j)$ that corresponds to the positions associated with $\bar{x}'$. Since $S(\bar{v}_j) \in I_j$, μ_j is defined over the 1-union $\{S(\bar{v}_j)\}$ of I_j. By the second condition in the definition of the winning strategy, there exists $h' \in \mu_j(\{S(\bar{v}_j)\})$ that is consistent with h. Since $S(h'(\bar{v}_j)) \in I$ and $S(h(\bar{v}_j)) = S(h'(\bar{v}_j))$, we conclude that $S(h(\bar{v}_j)) \in I$. Therefore, $\phi(h(\bar{u}_j), h(\bar{u}'_j)) \subseteq I$, as claimed above. Recall now that $I \models \Sigma$, which means that I satisfies τ_j. Thus, $\psi(h(\bar{u}_j), \bar{w}_h) \subseteq I$, where $\bar{w}_h$ is a tuple of terms of $\mathrm{dom}(I)$. In other words, every homomorphism $h \in \mu_j(\{R(\bar{u}_j, \bar{u}'_j)\})$ naturally gives rise to a homomorphism h^{ext} from $\psi(\bar{u}_j, \bot(\bar{z})) \subseteq I_{j+1}$ to $\psi(\bar{u}_j, \bar{w}_h) \subseteq I$. In particular, h^{ext} agrees with h over the terms of $\bar{u}_j$, and maps the nulls of $\bot(\bar{z})$ to their corresponding terms in $\bar{w}_h$. Clearly, h and h^{ext} are consistent.

Consider now an arbitrary $J \in U_{I_{j+1},k} \setminus U_{I_j,k}$. Observe that J can be partitioned into $\{J_1, J_2\}$, where $J_1 = J \cap I_j$ and $J_2 = J \setminus J_1$. Notice that J_2 is non-empty; otherwise, J would be a k-union of I_j. Therefore, $|J_1| \leq k - 1$, which implies that $(J_1 \cup \{R(\bar{u}_j, \bar{u}'_j)\}) \in U_{I_j,k}$. By the second condition in the definition of the winning strategy, if $h \in \mu_j((J_1 \cup \{R(\bar{u}_j, \bar{u}'_j)\}))$, then the restriction of h over $\mathrm{dom}(J_1)$, denoted $h_{|J_1}$, belongs to $\mu_j(J_1)$. Analogously, the restriction of h over the set of terms occurring in $R(\bar{u}_j, \bar{u}'_j)$, denoted $h_{|R(\bar{u}_j, \bar{u}'_j)}$, belongs to $\mu_j(\{R(\bar{u}_j, \bar{u}'_j)\})$. Consequently, from what we

discussed above, $h_{|R(\bar{u}_j,\bar{u}'_j)}$ gives rise to a homomorphism $(h_{|R(\bar{u}_j,\bar{u}'_j)})^{\text{ext}}$ from $\psi(\bar{u}_j, \bot(\bar{z}))$ to I, which is consistent with $h_{|R(\bar{u}_j,\bar{u}'_j)}$. As usual, we write $((h_{|R(\bar{u}_j,\bar{u}'_j)})^{\text{ext}})_{|J_2}$ for the restriction of $(h_{|R(\bar{u}_j,\bar{u}'_j)})^{\text{ext}}$ over $\text{dom}(J_2)$. Notice that $((h_{|R(\bar{u}_j,\bar{u}'_j)})^{\text{ext}})_{|J_2}$ is consistent with $h_{|J_1}$, since each term of $\text{dom}(J_1) \cap \text{dom}(J_2)$ belongs to $(\bar{u}_j, \bar{u}'_j)$. Therefore,

$$\lambda^h \;=\; h_{|J_1} \;\cup\; \left(\left(h_{|R(\bar{u}_j,\bar{u}'_j)}\right)^{\text{ext}}\right)_{|J_2}$$

is a well-defined homomorphism from $J = J_1 \cup J_2$ to I. We then define

$$\mu_{j+1}(J) \;=\; \{\lambda^h \mid h \in \mu_j(J_1 \cup \{R(\bar{u}_j, \bar{u}'_j)\})\}.$$

We proceed to show that μ_{j+1} satisfies (†) and (††). In fact, the condition (††), which states that $\mu_j(J) = \mu_{j+1}(J)$ for each $J \in U_{I_j,k}$, follows by construction since $(U_{I_j,k} \cap U_{I_{j+1},k}) = U_{I_j,k}$. It remains to show that (†) holds, that is, μ_{j+1} is a winning strategy for the duplicator in the existential k-cover game on $(I_{j+1}, \bot(\bar{x}))$ and $(I, \bar{t})$. Equivalently, we need to show that, for each $J \in U_{I_{j+1},k}$:

(1) $\mu_{j+1}(J) = \{h \in H_{I_{j+1}\to I} \mid h(J) \subseteq I \text{ and } \bot_{x_i} \in \text{dom}(J) \implies h(\bot_{x_i}) = t_i\}$, and
(2) for each $h \in \mu_{j+1}(J)$ and $J' \in U_{I_{j+1},k}$, there exists $h' \in \mu_{j+1}(J')$ such that h, h' are consistent.

We first prove that (1) holds. Fix an arbitrary $J \in U_{I_{j+1},k}$. We proceed by considering two cases:

- $J \in U_{I_j,k}$. Clearly, $\mu_j(J) = \mu_{j+1}(J)$, and the claim follows by induction hypothesis.
- $J \in U_{I_{j+1},k} \setminus U_{I_j,k}$. Clearly, J can be partitioned into $\{J_1, J_2\}$, where $J_1 = J \cap I_j$ and $J_2 = J \setminus J_1$. By construction, $\mu_{j+1} = \{\lambda^h \mid h \in \mu_j(J_1 \cup \{R(\bar{u}_j, \bar{u}'_j)\})\}$, where $\lambda^h = h_{|J_1} \cup ((h_{|R(\bar{u}_j,\bar{u}'_j)})^{\text{ext}})_{|J_2}$. Fix an arbitrary $\lambda^h \in \mu_{j+1}(J)$. As explained above, $\lambda^h(J) \subseteq I$. Assume now that $\bot_{x_i} \in \text{dom}(J)$. If $\bot_{x_i} \in \text{dom}(J_1)$, then, by induction hypothesis, $\lambda^h(\bot_{x_i}) = t_i$ since $h_{|J_1} \in \mu_j(J_1)$. Now, if $\bot_{x_i} \in \text{dom}(J_2) \setminus \text{dom}(J_1)$, then it necessarily belongs to $\bar{u}_j$ since the terms $\bot(\bar{z})$ are nulls generated by the chase procedure. Since $\lambda' = ((h_{|R(\bar{u}_j,\bar{u}'_j)})^{\text{ext}})_{|J_2}$ is consistent with h, we have that $\lambda'(\bot_{x_i}) = h(\bot_{x_i})$. By induction hypothesis, $h(\bot_{x_i}) = \bar{t}_i$, and thus, $\lambda^h(\bot_{x_i}) = \bar{t}_i$.

We finally show that (2) holds. Fix arbitrary $J, J' \in U_{I_{j+1},k}$ and $\lambda \in \mu_{j+1}(J)$. We show that there is $\lambda' \in \mu_{j+1}(J')$ that is consistent with λ by considering the following cases:

- $J, J' \in U_{I_j,k}$. The claim follows by induction hypothesis.
- $J \in U_{I_j,k}$ and $J' \in U_{I_{j+1},k}$. Clearly, $\lambda \in \mu_j(J)$ since J is a k-union of I_j. As usual, J' can be partitioned into $\{J'_1, J'_2\}$, where $J'_1 = J' \cap I_j$ and $J'_2 = J' \setminus J'_1$. Notice that $J'_2 \neq \varnothing$, which implies that $|J'_1| \leq k-1$. Thus, $J' \cup \{R(\bar{u}_j, \bar{u}'_j)\}$ is a k-union of I_j. By induction hypothesis and the second condition in the definition of the witness strategy, there exists a homomorphism $h \in \mu_j(J'_1 \cup \{R(\bar{u}_j, \bar{u}'_j)\})$ that is consistent with λ. By definition, the homomorphism $\lambda^h = h_{|J'_1} \cup ((h_{|R(\bar{u}_j,\bar{u}'_j)})^{\text{ext}})_{|J'_2}$ belongs to $\mu_{j+1}(J')$. Observe that λ and λ^h are consistent since every term that is shared by J and J'_2 occurs in $(\bar{u}_j, \bar{u}'_j)$. The claim follows with $\lambda' = \lambda^h$.
- $J \in U_{I_{j+1},k}$ and $J' \in U_{I_j,k}$. It suffices to show that there exists $\lambda' \in \mu_j(J')$ that is consistent with λ. We partition J into $\{J_1, J_2\}$, where $J_1 = J \cap I_j$ and $J_2 = J \setminus J_1$. Observe that $|J_1| \leq k-1$, and thus, $J_1 \cup \{R(\bar{u}_j, \bar{u}'_j)\}$ is a k-union of I_j. Clearly, λ is of the form $\lambda^h = h_{|J_1} \cup ((h_{|R(\bar{u}_j,\bar{u}'_j)})^{\text{ext}})_{|J_2}$ for some $h \in \mu_j(J_1 \cup \{R(\bar{u}_j, \bar{u}'_j)\})$. By induction hypothesis and the second condition in the definition of the witness strategy, there exists a homomorphism $\chi \in \mu_j(J')$ that is consistent with h. It is easy to see that $\chi_{|J_1} \in \mu_j(J)$ is consistent with λ^h since J' and J_2 share only terms of $(\bar{u}_j, \bar{u}'_j)$. The claim follows with $\lambda' = \chi_{|J_1}$.
- $J, J' \in U_{I_{j+1},k} \setminus U_{I_j,k}$. As above, J and J' can be partitioned into $\{J_1, J_2\}$ and $\{J'_1, J'_2\}$, respectively, such that $J_1 = J \cap I_j$ and $J_2 = J \setminus J_1$, and $J'_1 = J' \cap I_j$ and $J'_2 = J' \setminus J'_1$. It is easy to verify that $|J_1|, |J'_1| \leq k-1$, and thus, $J_1 \cup \{R(\bar{u}_j, \bar{u}'_j)\}$ and $J'_1 \cup \{R(\bar{u}_j, \bar{u}'_j)\}$ are k-unions of I_j. Clearly, λ

is of the form $\lambda^h = h_{|J_1} \cup ((h_{|R(\bar{u}_j,\bar{u}'_j)})^{\text{ext}})_{|J_2}$ for some $h \in \mu_j(J_1 \cup \{R(\bar{u}_j, \bar{u}'_j)\})$. By induction hypothesis and the second condition in the definition of the witness strategy, there exists a homomorphism $\chi \in \mu_j(J')$ that is consistent with h. By definition, the homomorphism $\lambda^\chi = \chi_{|J_1} \cup ((\chi_{|R(\bar{u}_j,\bar{u}'_j)})^{\text{ext}})_{|J_2}$ belongs to $\mu_{j+1}(J')$. It is easy to verify that λ^h and λ^χ are consistent. The claim follows with $\lambda' = \lambda^\chi$.

D ADDITIONAL PROOFS FROM SECTION 10

Lemma 10.3. *Let $q, q' \in \mathrm{GHW}_k$ over a schema σ, and $\Sigma \in \mathbb{C}$ over σ, where $\mathbb{C} \in \{\mathbb{G}, \mathbb{NR}, \mathbb{S}\}$, such that $q' \subseteq_\Sigma q$. There is a CQ $q'' \in \mathrm{GHW}_k$ over σ such that $q' \subseteq_\Sigma q'' \subseteq_\Sigma q$ and $|q''| \leq g_{\mathbb{C}}(q, \Sigma)$.*

Assume first that $\mathbb{C} = \mathbb{G}$. We define $q'_\Sigma(\bar{x})$, where $\bar{x}$ are the free variables of q', as the CQ obtained by considering the conjunction of atoms in $\mathrm{chase}(q', \Sigma)$, after renaming each null $\perp$ into a variable $v(\perp)$, with $v(\perp_x) = x$ for each $x \in \bar{x}$. By exploiting Lemma 2.5, it is easy to show that $q' \subseteq_\Sigma q$ implies $q'_\Sigma \subseteq q$. Since $q' \in \mathrm{GHW}_k$ and $\mathbb{G}$ has GHW-preserving chase (by Proposition 5.8), we conclude that $q'_\Sigma \in \mathrm{GHW}_k$. Hence, by Lemma 5.4, there exists a CQ $q'' \in \mathrm{GHW}_k$ over σ such that $q'_\Sigma \subseteq q'' \subseteq q$ and $|q''| \leq |q| \cdot (2k+1)$. By using Lemma 2.5, we can show that $q'_\Sigma \subseteq q''$ implies $q' \subseteq_\Sigma q''$. Moreover, $q'' \subseteq q$ implies $q'' \subseteq_\Sigma q$, and thus, $q' \subseteq_\Sigma q'' \subseteq_\Sigma q$.

Assume now that $\mathbb{C} \in \{\mathbb{NR}, \mathbb{S}\}$. Since $\mathbb{C}$ is UCQ rewritable, there exists a UCQ Q over σ such that $q' \subseteq Q$. Thus, there exists a disjunct $\hat{q}$ of Q such that $q' \subseteq \hat{q}$. By Lemma 5.4, there exists a CQ $q'' \in \mathrm{GHW}_k$ over σ such that $q' \subseteq q'' \subseteq \hat{q}$ and $|q''| \leq |\hat{q}| \cdot (2k+1) \leq f_{\mathbb{C}}(q, \Sigma) \cdot (2k+1)$. It is clear that $q' \subseteq q''$ implies $q' \subseteq_\Sigma q''$. Moreover, as shown in the proof of Proposition 6.5, $\hat{q} \subseteq_\Sigma q$. Consequently, $q' \subseteq_\Sigma q'' \subseteq_\Sigma q$, as needed.

Received ; revised ; accepted