Semantic Optimization of Conjunctive Queries

Citation for published version:

Digital Object Identifier (DOI):
10.1145/3424908

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
Journal of the ACM

Publisher Rights Statement:
© ACM, 2020. This is the author's version of the work. It is posted here by permission of ACM for your personal use. Not for redistribution. The definitive version was published in Journal of the ACM, (67, 6, October 2020) http://doi.acm.org/10.1145/3424908

General rights
Copyright for the publications made accessible via the Edinburgh Research Explorer is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact openaccess@ed.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.
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 hypertree-width 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:

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2020 Copyright held by the owner/author(s). Publication rights licensed to ACM.
0004-5411/2020/9-ART $15.00
https://doi.org/0000001.0000001

J. ACM, Vol. 1, No. 1, Article . Publication date: September 2020.
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 \( \text{GHW}_k \) 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 \( \text{GHW}_1 \) 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 \( \Sigma \):

\[
\text{PROBLEM: } \text{Semantic Generalized HypertreeWidth } k
\]

\[
\text{INPUT: } \text{A CQ } q \text{ and a finite set } \Sigma \text{ of constraints.}
\]

\[
\text{QUESTION: } \text{Is there a CQ } q' \in \text{GHW}_k \text{ such that } q \equiv_{\Sigma} q'？
\]

We study the above problem, from now on abbreviated as \( \text{SemGHW}_k \), for the two most important classes of database constraints:

1. Tuple-generating dependencies (TGDs), i.e., expressions of the form \( \forall x \forall y (\phi(x, y) \rightarrow \exists z \psi(x, z)) \), where \( \phi \) and \( \psi \) are conjunctions of atoms. TGDs subsume the important class of referential integrity constraints (or inclusion dependencies).
Equality-generating dependencies (EGDs), i.e., expressions of the form $\forall \bar{x}(\phi(\bar{x}) \rightarrow y = z)$, where $\phi$ 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].

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

$$q(x, y) = \exists z (\text{Interest}(x, z) \land \text{Class}(y, z) \land \text{Owns}(x, y)),$$

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:

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

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

$$q'(x, y) = \exists z (\text{Interest}(x, z) \land \text{Class}(y, z)).$$

Notice that $q$ and $q'$ are in fact equivalent over every database that satisfies $\tau$. ■

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$_k$. 

J. ACM, Vol. 1, No. 1, Article . Publication date: September 2020.
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 $SemGHW_k$ 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\(^2\), (ii) non-recursive, and (iii) sticky sets of TGDs. 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 $N\text{ExpTime}$ upper bound and an $\text{ExpTime}$ lower bound. For a fixed schema, or even for bounded arity, the problem is $\text{NP}$-complete.

Let us stress that the $\text{NP}$ 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 $\text{GHW}_k$ reformulation, if one exists.

**Results for EGDs:** By adapting the proof for showing that $\text{SemGHW}_1$ under full TGDs is undecidable, we show that $\text{SemGHW}_1$ 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 $\text{SemGHW}_1$ under keys is $\text{NP}$-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 $\text{SemGHW}_k$ 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 $\text{SemGHW}_k$ 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 $\text{SemGHW}_k$ 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 $\text{GHW}_k$ under a set of constraints $\Sigma$ that falls in $\mathbb{C}$ over a database $D$ that satisfies the constraints:

1. Convert $q$ into a CQ $q'$ in $\text{GHW}_k$ that is equivalent to $q$ under $\Sigma$.  
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 $\text{GHW}_k$ 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 $\text{GHW}_k$ 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 $\text{GHW}_k$ 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 $\text{GHW}_k$ under a set $\Sigma$ of constraints, we can exploit our proof techniques for TGD-based constraints in order to compute an approximation of $q$ under $\Sigma$ [6], that is, a CQ $q'$ in $\text{GHW}_k$ that is maximally contained in $q$ under $\Sigma$. 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\textsubscript{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\textsubscript{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\textsubscript{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\textsubscript{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\textsubscript{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\textsubscript{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\textsubscript{k}. In Section 4 we investigate the frontiers of decidability for SemGHW\textsubscript{k} under TGDs. The property of generalized hypertreewidth preserving chase is studied in Section 5, while UCQ rewritable classes in Section 6. SemGHW\textsubscript{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\textsubscript{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 C, N and 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 \(\sigma\) 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 \(\sigma\) is an expression of the form \(R(i)\), where \(R/n\) is a relation symbol in \(\sigma\) and \(I\) is an \(n\)-tuple of terms. An instance over \(\sigma\) is a (possibly infinite) set of atoms over \(\sigma\) that contains constants and nulls, while a database over \(\sigma\) is simply a finite instance over \(\sigma\). For an instance \(I\) we write \(\text{dom}(I)\) 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
The second condition above is generally known as the connectedness condition. The connectedness condition is violated if there are terms \( v \) such that \( \chi(v) \) contains all the terms in \( I \).

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

\[
R(c_1, c_2), P(c_3, c_1, c_4), S(c_5, c_4, c_5, c_3), R(c_5, c_6), R(c_6, c_7), P(c_2, c_8, c_2), R(c_8, c_7),
\]

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

A generalized hypertree decomposition of \( I \) is a triple \((T, \chi, \lambda)\), where \((T, \chi)\) is a tree decomposition of \( I \), and, assuming \( T = (V, E) \), \( \lambda \) is a labeling function \( V \to 2^{\text{dom}(I)} \), i.e., it assigns a subset of \( \text{dom}(I) \) to each node of \( T \), such that:

1. For each \( R(i) \in I \), there exists \( v \in V \) such that \( \chi(v) \) contains all the terms in \( I \).
2. For each \( t \in \text{dom}(I) \), 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.

For \( k \geq 1 \), we denote by \( \text{GHW}_k \) 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 \( \text{GHW}_1 \) 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 \( \text{GHW}_2 \).
Conjunctive queries. A conjunctive query (CQ) over $\sigma$ is a formula of the form:

$$q(\bar{x}) := \exists \bar{y}(R_1(t_1) \land \cdots \land R_m(t_m)),$$

where each $R_i(t_i)$ ($1 \leq i \leq m$) is an atom that contains only variables of $V$ over $\sigma$, each variable mentioned in the $t_i$’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 $C \cup \mathbb{N}$ such that $R(h(t)) \in I$, for each atom $R(t)$ 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 $t$ 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 hypertreewidth $k \geq 1$ 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 hypertreewidth $k$. We slightly abuse notation and we write $\text{GHW}_k$ for the class of CQs of generalized hypertreewidth bounded by $k \geq 1$. It would be clear from the context when $\text{GHW}_k$ 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

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

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 $\text{GHW}_2$. 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 $q \notin \text{GHW}_1$, i.e., it is not acyclic. However, it is easy to verify that the CQ

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

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 $\text{GHW}_1$. 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 $q \in \text{GHW}_k$ 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 $t \in q(D)$, given a database $D$, a CQ $q(\bar{x}) \in \text{GHW}_k$, and a tuple $t \in \text{dom}(D)^{|\bar{x}|}$, can be solved in time $O(||D||^{k+1} \cdot ||q||)$.

\footnote{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 $\sigma$ is an expression

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

where $\phi$ and $\psi$ are conjunctions of atoms over $\sigma$ that contain only variables of $V$. 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 $\land$ for conjoining atoms. Further, we assume that each variable in $\bar{x}$ is mentioned in some atom of $\psi$. We call $\phi$ and $\psi$ 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 $\sigma$ satisfies this TGD if and only if $q_{\phi}(I) \subseteq q_{\psi}(I)$. An instance $I$ satisfies a set $\Sigma$ of TGDs, denoted $I \models \Sigma$, if $I$ satisfies every TGD in $\Sigma$. Let us clarify that we work with finite sets of TGDs. Henceforth, whenever we refer to a set $\Sigma$ of TGDs, it is implicit that $\Sigma$ is finite.

An equality-generating dependency (EGD) over $\sigma$ is an expression

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

where $\phi$ is a conjunction of atoms over $\sigma$ 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 $\phi$ the body of the EGD. An instance $I$ over $\sigma$ 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 $\Sigma$ of EGDs, denoted $I \models \Sigma$, if $I$ satisfies every EGD in $\Sigma$. 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 $\sigma$ is an expression of the form $R[A \rightarrow B]$, where $R/n$ is a relation symbol in $\sigma$, 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 $\sigma$ and $\tau$ a TGD of the form $\phi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z} \psi(\bar{x}, \bar{z})$ over $\sigma$. We say that $\tau$ is applicable with respect to $I$ if there exists a tuple $(i, i')$ of terms in $I$ such that $\phi(i, i')$ holds in $I$. In this case, the result of applying $\tau$ over $I$ with $(i, i')$ is the instance $J$ that extends $I$ with every atom in $\psi(i, i'')$, where $i''$ 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:

$$I \xrightarrow{\tau_i(i, i')} J.$$

Let $I$ be an instance and $\Sigma$ a set of TGDs. A chase sequence for $I$ under $\Sigma$ is a sequence:

$$I_0 \xrightarrow{\tau_{t_0}(t_0, t_0')} I_1 \xrightarrow{\tau_{t_1}(t_1, t_1')} I_2 \ldots$$

of chase steps such that:

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

J. ACM, Vol. 1, No. 1, Article . Publication date: September 2020.
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 $\Sigma$, denoted chase($I, \Sigma$).\footnote{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.}

Moreover, for a CQ $q$ we write chase($q, \Sigma$) for the instance chase($D[q], \Sigma$). The key property of the chase procedure is that the instance chase($I, \Sigma$) is universal, i.e., for every instance $I$ such that $I \supseteq I$ and $J \models \Sigma$, there is a homomorphism from chase($I, \Sigma$) to $J$\footnote{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).} [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 $\sigma$ and an EGD $\rho$ of the form $\phi(\bar{x}) \rightarrow x_i = x_j$ over $\sigma$. We say that $\rho$ 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 $\rho$ 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 $\Sigma$ of EGDs. Such a non-failing sequence is finite and unique (up to null renaming); thus, we refer to the chase for $I$ under $\Sigma$, denoted chase($I, \Sigma$).

Observe that, if chase($I, \Sigma$) exists, then the chase sequence that leads to it gives rise to a homomorphism $h_{I,\Sigma} : \text{dom}(I) \rightarrow \text{dom}(\text{chase}(I, \Sigma))$, which is the identity on dom(chase($I, \Sigma$)), such that $h_{I,\Sigma}(I) = \text{chase}(I, \Sigma)$. For a CQ $q$ we write chase($q, \Sigma$) for 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 $\Sigma$ a set of TGDs or EGDs. Then $q$ is contained in $q'$ under $\Sigma$, 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 $\Sigma$, 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 $\Sigma$ 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 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 $\Sigma$ of TGDs or EGDs, is it the case that $q \subseteq_\Sigma q'$? Whenever $\Sigma$ is bound to belong to a certain class $C$ of sets of TGDs or EGDs, we denote this problem as Cont($C$). 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).\footnote{This restriction is satisfied when 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 $\mathcal{G}$ be the class of sets of guarded TGDs. Then:}

**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 $\mathcal{G}$ be the class of sets of guarded TGDs. Then:
Fig. 2. Stickiness and marking.

**Proposition 2.6.** [11] \(\text{Cont}(\mathcal{G})\) is \(2\text{ExpTime}\)-complete, \(\text{ExpTime}\)-complete if the arity of the schema is fixed, and \(\text{NP}\)-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 \(\mathbb{D}\) be the classes of sets of linear TGDs and inclusions dependencies, respectively. Then:

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

**Non-recursiveness:** A set \(\Sigma\) of TGDs is non-recursive if its predicate graph contains no directed cycles. Recall that the nodes of the predicate graph of \(\Sigma\) are the predicates occurring in \(\Sigma\), 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 \(\tau\). 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] \(\text{Cont}(\mathbb{NR})\) is complete for \(\text{NExpTime}\), even if the arity of the schema is fixed. It becomes \(\text{NP}\)-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 \(\tau\) but not in every head-atom of \(\tau\) 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 \(\Sigma\) of TGDs; w.l.o.g., we assume that the TGDs in \(\Sigma\) do not share variables. For notational convenience, given an atom \(R(f)\) and a variable \(x \in f\), \(\text{pos}(R(f), x)\) is the set of positions
Each one of the previous classes has an associated weak version, called \( \tau \in \Sigma \) (e.g., guarded, non-recursive, sticky, etc.) or sets of EGDs (e.g., arbitrary EGDs, FDs, etc.):

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 \( \Sigma \) of TGDs or EGDs. Whenever this is the case, we say that \( q \) is semantically in \( \text{GHW}_k \) under \( \Sigma \). In case \( k = 1 \), \( q \) is also called semantically acyclic under \( \Sigma \) since \( \text{GHW}_1 \) 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:** Sem\( \text{GHW}_k(\mathbb{C}) \)

**INPUT:** A CQ \( q \) and a finite set \( \Sigma \in \mathbb{C} \) of TGDs or EGDs, both over a schema \( \sigma \).

**QUESTION:** Is there a CQ \( q' \in \text{GHW}_k \) over \( \sigma \) such that \( q \equiv_{\Sigma} q' \)?

One may ask whether there exist CQs that are in general not semantically in \( \text{GHW}_k \), but are semantically in \( \text{GHW}_k \) 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 Sem\( \text{GHW}_k(\mathbb{C}) \), 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 \( \text{GHW}_k \) 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

\[
q = \exists x \exists y \exists z \left( R(x, y) \land S(y, z) \land S(z, x) \right).
\]

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 \( \Sigma \) of TGDs consisting of

\[
R(x, y) \rightarrow \exists z \ P(x, y, z), \quad P(x, y, z) \rightarrow S(y, z), \quad P(x, y, z) \rightarrow S(z, x).
\]

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

\[
R[1, 2] \subseteq P[1, 2], \quad P[2, 3] \subseteq S[1, 2], \quad P[3, 1] \subseteq S[1, 2].
\]
Moreover, observe that $\Sigma$ belongs to all the classes of TGDs mentioned in the previous section, i.e., $G$, $NR$ and $S$. Interestingly, $q$ is equivalent to the acyclic CQ

$$q' = \exists x \exists y \exists z \left( R(x, y) \land S(y, z) \land S(z, x) \land P(x, y, z) \right)$$

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

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

**Example 3.2.** Consider the CQ

$$q = \exists x \exists y \exists z \left( R(x, y) \land R(y, z) \land R(z, x) \land R(x, z) \right).$$

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 $\Sigma$ 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

$$q' = \exists y \left( R(y, y) \right)$$

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

The above discussion shows that $\text{SemGHW}_k(C)$, where $C$ 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 $\text{SemGHW}_k(C)$.  

### 3.1 Infinite Instances vs. Finite Databases

It is important to clarify that $\text{SemGHW}_k(C)$ asks for the existence of a CQ $q'$ in $GHW_k$ that is equivalent to $q$ under $\Sigma$ focusing 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 $\text{FinSemGHW}_k(C)$, which accepts as input a CQ $q$ and a set $\Sigma \in C$ of TGDs or EGDs, and asks whether there is a CQ $q' \in GHW_k$ such that $q(D) = q'(D)$ for every (finite) database $D$ that satisfies $\Sigma$. Interestingly, for all the classes of sets of TGDs discussed in the previous section, and the class of (arbitrary) EGDs, $\text{SemGHW}_k$ and $\text{FinSemGHW}_k$ 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 $\text{SemGHW}_k(C)$, instead of $\text{FinSemGHW}_k(C)$, 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 $\text{SemGHW}_k(C)$ is defined for TGDs or EGDs, i.e., $C$ 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].

4 SEMANTIC GENERALIZED HYPETREEWIDTH UNDER TGDS

It is clear that SemGHW\(_k(\Sigma)\) and Cont(\(\Sigma\)), where \(\Sigma\) 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 \(\Sigma \in \mathcal{C}\). In view of the fact that CQ containment under TGDS is undecidable [10], the question that comes up is whether the decidability of Cont(\(\Sigma\)) for a class \(\mathcal{C}\) of TGDS is a necessary condition for the decidability of SemGHW\(_k(\mathcal{C})\). Of course, it is also interesting to ask the converse, i.e., whether the decidability of Cont(\(\Sigma\)) is a sufficient condition for the decidability of SemGHW\(_k(\mathcal{C})\). 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 TGDSs 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 \(q = \exists y \wedge_1 \leq n R_i(i_t)\) is a Boolean connected CQ \(q' = \exists y' \wedge_1 \leq k R_i'(i_t)\), where \(y' \subseteq y, \{R_i'(i_t)\}_{1 \leq i \leq k} \subseteq \{R_i(i_t)\}_{1 \leq i \leq m}\), and the CQ obtained by adding an atom of \(\{R_i(i_t)\}_{1 \leq i \leq m} \setminus \{R_i'(i_t)\}_{1 \leq i \leq k}\) to \(q'\) is not connected. Finally, a TGD \(\tau\) is body-connected if its body, which can be seen as a CQ, is connected. The following holds:

**Lemma 4.1.** Consider a set \(\Sigma\) of body-connected TGDSs, and two Boolean CQs \(q, q'\), where \(q'\) is connected. If \(q \equiv_S q'\), then there exists a component \(q''\) of \(q\) such that \(q'' \equiv_S q'\).

**Proof.** By Lemma 2.5, it suffices to show that there exists a component \(q''\) of \(q\) such that \(q'(\text{chase}(q'', \Sigma)) \neq \emptyset\). By hypothesis and Lemma 2.5, \(q'(\text{chase}(q, \Sigma)) \neq \emptyset\). Let \(q_1, \ldots, q_k\), where \(k \geq 1\), be the components of \(q\). By exploiting the fact that \(\Sigma\) is body-connected, it is not difficult to show that \(\text{chase}(q, \Sigma)\) 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 \(R(i) \in I_i\), the chase derivation that derives \(R(i)\) starts from atoms of \(q_i\). Since \(q'\) is connected, \(q'(\text{chase}(q, \Sigma)) \neq \emptyset\) implies that there is \(i \in \{1, \ldots, k\}\) such that \(q'(I_i) \neq \emptyset\), i.e., \(q'(\text{chase}(q, \Sigma)) \neq \emptyset\), 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 TGDSs to SemGHW\(_k\), and show its correctness by exploiting the above lemma. Given two Boolean CQs \(q = \exists y \wedge_1 \leq n R_i(i_t)\) and \(q' = \exists z \wedge_1 \leq m R'_j(v_i)\), let \(\text{conj}(q, q')\) be the Boolean CQ

\[
\exists y \exists z' (R_1(i_t) \wedge \cdots \wedge R_n(i_t) \wedge R'_1(v_i) \wedge \cdots \wedge R'_m(v'_m)),
\]

where \(z'\) is obtained from \(z\) by replacing each \(z \in z\) by a fresh variable \(z'\), and for each \(i \in \{1, \ldots, m\}\), \(v'_i\) is obtained from \(v_i\) by replacing each \(v \in v_i\) by a fresh variable \(v'_i\). In other words, \(\text{conj}(q, q')\) 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:

\[\text{Note that unrestricted CQ containment under the non-conflicting combination of inclusion dependencies and key dependencies is decidable [14]. However, as discussed in Section 3.1, we are only concerned about finite databases.}\]
We define the following two sets of indices:

\[ \text{RestCont} \text{ and } \text{SemGHW} \]

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 \( C \) of sets of TGDs, \( \text{SemGHW}_k(C) \) can be reduced to \( \text{Cont}(C) \). This would immediately imply the decidability of \( \text{SemGHW}_k(C) \) whenever \( \text{Cont}(C) \) is decidable. However, as the next result shows, the picture is more complicated than this as \( \text{SemGHW}_1(\mathbb{F}) \) is undecidable, where \( \mathbb{F} \) is the class of full TGDs, a class which ensures the decidability of CQ containment:

\[ \text{Theorem 4.4.} \quad \text{SemGHW}_1(\mathbb{F}) \text{ is undecidable, even if we allow only unary and binary predicates.} \]

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

\[ \text{J. ACM, Vol. 1, No. 1, Article . Publication date: September 2020.} \]
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 \(u_1 \cdots u_m = v_1 \cdots v_m\). 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 \(\Sigma\) of full TGDs, both over the schema \(\sigma\) that consists of the binary predicates

\[
\{a, b, (\diamond i, i)_1 \leq i \leq n, (\diamond x, i)_1 \leq i \leq n, (\diamond i, j)_1 \leq i, j \leq n, \text{start, end, } \eta_1, \eta_2, \eta\},
\]

such that the PCP instance has a solution if and only if there exists a CQ \(q'\) ∈ \(\text{GHW}_1\) (i.e., an acyclic CQ) such that \(q \equiv_{\Sigma} q'\), or, equivalently, \(q\) is semantically in \(\text{GHW}_1\) under \(\Sigma\). Since \(\Sigma\) consists only of binary predicates, we can naturally represent CQs over \(\sigma\) as directed graphs in which edges are labeled with symbols from \(\sigma\). Intuitively speaking, if there exists a CQ \(q'\) ∈ \(\text{GHW}_1\) such that \(q \equiv_{\Sigma} q'\), then \(q'\) "encodes" a solution \(i_1 \ldots i_m\) to the PCP instance, i.e., \(u_1 \cdots u_m = v_1 \cdots v_m\). In particular, such a solution is "encoded" in \(q'\) in a directed path, which is labeled by a word \(w\) over the alphabet \(\{a, b, (\diamond i, i)_1 \leq i \leq n, (\diamond x, i)_1 \leq i \leq n, (\diamond i, j)_1 \leq i, j \leq n, \text{start, end}\}\) such that:

- after removing from \(w\) the symbols \(\diamond x, j\), for \(1 \leq j \leq n\), we obtain a word of the form:
  \[
  \text{start } u_1 \diamond x_{r_1} \cdots u_{i_m} \diamond x_{r_m} \text{ end},
  \]
  where \(r_j \in \{*, 1, \ldots, n\}\) for each \(1 \leq j \leq m\), and

- after removing from \(w\) the symbols \(\diamond x, j\), for \(1 \leq j \leq n\), we obtain a word of the form:
  \[
  \text{start } v_1 \diamond x_{r_1} \cdots v_{i_m} \diamond x_{r_m} \text{ end},
  \]
  where \(r_j \in \{*, 1, \ldots, n\}\) for each \(1 \leq j \leq m\).

In other words, the symbols \(\diamond i, i\) and \(\diamond x, i\), for \(1 \leq i \leq n\), mark how \(w\) is formed in terms of the \(u_i\)'s and \(v_j\)'s, respectively. The symbols \(\diamond i, j\), for \(1 \leq i, j \leq n\), are used to mark positions that represent synchronous occurrences of \(u_i\) and \(v_j\).

The predicates \(\eta_1, \eta_2\), and \(\eta\) in the set \(\Sigma\) 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 \(\diamond x, j\), for \(1 \leq j \leq n\), and, analogously, let \(w_2\) be the word that is obtained from \(w\) after removing the symbols \(\diamond j, x\), for \(1 \leq j \leq n\). Then the following statements hold:

1. The word \(w_1\) is of the form:
   \[
   \text{start } u_1 \diamond i_{r_1} \cdots u_{i_m} \diamond i_{r_m} \text{ end},
   \]
   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 \(\diamond i_{r_j}\) that marks occurrences of such word.

2. The word \(w_2\) is of the form:
   \[
   \text{start } v_{j_1} \diamond i_{j_1} \cdots v_{j_t} \diamond i_{j_t} \text{ end},
   \]
   for \(1 \leq j_1, \ldots, j_t \leq n\), and \(r_1, \ldots, r_t \in \{*, 1, \ldots, n\}\). That is, each word \(v_{j_j}\) is in fact followed by a symbol of the form \(\diamond i_{r_j}\) that marks occurrences of such word.

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

In order to ensure that the above conditions hold, we define the predicates \(\eta_1, \eta_2\), and \(\eta\) in the set \(\Sigma\) 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 $\diamond_{i,j}$. 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 $u_i \diamond_{i,j}$ 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 $v_i \diamond_{r,i}$ 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 $\text{SemGHW}_k$, 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 $C$ of sets of TGDs, the so-called generalized hypertreewidth preserving chase (or simply GHW-preserving chase), that ensures the decidability of $\text{SemGHW}_k(C)$. This criterion guarantees that, starting from an instance in $\text{GHW}_k$, 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 $\text{SemGHW}_k(\Sigma)$, and we pinpoint the complexity of the latter problem. Notice that non-recursiveness and stickiness do not fulfill 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 $C$ of sets of TGDs has generalized hypertreewidth preserving chase (or simply GHW-preserving chase) if, for every CQ $q \in \text{GHW}_k$, set $\Sigma \in C$, and chase sequence for $D[q]$ under $\Sigma$ of the form

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

it holds that the instance $\bigcup_{i \geq 0} I_i$ belongs to $\text{GHW}_k$.

Recall that the purpose of the above property is to ensure the decidability of $\text{SemGHW}_k(C)$, for a class $C$ of TGDs. Let us first show that it ensures the decidability of $\text{Cont}(\Sigma)$. To this end, we exploit the classical result that the satisfiability problem for a fragment $\mathcal{L}$ 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 $\mathcal{L}$ 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 $(T, \chi)$, with $T = (V, E)$, is the number $\max_{v \in V} \{|\chi(v)|\} - 1$. A fragment of first-order logic $\mathcal{L}$ enjoys the FTMP if, for every sentence $\Phi$ that falls in $\mathcal{L}$, if $\Phi$ has a model, then it has a model of finite treewidth – note that a model of $\Phi$ can be naturally seen as a relational instance. We can now show that:

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

**Proof.** Consider an instance of $\text{Cont}(C)$, that is, two CQs $q(\bar{x})$ and $q'(\bar{x}')$, and a set $\Sigma \in C$ of TGDs. By Lemma 2.5, we need to show that the problem of deciding whether $\bot(\bar{x})$ belongs
to the evaluation of $q'$ over $\text{chase}(q, \Sigma)$, 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 $\text{chase}(q, \Sigma)$, 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 $\text{chase}(q, \Sigma)$, with each null being interpreted as an existentially quantified variable, is a model of $q'(\bot(\bar{x}))$, i.e., the sentence $\Phi_{q, q', \Sigma} = \phi_{q, \Sigma} \land \neg q'(\bot(\bar{x}))$ is unsatisfiable. Consequently, to show that $\text{Cont}(\Sigma)$ 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($q, \Sigma$). Since $\Sigma$ falls in a class that has GHW-preserving chase, the generalized hypertreewidth of $M$ coincides 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 treewidth 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 $\text{SemGHW}_k$. We also need to show that, if the given query $q$ is semantically in GHW under the given set $\Sigma$ of TGDs, then there exists a CQ $q' \in \text{GHW}_k$ 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 $\Sigma$ of TGDs over a schema $\sigma$ that belongs to a class that has GHW-preserving chase, and a CQ $q$ over $\sigma$. If $q$ is semantically in GHW$_k$ under $\Sigma$, then there exists a CQ $q' \in \text{GHW}_k$ over $\sigma$, 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 $q' \in \text{GHW}_k$, both over a schema $\sigma$, such that $q' \subseteq q$. There exists a CQ $q'' \in \text{GHW}_k$ over $\sigma$ such that $q'' \subseteq q'$ and $|q''| \leq |q| \cdot (2k + 1)$.

**Proof.** Since, by hypothesis, $q' \in \text{GHW}_k$, the canonical instance of $q'$ admits a generalized hypertree decomposition $(T, \chi, \lambda)$ of width $k$. Our goal is, by exploiting the existence of $(T, \chi, \lambda)$ and the fact that $q' \subseteq q$, to first define a database $D \in \text{GHW}_k$ over $\sigma$ with at most $|q| \cdot (2k + 1)$ atoms, and then show that $D$ can be converted into a CQ $q''$ over $\sigma$ such that $q'' \subseteq q'$.
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 $(T, \chi, \lambda)$ 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 $(F, \chi', \lambda')$, where $\chi'$ (resp., $\lambda'$) is the restriction of $\chi$ (resp., $\lambda$) 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 $(F, \chi', \lambda')$ we can construct a triple $(F, \chi'', \lambda'')$ 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 $D'' \in \text{GHW}_k$. The construction of $(F, \chi'', \lambda'')$ from $(F, \chi', \lambda')$ 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 $\{t_{\ell+1}, \ldots, t_{\ell+p}\} = \emptyset$. Then $\chi''(v) = \{t_1, \ldots, t_{\ell+i}, \perp_{\ell+1}, \ldots, \perp_{\ell+p}\}$, where $\perp_{\ell+1}, \ldots, \perp_{\ell+p}$ 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 $\perp_{\ell+i}$. 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 $\chi''(v) = \{t_1, t_2, \perp_3, \perp_4\}$ and $\lambda''(v) = \{R(t_1, \perp_3), P(t_2, \perp_3, \perp_4)\}$. It is easy to verify that $(F, \chi'', \lambda'')$ is indeed a generalized hypertree decomposition of $D''$ with width $k$, and thus $D'' \in \text{GHW}_k$. 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 $\perp_x$ occurs in $D$. Let $q''(\bar{x}')$ be the CQ obtained by considering the conjunction of atoms in $D$, after renaming each null $\perp$ into a variable $v(\perp)$, with $v(\perp) = x$ for each $x \in \bar{x}'$. It is clear that $q'' \in \text{GHW}_k$ since $D'' \in \text{GHW}_k$, and $|q''| \leq |q| \cdot (2k + 1)$ since $D''$ contains at most $|q| \cdot (2k + 1)$ atoms. Moreover, $\perp(\bar{x}')$ 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 $\mu$ from $q''$ to the canonical instance of $q'$ such that $\mu(\bar{x}') = \perp(\bar{x}')$, and hence, by Lemma 2.5, $q' \subseteq q''$.  

J. ACM, Vol. 1, No. 1, Article . Publication date: September 2020.
The homomorphism $\mu$ is obtained by reversing the renaming substitutions applied during the construction of $(F, \chi'', \lambda'')$ from $(F, \chi', \lambda')$. Since $q' \subseteq q'' \subseteq q$ the claim follows. \hfill \qedsymbol

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 $q'' \in \text{GHW}_k$ 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 $\text{GHW}_k$ under $\Sigma$, there is a CQ $q''(x) \in \text{GHW}_k$ such that $q \equiv_{\Sigma} q''$. Let $q''_q$ be the (possibly infinite) CQ obtained by applying the chase procedure over the atoms of $q''$ using the TGDs of $\Sigma$. More formally, $q''_q(x)$ is the CQ obtained by considering the (possibly infinite) conjunction of atoms in chase($q''$, $\Sigma$), after renaming each null $\bot$ occurring in chase($q''$, $\Sigma$) into a variable $v(\bot)$, with $v(\bot_x) = x$ for each $x \in \bar{x}$. By using Lemma 2.5, we can show that $q'' \subseteq q$ implies $q''_q \subseteq q$. Since $q'' \in \text{GHW}_k$ and $\Sigma$ belongs to a class that has GHW-preserving chase, we conclude that $q''_q \in \text{GHW}_k$. Thus, by Lemma 5.4, there is a CQ $\hat{q} \in \text{GHW}_k$ such that $q''_q \subseteq \hat{q} \subseteq q$ and $|\hat{q}| \leq |q| \cdot (2k + 1)$. Since $q''_q \subseteq \hat{q}$ 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 $q' = \hat{q}$. \hfill \qedsymbol

The problem of determining whether a CQ $q$ belongs to $\text{GHW}_k$ 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 $\text{SemGHW}_k(\mathbb{C})$ whenever $\mathbb{C}$ has GHW-preserving chase. More precisely, given a CQ $q$ and a set $\Sigma \in \mathbb{C}$, both over a schema $\sigma$:

1. Guess a CQ $q'$ over $\sigma$ such that $|q'| \leq |q| \cdot (2k + 1)$.
2. If $q' \in \text{GHW}_k$, $q \subseteq_{\Sigma} q'$ 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, $\text{SemGHW}_k(\mathbb{C})$ is decidable.

### 5.1 Guardedness

We proceed to show that $\text{SemGHW}_k(\mathbb{G})$ is decidable and has the same complexity as $\text{Cont}(\mathbb{G})$. 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.** $\text{SemGHW}_k(\mathbb{G})$ is $2\text{ExpTime}$-complete, and it becomes $\text{ExpTime}$-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

$I = I_0 \xrightarrow{r_1(t_1, I'_1)} I_1 \xrightarrow{r_1(t_1, I'_1)} I_2 \ldots$

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

J. ACM, Vol. 1, No. 1, Article . Publication date: September 2020.
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., $\lambda$ 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 $\tau_i$ is satisfied by $\lambda(v)$.

Example 5.7. Consider the database $D = \{R(a,b), S(b)\}$, the set $\Sigma$ consisting of 
\[ \tau = R(x,y), S(y) \rightarrow \exists z R(z,x) \quad \tau' = R(x,y) \rightarrow S(x), \]
and the chase sequence for $D$ under $\Sigma$
\[
D \xrightarrow{\tau,(a,b)} D \cup \{R(\bot_1,a)\} \xrightarrow{\tau',(\bot_1,a)} D \cup \{R(\bot_1,a), S(a)\} \xrightarrow{\tau,(\bot_1,a)} \\
D \cup \{R(\bot_1,a), S(a)\} \xrightarrow{\tau',(\bot_1,a)} D \cup \{R(\bot_1,a), S(a), R(\bot_2, \bot_1), S(\bot_1)\} \ldots,
\]
where $\bot_1, \bot_2, \ldots \in \mathbb{N}$. The guarded chase forest for $D$ and $\Sigma$ is depicted in Figure 4.

We are now ready to show that:

Proposition 5.8. $\square$ has GHW-preserving chase.

Proof. Consider a CQ $q \in \text{GHW}_k$, and a set of TGDs $\Sigma \in \square$. We need to show that chase($q, \Sigma$), that is, the result of an arbitrary chase sequence
\[
D[q] = I_0 \xrightarrow{\tau_0,(i_0,i'_0) I_1} I_1 \xrightarrow{\tau_1,(i_1,i'_1) I_2} \ldots
\]
for $D[q]$ under $\Sigma$, belongs to $\text{GHW}_k$, 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 $\Sigma$. We first show that each connected component of $F$, which is a tree with its root labeled by an atom $\alpha$ of $D[q]$, denoted $T_{\alpha}$, enjoys the following property: For each term (constant or null) $t$ occurring in $T_{\alpha}$, the set \{\(\nu \in V \mid t \text{ occurs in } \lambda(\nu)\)\} induces a connected subtree of $T_{\alpha}$. Towards a contradiction, assume that the latter does not hold. This implies that there exists a path $v\nu_1 \ldots \nu_nu$ in $T_{\alpha}$, where $n \geq 1$, and a term $t$ that occurs in $\lambda(\nu)$ and $\lambda(\nu)$, but $t$ does not occur in $\lambda(\nu_i)$, for each $i \in \{1, \ldots, n\}$. Assume that $\lambda(\nu)$ was generated during the $i$-th application of the chase step, i.e., $I_{i+1} \setminus I_i = \{\lambda(\nu)\}$. Since $t$ does not occur in $\lambda(\nu_i)$, we conclude that $\tau_i$ is not guarded. But this contradicts our hypothesis that $\Sigma \in \square$; thus, $T_{\alpha}$ enjoys the desired property. This allows us to convert $T_{\alpha}$ 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 hypertree decomposition ($T_q, \chi', \lambda'$) of width at most $k$. Having ($T_{\alpha}, \chi, \lambda$), for each atom $\alpha \in D[q]$, and ($T_q, \chi', \lambda'$) in place, we can now construct a generalized hypertree decomposition ($T, \chi'', \lambda''$) of width at most $k$ for chase($q, \Sigma$) as follows: For each $\alpha \in D[q]$, we attach ($T_{\alpha}, \chi, \lambda$) to $v_\alpha$, where $v_\alpha$ is a node in $T_q$ such that $\chi'(v_\alpha)$ contains all the nulls in $\alpha$. Note that $v_\alpha$ always exists since ($T_q, \chi'$) is a tree decomposition for $q$. Moreover,

\[\text{[Here, we assume, w.l.o.g., that the chosen chase sequence is such that every chase step generates a different atom, i.e., for every } i > 0, I_{i+1} \setminus I_i \neq \emptyset.\text{ This allows us to refer to the chase step that generates an atom of chase}(I, \Sigma) \setminus I.\]
We proceed to show that the above complexity upper bounds are optimal. By Proposition 4.2, Assumee that $q$ and a set $\Sigma \in \mathcal{G}$, both over a schema $\sigma$, we simply need to guess a CQ $q'$ over $\sigma$ 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($\mathcal{G}$). Recall that checking whether a CQ belongs to GHW$_k$ is in NP [24]. Thus, Proposition 2.6 implies that SemGHW$_k(\mathcal{G})$ is in 2ExpTime, in ExpTime if the arity of the schema is fixed, and in NP if the schema is fixed; recall that for $C \in \{2\text{ExpTime}, \text{ExpTime}\}$, NP$^C = C$. At this point, one may ask why for a fixed schema the obtained upper bound is NP and not NP$^\Sigma = \Sigma_P^P$ since $C = \Sigma_P = \text{NP}$. 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$_k(\mathcal{G})$ can be reduced in linear time to SemGHW$_k(\mathcal{G})$. Thus, to obtain the desired lower bounds, it suffices to reduce in polynomial time Cont($\mathcal{G}$) to RestCont$_k(\mathcal{G})$. Interestingly, the lower bounds given in Section 2 for Cont($\mathcal{G}$) 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$_2(\mathcal{C})$ be the following problem: Given a Boolean CQ $q \in \text{GHW}_k$, a Boolean CQ $q'$, and a set $\Sigma \in \mathcal{C}$ of TGDs, is it the case that $q \equiv_{\Sigma} q'$?

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$_2$ 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 $\Sigma$ of TGDs. Assume that $q, q'$ are of the form $\exists \bar{y}(R_1(i_1) \land \cdots \land R_m(i_m))$. The application of the connecting operator on $(q, q', \Sigma)$ returns the triple $(c(q), c(q'), c(\Sigma))$, where

- $c(q)$ is the query $\exists \bar{y} \exists w(R^*_1(i_1, w) \land \cdots \land R^*_m(i_m, w) \land \text{Aux}(w, w))$, where \(w\) is a new variable not in \(q\), each \(R^*_i\) is a new predicate, and Aux is a new binary predicate not occurring in \(q, q'\) or \(\Sigma\).

- $c(q')$ is the query $\exists \bar{y} \exists w_1 \cdots \exists w_{2(k+1)}(R^*_1(i_1, w_1) \land \cdots \land R^*_m(i_m, w_1) \land \bigwedge_{1 \leq i < j \leq 2(k+1)} \text{Aux}(w_i, w_j))$, where \(w_1, \ldots, w_{2(k+1)}\) are new variables not in \(q\).

- Finally, $c(\Sigma) = \{c(\tau) \mid \tau \in \Sigma\}$, where for $\tau = \phi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z} \psi(\bar{x}, \bar{z})$, $c(\tau)$ is the TGD $\phi^*(\bar{x}, \bar{y}, w) \rightarrow \exists \bar{z} \psi^*(\bar{x}, \bar{z}, w)$, where $\phi^*(\bar{x}, \bar{y}, w)$ and $\psi^*(\bar{x}, \bar{z}, w)$ 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 $R^*(x_1, \ldots, x_n, w)$, where \(w\) is a new variable.
This concludes the definition of the connecting operator. A class $C$ of sets of TGDs is closed under connecting if, for every set $\Sigma \in C$, $c(\Sigma) \in C$. It is not difficult to see that $q \subseteq q'$ if and only if $c(q) \subseteq c(\Sigma) c(q')$. It is easy to verify that $c(q)$ belongs to GHW$_k$ and is connected, $c(\Sigma)$ is a set of body-connected TGDs, and $c(q')$ is connected. However, we need to argue a bit more why $c(q')$ is not semantically in GHW$_k$ under $c(\Sigma)$. 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 $q' \in$ 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 $c(q')$ contains a clique-query over $2(k + 1)$ variables in which only the binary predicate Aux is involved. This implies that $c(q')$ is not semantically in GHW$_k$ since the clique-query over $n$ variables belongs to GHW$_\lceil \frac{n}{2} \rceil$ but not to GHW$_\lceil \frac{n}{2} \rceil - 1$. Since the predicate Aux does not occur in $c(\Sigma)$, we get that $c(q')$ is not semantically in GHW$_k$ under $c(\Sigma)$. From the above discussion, it is clear that the connecting operator provides a polynomial time reduction from BoolCont$_k$ to RestCont$_k$, for every class $C$ of sets of TGDs that is closed under connecting. We can then easily show the following result:

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

**Proof.** It suffices to show that BoolCont$_k(C)$ can be reduced in polynomial time to SemGHW$_k(C)$. Since $C$ is closed under connecting, the connecting operator provides a polynomial time reduction from BoolCont$_k(C)$ to RestCont$_k(C)$. Moreover, by Proposition 4.2, we know that RestCont$_k(C)$ can be reduced in linear time to SemGHW$_k(C)$. By composing these two reductions, we get a polynomial time reduction from BoolCont$_k(C)$ to SemGHW$_k(C)$, 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 SemGHW$_k(C)$ stated in Theorem 5.6 follow from Proposition 2.6 and Proposition 5.9. Note that, although Proposition 2.6 refers to Cont($\mathcal{G}$), the lower bounds hold for BoolCont$_k(\mathcal{G})$; 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 (L), i.e., TGDs whose body consists of a single atom, which in turn subsumes the class of inclusion dependencies (ID). By exploiting the non-deterministic procedure employed for SemGHW$_k(\mathcal{G})$, and Proposition 2.7, we conclude that SemGHW$_k(C)$, where $C \in \{L, ID\}$, is in PSPACE, and in NP if the arity of the schema is fixed. Moreover, since, by construction, L and ID are closed under connecting, we get matching lower bounds, and the next complexity result follows:

**Theorem 5.10.** SemGHW$_k(C)$, where $C \in \{L, ID\}$, 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 SemGHW$_k(\mathcal{G})$, 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

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

J. ACM, Vol. 1, No. 1, Article . Publication date: September 2020.
where $q \in \text{GHW}_1$, and $\{\tau\}$ is non-recursive and sticky, but not guarded. In $\text{chase}(q, \{\tau\})$ the predicate $R$ holds all the pairs that can be formed using the nulls $\bot_{x_1}, \ldots, \bot_{x_n}$. More precisely,

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

Therefore, even though $q$ is acyclic, $\text{chase}(q, \{\tau\})$ is highly cyclic; in fact, the generalized hypertreewidth of $\text{chase}(q, \{\tau\})$ 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, $\mathbb{N}\mathbb{R}$ 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 $\sigma$ is an expression

$$Q(\vec{x}) = q_1(\vec{x}) \lor \cdots \lor q_n(\vec{x})$$

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

**Definition 6.2. (UCQ Rewritability)** We say that a class $\mathcal{C}$ of sets of TGDs is UCQ rewritable if, for every CQ $q$ and $\Sigma \in \mathcal{C}$, both over a schema $\sigma$, we can construct a UCQ $Q$ over $\sigma$ 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 $\mathcal{C}$ of sets of TGDs that is UCQ rewritable, $\text{Cont}(\mathcal{C})$ 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 $b \in \mathbb{C}$. 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 $P(a) \land \phi$, where $\phi$ is a conjunction of atoms of the form $R(c, d)$, where $c, d \in \mathbb{C}$. Recall that $q' \subseteq Q$ if and only if $Q(D[q']) \neq \emptyset$. This means that $Q$ can check whether in $D[q']$ there exists a sequence of atoms $R(\bot_{a_1}, \bot_1), R(\bot_{a_1}, \bot_2), \ldots, R(\bot_{a_n}, \bot_{b})$ 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 $\mathcal{C}$ of sets of TGDs, there exists a computable function $f_{\subset}(\cdot, \cdot)$ from the set of pairs $(q, \Sigma)$, where $q$ is a CQ and $\Sigma \in \mathcal{C}$, to the natural numbers, such that the following holds: For every CQ $q$, set $\Sigma \in \mathcal{C}$, and UCQ rewriting $Q$ of $q$ and $\Sigma$, the maximal size of its disjuncts is at most $f_{\subset}(q, \Sigma)$. The existence of the function $f_{\subseteq}$ follows by the definition of UCQ rewritability. We then show the following; in the rest of this section, we fix $k \geq 1$.
Proposition 6.5. Consider a set \( \Sigma \) of TGDs over a schema \( \sigma \) that belongs to a UCQ rewritable class \( \mathbb{C} \), and a CQ \( q \) over \( \sigma \). If \( q \) is semantically in \( \mathrm{GHW}_k \) under \( \Sigma \), then there exists a CQ \( q' \in \mathrm{GHW}_k \) over \( \sigma \), where \( |q'| \leq f_c(q, \Sigma) \cdot (2k + 1) \), such that \( q \equiv_{\Sigma} q' \).

Proof. Since, by hypothesis, \( q \) is semantically in \( \mathrm{GHW}_k \) under \( \Sigma \), there exists a CQ \( q'' \in \mathrm{GHW}_k \) such that \( q \equiv_{\Sigma} q'' \). The fact that \( \mathbb{C} \) is UCQ rewritable implies that there exists a UCQ \( q' \) over \( \sigma \) such that, for every (finite or infinite) CQ \( p \), \( p \subseteq_{\Sigma} q '' \), which is equivalent to \( q'' \subseteq Q \). 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 \( q' \in \mathrm{GHW}_k \) such that \( q'' \subseteq q' \subseteq \hat{q} \) and \( |q'| \leq |\hat{q}| \cdot (2k + 1) \leq f_c(q, \Sigma) \cdot (2k + 1) \). It remains to show that \( q \equiv_{\Sigma} q' \). By hypothesis, \( q \subseteq q'' \), and thus, \( q \subseteq q' \) 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 a point \( t \) such that \( I \models \Sigma \) and \( \hat{q}(I) \not\subseteq q(I) \), i.e., there is a tuple \( t \in \text{dom}(I)^{|x|} \) such that \( t \not\in \hat{q}(I) \) and \( t \not\in q(I) \). Without loss of generality, we assume that \( t = (\bot_{x_1}, \ldots, \bot_{x_n}) \), where \( x = (x_1, \ldots, x_n) \) are the free variables of \( \hat{q} \), with \( \bot_{x_1}, \ldots, \bot_{x_n} \) being nulls. Since \( I \models \Sigma \), \( \text{chase}(I, \Sigma) \) can be homomorphically mapped into \( I \) via a homomorphism that is the identity on \( \text{dom}(I) \), and thus, \( t \notin q(\text{chase}(I, \Sigma)) \); otherwise, \( t \in q(I) \) which contradicts the fact that \( t \notin q(I) \). Since \( \hat{q} \) is a disjunct of \( Q \), we get that \( t \in Q(I) \). 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 \( \nu(\bot) \), with \( \nu(\bot_x) = x \) for each \( x \in \bar{x} \). Clearly, \( \nu(\bar{x}) \notin q(\text{chase}(q_I, \Sigma)) \) and \( \nu(\bar{x}) \in Q(D(q_I)) \). 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 \( \text{SemGHW}_k(\mathbb{C}) \) whenever \( \mathbb{C} \) is UCQ rewritable. More precisely, given a CQ \( q \) and a set \( \Sigma \in \mathbb{C} \), both over a schema \( \sigma \), we can decide whether \( q \) is semantically in \( \mathrm{GHW}_k \) under \( \Sigma \) as follows:

1. Guess a CQ \( q' \) over \( \sigma \) such that \( |q'| \leq f_c(q, \Sigma) \cdot (2k + 1) \).
2. If \( q' \in \mathrm{GHW}_k \), \( q \subseteq_{\Sigma} q' \) 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, \( \text{SemGHW}_k(\mathbb{C}) \) is decidable.

6.1 Non-Recursiveness

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

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

The above result, together with Theorem 6.6, implies that \( \text{SemGHW}_k(\text{NR}) \) 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 \( f_{\text{NR}} \) is polynomial, and therefore, Proposition 6.7 guarantees the existence of a polynomially sized CQ in \( \text{GHW}_k \). In this case, the fact that checking whether a CQ falls in \( \text{GHW}_k \) is in NP together with the fact that \( \text{Cont}(\text{NR}) \) is in NP (by Proposition 2.8) immediately imply that \( \text{SemGHW}_k(\text{NR}) \) is in NP.

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

---

9Let us clarify that the work [31] does not explicitly consider the class \( \text{NR} \). However, the rewriting algorithm proposed in that paper works also for non-recursive sets of TGDs.
in \( \text{NExpTime} \). However, the fact that \( \text{Cont}(\mathbb{NR}) \) is in \( \text{NExpTime} \) (by Proposition 2.8) alone is not enough to conclude that \( \text{SemGHW}_k(\mathbb{NR}) \) is also in \( \text{NExpTime} \). 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

\[
D[q] = I_0 \xrightarrow{r_n(I_n,I_n)} I_1 \xrightarrow{r_n(I_1,I_2)} I_2 \ldots I_{n-1} \xrightarrow{r_n(I_{n-1},I_{n-1})} I_n
\]

for \( D[q] \) and \( \Sigma \), where \( n = |q'| \cdot (b_2)^O(p) \), 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 \( \Sigma \), 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 \( \text{SemGHW}_k(\mathbb{NR}) \) 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 \( \text{SemGHW}_k(\mathbb{NR}) \) is in \( \text{NExpTime} \).

The lower bounds are inherited from \( \text{BoolCont}_k(\mathbb{NR}) \) since \( \mathbb{NR} \) is closed under connecting (see Proposition 5.9). From the above discussion we conclude that:

**Theorem 6.8.** \( \text{SemGHW}_k(\mathbb{NR}) \) is \( \text{NExpTime} \)-complete, even if the arity of the schema is fixed, and it becomes \( \text{NP} \)-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 \( \Sigma \) of TGDs, let \( p_q, \Sigma \) be the number of predicates occurring in \( q \) and \( \Sigma \), 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 \( f_3(q, \Sigma) = p_{q, \Sigma} \cdot (a_{q, \Sigma} \cdot |q| + 1)^{a_{q, \Sigma}} \).

The above result, together with Theorem 6.6, implies that \( \text{SemGHW}_k(\mathbb{S}) \) 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 \( f_3 \) is polynomial. In fact, we show that \( \text{SemGHW}_k(\mathbb{S}) \) is \( \text{NP} \)-complete when the arity of the schema is fixed. The \( \text{NP} \)-hardness is inherited from \( \text{BoolCont}_k(\mathbb{S}) \) 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 \( f_3 \) is exponential, and thus, by following the usual guess and check approach we get that \( \text{SemGHW}_k(\mathbb{S}) \) is in \( \text{NExpTime} \), while Proposition 5.9 implies \( \text{ExpTime} \)-hardness. Summing up, our machinery based on UCQ rewritability shows that:

**Theorem 6.10.** \( \text{SemGHW}_k(\mathbb{S}) \) is in \( \text{NExpTime} \) and \( \text{ExpTime} \)-hard. It becomes \( \text{NP} \)-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 \( \text{GHW}_k \). It is clear that such a result would allow us to establish an \( \text{ExpTime} \) upper bound for \( \text{SemGHW}_k(\mathbb{S}) \). At this point, one may think that this can be achieved by showing that the function \( f_3 \) 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 \( \{\Sigma_n\}_{n \geq 0} \) of sticky sets of TGDs, where \( a_{q, \Sigma_n} = n + 2 \), such that \( f_3(q, \Sigma_n) \geq n^2 \). The set \( \Sigma_n \in \mathbb{S} \) consists of the following TGDs; for brevity, we write \( \bar{x}^i_n \) for the tuple of variables \( x_i, x_{i+1}, \ldots, x_j \):

\[
R_i(\bar{x}_1^{i-1}, z, \bar{x}_{i+1}^n, z, o), R_i(\bar{x}_1^{i-1}, o, \bar{x}_{i+1}^n, z, o) \rightarrow R_{i-1}(\bar{x}_1^{i-1}, z, \bar{x}_{i+1}^n, z, o), \quad 1 \leq i \leq n, \\
R_0(\underbrace{z, \ldots, z}_n, z, o) \rightarrow \text{Ans}(z, o).
\]
while \( q = \text{Ans}(0, 1) \). Let us clarify that \( \Sigma_n \) is indeed a sticky set of TGDs since, for every TGD \( \tau \in \Sigma_n \), all the variables in the body of \( \tau \) appear also in the head of \( \tau \). This means that there are no variables that are marked in \( \Sigma_n \), 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, \( Q(D[q']) \neq \emptyset \)). 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., \( f_\Sigma(q, \Sigma_n) \geq 2^n \), as needed. Therefore, we need a more refined property for stickiness than UCQ rewritability, which will allow us to close the complexity of \( \text{SemGHW}_k(\Sigma) \) when the arity is not fixed (unless the problem is \( \text{NExpTime} \)-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 \( \text{SemGHW}_1 \) 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 \( \text{SemGHW}_k \). 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 \( \sigma \) is an expression of the form \( R[A \rightarrow B] \), where \( R/n \) is a relation symbol in \( \sigma \), 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 \( \text{SemGHW}_k \) 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
\[
R(x, y) \land S(x, y, z) \land S(x, z, w) \land S(x, w, v) \land R(x, v).
\]
After applying on \( q \) the key \( R[1 \rightarrow 2] \)\(^{10}\) we obtain the CQ
\[
R(x, y) \land S(x, y, z) \land S(x, z, w) \land S(x, w, y),
\]
which clearly falls in \( \text{GHW}_2 \) but not in \( \text{GHW}_1 \).

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 \( R[\{1\} \rightarrow \{2\}] \).
Consider the Boolean CQs and key
\[ q = \exists x (R(x, x) \land P(x)) \quad \text{and} \quad \tau = R[1 \rightarrow 2]. \]
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 \( \exists x (R(x, x) \land \phi \land P(y)) \), where \( x, y \in \bar{x} \) and \( \phi \) 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 \( Q(D[q']) \neq \emptyset \). Hence, \( Q \) checks whether in \( D[q'] \) there is a sequence of atoms \( R(\bot, \bot), R(\bot, \bot_2), \ldots, R(\bot_{n-1}, \bot_q) \) 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 from \( K \), i.e., acyclic CQs. In other words, \( K \) has _acyclicity-preserving chase_, which is defined in the obvious way; roughly, given \( q \in GHW_1 \) and \( \Sigma \in K \), \( chase(q, \Sigma) \) is acyclic.

**Proposition 7.4.** For schemas with unary and binary predicates, \( K \) has acyclicity-preserving chase.

**Proof.** Consider a CQ \( q \in GHW_1 \) 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 \( \leq 2 \) 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 \( K \), 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 \( K \) 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, \( SemGHW_1(K) \) 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 \( K \) 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 \( SemGHW_k(K) \), 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 generalised 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 \( q_n \in GHW_2 \), while the instance \( chase(q_n, \Sigma) \) belongs to \( GHW_1[\frac{n}{2}] \). The CQ \( q_n \) is defined as the one in Figure 5. It is clear that \( chase(q_n, \Sigma) \) 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 \( chase(q_n, \Sigma) \) is of width at least \( \lceil \frac{n}{2} \rceil \) since it corresponds to an \( n \times n \) grid.
Assuming unary and binary predicates, can we show that $\text{SemGHW}_k(\mathcal{K})$, 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.

8 SEMANTIC GENERALIZED HYPETREEWIDTH UNDER KEYS

The goal of this section is to show that $\text{SemGHW}_k(\mathcal{K})$, 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, $\text{SemGHW}_k(\mathcal{K})$ 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 $q \equiv p$, 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 $\text{core}(q)$, is a subquery $p$ of $q$ of minimal size that is homomorphically equivalent to $q$, i.e., $q \equiv p$.

Recall that chase($q, \Sigma$), where $q$ is a BCQ and $\Sigma$ a set of keys, refers to the instance chase($D[q], \Sigma$), which can be naturally seen as a BCQ: simply consider the conjunction of the atoms occurring in chase($D[q], \Sigma$), and then rename each null $\perp$ into an existentially quantified variable $v(\perp)$. Notice that in this case 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 $\Sigma$ 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

Fig. 5. Depiction of $q_2, q_3, q_n$. 
$q \Rightarrow^* p$ 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 $\text{GHW}^k$ in terms of the existence of certain BCQs. In fact, we show that a BCQ $q$ is semantically in $\text{GHW}^k$ if and only if there is a BCQ in $\text{GHW}^k$ 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 $\Sigma$ of keys. We first define the set of BCQs

$$W_{q, \Sigma} = \{ p \mid \text{core}(\text{chase}(p, \Sigma)) \equiv \text{core}(\text{chase}(q, \Sigma)) \},$$

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 $\Sigma$ of keys. The following are equivalent:

1. $q$ is semantically in $\text{GHW}^k$ under $\Sigma$.
2. $W_{q, \Sigma} \cap \text{GHW}^k \neq \emptyset$.

**Proof.** For (1) $\Rightarrow$ (2), by hypothesis, there exists a BCQ $p \in \text{GHW}^k$ such that $q \equiv_\Sigma p$, and thus, $\text{chase}(q, \Sigma) \equiv \text{chase}(p, \Sigma)$ 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, $\text{core}(\text{chase}(p, \Sigma)) \equiv \text{core}(\text{chase}(q, \Sigma))$, which implies that $p \in W_{q, \Sigma} \cap \text{GHW}^k \neq \emptyset$, as needed. For (2) $\Rightarrow$ (1), by hypothesis, there exists a BCQ $p \in W_{q, \Sigma} \cap \text{GHW}^k$. 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 $W_{q, \Sigma} \cap \text{GHW}^k \neq \emptyset$ can be reduced to the satisfiability problem of MSO sentences over databases that belong to $\text{GHW}^k$. In particular, we want to show the following:

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

1. $W_{q, \Sigma} \cap \text{GHW}^k \neq \emptyset$;
2. $M_{q, \Sigma} \cap \text{GHW}^k \neq \emptyset$.

Furthermore, $M_{q, \Sigma}$ is definable via an MSO sentence $\xi_{q, \Sigma}$ that can be computed in single exponential time from $q$ and $\Sigma$, 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 $\text{SemGHW}^k(\mathbb{K})$. 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., $\ell$ is independent of $q$ and $\Sigma$), and hence the problem of checking whether $\xi_{q, \Sigma}$ has a model in $\text{GHW}^k$ can be solved in $\ell\text{-ExpTime}$ [40].

At this point, one might wonder whether Proposition 8.3 could be established by simply showing that the set $W_{q, \Sigma}$ is MSO-definable, that is, by showing that there exists an effective MSO sentence
Lemma 8.4. For a BCQ \( q \) and a set of keys \( \Sigma \), \( \{ p \mid \text{chase}(p, \Sigma) \equiv q \} \) 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 \( \Sigma = \{ R[1\to 2] \} \). Notice that for \( n > m \) we have that \( p_{n,m} \Rightarrow \Sigma p_{n-m,m} \) and \( \text{chase}(p_{n,m}, \Sigma) = q_n \). Thus, \( \text{chase}(p_{n,m}, \Sigma) \) essentially computes the greatest common divisor of \( n \) and \( m \), \( \text{GCD}(n, m) \), through the Euclidean algorithm, i.e., \( \text{chase}(p_{n,m}, \Sigma) \equiv q_{\text{GCD}(n,m)} \). Suppose now, for the sake of contradiction, that there exists an MSO sentence \( \phi \) of quantifier rank \( \ell \) such that, for every BCQ \( q \), it is the case that \( D[q] \models \phi \) iff \( \text{chase}(q, \Sigma) = q_1 \), 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 \( \ell \). 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 \( D[p_{s_j,(s_j-1)!}] \models \phi \), which is in contradiction with our assumption since \( \text{GCD}(p_j, (p_j - 1)!)) = 1 \) but \( \text{GCD}(p_i, (p_i - 1)!)) \neq 1 \).

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 \( \Sigma \), \( W_{q,\Sigma} \) 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 \( M_{q,\Sigma} \) as an MSO-definable superset of \( W_{q,\Sigma} \) in such a way that every BCQ \( p \in M_{q,\Sigma} \) can be modified into a BCQ \( \hat{p} \), without increasing its width, such that \( \hat{p} \in W_{q,\Sigma} \). 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 \( \Sigma \) 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 \( M_{q,\Sigma} \) (Section 8.3), we show that \( W_{q,\Sigma} \subseteq \)
M_q,\Sigma (Section 8.4), and we show that the expansion of every query from M_q,\Sigma belongs to 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 \( p|_V \) for the BCQ obtained by restricting \( p \) to the atoms that have a relation name occurring in \( \Sigma \).

### 8.2 The Expansion of a CQ

A \( \Sigma \)-cycle of a BCQ \( p \) is a subquery \( p' \subseteq p|_\Sigma \) consisting of a directed simple cycle over the relations of \( \Sigma \). Similarly, a \( \Sigma \)-path of \( p \) is a subquery induced by a directed simple path in \( p|_\Sigma \). Given a \( \Sigma \)-cycle \( C \) of \( p \), a node \( x \) of \( p \) in the same strongly connected component (SCC) as \( C \), and a (possibly empty) simple \( \Sigma \)-path \( \pi \) 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. \( \Sigma \)) of \((p, x)\) is the union of all \( P_{x,\pi,C} \) for all \( \Sigma \)-paths \( \pi \) of \( q \), and \( \Sigma \)-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.](image)

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. \( \pi' \in \text{GHW}_2 \).
2. \( \pi' \Rightarrow^*_\Sigma p \) with a provenance homomorphism that maps the variable \( x \) of \( \pi' \) to the variable \( x \) of \( p \).
   and every variable \( y \neq x \) of \( \pi' \) 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 \( \{x_1, \ldots, x_m\} \) 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 \( \pi_j \) be a simple path of \( p_{i-1} \) from \( x \) to \( x_j \). Observe that since, by inductive hypothesis, \( p'_i \Rightarrow^*_\Sigma p_{i-1} \), 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 \( \pi_j \) assures that the copy of \( x_j \) in \( P_{x,\pi_j,C_i} \) 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 \( h : p' \rightarrow p \). The expansion (w.r.t. \( h \) and \( \Sigma \)) of \( p' \) is the BCQ \( \hat{p}' \) obtained by attaching the flower decomposition (w.r.t. \( \Sigma \)) 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 \( \Sigma \)-expansion of \( (p', h) \).

\[
\begin{align*}
\text{(expansion)} \\
\hat{p} \\
\hat{h}
\end{align*}
\]

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 \( M_{q, \Sigma} \)

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. \( \Sigma \)) is either

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

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

(iii) a path of the form \( (x_1 \xrightarrow{R_1} y_1) \pi_1 \cdots \pi_{n-1} (x_n \xrightarrow{R_n} y_n) \), where,

- \( x_i = y_n \) and \( x_i \equiv_p y_j \) for every \( i, j \in [n] \),
- \( x_i \xrightarrow{R_i} y_i \) is either an edge \( x_i \xrightarrow{R_i} y_i \) or \( x_i \xleftarrow{R_i} y_i \) in \( p|_{\Sigma} \) for every \( i \in [n] \), and
- \( \pi_i \) 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 \( \pi \) and \( \pi' \) 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 \( \{x \xrightarrow{R} y, y \xleftarrow{R} x \mid x \xrightarrow{R} y \in p\} \) (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)
We proceed to show that $M$ variables $J. ACM, Vol. 1, No. 1, Article . Publication date: September 2020.

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 $\pi$ between two variables $x, y$ of $p'$, we say that $\pi$ is an $h$-regular cycling path if $h(\pi)$, that is, the path obtained after applying $h$ to every node of $\pi$, 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 chase($p', \Sigma$), provided $p'$ is an expansion.

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

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

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

**Lemma 8.7.** There exists an MSO sentence $\xi_{q, \Sigma}$ defining $M_{q, \Sigma}$. 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 chase($q, \Sigma$) can be computed in polynomial time, while the core of a CQ can be computed in exponential time. Thus, core(chase($q, \Sigma$)) is of polynomial size, while it can be computed in exponential time. Hence, we can assume that $q \equiv \text{core(chase}(q, \Sigma))$. 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 $\sim_h$ 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 $\sim_h$ 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 $\sim_h$-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 $g : q \to \hat{p}/\sim_h$. 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

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

J. ACM, Vol. 1, No. 1, Article . Publication date: September 2020.
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

1. \(z'_y = z'_y\) and \(f(z'_y) \xrightarrow{R} f(z'_y)\) in the flower decomposition of \((q, h(z'_y))\); or
2. \(z'_y \neq z'_y\), 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 \(p\).

\[\Box\]

8.4 Suitability of \(M_{q, \Sigma}\)

In this section we prove the following two statements:

(a) \(W_{q, \Sigma} \subseteq M_{q, \Sigma}\) (Lemma 8.9), and

(b) for every \(p \in M_{q, \Sigma} \cap \text{GHW}_k\) such that \(h : p \rightarrow \text{core}(\text{chase}(q, \Sigma))\), it holds that \(\hat{p} \in W_{q, \Sigma} \cap \text{GHW}_k\), where \((\hat{p}, \hat{h})\) is the \(\Sigma\)-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 \(h : p \rightarrow \text{core}(\text{chase}(q, \Sigma))\), then \(\text{chase}(\hat{p}, \Sigma) \equiv \hat{p}/\sim_{\hat{h}}\), where \((\hat{p}, \hat{h})\) is the \(\Sigma\)-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 \(W_{q, \Sigma} \subseteq M_{q, \Sigma}\).

**Proof.** Consider a BCQ \(p \in W_{q, \Sigma}\). We prove the following chain of homomorphic mappings

\[p \rightarrow \text{chase}(p, \Sigma) \rightarrow \text{chase}(q, \Sigma) \rightarrow \text{core}(\text{chase}(q, \Sigma)) \rightarrow \text{chase}(p, \Sigma) \rightarrow \text{chase}(\hat{p}, \Sigma) \rightarrow \hat{p}/\sim_{\hat{h}}\]

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

**Lemma 8.10.** For every BCQ \(p \in M_{q, \Sigma} \cap \text{GHW}_k\) with \(h : p \rightarrow \text{core}(\text{chase}(q, \Sigma))\), it holds that \(\hat{p} \in W_{q, \Sigma} \cap \text{GHW}_k\), where \((\hat{p}, \hat{h})\) is the \(\Sigma\)-expansion of \((p, h)\).

**Proof.** Let \(q' = \text{core}(\text{chase}(q, \Sigma))\). Since \(p \in M_{q, \Sigma}\), there is \(h : p \rightarrow q'\) such that \(q' \rightarrow \hat{p}/\sim_{\hat{h}}\), 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 \(\hat{p}/\sim_{\hat{h}} \equiv \text{chase}(\hat{p}, \Sigma)\) and thus

\[q' \rightarrow \text{chase}(\hat{p}, \Sigma) \tag{1}\]

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

\[\text{chase}(\hat{p}, \Sigma) \rightarrow q' \tag{2}\]

From (1) and (2) we obtain \(\text{chase}(\hat{p}, \Sigma) \equiv q'\), which shows \(\hat{p} \in W_{q, \Sigma}\). 

\[\Box\]
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 $\equiv_S$ 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 $\equiv_S$ as the equivalence classes of all variables that have been "glued together" after some steps of the chase.) Let $S^p_0$ 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 $\rho \equiv_S$ over the set of all partitions of the variables of $p$ in the following way: $S \equiv_S S'$ 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 $x \equiv_S x'$, $y \not\equiv_S z$ and $S' = S \setminus \{[y]_S, [z]_S\} \cup \{[y]_S \cup [z]_S\}$. The intuition is that $S \equiv_S S'$ 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 $\equiv_S$ and $\rho \equiv_S$.

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

1. $p \equiv^n_\Sigma p'$.
2. $p' \equiv p/\equiv_S$ for some $S$ such that $S^p_0 \equiv^n_\Sigma S$.

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 $h : p \to \text{core(chase}(q, \Sigma))$, and let $(\hat{p}, \hat{h})$ be the $\Sigma$- expansion of $(p, h)$. The following are equivalent:

1. $x \equiv_S x'$ for some $S$ such that $S^\hat{p}_0 \equiv^*_\Sigma S$.
2. $x \sim_{\hat{h}} x'$.

**Proof.** We first show that (1) implies (2). We proceed by induction on the number $n$ of $\equiv_\Sigma$ steps. The base case $n = 0$ is trivial since $x \sim_{\hat{h}} x$ for every $x$. Suppose now that $z' \equiv_S y'$ for $S^\hat{p}_0 \equiv^{n+1}_\Sigma S$.

We need to show that $z' \sim_{\hat{h}} y'$. Without loss of generality, assume that $S^\hat{p}_0 \equiv_\Sigma S' \equiv_\Sigma S$, 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:

\[ x \approx_{S} x' \text{ for some } S \text{ such that } S^\hat{p}_0 \equiv^*_\Sigma S. \]

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 $\text{core(chase}(q, \Sigma))$, while $\text{core(chase}(q, \Sigma))$ satisfies $\Sigma$. 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 $\hat{p}$: $\pi_{x_2, x_1}$ between $x_2$ and $x_1$, $\pi_{y,y'}$ between $y$ and $y'$, and $\pi_{z',z}$ between $z'$ and $z$, which exist by induction hypothesis. Now, observe that the path $\pi = \pi_{z',z}\cdot(x_1 \xrightarrow{R} y)\cdot \pi_{y,y'}$ between $z'$ and $y'$ is an $\hat{h}$-regular cycling path in $\hat{p}$. Indeed, there are two cases to consider:

1. $\hat{h}(x_1) = \hat{h}(x_2) \equiv_{\text{core(chase}(q, \Sigma))} \hat{h}(z') = \hat{h}(y) = \hat{h}(y')$.
2. $\hat{h}(x_1) = \hat{h}(x_2) \not\equiv_{\text{core(chase}(q, \Sigma))} \hat{h}(z') = \hat{h}(y) = \hat{h}(y')$.

J. ACM, Vol. 1, No. 1, Article . Publication date: September 2020.
These cases are covered, respectively, by items (ii) and (iii) in the definition of regular cycling path, and therefore, \( \pi \) is a valid \( \hat{h} \)-regular cycling path. Thus, \( y' \sim_{\hat{h}} z' \).

We now show that (2) implies (1). For brevity, let \( q' = \text{core}(\text{chase}(q, \Sigma)) \). Since \( \hat{p} \Rightarrow _{\Sigma} \) is monotone in the sense that it coarsens partitions, it is sufficient to prove the statement for the coarsest partition \( S \) such that \( S_0^0 \Rightarrow_{\Sigma} S \) (i.e., the partition \( S \) so that \( \text{chase}(\hat{p}, \Sigma) \equiv \hat{p} / \approx_S \) in view of Lemma 8.11). We proceed by structural induction on the \( \hat{h} \)-regular cycling path \( \pi \) of \( p \) between the variables \( x, x' \).

The base case (i) is the empty path \( \varepsilon \) 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 \( \hat{h}(\pi) = (y \xrightarrow{R} y')\pi(y' \xrightarrow{R} y) \) goes by induction hypothesis and the definition of \( \Rightarrow_{\Sigma} \). The only interesting case is (iii), where \( \hat{h}(\pi) = \text{of the form } (x_1 \xrightarrow{R_1 i_1} y_1) \pi_1 \cdots \pi_{n-1} (x_n \xrightarrow{R_n i_n} y_n) \), where, \( x_1 = y_n, x_i \equiv_{q'} y_j \) for every \( i, j \in [n] \), \( x_i \xrightarrow{R_i i} y_i \) is either an edge \( x_i \xrightarrow{R_i} y_i \) or \( x_i \xleftarrow{R_i} y_i \) in \( q' \mid z \) for every \( i \in [n] \), and \( \pi_i \) is a regular cycling path from \( y_i \) to \( x_{i+1} \) for every \( i \in [n - 1] \). Let \( \pi = (x_1 \xrightarrow{R_1 i_1} y_1) \pi_1 \cdots \pi_{n-1} (x_n \xrightarrow{R_n i_n} y_n) \) 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 \( \text{scq}(q', z) \) be the subquery of \( q' \mid z \) consisting of the strongly connected component of \( q' \mid z \) containing \( z \). In view of Lemma 8.11 and Lemma 8.6.2, note that for every node \( x \) of \( \hat{h}(\pi) \), \( \hat{h}(\pi) / \approx_S \) has attached a homomorphic copy of \( \text{scq}(q', \hat{h}(x)) \) —that is, there is a homomorphism \( \text{sc}(q', \hat{h}(x)) \rightarrow \hat{p} / \approx_S \) mapping \( \hat{h}(x) \) to \( [x] /_{\approx_S} \). Therefore, for every \( x \xrightarrow{R} y \) of \( \hat{h}(\pi) \), \( \hat{h}(x), \hat{h}(y) \) are in the same strongly connected component of \( q' \mid z \) and \( R[1 \rightarrow 2] \in \Sigma \), we have that the homomorphic copies of \( \text{scq}(q', \hat{h}(x)) \) and \( \text{scq}(q', \hat{h}(y)) \) 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 \( \text{scq}(q', \hat{h}(x)) \). This means that, since \( \hat{h}(x_i) = \hat{h}(x_n) \), 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 \( q' \mid z \), we have that \( x_i \approx_S x_n \), 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 \( \text{SemGHW}_k(\mathbb{K}) \), 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 \( \Sigma \) of keys. There is an MSO-definable set \( M_{q, \Sigma} \) of BCQs such that the following are equivalent:

1. \( W_{q, \Sigma} \cap \text{GHW}_k \neq \emptyset \);
2. \( M_{q, \Sigma} \cap \text{GHW}_k \neq \emptyset \).

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

**Proof.** The fact that (1) implies (2) is an immediate consequence of the fact that \( W_{q, \Sigma} \subseteq M_{q, \Sigma} \) by Lemma 8.9. For showing that (2) implies (1), consider an arbitrary BCQ \( p \in M_{q, \Sigma} \cap \text{GHW}_k \), where \( k > 1 \). Assuming that \( \hat{h} : p \rightarrow \text{core}(\text{chase}(q, \Sigma)) \), Lemma 8.10 implies that \( \hat{p} \in W_{q, \Sigma} \cap \text{GHW}_k \), where \( (\hat{p}, \hat{h}) \) is the \( \Sigma \)-expansion of \( (p, h) \). Therefore, \( W_{q, \Sigma} \cap \text{GHW}_k \neq \emptyset \), 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, \( \text{SemGHW}_k(\mathbb{K}) \) is decidable in elementary time.
We show that $\text{EvalSemGHW}_\Sigma$ which in turn, by Proposition 8.3, is equivalent to checking whether $q$ is semantically in $\text{GHW}_k$. As it has been noted in different scenarios in the absence of constraints, the property of being unary and binary predicates, is semantically in $\text{GHW}_k$, 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 $\mathcal{C}$ of constraints considered in the previous sections, the fact that we have a decision procedure for solving $\text{SemGHW}_k(\mathcal{C})$ allows us to show that the problem of evaluating a CQ $q$ that is semantically in $\text{GHW}_k$ under a set $\Sigma \in \mathcal{C}$ is fixed-parameter tractable. The crucial fact here is that our techniques for solving $\text{SemGHW}_k(\mathcal{C})$ yields an equivalent CQ $q' \in \text{GHW}_k$ in case $q$ is semantically in $\text{GHW}_k$ under $\Sigma$. 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 $\text{GHW}_k$ 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 $\mathcal{C}$ be a class of sets of TGDs or EGDs. We define:

**PROBLEM** : $\text{EvalSemGHW}_k(\mathcal{C})$

**INPUT** : A set $\Sigma \in \mathcal{C}$, a CQ $q(\bar{x})$ that is semantically in $\text{GHW}_k$ under $\Sigma$, a database $D$ such that $D \models \Sigma$, and a tuple $\bar{t} \in \text{dom}(D)^{|\bar{x}|}$.

**QUESTION** : Is $\bar{t} \in q(D)$?

### 9.1 Fixed-parameter Tractable Evaluation

We show that $\text{EvalSemGHW}_k(\mathcal{C})$, for $\mathcal{C} \in \{\mathcal{G}, \mathcal{NR}, \mathcal{S}\}$, as well as $\text{EvalSemGHW}_k(\mathcal{K})$ 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 $\Sigma$, respectively. Recall that fpt means that the problem can be solved in time

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

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

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

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

and thus is fixed-parameter tractable.

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

1. Compute a CQ $q'$ such that $q' \in \text{GHW}_k$ 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 $\mathcal{G}$ is polynomial, while for $\mathcal{NR}$ and $\mathcal{S}$ is exponential. Proposition 2.4 implies that the cost of computing $q'$ and checking if $\bar{t} \in q'(D)$ is:

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

Clearly, the running time of the algorithm is dominated by

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

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, EvalSemGHW$(k)(\mathcal{K})$ can be solved in time

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

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 EvalSemGHW$(k)(\mathcal{C})$ 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 $\mathcal{C}$ 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 $\mathcal{FD}$.

**Theorem 9.3.** EvalSemGHW$(k)(\mathcal{C})$, where $\mathcal{C} \in \{\mathcal{G}, \mathcal{FD}\}$, is in PTime.

The case of FDs is very interesting since, although EvalSemGHW$(k)(\mathcal{FD})$ 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 $\Sigma$ 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 $\Sigma$ of constraints that falls in $\mathcal{G}$ or $\mathcal{FD}$. 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 $\Sigma$ of TGDs or EGDs, and in addition $D \models \Sigma$, evaluating $q$ over $D$ amounts to checking whether chase$(q, \Sigma) \rightarrow_k D$ (see Proposition 9.5 below). Having this technical result in place, Theorem 9.3 is obtained as follows:
We proceed to formalize the above high-level description.

• If $\Sigma \in \mathcal{P}$, then $\text{chase}(q, \Sigma)$ can be computed in polynomial time. This implies that checking whether $\text{chase}(q, \Sigma) \rightarrow_k D$ is feasible in polynomial time, and thus, $\text{EvalSemGHW}_k(\mathcal{P})$ is in PTIME. In fact, this shows that $\text{EvalSemGHW}_k(\mathcal{C})$ is in PTIME for any class $\mathcal{C}$ of sets of TGDs or EGDs for which the chase can be computed in polynomial time.

• If $\Sigma \in \mathcal{G}$, then, although $\text{chase}(q, \Sigma)$ may be infinite, we can prove that:

$$\text{chase}(q, \Sigma) \rightarrow_k D \iff q \rightarrow_k D.$$ 

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, $\text{EvalSemGHW}_k(\mathcal{G})$ 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 $(I, i)$ and $(I', i')$, where $I$ and $I'$ are instances and $i$ and $i'$ are two equally long tuples of terms of $\text{dom}(I)$ and $\text{dom}(I')$, 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 $\bar{t}' = (t_1', \ldots, t_\ell')$

$$h = \{c_i \mapsto d_i\}_{1 \leq i \leq \ell} \cup \{t_i \mapsto t_i'\}_{1 \leq i \leq n}$$

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 $(I, i)$ and $(I', i')$ is denoted $(I, i) \rightarrow_k (I', i')$.

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

2. Given (finite) databases $D$ and $D'$, and tuples $\bar{t} \in \text{dom}(D)^n$ and $\bar{t}' \in \text{dom}(D')^n$, 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 $\Sigma$ of TGDs or EGDs over an instance $I$ that satisfies $\Sigma$ amounts to checking whether $\text{chase}(q, \Sigma) \rightarrow_k I$. 

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

$$\bar{t} \in q(I) \iff \text{chase}(q, \Sigma), \bot(\bar{x}) \rightarrow_k (I, \bar{t}).$$
Efficiently computable chase. A class \( \text{EvalSemGHW}_k \) to show that the same holds for \( \text{EvalSemGHW}_k \) of chase computation, then conclude that \( \overline{\bot}(\Sigma) \) is semantically in \( \text{EvalSemGHW}_k \). Thus, by Lemma 2.5, \( \bot(\Sigma) \in q'(\text{chase}(q, \Sigma)) \). 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 \( \text{EvalSemGHW}_k(C) \) for a class \( C \) of sets of TGDs or EGDs:

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

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

**Proposition 9.6.** For every class \( C \) of TGDs or EGDs that enjoys \textit{tractability of chase computation}, \( \text{EvalSemGHW}_k(C) \) is in PTIME.

Importantly, the class \( \text{FD} \) enjoys tractability of chase computation. This follows from two known facts. Given a CQ \( q \) and a set \( \Sigma \in \text{FD} \): (1) the length of an arbitrary chase sequence for \( D[q] \) under \( \Sigma \) 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 \( \epsilon \), determining all the pairs of atoms in a database that violate \( \epsilon \), and then applying \( \epsilon \) over every such pair, takes polynomial time. Then:

**Proposition 9.7.** \( \text{FD} \) enjoys \textit{tractability of chase computation}.

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

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

\[
(\text{chase}(q, \Sigma), \bot(\bar{x})) \rightarrow_k (I, \bar{t}) \iff (D[q], \bot(\bar{x})) \rightarrow_k (I, \bar{t}).
\]

From Propositions 9.4 and 9.5 we immediately get that:

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

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

**Proposition 9.9.** \( G \) enjoys \textit{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, \( \bar{t} = (t_1, \ldots, t_n) \in \text{dom}(I)^n \), and \( \bar{t}' = (t'_1, \ldots, t'_n) \in \text{dom}(I')^n \), for \( n \geq 0 \), that can be easily obtained...
from a characterization in [17]. An instance \( J \subseteq I \) is a \( k \)-\emph{union} of \( I \) if \( 1 \leq |J| \leq k \), i.e., is a nonempty subinstance of \( I \) with at most \( k \) atoms. For brevity, let \( U_{I,k} \) be the set of all \( k \)-unions of \( I \), and \( H_{I \rightarrow I'} = \bigcup_{J \subseteq I} \{ h \mid h \text{ is a homomorphism from } J \text{ to } I' \} \). Two homomorphisms \( h \) and \( h' \) are \textit{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 \( \mu : U_{I,k} \rightarrow H_{I \rightarrow I'} \) is called \textit{winning strategy} for the duplicator in the existential \( k \)-cover game on \(( I, \hat{t} \) and \( (I', \hat{t}') \) if, for each \( J \in U_{I,k} \):

1. \( \mu(J) = \{ h \in H_{I \rightarrow I'} \mid h(J) \subseteq I' \text{ and } t_i \in \text{dom}(J) \implies h(t_i) = t'_i \} \), 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.** \(( I, \hat{t} \rightarrow_k (I', \hat{t}') \) if and only if there exists \( \mu : U_{I,k} \rightarrow H_{I \rightarrow I'} \) that is a winning strategy for the duplicator in the existential \( k \)-cover game on \(( I, \hat{t} \) and \( (I', \hat{t}') \).**

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

\[
\text{(chase}(q, \Sigma), \sqsubseteq(\bar{x})) \rightarrow_k (I, \bar{t}) \iff \text{ (chase}(D[q], \sqsubseteq(\bar{x})) \rightarrow_k (I, \bar{t}).
\]

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

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

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

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

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

We show that there are functions \( \mu_j : U_{I_j, k} \rightarrow H_{I_j \rightarrow I_{j+1}} \) such that, for each \( j \geq 0 \):

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

Having the above sequence of functions in place, the claim follows with \( \mu' = \bigcup_{j \geq 0} \mu_j \). The proof that \( (\mu_j) \geq 0 \) exists 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 \( \mathcal{C} \) of TGDs considered above, i.e., \( \mathcal{NR} \) and \( \mathcal{S} \), the question whether \( \text{EvalSemGHW}_k(\mathcal{C}) \) 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 \( \text{GHW}_k \)-approximation of a CQ under a set of constraints. Given a CQ \( q \), and a set \( \Sigma \) of TGDs that falls in \( \mathcal{G} \) or \( \mathcal{NR} \) or \( \mathcal{S} \), we can exploit the techniques developed in Sections 5 and 6 in order to compute the CQs in \( \text{GHW}_k \), for \( k \geq 1 \), that are maximally contained in \( q \) and \( \Sigma \). Following the recent database literature, such CQs correspond to the \( \text{GHW}_k \)-approximations of \( q \) under \( \Sigma \); see, e.g., [6–8]. Computing and evaluating the \( \text{GHW}_k \)-approximations of \( q \) under \( \Sigma \) 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 $\Sigma$; in the rest of the section, we fix $k \geq 1$:

**Definition 10.1.** Consider a CQ $q$ and a set $\Sigma$ of TGDs, both over a schema $\sigma$. A $GHW_k$-approximation of $q$ under $\Sigma$ is a CQ $q' \in GHW_k$ over $\sigma$ such that:

1. $q' \subseteq_{\Sigma} q$, and
2. for every $q'' \in GHW_k$, $q' \subseteq_{\Sigma} q'' \subseteq_{\Sigma} q$ $\implies$ $q' \equiv_{\Sigma} q''$.

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

Intuitively, condition (1) corresponds to soundness, i.e., $q'$ only returns sound answers w.r.t. $q$ and $\Sigma$, while condition (2) corresponds to maximality, i.e., there is no CQ $q'' \in GHW_k$ that approximates $q$ better than $q'$ in terms of containment under $\Sigma$. Notice that whenever $q$ is semantically in $GHW_k$ under $\Sigma$, i.e., there is a CQ $q' \in GHW_k$ such that $q \equiv_{\Sigma} q'$, then the unique $GHW_k$-approximation of $q$ under $\Sigma$ 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 $\Sigma$ 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 $\Sigma$ of TGDs, let $g_C(q, \Sigma) = |q| \cdot (2k + 1)$, and $g_C(q, \Sigma) = f_C(q, \Sigma) \cdot (2k + 1)$ if $C \subseteq \{\text{NR}, \Sigma\}$; recall that the functions $f_\text{NR}$ and $f_\Sigma$ are given by Propositions 6.7 and 6.9, respectively.

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

1. $\text{approx}_{GHW_k}(q, \Sigma) \neq \emptyset$.
2. $|q'| \leq g_C(q, \Sigma)$, for each $q' \in \text{approx}_{GHW_k}(q, \Sigma)$.
3. $\text{approx}_{GHW_k}(q, \Sigma)$ 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 $q, q' \in GHW_k$ over a schema $\sigma$, and $\Sigma \in \mathcal{C}$ over $\sigma$, where $\mathcal{C} \subseteq \{G, \text{NR}, \Sigma\}$, such that $q' \subseteq_{\Sigma} q$. There is a CQ $q'' \in GHW_k$ over $\sigma$ such that $q' \subseteq_{\Sigma} q'' \subseteq_{\Sigma} q$ and $|q''| \leq g_C(q, \Sigma)$.

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 $\Sigma \in \mathcal{C}$ of TGDs, we write $\text{cont}_{GHW_k}(q, \Sigma)$ for the set of CQs $q' \in GHW_k$ such that $q' \subseteq_{\Sigma} q$ and $|q'| \leq g_C(q, \Sigma)$. Moreover, we write $\text{maximal}_{GHW_k}(q, \Sigma)$ for the $\subseteq_{\Sigma}$-maximal elements of $\text{cont}_{GHW_k}(q, \Sigma)$. In other words, $\text{maximal}_{GHW_k}(q, \Sigma)$ consists of the CQs $q' \in \text{cont}_{GHW_k}(q, \Sigma)$ for which there is no $q'' \in \text{cont}_{GHW_k}(q, \Sigma)$ such that $q' \subseteq_{\Sigma} q''$, where the latter notation means that $q' \subseteq_{\Sigma} q''$ and, in addition, there exists an instance $I$ that satisfies $\Sigma$ such that $q'(I) \subsetneq q''(I)$.

**Proof.** (of Theorem 10.2) We first observe that there exists a CQ $q' \in GHW_k$ 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

$$q'(x, \ldots, x) := R_1(x, \ldots, x) \land \cdots \land R_m(x, \ldots, x).$$

J. ACM, Vol. 1, No. 1, Article . Publication date: September 2020.
It is clear that $q'$ is acyclic, i.e., $q' \in \text{GHW}_1$, which implies that $q' \in \text{GHW}_k$. Moreover, $q' \subseteq \Sigma$ since $(\perp, \ldots, \perp)$ belongs to the evaluation of $q$ over chase$(q', \Sigma)$, and $|q'| \leq |q|$ since $m \leq |q|$. This immediately implies that the set $\text{cont}_{\text{GHW}_k}(q, \Sigma)$ is non-empty, and thus, maximal$_{\text{GHW}_k}(q, \Sigma)$ is non-empty. We now show that the set maximal$_{\text{GHW}_k}(q, \Sigma)$ consists of all the GHW$_k$-approximations of $q$ under $\Sigma$ (up to $\equiv_{\Sigma}$). Formally:

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

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

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

We now show that (†) holds. Consider a CQ $q'' \in \text{maximal}_{\text{GHW}_k}(q, \Sigma)$. We are going to show that $q' \in \text{approx}_{\text{GHW}_k}(q, \Sigma)$. Fix an arbitrary CQ $q'' \in \text{GHW}_k$. 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 $\tilde{q} \in \text{GHW}_k$ such that $q' \subseteq \Sigma q'' \subseteq \tilde{q} \subseteq \Sigma q$, and $|\tilde{q}| \leq g_{\Sigma}(q, \Sigma)$. Clearly, $\tilde{q} \in \text{cont}_{\text{GHW}_k}(q, \Sigma)$. Since $q' \in \text{maximal}_{\text{GHW}_k}(q, \Sigma)$, either $\tilde{q} \subseteq q'$, or $q' \not\subseteq \tilde{q}$ and $\tilde{q} \not\subseteq q'$. Notice that the latter case contradicts the fact that $q' \subseteq \Sigma \tilde{q}$. Thus, the only valid case is $\tilde{q} \subseteq q'$. Therefore, $q' \equiv_{\Sigma} \tilde{q}$, 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 maximal$_{\text{GHW}_k}(q, \Sigma)$ is non-empty since cont$_{\text{GHW}_k}(q, \Sigma)$ is non-empty. Thus, approx$_{\text{GHW}_k}(q, \Sigma) \neq \emptyset$ since maximal$_{\text{GHW}_k}(q, \Sigma)$ consists of all the GHW$_k$-approximations of $q$ under $\Sigma$ (up to $\equiv_{\Sigma}$), and (1) follows. Item (2) follows by definition, since each CQ $q' \in \text{maximal}_{\text{GHW}_k}(q, \Sigma)$ consists of at most $g_{\Sigma}(q, \Sigma)$ atoms. Finally, for item (3) it suffices to show that the set maximal$_{\text{GHW}_k}(q, \Sigma)$ can be computed in double-exponential time in $|q|| + ||\Sigma||$. This is done by simply enumerating all CQs $q'$ such that $|q'| \leq g_{\Sigma}(q, \Sigma)$, and for each one we check that (a) $q' \in \text{GHW}_k$, (b) $q' \subseteq \Sigma q$, and (c) there is no CQ $q'' \in \text{GHW}_k$ such that $q' \subseteq \Sigma q'' \subseteq \Sigma q$ and $|q''| \leq g_{\Sigma}(q, \Sigma)$. In general, we have to consider double-exponentially many queries (in fact, exponentially many when $\Sigma = \emptyset$), 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 $\Sigma \in \mathfrak{C}$ of TGDs, where $\mathfrak{C} \in \{\emptyset, \mathfrak{N}, \mathfrak{S}\}$, a database $D$ such that $D \models \Sigma$, and a tuple $\bar{t} \in \text{dom}(D)^{|\bar{x}|}$, decide whether there exists a GHW$_k$-approximation $q'$ of $q$ under $\Sigma$ such that $\bar{t} \in q'(D)$. Since each such $q'$ is contained in $q$ under $\Sigma$, we can then be sure that $\bar{t}$ is a sound answer to $q$ over $D$. By Theorem 10.2, the set approx$_{\text{GHW}_k}(q, \Sigma)$ can be computed in double-exponential time in $|q|| + ||\Sigma||$, while each GHW$_k$-approximation of $q$ under $\Sigma$ 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 $\Sigma$ takes time:

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

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 $\Sigma$ of TGDs that falls in $\mathfrak{G}$ or $\mathfrak{N}$ or $\mathfrak{S}$ is fixed-parameter tractable. Thus, as said above, computing and evaluating the GHW$_k$-approximations of $q$ under $\Sigma$ 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.

J. ACM, Vol. 1, No. 1, Article . Publication date: September 2020.
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 \( \text{GHW}_k \), 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 \( \text{GHW}_k \) over all those databases that satisfy the given set of constraints. This problem has been dubbed \( \text{SemGHW}_k \).

Surprisingly, we have shown that there are classes of constraints for which containment is decidable, while \( \text{SemGHW}_1 \) 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 \( \text{SemGHW}_k \) is decidable, and obtained several complexity results. We have also shown that \( \text{SemGHW}_k \) 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 \( \text{GHW}_k \) under a set \( \Sigma \) of constraints over a database that satisfies \( \Sigma \). We have shown that whenever \( \Sigma \) 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 \( \text{GHW}_k \), via a CQ in \( \text{GHW}_k \), 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 \( \text{SemGHW}_k \) 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 \( \text{SemGHW}_k \) under sticky sets of TGDs is still unknown. Recall that we have a \( \text{NExpTime} \) upper bound and an \( \text{ExpTime} \) 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 \( \text{SemGHW}_k \) under sticky sets of TGDs is in \( \text{ExpTime} \).
2. The exact complexity of \( \text{SemGHW}_k \) 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 \( 2\text{ExpTime} \) by exploiting tree-walking automata.
3. The decidability status of \( \text{SemGHW}_k \) under arbitrary key or functional dependencies, without any assumption on the underlying schema, is still open. We conjecture that in this case \( \text{SemGHW}_1 \) is undecidable. However, it seems that this is a highly non-trivial result. Let us stress that the EGDs employed in the proof that \( \text{SemGHW}_1 \) 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 \( \text{GHW}_k \) 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 \( \text{GHW}_k \), via a CQ that falls in \( \text{GHW}_k \) 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

Semantic Optimization of Conjunctive Queries


A ADDITIONAL PROOFS FROM SECTION 4

THEOREM A.4. $\text{SemGHW}_1(\mathbb{F})$ 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 $\sigma$ described above, 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 $q \equiv_{\Sigma} q'$. 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 $\sigma$, i.e., $D[q] = \text{chase}(q, \sigma)$ (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 $\text{GHW}_1$. The proof exploits the syntactic properties of $q$ summarized in Claim A.1, and the fact that $D[q] = \text{chase}(q, \sigma)$ 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 $2(n^2+2n+2)$:

$$\bigwedge_{1 \leq i \leq 2(n^2+2n+2)} \text{start}(y, x_i) \land \text{end}(x_i, z) \land \bigwedge_{1 \leq i, j \leq 2(n^2+2n+2)} \eta_1(x_i, x_j) \land \eta_2(x_i, x_j) \land \eta(x_i, x_j) \land a(x_i, x_{i+1}) \land a(x_{i+2}, x_i) \land b(x_i, x_{i+3}) \land b(x_{i+4}, x_i) \land \bigwedge_{1 \leq i \leq 2(n^2+2n+2)} \bigvee_{1 \leq j \leq n} \bigwedge_{1 \leq j \leq 2(n^2+2n+2)} \bigvee_{1 \leq j, \ell \leq n} \delta_j, \ell(x_i, x_{i+2(j-1)n+2\ell+4n+4}, x_i)$$

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 $\eta_1, \eta_2$, and $\eta$. Finally, each variable $x_i$, for $i \in \{1, \ldots, 2(n^2+2n+2)\}$, has outgoing edges labeled $a$, $b$, $\delta_j, \ell$, $\phi_{j, \ell}$, 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 $\{a, b, (\delta_{k, \ell})_{1 \leq k \leq n}, (\phi_{k, \ell})_{1 \leq k \leq n}, (\phi_{k, \ell})_{1 \leq k, \ell \leq n}\}$, 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 $\{a, b, (\delta_{k, \ell})_{1 \leq k \leq n}, (\phi_{k, \ell})_{1 \leq k \leq n}, (\phi_{k, \ell})_{1 \leq k, \ell \leq n}\}$. In other words, $q$
We denote by \( u \) the restriction of \( \mu \) to the alphabet \( \{ a, b \} \), by \( \eta \), \( \eta_1 \), and \( \eta_2 \) 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

\[
\bigcup_{1 \leq i < j \leq 2(n^2+2n+2)} \{ R(x_i, x_j), R(x_j, x_i) \mid R \in \{ a, b, (\diamond k), 1 \leq k \leq n, (\diamond k, \ell), 1 \leq k, \ell \leq n \} \}.
\]

(3) Each self-loop is labeled only with \( \eta_1 \), \( \eta_2 \), and \( \eta \). 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 \( \Sigma \).

**The Set \( \Sigma \) 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 \(\{\diamond_j, \ast \mid 1 \leq j \leq n\}\) at non-consecutive positions.

Let \(w\) be a word of the form \(t_1 \ldots t_m\), where each \(t_j\), for \(1 \leq j \leq m\), is a symbol in
\[
\{a, b, \diamond_{s,k}, \diamond_{k,s}, \diamond_{k,l}, \text{start, end} \mid 1 \leq k, l \leq n\}.
\]

In our TGDs, we write \(w(x, y)\) as an abbreviation for \(t_1(z_1), \ldots, t_m(z_{m-1}, y)\), 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:
\[
\text{start}(x, y), \eta_1(y, z), \eta_2(y, z') \rightarrow \eta(z, z'), \eta(z', z).
\]

That is, if a variable \(y\) has an incoming edge labeled start and outgoing edges labeled \(\eta_1\) and \(\eta_2\) pointing to variables \(z\) and \(z'\), respectively, then \(z\) and \(z'\) are linked to each other via edges labeled \(\eta\). The recursive rules that define the predicate \(\eta\), 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 \(r, r' \in \{\ast, 1, \ldots, n\}\), 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 \(u'_i\) and \(v'_i\), 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 Extended\((u_i)\) instead of \(u_i\) itself (resp., a word \(v'_i\) in Extended\((v_i)\) instead of \(v_i\)) is because the reading of \(u_i\) from \((x_1, y_1)\) might need to “jump” over several occurrences of symbols of the form \(\diamond_{i,j}\) (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 Extended\((u_i)\) (resp., to 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:
\[
\text{start}(y, x), \eta_1(z_1, z'_1), \eta_2(z_2, z'_2), \eta(z'_1, z'_2), u'_i(z_1, x'_1), v'_i(z_2, x'_2), \diamond_{i,i}(x'_1, x_1),
\]
\[
a(x_1, x_2), \ldots, a(x_2(n^2+2n+2)−1, x_2(n^2+2n+2)), \text{end}(x_2(n^2+2n+2), z) \rightarrow q,
\]

where \(q\) is an abbreviation for the set of atoms in \(q\) (which are defined over the variables \(y, z, x_1, \ldots, x_2(n^2+2n+2)\), 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 \(u_i\) and \(v_i\) 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 \(\Sigma\) can be effectively constructed since, as explained above, for each \(i \in \{n\}\), the sets Extended\((u_i)\) and Extended\((v_i)\) are finite. We now establish a simple claim, which states that \(q\) is closed under the applications of the TGDs in \(\Sigma\), that will be used in the correctness proof.

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

J. ACM, Vol. 1, No. 1, Article . Publication date: September 2020.
We first explain how to represent solutions to the PCP instance with a word over the alphabet $z$, $w$, and $s$. We need to show that $h$ also maps $q$ to $D[q]$. It is clear that $h(y) = \bot_y$ and $h(z) = \bot_z$. The procedure is defined. Then, we consider three cases:

- If $c_1 = \circ_{i,*}$ and $c_2 = \circ_{s,i}$, for $1 \leq i, j \leq n$, then we write the symbol $\circ_{i,j}$ to the output tape and move all heads one cell to the right. This implies that there exists $1 \leq j \leq 2(n^2 + n + 2)$ such that, for each $1 \leq i \leq 2(n^2 + n + 2)$, $h(x_1) = \bot_{x_{i+1}}$. It is also not difficult to see that, for each $1 \leq j \leq 2(n^2 + n + 2)$, there is an isomorphism from $q$ to $D[q]$ that maps $y$ and $z$ to $\bot_y$ and $\bot_z$, respectively, and, for each $1 \leq i \leq 2(n^2 + n + 2)$, it maps $x_i$ to $\bot_{x_{i+1}}$. 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 $GWH_1$ under $\Sigma$.

The Correctness Proof

We first explain how to represent solutions to the PCP instance with a word over the alphabet $\{a, b, (\circ_{i,*})_{1 \leq i \leq n}, (\circ_{s,i})_{1 \leq i \leq n}, (\circ_{l,j})_{1 \leq i, j \leq n}\}$. 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:

$$w_1^s := u_{i_1} \circ_{i_1,*} \ldots u_{i_m} \circ_{i_m,*} \quad \text{and} \quad w_2^s := v_{i_1} \circ_{s,i_1} \ldots v_{i_m} \circ_{s,i_m}.$$ 

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 $\circ_{i,*}$, or the symbol $c_2$ read on the second tape is of the form $\circ_{s,i}$, 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 $c_1 = \circ_{l,*}$ and $c_2 = \circ_{s,j}$, for $1 \leq i, j \leq n$, then we write the symbol $\circ_{l,j}$ to the output tape and move all heads one cell to the right.
- If $c_1 = \circ_{l,*}$, for $1 \leq i \leq n$, while $c_2$ is either $a$ or $b$, then we write the symbol $\circ_{l,*}$ 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 $c_2 = \circ_{s,j}$, for $1 \leq i \leq n$, while $c_1$ is either $a$ or $b$, then we write the symbol $\circ_{s,j}$ 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:

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

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

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

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

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

J. ACM, Vol. 1, No. 1, Article . Publication date: September 2020.
The key ingredient of the correctness proof is Lemma A.3 below. We first need some definitions.

Let \( w \) be a word over \( \{a, b, (\bullet_{i,j})_{1 \leq i \leq n}, (\circ_{i,j})_{1 \leq i \leq n}, (\diamond_{i,j})_{1 \leq i, j \leq n}\} \) and consider a directed path \( \begin{array}{c}
\text{start} \\
\downarrow \eta_1
\end{array} \begin{array}{c} y \\
\rightarrow \begin{array}{c} x_1 \rightarrow x_2 \rightarrow \cdots \rightarrow x_{2(n^2+2n+2)-1} \rightarrow \end{array} \begin{array}{c} a \\
\rightarrow \begin{array}{c} x_{2(n^2+2n+2)} \\
\end{array} \end{array} \end{array} \begin{array}{c} \text{end} \\
\downarrow \eta_n \\
z
\end{array} \)

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:

- \( x \xrightarrow{\eta_1} x_1 \) and \( x \xrightarrow{\eta_2} x, \) where \( x_1 \) and \( x_2 \) are fresh variables.
- For each variable \( p \) in the path from \( x \) to \( x_1 \) with an incoming edge labeled \( \bullet_{i,j}, \circ_{i,j}, \) or \( \diamond_{i,j}, \) for \( 1 \leq i, j \leq n \), we add the edges \( p \xrightarrow{\eta_1} p_1 \) and \( p \xrightarrow{\eta_2} p, \) where \( p_1 \) and \( p_2 \) 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, i.e., is acyclic. Figure 12 shows a graphical depiction of the initial part of \( q[w], \) from \( y \) to \( x_1 \), for \( w = bb \circ_{1,1} ab \circ_{2,1} b \circ_{3,2} b \circ_{3,3} a \circ_{4,3} b \circ_{4,4}. \)

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 \( S = \text{chase}(q[w_1^1 \otimes w_2^1], \Sigma), \) where \( \text{dom}(S) = \{y, z, x_1, \ldots, x_{2(n^2+2n+2)}\}, \) and a bijection \( i : \text{dom}(S) \rightarrow \text{dom}(D[q]), \) such that \( i(S) = D[q]. \)
2. Let \( w \) be a word over the alphabet \( \{a, b, (\bullet_{i,j})_{1 \leq i \leq n}, (\circ_{i,j})_{1 \leq i \leq n}, (\diamond_{i,j})_{1 \leq i, j \leq n}\}, \) and a bijection \( i : \text{dom}(S) \rightarrow \text{dom}(D[q]), \) such that \( i(S) = D[q], \) then the PCP instance has a solution.

**Proof**. We first prove (1). Assume that the solution \( s \) is given by the sequence \( 1 \leq i_1 \ldots i_m \leq n. \) It is clear that \( \eta(\bot_{x_{i_1}}, \bot_{x_{i_2}}, \ldots, \bot_{x_{i_m}}) \in \text{chase}(q[w_1^1 \otimes w_2^1], \Sigma), \) due to the initialization rule, since the atoms start(\( \bot_{x_1}, \bot_{x_2} \)) and \( \eta(\bot_{x_1}, \bot_{x_2}) \) occur in the canonical instance of \( q[w_1^1 \otimes w_2^1]. \) From such an atom, the matching rules start iterating over the path (in the canonical instance of \( q[w_1^1 \otimes w_2^1] \)) from \( \bot_x \) to \( \bot_{y_j}, \) labeled \( w_1^1 \otimes w_2^1. \) We claim the following: Over such a path, the matching rules generate all the atoms of the form \( \eta(\bot_{y_{j_1}}, \bot_{y_{j_2}}), \) for \( 1 \leq j \leq m, \) where the nulls \( \bot_{y_{j_1}} \) and \( \bot_{y_{j_2}} \) are the ones that satisfy the following conditions. Suppose that the path from \( \bot_x \) to \( \bot_{y_1} \) is labeled \( \lambda, \) and the path from \( \bot_x \) to \( \bot_{y_{j+1}} \) is labeled \( \lambda', \) then:

- The restriction of \( \lambda \) to the alphabet \( \{a, b\} \) is precisely \( u_{i_1} \ldots u_{i_{r-1}} \), and the last symbol of \( \lambda \) is of the form \( \diamond_{r, r}, \) for \( r \in \{1, \ldots, n\}. \)
- The restriction of \( \lambda' \) to the alphabet \( \{a, b\} \) is precisely \( v_{i_1} \ldots v_{i_{n+1}}, \) and the last symbol of \( \lambda' \) is of the form \( \diamond_{r, r}, \) for \( r' \in \{1, \ldots, n\}. \)

J. ACM, Vol. 1, No. 1, Article . Publication date: September 2020.
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 \(w_1^j \otimes w_2^j\) that are naturally associated with \(\perp y_j\) and \(\perp z_j\), respectively).

**Base case.** We assume that \(u_{i_j}\) is shorter than \(u_{i_i}\) – the other case can be treated in an analogous way. By construction, \(w_1^j \otimes w_2^j\) has a prefix of the form \(u_{i_i} \otimes_{i_i} u'_{i_i} \otimes_{r, i_i}\), where \(u_{i_i} \otimes_{i_i} u'_{i_i} \in \text{Extended}(u_{i_i})\) and \(r \in \{*, 1, \ldots, n\}\). Therefore, the path from \(\perp_x\) to \(\perp y_1\), in the canonical instance of \(q[w_1^j \otimes w_2^j]\), contains an initial path of the form:

\[
\perp_x u_{i_i} \otimes_{i_i} \star \perp y_1 u'_{i_i} \otimes_{r, i_i} \perp z_i.
\]

Observe that the the nulls \(\perp y_1\) and \(\perp z_i\) satisfy the conditions stated above. Moreover, since \(u_{i_i} \otimes_{i_i} u'_{i_i}\) belongs to \(\text{Extended}(u_{i_i})\), the matching rule:

\[
\eta_1(x, x_1), \eta_2(x, x_2), \eta(x_1, x_2), u_{i_i}(x, y_1), \otimes_{i_i} (y_1, y_2), \eta_1(y_1, y_2), [u_{i_i} \otimes_{i_i} u'_{i_i} (x, z_1)] \rightarrow \eta(y_1, z_1)
\]

is triggered, and the atom \(\eta(\perp y_1, \perp z_i)\) is generated.

**Inductive step.** By induction hypothesis, the atom \(\eta(\perp y_1, \perp z_i)\) has been generated during the chase. By the construction of \(w_1^j \otimes w_2^j\), and the definition of \(\perp y_j\) and \(\perp z_j\), the subword of \(w_1^j \otimes w_2^j\) from position \(y_j\) to \(y_{j+1}\) is the word \(u'_{i_{j+1}} \otimes_{i_{j+1}, r}\) such that \(u'_{i_{j+1}} \in \text{Extended}(u_{i_{j+1}})\) and \(r \in \{*, 1, \ldots, n\}\). Analogously, the subword of \(w_1^j \otimes w_2^j\) from \(z_j\) to \(z_{j+1}\) is the word \(v'_{i_{j+1}} \otimes_{r', i_{j+1}}\) such that \(v'_{i_{j+1}} \in \text{Extended}(v_{i_{j+1}})\) and \(r' \in \{*, 1, \ldots, n\}\). Therefore, the matching rule:

\[
\eta_1(y_j, y_{j+1}), \eta_2(z_j, z_{j+1}), \eta(y_{j+1}, z_{j+1}), u'_{i_{j+1}}(y_j, y_{j+1}), \otimes_{i_{j+1}, r}(y_{j+1}, y_{j+2}), \eta_1(y_{j+1}, y_{j+2})
\]

\[\rightarrow \eta(y_{j+1}, z_{j+1})\]

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

From the above discussion, we conclude that \(\eta(\perp y_{m}, \perp z_{m})\) is in \(\text{Chase}(q[w_1^j \otimes w_2^j], \Sigma)\). We proceed to show that the finalization rule will be triggered and generate an isomorphic copy \(\Sigma\) of \(D[q]\) with \(\text{dom}(\Sigma) = \{\perp y, \perp z, \perp x_1, \ldots, \perp x_{2(n^2+2n+2)}\}\). We assume that \(u_{i_{m}}\) is shorter than \(u_{i_{m}}\) (the other case is analogous). By the construction of \(w_1^j \otimes w_2^j\), and the definition of \(\perp y_{m}\) and \(\perp z_{m}\), the subword of \(w_1^j \otimes w_2^j\) from \(y_{m}\) to the last position of \(w_1^j \otimes w_2^j\) is the word \(u'_{i_{m}} \otimes_{i_{m}, r}\) such that \(u'_{i_{m}} \in \text{Extended}(u_{i_{m}})\) and \(r \in \{*, 1, \ldots, n\}\). Analogously, the subword of \(w_1^j \otimes w_2^j\) from \(z_{m}\) to the last position of \(w_1^j \otimes w_2^j\) is the word \(v'_{i_{m}} \otimes_{r', i_{m}}\) such that \(r' \in \{*, 1, \ldots, n\}\). We conclude that \(r = r' = i_{m}\). Therefore, the finalization rule:

\[
\text{start}(y_{m}, x_{m}), \eta_1(y_{m}, y_{m}), \eta_2(z_{m}, z_{m}), \eta(y_{m}, z_{m}), u'_{i_{m}}(y_{m}, x_{m}), v'_{i_{m}}(m, x_{m}, z_{m}), \otimes_{i_{m}, i_{m}}(x_{m}, x_{m}), a(x_{m}, x_{m}), \ldots, a(x_{2(n^2+2n+2)+1}, x_{2(n^2+2n+2)+2})\]

\[\rightarrow q_{m}, \text{end}(x_{2(n^2+2n+2)+2}, z) \rightarrow q_{m}\]

is triggered, and generates in \(\text{Chase}(q[w_1^j \otimes w_2^j], \Sigma)\) an isomorphic copy \(\Sigma\) of \(D[q]\), where \(\text{dom}(\Sigma) = \{\perp y, \perp z, \perp x_1, \ldots, \perp x_{2(n^2+2n+2)}\}\), as needed.

We now prove (2). Assume there is an isomorphic copy \(\Sigma\) of \(D[q]\), where \(\text{dom}(\Sigma) = \{\perp y, \perp z, \perp x_1, \ldots, \perp x_{2(n^2+2n+2)}\}\), in \(\text{Chase}(q[w], \Sigma)\). This implies that the finalization rule was triggered during the construction of \(\text{Chase}(q[w], \Sigma)\). Let \(w_1\) be the word that is obtained from \(w\) by (i) removing each occurrence of a symbol of the form \(\otimes_{i, i}\), and (ii) changing each occurrence of a symbol of the form \(\otimes_{i, j}\) by \(\otimes_{j, j}\). 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 \( \diamond_{i,j} \), and (ii) changing each occurrence of a symbol of the form \( \diamond_{i,*} \). We claim that \( w_1 \) and \( w_2 \) are of the following form:

\[
w_1 = u_{i_1} \diamond_{i_1,*} \cdots u_{i_m} \diamond_{i_m,*} \quad \text{and} \quad w_2 = u_{i_1} \diamond_{*,i_1} \cdots u_{i_m} \diamond_{*,i_m},
\]

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 \( \perp_x \) to \( \perp_{x_1} \), 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 \( \mathsf{GHW}_1 \) under \( \Sigma \).

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

\[
\begin{align*}
\text{start } w_1 \otimes w_2 & \quad a a a \cdots a \quad \text{end.} \\
& 2(n^2 + 2n + 2) \text{ times}
\end{align*}
\]

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

\[
\begin{align*}
\text{start } w_1 \otimes w_2 & \quad a a a \cdots a \\
& 2(n^2 + 2n + 2) \text{ times}
\end{align*}
\]

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 \( \eta_1 \) and \( \eta_2 \) 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 \( \text{chase}(q, \Sigma) \). On the other hand, by Lemma A.3, \( \text{chase}(q[w_1^s \otimes w_2^s], \Sigma) \) contains an isomorphic copy of \( D[q] \), which immediately implies that \( D[q] \) can be homomorphically mapped to \( \text{chase}(q[w_1^s \otimes w_2^s], \Sigma) \).

\[(\Leftarrow) \text{ Assume that there is an acyclic CQ } q' \text{ such that } q \equiv_{\Sigma} q'. \text{ To show that the PCP instance has a solution, it suffices to show that } q' \text{ contains a subquery of the form } q[w], \text{ where } w \text{ 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 \leq j \leq n}\}, \text{ such that } \text{chase}(q[w], \Sigma) \text{ contains an isomorphic copy } S \text{ of } D[q] \text{ with } \text{dom}(S) = \{\perp_y, \perp_z, \perp x_1, \ldots, \perp x_{2(n^2 + 2n + 2)}\}. \text{ 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 \( \text{chase}(q', \Sigma) \), and \( D[q'] \) can be homomorphically mapped to \( \text{chase}(q, \Sigma) \). Since, by Claim A.2, \( D[q] = \text{chase}(q, \Sigma) \), 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 \( \text{chase}(q', \Sigma) \), we get that \( \text{chase}(q', \Sigma) \) 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 \( \Sigma \), it is the case that \( \text{chase}(q', \Sigma) \) 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 \( \perp_y \), while an edge in \( D[q] \) is labeled end if and only if it reaches the sink node \( \perp_z \). Therefore, since there is a homomorphism from \( D[q'] \) to \( D[q] \), an edge in \( D[q'] \) is labeled start if

\[\text{J. ACM, Vol. 1, No. 1, Article . Publication date: September 2020.}\]
and only it leaves the root element \( \bot_r \). 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(\( q', \Sigma \)). Then, \( h(\bot_x) \) must correspond to a leaf \( \bot_{z'} \) 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 \( h(\bot_{x_i}) \) is a term of \( \text{dom}(D[q']) \) that has (i) an outgoing edge labeled end pointing out to \( \bot_{z'} \), 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

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

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 \( \bot_r \) to the leaf \( \bot_{z'} \) is of the form:

\[
\bot_r \xrightarrow{w'} \bot_{z_2} \xrightarrow{v_i'} \bot_{z_1} \xrightarrow{u_i} \bot_{x_1} \xrightarrow{a} \bot_{x_2} \ldots \xrightarrow{a} \bot_{x_{2(n^2 + 2n + 2) - 1}} \xrightarrow{a} \bot_{x_{2(n^2 + 2n + 2)}}, 
\]

assuming without loss of generality that \( u_1 \) is shorter than \( v_1 \) and that \( v_1' = v_2''u_1 \). 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 \( \eta \), 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 \( \eta_1(\bot_{z_1}, \bot_{z_1'}) \) and \( \eta_2(\bot_{z_2}, \bot_{z_1'}) \). But the atom \( \eta(\bot_{z_1}, \bot_{z_1'}) \) also belongs to chase(\( q', \Sigma \)), and this atom could have only been generated during the chase. Assume otherwise, i.e., \( \eta(\bot_{z_1}, \bot_{z_1'}) \in D[q'] \). Then \( D[q'] \) contains a path from \( \bot_{z_2} \) to \( \bot_{z_1} \) labeled \( v_1'' \), and another one labeled \( \eta_1(\eta_2)^- \), where \( (\eta_2)^- \) represents a backward traversal of an edge labeled \( \eta_2 \). These paths are disjoint, apart from the external nodes. This contradicts the fact that \( D[q'] \) is acyclic. Therefore, \( \eta(\bot_{z_1}, \bot_{z_1'}) \) was either generated by the initialization rule, or the recursive matching rules. By recursively applying this reasoning, we get that the unique path from \( \bot_r \) to \( \bot_{z'} \) in \( D[q'] \) is

\[
\bot_r \xrightarrow{w} \bot y \xrightarrow{a} \bot x \xrightarrow{a} \bot_{x_{2(n^2 + 2n + 2)}} \xrightarrow{a} \bot_{x_{2(n^2 + 2n + 2)}}, 
\]

where \( w \) is a word over the alphabet \( \{a, b, (\eta_1, \eta_2), \eta_1, \eta_2, (\eta_1, \eta_2)\} \). Moreover, we can assume, without loss of generality, that each term in the path from \( \bot_X \) to \( \bot_Y \) with an incoming edge labeled over \( \{(\eta_1, \eta_2), (\eta_1, \eta_2), (\eta_1, \eta_2)\} \) has outgoing edges labeled \( \eta_1 \) and \( \eta_2 \) 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 \( \{a, b, (\eta_1, \eta_2), (\eta_1, \eta_2), (\eta_1, \eta_2)\} \), 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 \( \{a, b, (\eta_1, \eta_2), (\eta_1, \eta_2), (\eta_1, \eta_2)\} \). 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 \( \eta_1 \), \( \eta_2 \), or \( \eta \).

The problem with our “backchase” analysis now is that \( D[q'] \) might contain “artificial” matching edges labeled \( \eta \), 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 \( \eta_1, \eta_2 \) or \( \eta \) (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.** \( \text{SemGHW}_1(\mathcal{BD}) \) 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:

\[
\text{start}(x, y), \eta_1(y, z), \eta_2(y, w) \rightarrow \eta(z, w), \eta(w, z).
\]

by the initialization EGD:

\[
\text{start}(x, y), \eta_1(y, z), \eta_2(y, w) \rightarrow z = w,
\]

and each matching rule of the form:

\[
\eta_1(x_1, x), \eta_2(y_1, y), \eta(x, y), u'_1(x_1, x'), \odot_{i,r}(x^*, x_2), \eta_1(x_2, x'), \nu'_1(y_1, y'), \odot_{r,i}(y^*, y_2), \eta_2(y_2, y') \rightarrow \eta(x', y').
\]

by a matching EGD of the form:

\[
\eta_1(x_1, x), \eta_2(y_1, x), u'_1(x_1, x'), \odot_{i,r}(x^*, x_2), \eta_1(x_2, x'), \nu'_1(y_1, y'), \odot_{r,i}(y^*, y_2), \eta_2(y_2, y') \rightarrow x' = y'.
\]

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:

\[
\begin{align*}
\text{start}(y, x), \eta_1(z_1, z'_1), \eta_2(z_2, z'_2), \eta(z'_1, z'_2), u'_1(z_1, x'_1), \nu'_1(z_2, x'_2), \odot_{i,l}(x'_1, x_1), \\
a(x_1, x_2), \ldots, a(x_2(n^2+2n+2)-1, x_2(n^2+2n+2)), \eta_1(x_2(n^2+2n+2), z) \rightarrow q,
\end{align*}
\]

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

Assume, on the other hand, that there is an acyclic CQ \( q' \) such that \( q \equiv_{\Sigma^\omega} q' \). As before, we show that \( q' \) must contain a subquery of the form \( q[w] \), where \( w \) is a word over the alphabet \( \{a, b, (\odot_{i,r})_{1 \leq i \leq m}, (\odot_{r,i})_{1 \leq i \leq m}, (\odot_{l,i})_{1 \leq i \leq m}, (\odot_{r,l})_{1 \leq i \leq m}\} \), such that 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 \( \eta_1 \) and \( \eta_2 \), respectively, are
We omit relation symbols \( \eta, \eta_1, \) and \( \eta_2 \) 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 straight-
forward 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 \( \Sigma^\eta \) with each EGD of the form

\[
R(x, z), R(y, z) \rightarrow x = y, \quad \text{for } R \in \{a, b, (\phi_{k,r})_{1 \leq k \leq n}, (\phi_{r,k})_{1 \leq k \leq n}, (\phi_{k,\ell})_{1 \leq k, \ell \leq n}\}.
\]

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

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 \( \phi_{k,r} \) and
\( \phi_{k'\ell}, \) or of the form \( \phi_{r,k} \) and \( \phi_{r',k'} \), for \( 1 \leq k < k' \leq n \) and \( r, r' \in \{\ast, 1, \ldots, n\} \). In fact, consider
for instance that one of the edges is labeled \( \phi_{r,k} \) 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 \( \phi_{k,r} \), 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 \( \phi_{k,r} \) and the other one is of
the form \( \phi_{r',k'} \), for \( 1 \leq k < k' \leq n \) and \( r, r' \in \{\ast, 1, \ldots, n\} \). Then the path that enters \( z \) via an edge
labeled \( \phi_{k,r} \) uses an edge labeled \( \eta_1 \) for synchronization, while the other path uses an edge labeled
\( \eta_2 \). 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 \( \phi_{k,r} \) and \( \phi_{k',r'} \),
for \( 1 \leq k < k' \leq n \) and \( r, r' \in \{\ast, 1, \ldots, n\} \), as the other case is analogous. An easy way to solve
this problem is by using different dangling edges \( \eta_1^k \) and \( \eta_1^{k'} \) for each \( 1 \leq k \leq n \). Hence, now the
element \( z \) will have two different dangling edges \( \eta_1^k \) and \( \eta_1^{k'} \) : the first one corresponding to the

Fig. 13. A part of CQ \( q \) showing variables \( y, z \), and how variable \( x_j \) is connected to all variables \( x_i \), with \( i < j \).
incoming edge labeled \( \diamond_{k,r} \) and the second one to the incoming edge labeled \( \diamond_{k',r'} \). The dangling edge labeled \( \eta_1^k \) can only “match” with another dangling edge labeled \( \eta_2^k \), 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 \( \eta_1^k \) is “matched” with at most one dangling edge labeled \( \eta_2^k \). The cost to be paid is that we now have to extend our query \( q \) with extra symbols \( \eta_1^k \) and \( \eta_2^k \), for each \( 1 \leq k \leq n \), interpreted exactly as \( \eta_1 \) and \( \eta_2 \) before. We also need to adapt the matching and finalization rules in \( \Sigma' \); e.g., now matching rules have to use \( \eta_1^k \) and \( \eta_2^k \) as opposed to simply \( \eta_1 \) and \( \eta_2 \) in the following way:

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

C ADDITIONAL PROOFS FROM SECTION 9

**Proposition 9.9.** \( \emptyset \) 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 \( (\mu_j : U_{I_j,k} \rightarrow H_{I_j-1})_{j \geq 0} \) such that, for each \( j \geq 0 \):

1. the function \( \mu_j \) 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. \( \forall j \in U_{I_{j+1}, k}, \mu_j(J) = \mu_{j+1}(J) \).

We proceed by induction on \( j \geq 0 \).

**Base case.** For \( j = 0 \), we have that \( D[q] = I_0 \). Thus, we can define \( \mu_0 \) to be \( \mu \).

**Inductive step.** By induction hypothesis, there exists \( \mu_j : U_{I_j,k} \rightarrow H_{I_j-1} \) 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 \( j \in U_{I_{j+1}, k} \), let \( \mu_{j+1}(J) = \mu_j(J) \). We explain next how to define \( \mu_{j+1}(J) \), for each \( j \in U_{I_{j+1}, k} \). We recall that \( I_{j+1} \) is the result of applying the TGD \( \tau_j \) over \( I_j \) with \( (\bar{u}_j, \bar{u}_j') \). Assume that \( \tau_j \) 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 \( \bar{z} \in \bar{z} \) with a fresh null \( \bot \).

Suppose that the guard of \( \tau_j \) is \( R(\bar{x}, \bar{y}) \). Since \( \phi(\bar{u}_j, \bar{u}_j') \subseteq I_j \), we have that \( R(\bar{u}_j, \bar{u}_j') \subseteq I_j \) and, therefore, \( \mu_j \) 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 \( R(h(\bar{u}_j), h(\bar{u}_j')) \subseteq I \). Consider any other atom \( S(x') \) in the body of \( \tau_j \). Since \( \tau_j \) is guarded, all the variables in \( S(x') \) occur also in the guard atom \( R(\bar{x}, \bar{y}) \) of \( \tau_j \), i.e., \( x' \subseteq \bar{x} \cup \bar{y} \). Let \( \bar{v}_j \) be the reduction of the tuple \( (\bar{u}_j, \bar{u}_j') \) that corresponds to the positions associated with \( x' \). Since \( S(\bar{v}_j) \in I_j \), \( \mu_j \) 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 \( S(h'(\bar{v}_j)) \subseteq I \) and \( S(h(\bar{v}_j)) = S(h'(\bar{v}_j)) \), we conclude that \( S(h(\bar{v}_j)) \subseteq I \). 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 \( \tau_j \). Thus, \( \psi(h(\bar{u}_j), \bar{w}_h) \subseteq I \), where \( \bar{w}_h \) is a tuple of terms of \( \text{dom}(I) \). In other words, every homomorphism \( h \in \mu_j((R(\bar{u}_j, \bar{u}_j'))) \) naturally gives rise to a homomorphism \( h^{\text{ext}} \) from \( \psi(\bar{u}_j, \bot(\bar{z})) \subseteq I_{j+1} \) to \( \psi(\bar{u}_j, \bar{w}_h) \subseteq I \). In particular, \( h^{\text{ext}} \) agrees with \( h \) over the terms of \( \bar{u}_j \), and maps the nulls of \( \bar{z} \) to their corresponding terms in \( \bar{w}_h \). Clearly, \( h \) and \( h^{\text{ext}} \) 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 \( (J_1 \cup \{R(\bar{u}_j, \bar{u}_j')\}) \subseteq U_{I_{j+1}, k} \). 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 \( \text{dom}(J_1) \), 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, $h_{|R(a_j, a'_j)}$ gives rise to a homomorphism $(h_{|R(a_j, a'_j)})^\text{ext}$ from $\psi(\bar{a}_j, \bot(\bar{x}))$ to $I$, which is consistent with $h_{|R(a_j, a'_j)}$. As usual, we write $((h_{|R(a_j, a'_j)})^\text{ext})_{J_2}$ for the restriction of $(h_{|R(a_j, a'_j)})^\text{ext}$ over $\text{dom}(J_2)$. Notice that $((h_{|R(a_j, a'_j)})^\text{ext})_{J_2}$ is consistent with $h_{J_2}$, since each term of $\text{dom}(J_1) \cap \text{dom}(J_2)$ belongs to $(\bar{a}_j, \bar{a}'_j)$. Therefore,

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

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

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

We proceed to show that $\mu_{j+1}$ satisfies $(\dagger)$ and $(\dagger\dagger)$. In fact, the condition $(\dagger\dagger)$, which states that $\mu_j(J) = \mu_{j+1}(J)$ for each $J \in U_{j,k}$, follows by construction since $U_{j,k} \cap U_{j+1,k} = U_{j,k}$. It remains to show that $(\dagger)$ holds, that is, $\mu_{j+1}$ 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 $J \in U_{j+1,k}$:

1. $\mu_{j+1}(J) = \{ h \in H_{j+1,k} \mid h(J) \subseteq I \text{ and } \bot_{j+1} \in \text{dom}(J) \Rightarrow h(\bot_{j+1}) = t_1 \}$, and
2. for each $h \in \mu_{j+1}(J)$ and $J' \in U_{j+1,k}$, there exists $h' \in \mu_{j+1}(J')$ such that $h, h'$ are consistent.

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

- **Case 1:** $J \subseteq U_{j,k}$. Clearly, $\mu_{j+1}(J) = \mu_j(J)$, and the claim follows by induction hypothesis.
- **Case 2:** $J \not\subseteq U_{j,k}$. Clearly, $J$ can be partitioned into $\{J_1, J_2\}$, where $J_1 = J \cap I_1$ and $J_2 = J \setminus J_1$. By construction, $\mu_{j+1} = \{ \lambda^h \mid h \in \mu_j(J_1 \cup \{R(\bar{a}_j, \bar{a}'_j)\}) \}$, where $\lambda^h = h_{|J_1} \cup ((h_{|R(a_j, a'_j)})^\text{ext})_{J_2}$. Fix an arbitrary $\lambda^h \in \mu_{j+1}(J)$. As explained above, $\lambda^h(J) \subseteq I$. Assume now that $\bot_{j+1} \in \text{dom}(J)$. If $\bot_{j+1} \in \text{dom}(J_1)$, then, by induction hypothesis, $\lambda^h(\bot_{j+1}) = t_1$ since $h_{|J_1} \in \mu_j(J_1)$. Now, if $\bot_{j+1} \in \text{dom}(J_2) \setminus \text{dom}(J_1)$, then it necessarily belongs to $\bar{u}_j$ since the terms $\bot(\bar{x})$ are nulls generated by the chase procedure. Since $\lambda' = ((h_{|R(a_j, a'_j)})^\text{ext})_{J_2}$ is consistent with $h$, we have that $\lambda'(\bot_{j+1}) = h(\bot_{j+1})$. By induction hypothesis, $h(\bot_{j+1}) = \bar{t}_i$, and thus, $\lambda^h(\bot_{j+1}) = \bar{t}_i$.

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

- **Case 1:** $J, J' \subseteq U_{j,k}$. The claim follows by induction hypothesis.
- **Case 2:** $J \not\subseteq U_{j,k}$ and $J' \not\subseteq U_{j,k}$. Clearly, $\lambda \in \mu_j(J)$ since $J$ is a $k$-union of $I_1$. As usual, $J'$ can be partitioned into $\{J'_1, J'_2\}$, where $J'_1 = J' \cap I_1$ and $J'_2 = J' \setminus J'_1$. Notice that $J'_2 \neq \emptyset$, which implies that $|J'_2| \leq k - 1$. Thus, $J' \cup \{R(\bar{a}_j, \bar{a}'_j)\}$ is a $k$-union of $I_1$. By induction hypothesis and the second condition in the definition of the witness strategy, there exists a homomorphism $h \in \mu_j(J'_1 \cup \{R(\bar{a}_j, \bar{a}'_j)\})$ that is consistent with $\lambda$. By definition, the homomorphism $\lambda^h = h_{|J'_1} \cup ((h_{|R(a_j, a'_j)})^\text{ext})_{J'_2}$ belongs to $\mu_{j+1}(J')$. Observe that $\lambda$ and $\lambda^h$ are consistent since every term that is shared by $J$ and $J'_2$ occurs in $(\bar{a}_j, \bar{a}'_j)$. The claim follows with $\lambda' = \lambda^h$.

- **Case 3:** $J \not\subseteq U_{j+1,k}$ and $J' \subseteq U_{j,k}$. It suffices to show that there exists $\lambda' \in \mu_j(J')$ that is consistent with $\lambda$. We partition $J$ into $\{J_1, J_2\}$, where $J_1 = J \cap I_1$ and $J_2 = J \setminus J_1$. Observe that $|J_1| \leq k - 1$, and thus, $J_1 \cup \{R(\bar{a}_j, \bar{a}'_j)\}$ is a $k$-union of $I_1$. Clearly, $\lambda$ is of the form $\lambda^h = h_{|J_1} \cup ((h_{|R(a_j, a'_j)})^\text{ext})_{J_2}$. 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 $\lambda^h$ since $J'$ and $J_2$ share only terms of $(\bar{a}_j, \bar{a}'_j)$. The claim follows with $\lambda' = \chi_{|J_1}$.

- **Case 4:** $J, J' \not\subseteq U_{j+1,k} \setminus U_{j,k}$. As above, $J$ and $J'$ can be partitioned into $\{J_1, J_2\}$ and $\{J'_1, J'_2\}$, respectively, such that $J_1 = J \cap I_1$ and $J_2 = J \setminus J_1$, and $J'_1 = J' \cap I_1$ and $J'_2 = J' \setminus J'_1$. It is easy to verify that $|J_1|, |J'_1| \leq k - 1$, and thus, $J_1 \cup \{R(\bar{a}_j, \bar{a}'_j)\}$ and $J'_1 \cup \{R(\bar{a}_j, \bar{a}'_j)\}$ are $k$-unions of $I_1$. Clearly, $\lambda$
is of the form $\lambda^h = h_{|J_1} \cup ((h_{|R(\bar{a}_j, \bar{a}'_j)})^{ext})_{|J_2}$ for some $h \in \mu_j(J_1 \cup \{R(\bar{a}_j, \bar{a}'_j)\})$. 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 $\lambda^x = \chi_{|J_1} \cup ((\chi_{|R(\bar{a}_j, \bar{a}'_j)})^{ext})_{|J_2}$ belongs to $\mu_{j+1}(J')$. It is easy to verify that $\lambda^h$ and $\lambda^x$ are consistent. The claim follows with $\lambda' = \lambda^x$.

D ADDITIONAL PROOFS FROM SECTION 10

Lemma 10.3. Let $q, q' \in \text{GHW}_k$ over a schema $\sigma$, and $\Sigma \in C$ over $\sigma$, where $C \in \{G, NR, S\}$, such that $q' \subseteq \Sigma q$. There is a CQ $q'' \in \text{GHW}_k$ over $\sigma$ such that $q' \subseteq \Sigma q'' \subseteq \Sigma q$ and $|q''| \leq g_C(q, \Sigma)$.

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

Assume now that $C \in \{NR, S\}$. Since $C$ is UCQ rewritable, there exists a UCQ $Q$ over $\sigma$ 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 $q'' \in \text{GHW}_k$ over $\sigma$ such that $q' \subseteq q'' \subseteq \hat{q}$ and $|q''| \leq |\hat{q}| \cdot (2k + 1) \leq f_C(q, \Sigma) \cdot (2k + 1)$. 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