Conservative Extensions for Existential Rules

Jean Christoph Jung[1], **Carsten Lutz**[2], **Jerzy Marcinkowski**[3]
[1]Institute of Computer Science, University of Hildesheim, Germany
[2]Institute of Computer Science, Leipzig University, Germany
[3]Institute of Computer Science, University of Wrocław, Poland
jungj@uni-hildesheim.de, clu@informatik.uni-leipzig.de, jma@cs.uni.wroc.pl

Abstract

We study the problem to decide, given sets T_1, T_2 of tuple-generating dependencies (TGDs), also called existential rules, whether T_2 is a conservative extension of T_1. We consider two natural notions of conservative extension, one pertaining to answers to conjunctive queries over databases and one to homomorphisms between chased databases. Our main results are that these problems are undecidable for linear TGDs, undecidable for guarded TGDs even when T_1 is empty, and decidable for frontier-one TGDs.

1 Introduction

Tuple-generating dependencies (TGDs) are an expressive constraint language that emerged in database theory, where it has various important applications (Abiteboul, Hull, and Vianu 1995). In knowledge representation, TGDs are used as an ontology language under the names of existential rules and Datalog$^\pm$ (Baget et al. 2011; Calì et al. 2010). For the purposes of this paper, however, we stick with the name of 'TGDs'. A major application of TGDs in KR is ontology-mediated querying where a database query is enriched with an ontology, aiming to deliver more complete answers and to extend the vocabulary available for query formulation (Bienvenu et al. 2014; Bienvenu and Ortiz 2015; Calvanese et al. 2009). The semantics of ontology-mediated querying can be given in terms of homomorphisms and the widely known chase procedure that makes explicit the logical consequences of a set of TGDs and a database.

As the use of unrestricted TGDs makes the evaluation of ontology-mediated queries undecidable, various computationally more well-behaved fragments have been identified. We consider linear TGDs, guarded TGDs, and frontier-one TGDs (Calì, Gottlob, and Lukasiewicz 2012; Baget et al. 2011; Calì, Gottlob, and Kifer 2013). For all of these, ontology-mediated query evaluation is decidable. Deferring a formal definition to Section 2 of this paper, we remark that guarded generalizes linear, and that frontier-one is orthogonal to both linear and guarded. Moreover, linear TGDs generalize description logics (DLs) of the DL-Lite family (Artale et al. 2009), while both guarded and frontier-one TGDs generalize DLs of the $\mathcal{ELI}$ family (Baader et al. 2017).

On top of bare-bones query evaluation, there are other natural problems that are suggested by the framework of ontology-mediated querying. Consider the following: given sets of TGDs T_1 and T_2 (formulated in any, potentially different schemas), a database schema Σ_D, and a query schema Σ_Q, decide whether T_2 is a Σ_D, Σ_Q-*CQ-conservative extension* of T_1, that is, whether for all Σ_D-databases D and conjunctive queries (CQ) $q(\bar{x})$ in schema Σ_Q, every tuple $\bar{c}$ that is an answer to q on D given T_1 is also an answer to q on D given T_2 (Botoeva et al. 2016). Note that this is a very relevant problem. If, for instance, T_2 is a Σ_D, Σ_Q-CQ-conservative extension of T_1 and vice versa, then we can safely replace T_1 with T_2 in any application where databases are formulated in schema Σ_D and queries in schema Σ_Q. CQ-conservative extensions have been studied for various DLs and are decidable for many members of the DL-Lite and $\mathcal{ELI}$ families (Konev et al. 2011; Jung et al. 2020). In this paper, we address the naturally emerging question whether decidability extends to the more general settings of linear, guarded, and frontier-one TGDs.

A natural problem related to CQ-conservative extensions is Σ_D, Σ_Q-*hom-conservative extension* which asks whether for every Σ_D-database, there is a Σ_Q-homomorphism[1] from the chase $\mathsf{chase}_{T_2}(D)$ of D with T_2 to $\mathsf{chase}_{T_1}(D)$ that is the identity on all constants in D. In fact, this problem corresponds to CQ-conservative extensions when CQs may be infinitary, and it is known that these two problems do not coincide even in the case of DLs (Botoeva et al. 2016). We study hom-conservative extensions along with CQ-conservative extensions. In addition, we consider the variant of CQ/hom-conservative extensions where the set of TGDs T_1 is required to be empty. We refer to this as Σ_D, Σ_Q-*CQ/hom-triviality*. Note that triviality is also a very natural problem as it asks whether the given set of TGDs T_2 says *anything at all* about Σ_D-databases as far as conjunctive queries and homomorphisms over schema Σ_Q are concerned. We observe that Σ_D, Σ_Q-CQ-triviality and Σ_D, Σ_Q-hom-triviality coincide even for unrestricted TGDs, and thus we only speak of Σ_D, Σ_Q-*triviality*. Our main results are as follows.

1. For linear TGDs, CQ- and hom-conservative extensions are undecidable, but triviality is decidable.
2. For guarded TGDs, triviality is undecidable.
3. For frontier-one TGDs, CQ- and hom-conservative extensions are decidable.

[1]A homomorphism that disregards symbols outside of Σ_Q.

We consider it remarkable that undecidability already appears for a class as restricted as linear TGDs. Regarding Point 1, we also determine the exact complexity of triviality for linear TGDs as being PSPACE-complete, and coNP-complete when the arity of relation symbols is bounded by a constant. Regarding Point 3, our algorithms yield 3EXPTIME upper bounds, while 2EXPTIME lower bounds can be imported from the DL $\mathcal{ELI}$, a fragment of frontier-one TGDs (Gutiérrez-Basulto, Jung, and Sabellek 2018; Jung et al. 2020). The exact complexity remains open.

Our undecidability results are proved by reductions from a convergence problem that concerns Conway functions (Conway 1972). In a database theory context, such a technique has been used in (Gogacz and Marcinkowski 2014). As the reader shall see, the reductions take place in the setting of Pyramus and Thisbe (Ovid 2008), a mythological couple that could only communicate through a crack in the wall and whose fate it was to never meet again in person. Bring some popcorn. The decidability result for hom-conservative extensions for frontier-one TGDs rests on the observation that whenever there is a database that witnesses non-conservativity, then there is such a database of bounded treewidth. This enables a decision procedure based on alternating tree automata. The case of CQ-conservative extensions is more intricate as it requires the use of *homomorphism limits*, that is, families of homomorphisms that can only look n steps 'into the model', for any n. It is not clear how the existence of homomorphism limits can be verified by tree automata. Our solution generalizes the approach to CQ-conservative extensions in $\mathcal{ELI}$ pursued in (Jung et al. 2020). In short, the idea is to push the use of homomorphism limits to parts of the chase that are Σ_Q-disconnected from the database and regular in shape, and to then characterize homomorphism limits from/into such regular (infinite) databases in terms of unbounded homomorphisms.

Related Work. We already mentioned the work on DLs from the DL-Lite and $\mathcal{ELI}$ families (Konev et al. 2011; Jung et al. 2020). For description logics such as $\mathcal{ALC}$ that support negation and disjunction, CQ- and hom-conservative extensions are undecidable (Botoeva et al. 2019). A different kind of conservative extension is obtained by replacing databases and query answers with logical consequences formulated in the ontology language (Ghilardi, Lutz, and Wolter 2006). While such conservative extensions are decidable in $\mathcal{ALC}$ (Ghilardi, Lutz, and Wolter 2006; Lutz, Walther, and Wolter 2007), they are undecidable in the guarded fragment and in the two-variable fragment of first-order logic (Jung et al. 2017). For existential rule languages, the difference between this version of conservative extensions and CQ-conservative extensions tends to be small (depending on the class of rules considered).

2 Preliminaries

Relational Databases. Fix countably infinite and pairwise disjoint sets of *constants* $\mathbf{C}$ and $\mathbf{N}$ and variables $\mathbf{V}$. We refer to the constants in $\mathbf{N}$ as *nulls*. A *schema* Σ is a set of relation symbols R with associated arity $\mathsf{ar}(R) \geq 1$. A Σ-*fact* is an expression of the form $R(\bar{c})$ with $R \in \Sigma$ and $\bar{c}$ is an $\mathsf{ar}(R)$-tuple of constants from $\mathbf{C} \cup \mathbf{N}$. A Σ-*instance* is a possibly infinite set of Σ-facts, and a Σ-*database* is a finite Σ-instance that uses only constants from $\mathbf{C}$. We write $\mathsf{adom}(I)$ for the set of constants from $\mathbf{C} \cup \mathbf{N}$ used in instance I. For an instance I and a schema Σ, $I|_\Sigma$ denotes the restriction of I to Σ, that is, the set of all facts in I that use a relation symbol from Σ. We say that I is *connected* (resp., Σ-*connected*) if the Gaifman graph of I (resp., $I|_\Sigma$) is connected and that I is of *finite degree* if the Gaifman graph of I has finite degree.

For a schema Σ, a Σ-*homomorphism* from instance I to instance J is a function $h : \mathsf{adom}(I) \to \mathsf{adom}(J)$ such that $R(h(\bar{c})) \in J$ for every $R(\bar{c}) \in I$ with $R \in \Sigma$. We say that h is *database-preserving* if it is the identity on all constants from $\mathbf{C}$ (but not necessarily from $\mathbf{N}$) and write $I \to_\Sigma J$ if there is a database-preserving Σ-homomorphism from I to J.

Conjunctive Queries. A *conjunctive query* (CQ) over a schema Σ takes the form $\exists \bar{y}\, \phi(\bar{x}, \bar{y})$ where $\bar{x}$ and $\bar{y}$ are tuples of variables from $\mathbf{V}$, ϕ is a set of *atoms* $R(\bar{z})$ with $R \in \Sigma$ and $\bar{z}$ a tuple of variables of length $\mathsf{ar}(R)$. We refer to the variables in $\bar{x}$ as the *answer variables* of q and denote a CQ with $q(\bar{x})$ to emphasize that it has answer variables $\bar{x}$. The *arity* of q is the length $|\bar{x}|$ of $\bar{x}$, and q is *Boolean* if it is of arity 0.

Every CQ $q(\bar{x})$ gives rise to a database D_q, known as the *canonical database* of q, by viewing variables as constants and atoms as facts. A Σ-*homomorphism* h from q to an instance I is a Σ-homomorphism from D_q to I. A tuple $\bar{c} \in \mathsf{adom}(I)^{|\bar{x}|}$ is an *answer* to q on I if there is a homomorphism h from q to I with $h(\bar{x}) = \bar{c}$. The *evaluation of* $q(\bar{x})$ *on* I, denoted $q(I)$, is the set of all answers to q on I.

For a CQ q, but also for any other syntactic object q, we use $||q||$ to denote the number of symbols needed to write q encoded as a word over a suitable alphabet.

TGDs. A *tuple-generating dependency* (TGD) ϑ is a first-order sentence $\forall \bar{x} \forall \bar{y}\, \big(\phi(\bar{x}, \bar{y}) \to \exists \bar{z}\, \psi(\bar{x}, \bar{z})\big)$ such that $q_\varphi = \exists \bar{y}\, \phi(\bar{x}, \bar{y})$ and $q_\psi = \exists \bar{z}\, \psi(\bar{x}, \bar{z})$ are CQs. We call ϕ and ψ the *body* and *head* of ϑ. The body may be the empty conjunction, that is, logical truth. The variables in $\bar{x}$ are the *frontier variables*. We may write ϑ as $\phi(\bar{x}, \bar{y}) \to \exists \bar{z}\, \psi(\bar{x}, \bar{z})$. An instance I *satisfies* ϑ, denoted $I \models \vartheta$, if $q_\phi(I) \subseteq q_\psi(I)$. It *satisfies* a set of TGDs T if $I \models \vartheta$ for each $\vartheta \in T$. We then also say that I is a *model* of T.

A TGD ϑ is *frontier-one* if it has exactly one frontier variable (Baget et al. 2011). It is *guarded* if its body is empty or contains a *guard atom* α that contains all variables in the body (Calì, Gottlob, and Kifer 2013). A TGD is *linear* if its body contains at most one atom. Clearly, every linear TGD is guarded. The *body width* of a set T of TGDs is the maximum number of variables in a rule body of a TGD in T, and the *head width* is defined accordingly.

Throughout this paper, we are going to make use of the well-known chase procedure for making explicit the consequences of a set of TGDs (Johnson and Klug 1984; Fagin et al. 2005; Calì, Gottlob, and Kifer 2013). Let I be an instance and T a set of TGDs. A TGD $\phi(\bar{x}, \bar{y}) \to \exists \bar{z}\, \psi(\bar{x}, \bar{z}) \in T$ is *applicable* at a tuple $\bar{c}$ of constants in I if $\phi(\bar{c}, \bar{c}') \subseteq I$ for some $\bar{c}'$ and there is no homomorphism h from $\psi(\bar{x}, \bar{z})$ to I such that $h(\bar{x}) = \bar{c}$. In this case, the *result* *of applying the TGD in I at $\bar{c}$* is the instance $I \cup \{\psi(\bar{c}, \bar{c}'')\}$ where $\bar{c}''$ is the tuple obtained from $\bar{z}$ by replacing each variable z with a fresh null, that is, a null that does not occur in I. We also refer to such an application as a *chase step*.

A *chase sequence for I with T* is a sequence of instances $I_0, I_1, \dots$ such that $I_0 = I$ and each I_{i+1} is the result of a chase step from I_i. The *result* of the chase sequence is the instance $J = \bigcup_{i \geq 0} I_i$. The chase sequence is *fair* if whenever a TGD from T is applicable to a tuple $\bar{c}$ in some I_i, then this application is a chase step in the sequence. Every fair chase sequence for I with T has the same result, up to homomorphic equivalence. Since for our purposes all results are equally useful, we use $\mathsf{chase}_T(I)$ to denote the result of an arbitrary, but fixed chase sequence for I with T and call $\mathsf{chase}_T(I)$ the *result of chasing I with T*. This version of the chase is often called the *restricted chase* and it ensures that $\mathsf{chase}_T(D)$ has finite degree, which shall be important for our proofs.

Lemma 1. *Let T be a set of TGDs and I an instance. Then for every model J of T with $I \subseteq J$, there is a homomorphism h from $\mathsf{chase}_T(I)$ to J that is the identity on $\mathsf{adom}(I)$.*

Note that if T is a set of frontier-one TGDs, then for any database D the instance $\mathsf{chase}_T(D)$ can be obtained from D by 'glueing' a (potentially infinite) instance onto each constant $c \in \mathsf{adom}(D)$. We denote this instance with $\mathsf{chase}_T(D)|_c^{\downarrow}$. A precise definition is given in the appendix.

Let T be a set of TGDs, $q(\bar{x})$ a CQ and D a database. A tuple $\bar{c} \in \mathsf{adom}(D)^{|\bar{x}|}$ is an *answer* to q on D w.r.t. T, written $D, T \models q(\bar{c})$, if $q(\bar{c})$ is logically follows from $D \cup T$ or, equivalently, if there is a homomorphism h from q to $\mathsf{chase}_T(D)$ with $h(\bar{x}) = \bar{c}$. The *evaluation of q on D w.r.t. T*, denoted $q_T(D)$, is the set of all answers to q on D w.r.t. T.

3 Conservative Extensions

We introduce the notions of conservative extension that are studied in this paper and the associated decision problems.

Definition 1. *Let T_1, T_2 be sets of TGDs and let Σ_D, Σ_Q be schemas called the* data schema *and* query schema*. Then*

- T_2 is Σ_D, Σ_Q-hom-conservative over T_1, *written* $T_1 \models^{hom}_{\Sigma_D,\Sigma_Q} T_2$, *if there is a database-preserving Σ_Q-homomorphism from $\mathsf{chase}_{T_2}(D)$ to $\mathsf{chase}_{T_1}(D)$ for all Σ_D-databases D;*
- T_2 is Σ_D, Σ_Q-CQ-conservative over T_1, *written* $T_1 \models^{CQ}_{\Sigma_D,\Sigma_Q} T_2$, *if $q_{T_2}(D) \subseteq q_{T_1}(D)$ for all Σ_D-databases D and all CQs q over schema Σ_Q.*
- T_1 *is* Σ_D, Σ_Q-hom-trivial *if T_1 is Σ_D, Σ_Q-hom-conservative over the empty set of TGDs, and likewise for* Σ_D, Σ_Q-CQ-triviality.

It is easy to see that logical entailment $T_1 \models T_2$ implies $T_1 \models^{\text{hom}}_{\Sigma_D,\Sigma_Q} T_2$ for all schemas Σ_D and Σ_Q, and that Σ_D, Σ_Q-hom-conservativity implies Σ_D, Σ_Q-CQ-conservativity. The following example from (Botoeva et al. 2016) shows that the converse fails.

Example 1. *Consider the following sets of TGDs that are both linear and frontier-one:*

$$\begin{aligned} T_1 &= \{\, A(x) \rightarrow \exists y\, S(x,y), B(y), \\ &\qquad B(x) \rightarrow \exists y\, R(x,y), B(y) \,\} \\ T_2 &= \{\, A(x) \rightarrow \exists y\, S(x,y), B(y), \\ &\qquad B(x) \rightarrow \exists y\, R(y,x), B(y) \,\}. \end{aligned}$$

Let $\Sigma_D = \{A\}$ and $\Sigma_Q = \{R\}$. We recommend to the reader to verify that T_2 is not Σ_D, Σ_Q-hom-conservative over T_1 by trying to find a database-preserving homomorphism from $\mathsf{chase}_{T_2}(D)$ to $\mathsf{chase}_{T_1}(D)$, and that it is Σ_D, Σ_Q-CQ-conservative.

However, Σ_D, Σ_Q-hom-conservativity is equivalent to Σ_D, Σ_Q-CQ-conservativity with infinitary CQs. We refrain from making this precise and instead consider the converse, that is, Σ_D, Σ_Q-CQ-conservativity is equivalent to Σ_D, Σ_Q-hom-conservativity when the latter is defined in terms of a finitary version of homomorphisms that we introduce next.

Let I_1, I_2 be instances and $n \geq 0$, and let Σ be a schema. We write $I_1 \rightarrow^n_\Sigma I_2$ if for every induced subinstance I of I_1 with $|\mathsf{adom}(I)| \leq n$, there is a database-preserving Σ-homomorphism from I to I_2. We further write $I_1 \rightarrow^{\mathsf{lim}}_\Sigma I_2$ if $I_1 \rightarrow^n_\Sigma I_2$ for all $n \geq 1$.

Theorem 1. *Let T_1 and T_2 be sets of TGDs and Σ_D, Σ_Q schemas. Then $T_1 \models^{CQ}_{\Sigma_D,\Sigma_Q} T_2$ iff $\mathsf{chase}_{T_2}(D) \rightarrow^{\mathsf{lim}}_{\Sigma_Q} \mathsf{chase}_{T_1}(D)$.*

For triviality, the hom- and CQ-version coincide.

Lemma 2. *Let T_1, T_2 be sets of TGDs and Σ_D, Σ_Q schemas. Then T_1 and T_2 are Σ_D, Σ_Q-hom-trivial if and only if they are Σ_D, Σ_Q-CQ-trivial.*

Because of Lemma 2, we from now on disregard Σ_D, Σ_Q-CQ-triviality and refer to Σ_D, Σ_Q-hom-triviality simply as Σ_D, Σ_Q-*triviality*. We thus obtain the three decision problems *hom-conservativity*, *CQ-conservativity*, and *triviality*, defined in the obvious way. For instance, hom-conservativity means to decide, given finite sets of TGDs T_1, T_2 and finite schemas Σ_D, Σ_Q, whether T_2 is Σ_D, Σ_Q-hom-conservative over T_1.

We note that Lemma 2 is an immediate consequence of Theorem 1 and the following observation.

Lemma 3. *Let I_1, I_2 be instances such that I_1 is countable and I_2 is finite, and let Σ be a schema. If $I_1 \rightarrow^{\mathsf{lim}}_\Sigma I_2$, then $I_1 \rightarrow_\Sigma I_2$.*

We sketch the proof of Lemma 3, details are in the appendix. If $I_1 \rightarrow^{\mathsf{lim}}_\Sigma I_2$, then we find database-preserving Σ-homomorphisms $h_1, h_2, \dots$ from finite subinstances $J_1 \subseteq J_2 \subseteq \dots$ of I_1 to I_2 such that $I_1 = \bigcup_{i \geq 1} J_i$. If $h_1, h_2, \dots$ are compatible in the sense that $h_i(c) = h_j(c)$ whenever $h_i(c), h_j(c)$ are both defined, then $\bigcup_{i \geq 1} h_i$ is a Σ-homomorphism that witnesses $I_1 \rightarrow_\Sigma I_2$. If this is not the case, however, we can still manipulate $h_1, h_2, \dots$ into a compatible sequence $g_1, g_2, \dots$ by 'skipping homomorphisms', which is used in several proofs in this paper. We start with h_1 and observe that since J_1 and I_2 are finite, there are only

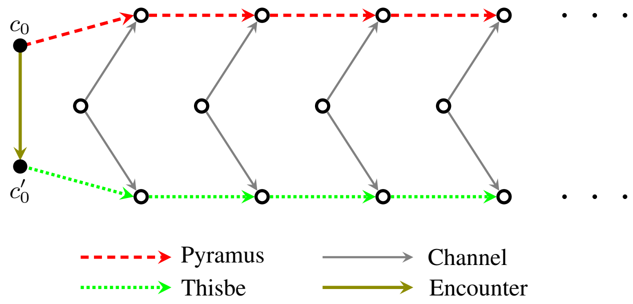

Figure 1: Chase generated by T_{myth}.

finitely many homomorphisms h from J_1 to I_2. Some such homomorphism must occur infinitely often in the restrictions of $h_1, h_2, \ldots$ to $\mathsf{adom}(J_1)$ and thus we find a subsequence $h_1', h_2', \ldots$ of $h_1, h_2, \ldots$ in which h_1' is compatible with all of $h_2', h_3', \ldots$. We proceed in the same way for h_2', then for h_3', ad infinitum, finding the desired sequence $g_1, g_2, \ldots$.

4 Undecidability

The aim of this section is to prove the following results.

Theorem 2. *The following problems are undecidable:*

1. *hom-conservativity for linear TGDs;*
2. *CQ-conservativity for linear TGDs;*
3. *triviality for guarded TGDs.*

We give a single proof that establishes Points 1 and 2. Attaining Point 3 requires a non-trivial modification of the proof. We start with the former, first highlighting the main mechanism that we use in our reduction.

4.1 The Main Mechanism

Consider the set of rules T_{myth}. It comprises three TGDs:

$$\begin{array}{rcl}\mathsf{Encounter}(p,t) & \rightarrow & \exists p', c, t'\ M(p,p',c,t',t) \\ M(p,p',c,t',t) & \rightarrow & \exists p'', c', t''\ M(p',p'',c',t'',t') \\ M(p,p',c,t',t) & \rightarrow & \mathsf{Pyramus}(p,p'), \mathsf{Thisbe}(t,t'), \\ & & \mathsf{Channel}(c,p'), \mathsf{Channel}(c,t').\end{array}$$

Now consider the database $D = \{\mathsf{Encounter}(c_0, c_0')\}$. The instance $\mathsf{chase}_{T_{\mathsf{myth}}}(D)$, shown in Figure 1, will play an important role. Its intuitive meaning is that '*after an initial brief encounter, Pyramus and Thisbe have never met again, but forever remained able to connect via an (indirect) channel*.' Notice that we do not explicitly show relation M in Figure 1 as M is only a construction aid, needed to ensure that the TGDs in T_{myth} are linear. As Σ_Q, we will use the set of relation symbols in T_{myth} except M, plus a unary relation symbol Mouth. We advise the reader to not worry about the schema Σ_D at this point (it will actually be empty).

Let $\kappa = \langle [p_1, \ldots, p_n], [t_1, \ldots, t_n] \rangle$ be a pair of sequences of positive integers of the same length n. By River_κ, we mean the database that contains the following facts, an example being displayed in Figure 2:

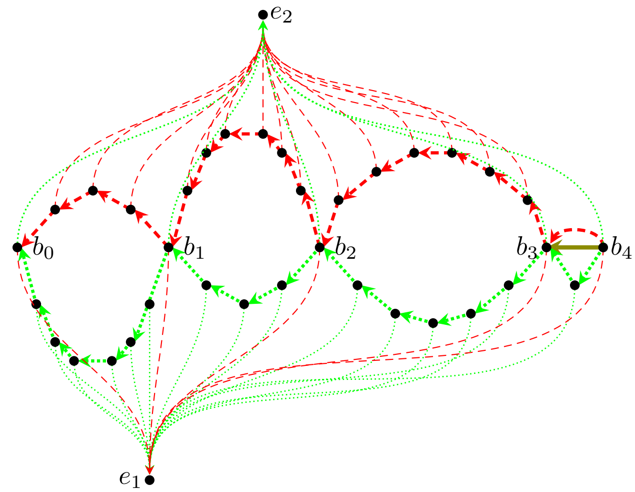

Figure 2: The database River_κ for $\kappa = \langle [4,7,7,1], [7,4,6,2] \rangle$.

- There are 3 kinds of constants. The *eternities* are e_1 and e_2. The *channel* is c, not shown in the picture. All remaining constants are called *worldly*.
- For $1 \leq i \leq n$ there is a Pyramus path of length p_i from b_i to b_{i-1} as well as a Thisbe path of length t_i from b_i to b_{i-1}. Constants b_i are called *bridges*.
- There is $\mathsf{Thisbe}(a, e_1)$ for each non-bridge constant a on each of the Thisbe-paths and there is $\mathsf{Pyramus}(a, e_2)$ for each non-bridge constant a on each of the Pyramus-paths. There are also $\mathsf{Thisbe}(a, e_2)$ and $\mathsf{Pyramus}(a, e_1)$ for each bridge constant a. In addition (and not in Figure 2), there are $\mathsf{Pyramus}(e_i, e_i)$ and $\mathsf{Thisbe}(e_i, e_i)$ for $i \in \{1, 2\}$.
- For each *worldly* constant a, there is $\mathsf{Channel}(c, a)$. Moreover, there are facts $\mathsf{Channel}(e_i, e_i)$, for $i \in \{1, 2\}$. These facts are not shown in Figure 2.
- There are $\mathsf{Encounter}(b_n, b_{n-1})$ and $\mathsf{Mouth}(b_0)$.

It is easy to see that $\mathsf{chase}_{T_{\mathsf{myth}}}(\mathsf{River}_\kappa)$ is obtained from River_κ by adding a copy of the instance shown in Figure 1, glueing the Encounter fact to the Encounter fact in River_κ (and adding some M-facts that are not important here). Now, let us leave to our readers the pleasure to notice that:

Observation 1. *There is a database-preserving Σ_Q-homomorphism from* $\mathsf{chase}_{T_{\mathsf{myth}}}(\mathsf{River}_\kappa)$ *to* River_κ *if and only if there exists $m \in \{1, \ldots, n-1\}$ such that $t_m \neq p_{m+1}$.*

Hint: As long as Pyramus and Thisbe walk down their respective river banks they are connected via the constant c. But for their union to last forever they need, at some point, to enter one of the eternities. Since eternity has no channel with the worldly constants (and the two eternities are not connected by a channel either), Pyramus and Thisbe both need to enter the same eternity, and they need to do it simultaneously. But this can only happen when one of them is in a bridge constant and the other in a non-bridge. □

That's nice, isn't it? But where could undecidability be lurking here?

4.2 Conway Functions

Let $\gamma, \alpha_0, \beta_0, \ldots, \alpha_{\gamma-1}, \beta_{\gamma-1}$ be positive integers such that $\beta_k|\gamma$ and $\beta_k|k\alpha_k$ for $0 \leq k < \gamma$ where as usual '|' denotes divisibility without remainder. For a positive integer n, define $\mathcal{F}(n)$ by setting $\mathcal{F}(n) = n\alpha_k/\beta_k$ for $k = n \bmod \gamma$. Thus, the remainder of n when dividing by γ determines the pair (α_k, β_k) used to compute the value $\mathcal{F}(n)$. Note that due to the two divisibility conditions, the range of $\mathcal{F}$ contains only positive integers.

The function $\mathcal{F}$ is called the *Conway function defined by* $\gamma, \alpha_0, \beta_0, \ldots, \alpha_{\gamma-1}, \beta_{\gamma-1}$. We say that $\mathcal{F}$ *stops* if there exists an $n \in \mathbb{N}$ such that $\mathcal{F}^n(2) = 1$, where $\mathcal{F}^n$ is $\mathcal{F}$ composed with itself, n times. There is no special meaning to the numbers 1 and 2 used here, we could choose otherwise. The following is well-known, see also (Gogacz and Marcinkowski 2014).

Theorem 3. *It is undecidable whether the Conway function defined by a given sequence* $\gamma, \alpha_0, \beta_0, \ldots, \alpha_{\gamma-1}, \beta_{\gamma-1}$ *stops.*

Take a sequence $\gamma, \alpha_0, \beta_0, \ldots, \alpha_{\gamma-1}, \beta_{\gamma-1}$ defining a Conway function $\mathcal{F}$. We prove Points 1 and 2 of Theorem 2 by showing how to compute, given the sequence, sets T_1 and T_2 of linear TGDs along with schemas Σ_D, Σ_Q such that $\mathcal{F}$ does not stop if and only if T_2 is Σ_D, Σ_Q-hom-conservative over T_1 if and only if T_2 is Σ_D, Σ_Q-CQ-conservative over T_1. We assume without loss of generality that $\mathcal{F}(1) = 1$ and $\mathcal{F}(2) = 3$.

4.3 The Reduction

We say that $\kappa = \langle [p_1, \ldots, p_n], [t_1, \ldots, t_n] \rangle$ (or River_κ) is

- *locally correct* if the following conditions hold:
  1. $p_1 = 2$ and $p_n = 1$;
  2. $\mathcal{F}(p_i) = t_i$ for $1 \leq i < n$;
- *correct* if it is locally correct and $t_i = p_{i+1}$ for $1 \leq i < n$.

The database River_κ shown in Figure 2 is not locally correct because $p_1 \neq 2$ and $t_n \neq 1$ (which must be the case as we assume $\mathcal{F}(1) = 1$).

Clearly, $\mathcal{F}$ does not stop if and only if every locally correct River_κ is incorrect, and by Observation 1 this is the case if and only if for each locally correct sequence κ there exists a database-preserving Σ_Q-homomorphism from $\mathsf{chase}_{T_{\mathsf{myth}}}(\mathsf{River}_\kappa)$ to River_κ.

Now the plan is as follows. Take $\Sigma_D = \emptyset$. We define T_1 such that $\mathsf{chase}_{T_1}(\emptyset)$ is the 'disjoint union' of all locally correct databases River_κ. Our T_2 will be the union of T_1 and T_{myth}. A careful reader can notice that if this plan succeeds, then the proof of Point 1 of Theorem 2 will be completed. And it will indeed succeed, but not without one little nuance. This is the reason why we used quotations mark around the term 'disjoint union' above.

The set of TGDs T_1 is the union of two sets of linear TGDs T_{rec} and T_{proj}. As intended, T_1 generates the union of all locally correct databases River_κ. The mentioned nuance is that the union is not disjoint, but massively overlapping. However, this does not compromise correctness of the reduction.

The rules of T_{rec} will not mention symbols from Σ_Q. They instead use a schema $\Sigma_{\mathcal{F}}$ that consists of high arity relation symbols used as construction aids. We later use T_{proj} to relate these symbols to those in Σ_Q. More precisely, $\Sigma_{\mathcal{F}}$ contains relation symbols Start of arity 8, End of arity 5, Bridge of arity 4, WH_k^i (for WorkHorse) of arity $\alpha_k + \beta_k + 5$ for $0 \leq k, i < \gamma$, and BH_k (for BridgeHead) of arity $\alpha_k + \beta_k + 5$ for $0 \leq k < \gamma$. In what follows, we use $\dagger$ to denote the list of variables 'c, e_1, e_2'. With $+_\gamma$ and $-_\gamma$, we denote addition and subtraction in the ring $\mathbb{Z}_\gamma$.

Since $\Sigma_D = \emptyset$, first of all we need a rule that will create something out of nothing:

$$\rightarrow \ \exists \dagger, b_0, x_1, y_1, y_2, b_1 \ \mathsf{Start}(\dagger, b_0, x_1, y_1, y_2, b_1).$$

Later, T_{proj} will generate a Pyramus-path from b_1 via x_1 to b_0 and a Thisbe-path from b_1 via y_2 and y_1 to b_0, determining the lengths $p_1 = 2$ and $t_1 = 3$ of the river. Recall that local correctness prescribes $p_1 = 2$ and we assume $\mathcal{F}(2) = 3$. We need to know that b_1 is a bridge:

$$\mathsf{Start}(\dagger, b_0, x_1, y_1, y_2, b_1) \ \rightarrow \ \mathsf{Bridge}(\dagger, b_1).$$

We now put our horses to work by adding, for $0 \leq k < \gamma$:

$$\mathsf{Bridge}(\dagger, b) \rightarrow \exists x_1, \ldots, x_{\beta_i}, y_1, \ldots, y_{\alpha_i} \quad \mathsf{WH}_k^{\beta_k}(\dagger, b, x_1, \ldots, x_{\beta_k}, b, y_1, \ldots, y_{\alpha_k})$$

and for $0 \leq k, i < \gamma$:

$$\begin{aligned} &\mathsf{WH}_k^i(\dagger, x_0, x_1, \ldots, x_{\beta_k}, y_0, y_1, \ldots, y_{\alpha_k}) \rightarrow \\ &\quad \exists z_1, \ldots, z_{\beta_k}, u_1, \ldots, u_{\alpha_k} \\ &\quad \mathsf{WH}_k^{i +_\gamma \beta_k}(\dagger, x_{\beta_k}, z_1, \ldots, z_{\beta_k}, y_{\alpha_k}, u_1, \ldots, u_{\alpha_k}). \end{aligned}$$

$\mathsf{WH}_k^i(\dagger, c_0, c_1, \ldots, x_{\beta_k}, y_0, y_1, \ldots, y_{\alpha_k})$ promises to generate, via T_{proj}, a Pyramus-path of length β_k from x_{β_k} to x_0 and a Thisbe-path of length α_k from y_{α_k} to y_0. The above two rules thus patiently produce Pyramus- and Thisbe-paths that lead to b. The superscript $\cdot^i$ remembers how many Pyramus-edges have been produced since the last bridge, modulo γ, and the subscript $\cdot_k$ chooses a remainder class, that is, it expresses the promise that the Pyramus-path between the two bridges is of length n, for some number n with $n \bmod \gamma = k$.

Then, at some point, the next bridge can be reached:

$$\begin{aligned} &\mathsf{WH}_k^{k -_\gamma \beta_k}(\dagger, x_0, x_1, \ldots, x_{\beta_k}, y_0, y_1, \ldots, y_{\alpha_k}) \rightarrow \\ &\quad \exists z_1, \ldots, z_{\beta_k - 1}, u_1, \ldots, u_{\alpha_k - 1}, b \\ &\quad \mathsf{BH}_k(\dagger, x_{\beta_k}, z_1, \ldots, z_{\beta_k - 1}, b, y_{\alpha_k}, u_1, \ldots, u_{\alpha_k - 1}, b) \end{aligned}$$

$$\begin{aligned} &\mathsf{BH}_k(\dagger, x_{\beta_k}, z_1, \ldots, z_{\beta_k - 1}, b, y_{\alpha_k}, u_1, \ldots, u_{\alpha_k - 1}, b) \rightarrow \\ &\quad \mathsf{Bridge}(\dagger, b). \end{aligned}$$

In the first rule above, relation $\mathsf{WH}_k^{k -_\gamma \beta_k}$ indicates that we have seen m Pyramus-edges, for some m with $m \bmod \gamma = k -_\gamma \beta_k$, and that BH_k will generate β_k more Pyramus-edges, thus arriving at the promised remainder of k. It is also easy to see that if the chosen remainder class was k and the length of the Pyramus-path between two bridges produced by the above rules is n, then the length of the Thisbe-path is $\mathcal{F}(n) = n\alpha_k/\beta_k$. Thus, Point 2 of local correctness is satisfied.

Finally, we want to produce the last[2] segment of the river:

$$\mathsf{Bridge}(\dagger, b) \rightarrow \exists b' \ \mathsf{End}(\dagger, b, b')$$

[2]Orographically the first, as we generate the river from the mouth to the source.

This will generate direct Pyramus- and Thisbe-edges from b' to b (recall that $\digamma(1) = 1$).

The TGDs in T_{rec} generate the actual rivers as projections of the template produced by T_{rec}. We start at the mouth:

$$\begin{aligned}&\mathsf{Start}(\dagger, b_0, x_1, y_1, y_2, b_1) \rightarrow\\ &\quad \mathsf{Mouth}(b_0),\\ &\quad \mathsf{Pyramus}(x_1, b_0), \mathsf{Pyramus}(b_1, x_1), \mathsf{Pyramus}(x_1, e_2),\\ &\quad \mathsf{Thisbe}(y_1, b_0), \mathsf{Thisbe}(y_2, y_1), \mathsf{Thisbe}(b_1, y_2),\\ &\quad \mathsf{Thisbe}(y_1, e_1), \mathsf{Thisbe}(y_2, e_1),\\ &\quad \mathsf{Channel}(c, x_1), \mathsf{Channel}(c, y_1), \mathsf{Channel}(c, y_2),\\ &\quad \mathsf{Channel}(e_1, e_1), \mathsf{Pyramus}(e_1, e_1), \mathsf{Thisbe}(e_1, e_1)\\ &\quad \mathsf{Channel}(e_2, e_2), \mathsf{Pyramus}(e_2, e_2), \mathsf{Thisbe}(e_2, e_2).\end{aligned}$$

The rules for WH^i_k are then as expected:

$$\begin{aligned}&\mathsf{WH}^i_k(\dagger, x_0, x_1, \dots, x_{\beta_k}, y_0, y_1, \dots, y_{\alpha_k}) \rightarrow\\ &\quad \mathsf{Pyramus}(x_1, x_0), \dots, \mathsf{Pyramus}(x_{\beta_k}, x_{\beta_k - 1}),\\ &\quad \mathsf{Pyramus}(x_1, e_2), \dots, \mathsf{Pyramus}(x_{\beta_k}, e_2),\\ &\quad \mathsf{Thisbe}(y_1, y_0), \dots, \mathsf{Thisbe}(y_{\alpha_k}, y_{\alpha_k - 1}),\\ &\quad \mathsf{Thisbe}(y_1, e_1), \dots, \mathsf{Thisbe}(y_{\alpha_k}, e_1),\\ &\quad \mathsf{Channel}(c, x_1), \dots, \mathsf{Channel}(c, x_{\beta_k}),\\ &\quad \mathsf{Channel}(c, y_1), \dots, \mathsf{Channel}(c, y_{\alpha_k}).\end{aligned}$$

Rules for the relations BH_i are analogous, so we skip them. There are also rules for projecting relations Bridge and End:

$$\begin{aligned}&\mathsf{Bridge}(\dagger, b) \rightarrow \mathsf{Channel}(c, b), \mathsf{Pyramus}(b, e_1), \mathsf{Thisbe}(b, e_2)\\ &\mathsf{End}(\dagger, b, b') \rightarrow \mathsf{Pyramus}(b', b), \mathsf{Thisbe}(b', b),\\ &\qquad\qquad \mathsf{Encounter}(b, b').\end{aligned}$$

In the appendix, we show that:

Lemma 4. *$\digamma$ does not stop iff $T_2 = T_1 \cup T_{\mathsf{myth}}$ is Σ_Q, Σ_D-hom-conservative over $T_1 = T_{\mathsf{rec}} \cup T_{\mathsf{proj}}$.*

This establishes Point 1 of Theorem 2. For the "if" direction, one shows that if $\mathsf{chase}_{T_2}(\emptyset) \rightarrow_{\Sigma_Q} \mathsf{chase}_{T_1}(\emptyset)$, then every locally correct river is incorrect, and thus $\digamma$ stops. Since rivers may be long, but are finite, it actually suffices that $\mathsf{chase}_{T_2}(\emptyset) \rightarrow^{\mathsf{lim}}_{\Sigma_Q} \mathsf{chase}_{T_1}(\emptyset)$ for $\digamma$ to stop, which by Theorem 1 gives Point 2 of Theorem 2.

For Point 3 of Theorem 2, we again want to use the toolkit above, in particular T_{myth} and Observation 1. But the situation is a bit different now. In the above reduction, we had at our disposal T_1 which was able to produce, from nothing, all the rivers we needed. So we could afford to have $\Sigma_D = \emptyset$. Now, however, we no longer have T_1, but only T_2, and our strategy is as follows. Recall that $\digamma$ stops if and only if there is a locally correct River_κ that is correct, and that River_κ is correct if there is no database-preserving Σ_Q-homomorphism from $\mathsf{chase}_{T_{\mathsf{myth}}}(\mathsf{River}_\kappa)$ to River_κ. We use the database D to guess a River_κ that admits no such homomorphism. More precisely, we design T_2 so that it verifies the existence of a (single) locally correct river in D and only if successful generates a chase with T_{myth} at the $\mathsf{Encounter}$ fact of that river. Details are in the appendix.

5 Triviality for Linear TGDs

We show that for linear TGDs, Σ_D, Σ_Q-triviality is decidable and PSPACE-complete, while it is only coNP-complete when the arity of relation symbols is bounded by a constant. The upper bounds crucially rely on the observation that non-triviality is always witnessed by a *singleton database*, that is, a database that contains at most one fact. This was first noted (for CQ-conservative extensions) in the context of the decription logic DL-Lite (Konev et al. 2011).

Lemma 5. *Let T be a set of linear TGDs and Σ_D, Σ_Q schemas. Then T is Σ_D, Σ_Q-trivial iff $\mathsf{chase}_T(D) \rightarrow_{\Sigma_Q} D$ for all singleton Σ_D-databases D.*

So an important part of deciding triviality is to decide, given a set of TGDs T and a singleton database D, whether $\mathsf{chase}_T(D) \not\rightarrow_{\Sigma_Q} D$. The basis for this is the subsequent lemma.

Lemma 6. *Let T be a set of linear TGDs and D a singleton database. Then $\mathsf{chase}_T(D) \not\rightarrow_{\Sigma_Q} D$ implies that there is a connected database $C \subseteq \mathsf{chase}_T(D)$ that contains at most two facts and such that $C \not\rightarrow_{\Sigma_Q} D$.*

Lemmas 5 and 6 provide us with a decision procedure for triviality for linear TGDs. Given a finite set of linear TGDs T and finite schemas Σ_D and Σ_Q, all we have to do is iterate over all singleton Σ_D-databases D and over all $C \subseteq \mathsf{chase}_T(D)$ that contain at most two facts and check (in polynomial time) whether $C \rightarrow_{\Sigma_Q} D$. To identify the sets C, we can iterate over all exponentially many candidates and check for each of them whether $D, T \models q_C$, where q_C is C viewed as a Boolean CQ. This entailment check is possible in PSPACE (Gottlob, Manna, and Pieris 2015). This yields the PSPACE upper bound in the following result.

Theorem 4. *For linear TGDs, triviality is PSPACE-complete. It is coNP-complete if the arity of relation symbols is bounded by a constant.*

To obtain the coNP upper bound, we recall that when the arity of relation symbols is bounded by a constant, then the entailment check '$D, T \models q_C$' is in NP (Gottlob et al. 2014). To decide non-triviality, we may thus guess D and C and verify in polynomial time that $C \not\rightarrow_{\Sigma_Q} D$ and in NP that $D, T \models q_C$. For the lower bounds, we reduce entailments of the form $D, T \models \exists x\, A(x)$, with T a set of linear TGDs, to non-triviality for linear TGDs. This problem is PSPACE-hard in general (Casanova, Fagin, and Papadimitriou 1984) and it is common knowledge that it is NP-hard when the arity of relation symbols is bounded by a constant. The reduction goes as follows. Let D, T, and $\exists x\, A(x)$ be given. Introduce a fresh binary relation symbol R, set $\Sigma_D = \Sigma_Q = \{R\}$, and let T' be the extension of T with the TGDs

$$\begin{aligned} &\rightarrow \quad q_D\\ A(u) &\rightarrow \quad \exists x \exists y \exists z\, R(x, y), R(y, z)\end{aligned}$$

where q_D is D viewed as a Boolean CQ. Note that there is no homomorphism from $R(x, y), R(y, z)$ into the singleton Σ_D-database $\{R(c, c')\}$. Based on this, it is easy to verify that T' is Σ_D, Σ_Q-trivial iff $D, T \not\models \exists x\, A(x)$.

6 Frontier-One TGDs

The purpose of this section is to show the following.

Theorem 5. *For frontier-one TGDs, CQ-conservativity and hom-conservativity are decidable in* 3ExpTime *(and* 2ExpTime*-hard).*

2ExpTime lower bounds carry over from the description logic $\mathcal{ELI}$, see (Gutiérrez-Basulto, Jung, and Sabellek 2018) for hom-conservativity and (Jung et al. 2020) for CQ-conservativity. They already apply when only unary and binary relation symbols are admitted. In the remainder of the section, we thus concentrate on the upper bounds.

Both in the case of hom-conservativity and CQ-conservativity, we first provide a suitable model-theoretic characterization and then use it to find a decision procedure based on tree automata. The case of CQ-conservativity is significantly more challenging because of the appearance of homomorphism limits.

6.1 Deciding Hom-Conservativity

We show that to decide hom-conservativity, it suffices to consider databases of bounded treewidth. Instead of using the standard notion of a tree decomposition, however, it is more convenient for us to work with so-called tree-like databases. Variations of these have been used for instance in (Benedikt, Bourhis, and Senellart 2012; Jung et al. 2018).

A Σ-*instance tree* is a triple $\mathcal{T} = (V, E, B)$ with (V, E) a directed tree and B a function that assigns a Σ-database $B(v)$ to every $v \in V$ such that the following conditions hold:

1. for every $a \in \bigcup_{v \in V} \mathsf{adom}(B(v))$, the restriction of (V, E) to the nodes $v \in V$ such that $a \in \mathsf{adom}(B(v))$ is a tree of depth at most one;
2. for every $(u, v) \in E$, $|\mathsf{adom}(B(u)) \cap \mathsf{adom}(B(v))| \leq 1$.

The *width* of the instance tree is the supremum of the cardinalities of $\mathsf{adom}(B(v))$, $v \in V$. A Σ-instance tree $\mathcal{T}$ defines an associated instance $I_{\mathcal{T}} = \bigcup_{v \in V} B(v)$. A Σ-instance I is *tree-like of width* k if there is a Σ-instance tree $\mathcal{T}$ of width k with $I = I_{\mathcal{T}}$.

Instance trees of width k are closely related to tree decompositions of width k in which the bags overlap in at most one constant. Condition 1, however, strengthens the usual connectedness condition to trees of depth 1. This strengthening is crucial for our constructions and not possible for other classes of TGDs such as guarded TGDs.

Theorem 6. *Let T_1 and T_2 be sets of frontier-one TGDs, and Σ_D and Σ_Q schemas. Let k be the body width of T_1. Then the following are equivalent:*

1. $T_1 \models^{hom}_{\Sigma_D, \Sigma_Q} T_2$*;*
2. $\mathsf{chase}_{T_2}(D) \rightarrow_{\Sigma_Q} \mathsf{chase}_{T_1}(D)$*, for all tree-like Σ_D-databases D of width at most k.*

The "1 ⇒ 2"-direction is a direct consequence of the definition of hom-conservativity. For the "2 ⇒ 1"-direction, let D be a Σ_D-database witnessing $T_1 \not\models^{\text{hom}}_{\Sigma_D, \Sigma_Q} T_2$, that is, $\mathsf{chase}_{T_2}(D) \not\rightarrow_{\Sigma_Q} \mathsf{chase}_{T_1}(D)$. We show in the appendix that the unraveling U of D into a (potentially infinite) tree-like Σ_D-instance of width k also satisfies $\mathsf{chase}_{T_2}(U) \not\rightarrow_{\Sigma_Q} \mathsf{chase}_{T_1}(U)$. Compactness then yields a finite subset U' of U that still satisfies $\mathsf{chase}_{T_2}(U') \not\rightarrow_{\Sigma_Q} \mathsf{chase}_{T_1}(U')$.

We show in the appendix how Theorem 6 can be used to reduce Σ_D, Σ_Q-hom-conservativity to the ExpTime-complete emptiness problem of two-way alternating tree automata (2ATAs) and in this way obtain a 3ExpTime upper bound. Here, we only give a sketch. Let T_1 and T_2 be sets of frontier-one TGDs, Σ_D and Σ_Q schemas, k the body width of T_1, and ℓ the head width of T_1.

The 2ATA works on input trees that encode a tree-like database D of width at most k along with a tree-like model I_0 of D and T_1 of width at most $\max\{k, \ell\}$. It verifies that $\mathsf{chase}_{T_2}(D) \not\rightarrow_{\Sigma_Q} I_0$. If such an I_0 is found, then $\mathsf{chase}_{T_2}(D) \not\rightarrow_{\Sigma_Q} \mathsf{chase}_{T_1}(D)$ because $\mathsf{chase}_{T_1}(D) \rightarrow I_0$. The converse is also true since $\mathsf{chase}_{T_1}(D)$ is tree-like of width $\max\{k, \ell\}$. In fact, the instance $\mathsf{chase}_{T_1}(D)|^{\downarrow}_c$ that the chase generates below each $c \in \mathsf{adom}(D)$ (see Section 2) is tree-like of width ℓ.

Since our homomorphisms are database-preserving and T_2 is a set of frontier-one TGDs, $\mathsf{chase}_{T_2}(D) \not\rightarrow_{\Sigma_Q} I_0$ if and only if there is a $c \in \mathsf{adom}(D)$ such that $\mathsf{chase}_{T_2}(D)|^{\downarrow}_c \not\rightarrow_{\Sigma_Q} I_0$. The 2ATA may thus check this latter condition, which it does by relying on the notion of a type. Since types also play a role in the subsequent sections, we make this precise.

Let T be a set of frontier-one TGDs. We use $\mathsf{bodyCQ}(T)$ to denote the set of unary or Boolean CQs that can be obtained by starting with the Boolean CQ $\exists y \exists \bar{z}\, \phi(y, \bar{z})$ with $\phi(y, \bar{z})$ the body of some TGD in T, then dropping any number of atoms, and then identifying variables. Finally, we may choose a variable as the answer variable and rename it to the fixed variable x (or stick with a Boolean CQ). A T-*type* is a subset $t \subseteq \mathsf{bodyCQ}(T)$ such that for some instance I that is a model of T and some $c \in \mathsf{adom}(I)$,

1. $q(x) \in t$ iff $c \in q(I)$ for all unary $q(x) \in \mathsf{bodyCQ}(T)$ and
2. $q \in t$ iff $I \models q$ for all Boolean $q \in \mathsf{bodyCQ}(T)$.

We then also use $\mathsf{tp}_T(I, c)$ to denote t. We assume that every type contains the additional formula $\mathsf{true}(x)$ (so that x is guaranteed to occur free in t). We may then view t as a unary CQ with free variable x and thus as a (canonical) database. For brevity, we use t also to denote both of these. $\mathsf{TP}(T)$ is the set of all T-types. Note that the number of types is double exponential in $||T||$. The type $\mathsf{tp}_T(\mathsf{chase}_T(D), c)$ tells us everything we need to know about c in the chase of a database D with T, as follows.

Lemma 7. *Let T be a set of frontier-one TGDs, I an instance, and $c \in \mathsf{adom}(I)$. Then $\mathsf{chase}_T(I)|^{\downarrow}_c$ and $\mathsf{chase}_T(J)|^{\downarrow}_c$ are homomorphically equivalent, where J is obtained from $\mathsf{tp}_T(\mathsf{chase}_T(I), c)$ by replacing the free variable x with c.*

The proof of Lemma 7 is straightforward by reproducing chase steps from the construction of $\mathsf{chase}_T(I)$ in $\mathsf{chase}_T(J)$ and vice versa. Details are omitted.

So to verify that $\mathsf{chase}_{T_2}(D) \not\rightarrow_{\Sigma_Q} I_0$, a 2ATA may guess a constant c in the database D represented by the input tree, and it may also guess the type $\mathsf{tp}_{T_2}(\mathsf{chase}_{T_2}(D), c)$. It then goes on to verify that $\mathsf{tp}_{T_2}(\mathsf{chase}_{T_2}(D), c)$ was guessed correctly (which is not entirely trivial as $\mathsf{chase}_{T_2}(D)$ is *not* encoded in the input). Exploiting Lemma 7, it then starts from type $\mathsf{tp}_{T_2}(\mathsf{chase}_{T_2}(D), c)$ to construct 'in its states' the instance $\mathsf{chase}_{T_2}(D)|_c^{\downarrow}$, simultaneously walking through the instance I_0 encoded by the input tree to verify that, as desired, $\mathsf{chase}_{T_2}(D)|_c^{\downarrow} \not\to_{\Sigma_Q} I_0$ (we actually build a 2ATA for verifying $\mathsf{chase}_{T_2}(D)|_c^{\downarrow} \to_{\Sigma_Q} I_0$ and then complement).

6.2 Deciding CQ-Conservativity

We start with showing that, also for deciding CQ-conservativity, it suffices to consider tree-like databases. In addition, it suffices to consider CQs q of arity 0 or 1.

Theorem 7. *Let T_1 and T_2 be sets of frontier-one TGDs, and Σ_D and Σ_Q schemas. Let k be the body width of T_1. Then the following are equivalent:*

1. $T_1 \models^{CQ}_{\Sigma_D,\Sigma_Q} T_2$;
2. $q_{T_2}(D) \subseteq q_{T_1}(D)$, *for all tree-like Σ_D-databases D of width at most k and connected Σ_Q-CQs q of arity 0 or 1.*

The proof of Theorem 7 first concentrates on restricting the shape of the database, using unraveling and compactness as in the proof of Theorem 6. In a second step, it is then not difficult to restrict also the shape of the CQ.

The following refinement of Theorem 1 is a straightforward consequence of Theorem 7.

Theorem 8. *Let T_1 and T_2 be sets of frontier-one TGDs, Σ_D and Σ_Q schemas, and k the body width of T_1. Then the following are equivalent:*

1. $T_1 \models^{CQ}_{\Sigma_D,\Sigma_Q} T_2$;
2. $\mathsf{chase}_{T_2}(D) \to^{\mathsf{lim}}_{\Sigma_Q} \mathsf{chase}_{T_1}(D)$, *for all tree-like Σ_D-databases D of width at most k.*

Although Theorem 8 looks very similar to Theorem 6, it does not directly suggest a decision procedure. In particular, it is not clear how tree automata can deal with homomorphism limits. We next work towards a more operative characterization that pushes the use of homomorphism limits to parts of the chase that are Σ_Q-disconnected from the database and regular in shape. As we shall see, this allows us to get to grips with homomorphism limits.

For a database D, with $\mathsf{chase}_T(D)|^{\mathsf{con}}_{\Sigma}$ we denote the union of all maximally Σ-connected components of $\mathsf{chase}_T(D)$ that contain at least one constant from $\mathsf{adom}(D)$.

Theorem 9. *Let T_1 and T_2 be sets of frontier-one TGDs, Σ_D and Σ_Q schemas, and k the body width of T_1. Then $T_1 \models^{CQ}_{\Sigma_D,\Sigma_Q} T_2$ iff for all tree-like Σ_D-databases D of width at most k, the following holds:*

1. $\mathsf{chase}_{T_2}(D)|^{\mathsf{con}}_{\Sigma_Q} \to_{\Sigma_Q} \mathsf{chase}_{T_1}(D)$;
2. *for all maximally Σ_Q-connected components I of $\mathsf{chase}_{T_2}(D) \setminus \mathsf{chase}_{T_2}(D)|^{\mathsf{con}}_{\Sigma_Q}$, one of the following holds:*
   (a) $I \to_{\Sigma_Q} \mathsf{chase}_{T_1}(D)$;
   (b) $I \to^{\mathsf{lim}}_{\Sigma_Q} \mathsf{chase}_{T_1}(D)|_c^{\downarrow}$ *for some $c \in \mathsf{adom}(D)$.*

The subsequent example illustrates the theorem.

Example 2. *Consider the sets of TGDs T_1, T_2 and the schemas Σ_D, Σ_Q from Example 1. Recall that T_2 is Σ_D, Σ_Q-CQ-conservative over T_1. Since Σ_D contains only the unary relation A, we may w.l.o.g. concentrate on the Σ_D-database $D = \{A(c)\}$. Clearly, Point 1 of Theorem 9 is satisfied.*

For Point 2, observe that $\mathsf{chase}_{T_2}(D) \setminus \mathsf{chase}_{T_2}(D)|^{\mathsf{con}}_{\Sigma_Q}$ contains only one maximally Σ_Q-connected component, which is of the form

$$I = \{R(c_1, c_0), R(c_2, c_1), \dots\}.$$

Moreover, $I \to^{\mathsf{lim}}_{\Sigma_Q} \mathsf{chase}_{T_1}(D)|_c^{\downarrow}$ and thus Point 2(b) is satisfied. Point 2(a) is not satisfied since $I \not\to_{\Sigma_Q} \mathsf{chase}_{T_1}(D)$.

The easier 'if' direction of the proof of Theorem 9 relies on the fact that, as per Theorem 7, we can concentrate on connected CQs of arity 0 or 1. The interesting direction is 'only if', distinguishing several cases and using several 'skipping homomorphism' arguments (see Lemma 3).

Points 1 and 2(a) of Theorem 9 are amenable to the same tree automata techniques that we have used for hom-conservativity. Point 2(b) achieves the desired expulsion of homomorphism limits, away from the database D to instances of regular shape. In fact, the number of possible T_1-types is independent of D and thus by Lemma 7 the number of distinct instances $\mathsf{chase}_{T_1}(D)|_c^{\downarrow}$ in Point 2(b) that have to be considered is also independent of D. Moreover, these instances are purely chase-generated and thus regular in shape. The same is true for the instances I in Point 2. We next take a closer look at the latter.

Let T be a set of frontier-one TGDs. A *T-labeled database* is a pair $A = (D, \mu)$ with D a database and $\mu : \mathsf{adom}(D) \to \mathsf{TP}(T)$. We associate A with a database D_A that is obtained by starting with D and then adding, for each $c \in \mathsf{adom}(D)$, a disjoint copy D' of the type $\mu(c)$ viewed as a database and glueing the copy of x in D' to c in D_A. We use T-labeled databases to describe fragments of chase-generated instances, and thus assume that D_A contains only null constants. We also associate A with a Boolean CQ q_A, obtained by viewing D_A as such a CQ.

A *labeled Σ-head fragment of T_2* is a T_2-labeled database (F, μ) such that F can be obtained by choosing a TGD $\phi(x, \bar{y}) \to \exists \bar{z}\, \psi(x, \bar{z}) \in T_2$ and taking a maximally Σ-connected component of ψ that does not contain the frontier variable. The following lemma follows from an easy analysis of the chase procedure. Proof details are omitted.

Lemma 8. *Let I be a maximally Σ_Q-connected component of $\mathsf{chase}_{T_2}(D) \setminus \mathsf{chase}_{T_2}(D)|^{\mathsf{con}}_{\Sigma_Q}$, as in Point 2 of Theorem 9. Then for some labeled Σ_Q-head fragment $A = (F, \mu)$ of T_2,*

1. $\mathsf{chase}_{T_2}(D) \models q_A$, *and*
2. *I is homomorphically equivalent to $\mathsf{chase}_T(D_A)|^{\mathsf{con}}_{\Sigma_Q}$.*

Clearly, the number of labeled Σ-head fragments of T_2 is independent of D, just like the number of T_1-types. It thus follows from Lemma 8 and what was said before it that the number of checks '$I \to^{\mathsf{lim}}_{\Sigma_Q} \mathsf{chase}_{T_1}(D)|_c^{\downarrow}$' in Point 2(b) of Theorem 9 does not depend on D: there is at most one such check for every labeled Σ-head fragment of T_2 and every T_1-type. We can do all these checks in a preprocessing step, before starting to build 2ATAs for CQ-conservativity that implement the characterization provided by Theorem 9. Whenever the 2ATA needs to carry out a check '$I \to^{\mathsf{lim}}_{\Sigma_Q}$

$\mathsf{chase}_{T_1}(D)|^{\downarrow}_c$' to verify Point 2(b), we can simply look up the precomputed result and let the 2ATA reject immediately if it is negative. Thus, the automata are completely freed from dealing with homomorphism limits.

6.3 Precomputing Homomorphism Limits

It remains to show how to actually achieve the precomputation of the tests '$I \rightarrow^{\mathsf{lim}}_{\Sigma_Q} \mathsf{chase}_{T_1}(D)|^{\downarrow}_c$' in Point 2(b) of Theorem 9. This is where we finally deal with homomorphism limits. The following theorem makes precise the problem that we actually have to decide.

Theorem 10. *Given two sets of frontier-one TGDs T_1 and T_2, a schema Σ, a labeled Σ-head fragment $A = (D, \mu)$ for T_2, and a T_1-type $\widehat{t}$, it can be decided in time triple exponential in $||T_1|| + ||T_2||$ whether $\mathsf{chase}_{T_2}(D_A)|^{\mathsf{con}}_{\Sigma} \rightarrow^{\mathsf{lim}}_{\Sigma} \mathsf{chase}_{T_1}(\widehat{t})$.*

We invite the reader to compare the decision problem formulated in Theorem 10 with Point 2(b) of Theorem 9 in the light of Lemmas 7 and 8. The decision procedure used to prove Theorem 10 is again based on tree automata. To enable their use, however, we first rephrase the decision problem in Theorem 10 in a way that replaces homomorphism limits with unbounded homomorphisms.

Let T_1, T_2, Σ, $A = (D, \mu)$, and $\widehat{t}$ be as in Theorem 10. Recall that A is associated with a database D_A and a Boolean CQ q_A. Here, we additionally use unary CQs q^c_A, for every $c \in \mathsf{adom}(D)$, which are defined exactly like q_A except that c is now the answer variable.

The main idea for proving Theorem 10 is to replace homomorphism limits into $\mathsf{chase}_{T_1}(\widehat{t})$ with homomorphisms into a class of instances $\mathcal{R}(T_1, \widehat{t})$ whose disjoint union should be viewed as a relaxation of $\mathsf{chase}_{T_1}(\widehat{t})$. In particular, this relaxation admits a homomorphism limit to $\mathsf{chase}_{T_1}(\widehat{t})$, but not a homomorphism. Let us make this precise.

We again use instance trees. This time, however, they are not based on directed trees, but on *directed pseudo-trees*, that is, finite or infinite directed graphs $G = (V, E)$ such that every node $v \in V$ has at most one incoming edge and G is connected and contains no cycle.[3] Note that infinite directed pseudo-trees need not have a root. For example, a two-way infinite path qualifies as a directed pseudo-tree.

A T_1-*labeled instance tree* has the form $\mathcal{T} = (V, E, B, \mu)$ with $\mathcal{T}' = (V, E, B)$ an instance tree (based on a directed pseudo-tree) and $\mu : \mathsf{adom}(I_{\mathcal{T}'}) \rightarrow \mathsf{TP}(T_1)$ a function that assigns a T_1-type to every element in $I_{\mathcal{T}'}$. For $v \in V$, we use μ_v to denote the restriction of μ to $\mathsf{adom}(B(v))$. Moreover, we set $I_{\mathcal{T}} = I_{\mathcal{T}'}$. We say that $\mathcal{T}$ is $\widehat{t}$-*proper* if the following conditions are satisfied:

1. for every $v \in V$, one of the following holds:
   (a) v is the root of (V, E), $B(v)$ has the form $\{\mathsf{true}(c_0)\}$, and $\mu(c_0) = \widehat{t}$;
   (b) there is a TGD ϑ in T_1 such that $B(v)$ is isomorphic to the head of ϑ and $\widehat{t}, T_1 \models q_{(B(v),\mu_v)}$;
2. for every $(u, v) \in E$ such that $B(u) \cap B(v)$ contains a (single) constant c, we have $\mu_u(c), T_1 \models q^c_{(B(v),\mu_v)}(x)$. That is: the constant x from the type $\mu_u(c)$ viewed as a database is an answer to the unary CQ $q^c_{(B(v),\mu_v)}$ w.r.t. T_1.

[3] Neither in the directed nor in the undirected sense, which is equivalent if every node has at most one incoming edge.

The announced class $\mathcal{R}(T_1, \widehat{t})$ consists of all instances I such that $I = I_{\mathcal{T}}$ for some $\widehat{t}$-proper T_1-labeled instance tree $\mathcal{T}$. It is easy to see that $\mathsf{chase}_{T_1}(\widehat{t}) \in \mathcal{R}(T_1, \widehat{t})$ as there is a $\widehat{t}$-proper T_1-labeled instance tree $\mathcal{T}$ such that $I_{\mathcal{T}} = \mathsf{chase}_{T_1}(\widehat{t})$. However, there are also instances $I \in \mathcal{R}(T_1, \widehat{t})$ that do not admit a homomorphism to $\mathsf{chase}(\widehat{t}, T_1)$. The following example illustrates their importance.

Example 3. *Consider $T_1, T_2, \Sigma_D, \Sigma_Q$ from Example 1 and I, D, c from Example 2. Let $\widehat{t} = \mathsf{tp}_{T_1}(\mathsf{chase}_{T_1}(D), c)$, that is, $\widehat{t} = \{A(x), \exists x\, A(x), \exists x\, B(x)\}$. Then $I \not\rightarrow \mathsf{chase}_{T_2}(\widehat{t})$. However, we find a $\widehat{t}$-proper T_1-labeled instance tree $\mathcal{T} = (V, E, B, \mu)$ such that $I \rightarrow I_{\mathcal{T}}$.*

We may construct $\mathcal{T}$ by starting with a single node v_0, $B(v_0) = \{R(c_1, c_0)\}$, and

$$\mu(c_0) = \mu(c_1) = \{B(x), \exists x\, A(x), \exists x\, B(x)\}.$$

Then repeatedly add a predecessor v_{i+1} of v_i, with $B(v_{i+1}) = \{R(c_{i+1}, c_i)\}$ and $\mu(c_{i+1}) = \mu(c_0)$, ad infinitum. The resulting tree $\mathcal{T}$ is $\widehat{t}$-proper and satisfies $I_{\mathcal{T}} = I$. Note that it does not have a root.

The next lemma is the core ingredient to the proof of Theorem 10. Informally, it states that when replacing $\mathsf{chase}_{T_1}(\widehat{t})$ with instances from $\mathcal{R}(T_1, \widehat{t})$, we may also replace homomorphism limits with homomorphisms.

Lemma 9. *Let I be a countable Σ-connected instance such that $\mathsf{adom}(I)$ contains only nulls. Then $I \rightarrow^{\mathsf{lim}}_{\Sigma} \mathsf{chase}_{T_1}(\widehat{t})$ iff there is an $\widehat{I} \in \mathcal{R}(T_1, \widehat{t})$ with $I \rightarrow \widehat{I}$.*

In the proof of Lemma 9, the laborious direction is 'only if', where one assumes that $I \rightarrow^{\mathsf{lim}}_{\Sigma} \mathsf{chase}_{T_1}(\widehat{t})$ and then uses finite subinstances $J_1 \subseteq J_2 \subseteq \cdots$ of I with $I = \bigcup_{i \geq 1} J_i$ and homomorphisms h_i from J_i to $\mathsf{chase}_{T_1}(\widehat{t})$, $i \geq 1$, to identify the desired instance $\widehat{I} \in \mathcal{R}(T_1, \widehat{t})$. This again involves several 'skipping homomorphisms' type of arguments.

Using Lemma 9, we give a decision procedure based on 2ATAs that establishes Theorem 10. The 2ATA accepts input trees encoding an instance $I \in \mathcal{R}(T_1, \widehat{t})$ that admits a Σ-homomorphism from $\mathsf{chase}_{T_2}(D_A)|^{\mathsf{con}}_{\Sigma}$.

7 Future Work

It would be interesting to determine the exact complexity of hom- and CQ-conservativity for frontier-one TGDs. We tend to think that these problems are 3EXPTIME-complete. Note that in the description logic $\mathcal{ELI}$, they are 2EXPTIME-complete (Jung et al. 2020).

It would also be interesting to study conservative extensions and triviality for other classes of TGDs that have been proposed in the literature. Of course, it would be of particular interest to identify decidable cases. Classes for which undecidability does not follow from the results in this paper include acyclic and sticky TGDs, which exist in several forms, see for instance (Calì, Gottlob, and Pieris 2010).

Acknowledgements

Carsten Lutz was supported by the DFG project LU 1417/3-1 QTEC and Jery Marcinkowski by the Polish National Science Centre (NCN) grant 2016/23/B/ST6/01438.

References

Abiteboul, S.; Hull, R.; and Vianu, V. 1995. *Foundations of Databases*. Addison-Wesley.

Artale, A.; Calvanese, D.; Kontchakov, R.; and Zakharyaschev, M. 2009. The DL-Lite family and relations. *J. Artif. Intell. Res.* 36:1–69.

Baader, F.; Horrocks, I.; Lutz, C.; and Sattler, U. 2017. *An Introduction to Description Logic*. Cambridge University Press.

Baget, J.; Mugnier, M.; Rudolph, S.; and Thomazo, M. 2011. Walking the complexity lines for generalized guarded existential rules. In *Proc. of IJCAI*, 712–717.

Benedikt, M.; Bourhis, P.; and Senellart, P. 2012. Monadic datalog containment. In *Proc. of ICALP*, volume 7392 of *LNCS*, 79–91. Springer.

Bienvenu, M., and Ortiz, M. 2015. Ontology-mediated query answering with data-tractable description logics. In *Reasoning Web*, 218–307.

Bienvenu, M.; ten Cate, B.; Lutz, C.; and Wolter, F. 2014. Ontology-based data access: A study through disjunctive Datalog, CSP, and MMSNP. *ACM Transactions on Database Systems* 39(4):33:1–33:44.

Botoeva, E.; Konev, B.; Lutz, C.; Ryzhikov, V.; Wolter, F.; and Zakharyaschev, M. 2016. Inseparability and conservative extensions of description logic ontologies: A survey. In *Proc. of Reasoning Web*, volume 9885 of *LNCS*, 27–89. Springer.

Botoeva, E.; Lutz, C.; Ryzhikov, V.; Wolter, F.; and Zakharyaschev, M. 2019. Query inseparability for $\mathcal{ALC}$ ontologies. *Artif. Intell.* 272:1–51.

Calì, A.; Gottlob, G.; Lukasiewicz, T.; Marnette, B.; and Pieris, A. 2010. Datalog+/-: A family of logical knowledge representation and query languages for new applications. In *Proc. of LICS*, 228–242. IEEE Computer Society.

Calì, A.; Gottlob, G.; and Kifer, M. 2013. Taming the infinite chase: Query answering under expressive relational constraints. *J. Artif. Intell. Res.* 48:115–174.

Calì, A.; Gottlob, G.; and Lukasiewicz, T. 2012. A general datalog-based framework for tractable query answering over ontologies. *J. Web Semant.* 14:57–83.

Calì, A.; Gottlob, G.; and Pieris, A. 2010. Advanced processing for ontological queries. *Proc. VLDB Endow.* 3(1):554–565.

Calvanese, D.; Giacomo, G. D.; Lembo, D.; Lenzerini, M.; Poggi, A.; Rodriguez-Muro, M.; and Rosati, R. 2009. Ontologies and databases: The DL-Lite approach. In *Reasoning Web*, volume 5689 of *LNCS*, 255–356.

Casanova, M. A.; Fagin, R.; and Papadimitriou, C. H. 1984. Inclusion dependencies and their interaction with functional dependencies. *J. Comput. Syst. Sci.* 28(1):29–59.

Comon, H.; Dauchet, M.; Gilleron, R.; Löding, C.; Jacquemard, F.; Lugiez, D.; Tison, S.; and Tommasi, M. 2007. Tree automata techniques and applications. Available at http://www.grappa.univ-lille3.fr/tata. October, 12th 2007.

Conway, J. 1972. Unpredictable iterations. In *Proc. of 1972 Number Theory Conference*, 49–52.

Fagin, R.; Kolaitis, P. G.; Miller, R. J.; and Popa, L. 2005. Data exchange: semantics and query answering. *Theor. Comput. Sci.* 336(1):89–124.

Ghilardi, S.; Lutz, C.; and Wolter, F. 2006. Did I damage my ontology? A case for conservative extensions in description logics. In *Proc. of KR*, 187–197. AAAI Press.

Gogacz, T., and Marcinkowski, J. 2014. All-instances termination of chase is undecidable. In *Proc. of ICALP*, volume 8573 of *LNCS*, 293–304. Springer.

Gottlob, G.; Kikot, S.; Kontchakov, R.; Podolskii, V. V.; Schwentick, T.; and Zakharyaschev, M. 2014. The price of query rewriting in ontology-based data access. *Artif. Intell.* 213:42–59.

Gottlob, G.; Manna, M.; and Pieris, A. 2015. Polynomial rewritings for linear existential rules. In *Proc. of IJCAI*, 2992–2998. AAAI Press.

Grädel, E., and Walukiewicz, I. 1999. Guarded fixed point logic. In *Proc. of LICS*, 45–54.

Gutiérrez-Basulto, V.; Jung, J. C.; and Sabellek, L. 2018. Reverse engineering queries in ontology-enriched systems: The case of expressive horn description logic ontologies. In *Proc. of IJCAI*, 1847–1853. ijcai.org.

Johnson, D. S., and Klug, A. C. 1984. Testing containment of conjunctive queries under functional and inclusion dependencies. *J. Comput. Syst. Sci.* 28(1):167–189.

Jung, J. C.; Lutz, C.; Martel, M.; Schneider, T.; and Wolter, F. 2017. Conservative extensions in guarded and two-variable fragments. In *Proc. of ICALP*, LIPIcs, 108:1–108:14.

Jung, J. C.; Lutz, C.; Martel, M.; and Schneider, T. 2018. Querying the unary negation fragment with regular path expressions. In *Proc. of ICDT*, volume 98 of *LIPIcs*, 15:1–15:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Jung, J. C.; Lutz, C.; Martel, M.; and Schneider, T. 2020. Conservative extensions in horn description logics with inverse roles. *J. Artif. Intell. Res.* 68:365–411.

Jung, J. C.; Lutz, C.; and Marcinkowski, J. 2022. Conservative extensions for existential rules. *CoRR* abs/2202.05689.

Konev, B.; Kontchakov, R.; Ludwig, M.; Schneider, T.; Wolter, F.; and Zakharyaschev, M. 2011. Conjunctive query inseparability of OWL 2 QL TBoxes. In *Proc. of AAAI*. AAAI Press.

Lutz, C.; Walther, D.; and Wolter, F. 2007. Conservative extensions in expressive description logics. In *Proc. of IJCAI*, 453–458.

Ovid. 2008. *Metamorphoses, first edition 8 AD, Translated by A. D. Melville. Introduction and notes by Edward John Kenney.* Oxford University Press.

Vardi, M. Y. 1998. Reasoning about the past with two-way automata. In *Proc. of ICALP*, 628–641.

A Proofs for Section 3

Theorem 1. *Let T_1 and T_2 be sets of TGDs and Σ_D, Σ_Q schemas. Then $T_1 \models^{CQ}_{\Sigma_D,\Sigma_Q} T_2$ iff $\mathsf{chase}_{T_2}(D) \rightarrow^{\mathsf{lim}}_{\Sigma_Q} \mathsf{chase}_{T_1}(D)$.*

Proof. The 'if' direction should be clear. For the 'only if' direction, take a Σ_D-database D and any finite induced subinstance I of $\mathsf{chase}_{T_2}(D)$. We may view I as a CQ q with the constants from $\mathbf{C}$ as answer variables and those from $\mathbf{N}$ as quantified variables. The identity is a homomorphism from q to I, giving rise to an answer $\bar{a}$ to q on D w.r.t. T_2 that consists of the constants in I that are from $\mathbf{C}$. Since $T_1 \models^{\mathsf{CQ}}_{\Sigma_D,\Sigma_Q} T_2$, $\bar{a}$ is an answer to q on D w.r.t. T_1, and this gives the desired homomorphism from I (which is q) to $\mathsf{chase}_{T_1}(D)$. □

Lemma 3. *Let I_1, I_2 be instances such that I_1 is countable and I_2 is finite, and let Σ be a schema. If $I_1 \rightarrow^{\mathsf{lim}}_{\Sigma} I_2$, then $I_1 \rightarrow_{\Sigma} I_2$.*

Proof. Assume that $I_1 \rightarrow^{\mathsf{lim}}_{\Sigma} I_2$. We need to find a database-preserving Σ-homomorphism h from I_1 to I_2. Let $\mathsf{adom}(I_1) = \{c_1, c_2, \dots\}$ (finite or infinite) and for $i \geq 1$, let A_i be the set of the first $\min\{i, |\mathsf{adom}(I_1)|\}$ constants from the sequence $c_1, c_2, \dots$. Since $I_1 \rightarrow^{\mathsf{lim}}_{\Sigma} I_2$, we find for each $i \geq 0$ a database-preserving Σ-homomorphism h_i from $I_1|_{A_i}$ to I_2. For $A \subseteq A_i$, we use $h_i|_A$ to denote the restriction of h_i to domain A. We would be done if we knew that the sequence $h_1, h_2, \dots$ satisfied the following uniformity condition:

$(*)$ $h_j|_{A_i} = h_i$ for $j > i > 0$,

as then we could simply take $h = \bigcup_{i\geq 1} h_i$. We show how to extract from $h_1, h_2, \dots$ a sequence $g_1, g_2, \dots$ that satisfies Condition $(*)$, and then define $h = \bigcup_{i\geq 1} g_i$. During the construction of the sequence $g_1, g_2, \dots$, we shall take care that for every g_i, there are infinitely many $j > i$ with $h_j|_{A_i} = g_i$.

To start, we need a homomorphism from $I_1|_{A_1}$ to I_2. Since A_1 and I_2 are finite, there are only finitely many mappings $f : A_1 \rightarrow \mathsf{adom}(I_2)$ and thus there must be such a mapping f such that $h_i|_{A_1} = f$ for infinitely many $i \geq 1$. Set $g_1 = f$.

Now assume that $g_1, \dots, g_n$ have already been defined. Then there is an infinite set Γ of indices $j > i$ such that for all $j \in \Gamma$, $h_j|_{A_n} = g_n$. Since I_2 is finite, there are only finitely many extensions $f : A_{n+1} \rightarrow \mathsf{adom}(I_2)$ of g_n an thus there must be such an extension f such that $h_j|_{A_{n+1}} = f$ for infinitely many $j \in \Gamma$. Set $g_{n+1} = f$.

The resulting sequence clearly satisfies $(*)$ and thus the proof is done. □

B Proofs for Section 4

B.1 Proof of Lemma 4

Before proving Lemma 4, we analyse $\mathsf{chase}_{T_1}(\emptyset)$ and $\mathsf{chase}_{T_2}(\emptyset)$. Recall that $T_1 = T_{\mathsf{rec}} \cup T_{\mathsf{proj}}$ and $T_2 = T_1 \cup T_{\mathsf{myth}}$. It is easy to notice that:

Lemma 10. $\mathsf{chase}_{T_2}(\emptyset) = \mathsf{chase}_{T_{\mathsf{myth}}}(\mathsf{chase}_{T_1}(\emptyset))$.

Proof. The claim follows since all the facts T_{myth} can possibly produce are from Σ_Q. And all the facts in all the bodies of rules from T_1 are from Σ_F. □

Now we are going to study Σ_Q-homomorphisms from $\mathsf{chase}_{T_2}(\emptyset)$ to $\mathsf{chase}_{T_1}(\emptyset)$. First of all notice that:

Lemma 11. *Identity is the only Σ_Q-homomorphism from $\mathsf{chase}_{T_1}(\emptyset)$ to $\mathsf{chase}_{T_1}(\emptyset)$.*

Proof. Suppose h is a Σ_Q-homomorphism from $\mathsf{chase}_{T_1}(\emptyset)$ to $\mathsf{chase}_{T_1}(\emptyset)$. It is easy to notice that $h(x) = x$ if

(i) x is one of the constants introduced by the initial 'something out of nothing' rule for the existentially quantified variables c, e_1, e_2 or

(ii) $\mathsf{Mouth}(x)$ holds.

We may use (i) to prove that x is a bridge if and only if $h(x)$ is a bridge or, in other words, that

(iii) h maps bridges to bridges and non-bridges to non-bridges.

As the next step, imagine two bridges b and b', such that there is a Pyramus path from b' to b that does not visit any other bridge. For such b and b', define $dist(b', b)$ to be the length of this Pyramus path. Now it follows from the construction of T_1 that if for bridges b, b', b'' it holds that $dist(b', b) = dist(b'', b)$ then $b' = b''$. Since distance is preserved by homomorphisms, this implies that

(iv) if x is a bridge then $h(x) = x$

The detailed proof would be by induction, with (ii) serving as induction basis and (iii) used in the induction step.

It then easily follows from (iv) that the lemma also holds true for the non-bridge constants. □

Let $\mathcal{C}$ denote the chase of $\{\mathsf{Encounter}(\mathsf{c}_0, \mathsf{c}_0')\}$ shown in Figure 1. For an instance I and a fact $F = \mathsf{Encounter}(\mathsf{c}, \mathsf{c}') \in I$, we denote with $I \cup_F \mathcal{C}$ the instance obtained from I by adding a disjoint copy of $\mathcal{C}$ to I and identifying c_0 with c and c_0' with c'. It is easy to see that:

Lemma 12. *Let $E = \mathsf{chase}_{T_1}(\emptyset)|_{\{\mathsf{Encounter}\}}$ be the set of all facts for the relation symbol $\mathsf{Encounter}$ in chase_{T_1}. Then:*

$$\mathsf{chase}_{T_2}(\emptyset) = \bigcup_{F\in E} \mathsf{chase}_{T_1}(\emptyset) \cup_F \mathcal{C}$$

Informally, the lemma says that $\mathsf{chase}_{T_2}(\emptyset)$ is $\mathsf{chase}_{T_1}(\emptyset)$ with one new copy of the chase $\mathcal{C}$ from Figure 1, attached to every fact F of the relation $\mathsf{Encounter}$ in $\mathsf{chase}_{T_1}(\emptyset)$.

It follows easily from Lemma 12 and Lemma 11 that:

Lemma 13. *The following two conditions are equivalent:*

(i) There exists a Σ_Q-homomorphism h from $\mathsf{chase}_{T_2}(\emptyset)$ to $\mathsf{chase}_{T_1}(\emptyset)$.

(ii) For each fact F of the relation $\mathsf{Encounter}$ in $\mathsf{chase}_{T_1}(\emptyset)$ there exists a Σ_Q-homomorphism h_F from $\mathsf{chase}_{T_1}(\emptyset) \cup_F \mathcal{C}$ to $\mathsf{chase}_{T_1}(\emptyset)$.

Proof. The (i) $\Rightarrow$ (ii) direction is trivial. For the opposite direction, assume (ii), consider all the homomorphisms h_F and notice that (by Lemma 11) they all agree on $\mathsf{chase}_{T_1}(\emptyset)$. Now take h as the union of all h_F. □

Let us now analyze the structure of $\mathsf{chase}_{T_1}(\emptyset)$. Notice that for each[4] fact F_1 of $\mathsf{chase}_{T_1}(\emptyset)|_{\Sigma_F}$, there exists exactly one *parent of* F_1, a fact F_0 of $\mathsf{chase}_{T_1}(\emptyset)|_{\Sigma_F}$ such that F_1 was created directly from F_0 by an application of a TGD from T_{rec}. We write $F_0 \to F_1$ to say that F_0 is a parent of F_1. By $\overset{*}{\to}$, we denote the reflexive and transitive closure of $\to$. Notice that if F is a fact of relation End, then $F \to G$ is never true.

For a fact $F \in \mathsf{chase}_{T_1}(\emptyset)|_{\Sigma_F}$, define:

$$\mathsf{Ancestors}(F) = \{G \in \mathsf{chase}_{T_1}(\emptyset)|_{\Sigma_F} : G \overset{*}{\to} F\}$$

In natural language, $\mathsf{Ancestors}(F)$ comprises all the facts of $\mathsf{chase}_{T_1}(\emptyset)|_{\Sigma_F}$ which were necessary to produce F (with fact F included).

Finally, let $\mathsf{Ancestors}_{\Sigma_Q}(F)$ be the set of all Σ_Q-facts that can be produced from some fact in $\mathsf{Ancestors}(F)$ by using a rule from T_{proj}.

Now, it follows from the construction of T_1 that:

Lemma 14. $\mathsf{chase}_{T_1}(\emptyset)$ *satisfies the following:*

(i) For a fact $F \in \mathsf{chase}_{T_1}(\emptyset)|_{\Sigma_F}$ *the database* $\mathsf{Ancestors}_{\Sigma_Q}(F)$ *is a* locally correct *river if and only if* F *is an fact of the relation* End.

(ii) For each locally correct sequence κ *there exists a fact* $F \in \mathsf{chase}_{T_1}(\emptyset)$ *(of the relation* End*) such that* $\mathsf{Ancestors}_{\Sigma_Q}(F)$ *is (isomorphic to)* River_κ.

We are now in the position to prove Lemma 4, restated here for convenience:

Lemma 4. $\digamma$ *does not stop iff* $T_2 = T_1 \cup T_{\mathsf{myth}}$ *is* Σ_Q, Σ_D*-hom-conservative over* $T_1 = T_{\mathsf{rec}} \cup T_{\mathsf{proj}}$.

The "$\Rightarrow$"-direction of this equivalence is now easy to show. If $\digamma$ does not stop then every locally correct river is incorrect. This means (using Lemma 14 (i) and Observation 1) that for each fact G of relation Encounter in $\mathsf{chase}_{T_1}(\emptyset)$, which was created by projecting some fact F of the relation End, the instance $\mathsf{chase}_{T_1}(\emptyset) \cup_G \mathcal{C}$ (connected to $\mathsf{chase}_{T_1}(\emptyset)$ via G) can be homomorphically embedded in $\mathsf{Ancestors}_{\Sigma_Q}(F)$. So, by Lemma 13, there exists a Σ_Q-homomorphism h from $\mathsf{chase}_{T_2}(\emptyset)$ to $\mathsf{chase}_{T_1}(\emptyset)$.

What about the "$\Leftarrow$"-direction? Suppose $\digamma$ stops. Then there is an n such that $\digamma^n(2) = 1$. Let $\kappa = \langle[p_1, \ldots, p_n], [t_1, \ldots, t_n]\rangle$ with $p_1 = 2$, $t_n = 1$, and $p_{i+1} = \digamma(p_i)$ as well as $t_i = p_{i+1}$ for $1 \le i < n$. It is easy to see that κ is correct. We know, from Lemma 14 (ii), that there is an F in $\mathsf{chase}_{T_1}(\emptyset)$ such that $\mathsf{Ancestors}_{\Sigma_Q}(F)$ is River_κ. Hence (using Observation 1 once again) we know that there is no homomorphism from $\mathsf{Ancestors}_{\Sigma_Q}(F) \cup_G \mathcal{C}$ to $\mathsf{Ancestors}_{\Sigma_Q}(F)$ (where again, G is a fact of relation Encounter resulting from projecting F). By Lemma 13, it suffices to prove is that there is no homomorphism from $\mathsf{chase}_{T_1}(\emptyset) \cup_G \mathcal{C}$ to $\mathsf{chase}_{T_1}(\emptyset)$. Now, observe that $\mathsf{chase}_{T_1}(\emptyset)$ is a union of all possible locally correct instances River_κ but it is *not* a disjoint union: they all share the mouth, and there is a lot of overlap between them. So maybe it is possible that one could homomorphically embed, in $\mathsf{chase}_{T_1}(\emptyset)$, the copy of $\mathcal{C}$ attached to G, using facts of $\mathsf{chase}_{T_1}(\emptyset)$ which are not in $\mathsf{Ancestors}_{\Sigma_Q}(F)$?

Our last lemma says that there is no such embedding:

Lemma 15. *If there exists a homomorphism from* $\mathsf{Ancestors}_{\Sigma_Q}(F) \cup_G \mathcal{C}$ *to* $\mathsf{chase}_{T_1}(\emptyset)$ *then there exists a homomorphism from* $\mathsf{Ancestors}_{\Sigma_Q}(F) \cup_G \mathcal{C}$ *to* $\mathsf{Ancestors}_{\Sigma_Q}(F)$.

Proof. Note that the homomorphism of $\mathcal{C}$ into $\mathsf{chase}_{T_1}(\emptyset)$ starts at some fact of relation Encounter, and then follows essentially facts $\mathsf{Pyramus}(c, c')$ or $\mathsf{Thisbe}(c, c')$ in this direction. So it remains to observe that sets $\mathsf{Ancestors}_{\Sigma_Q}(F)$ are closed under taking successors along relations Pyramus and Thisbe, by construction of T_1.

More precisely, suppose x is a constant of $\mathsf{Ancestors}_{\Sigma_Q}(F)$ and y is a constant of $\mathsf{chase}_{T_1}(\emptyset)$. Suppose also that the fact $\mathsf{Pyramus}(x, y)$ or the fact $\mathsf{Thisbe}(x, y)$ is in $\mathsf{chase}_{T_1}(\emptyset)$. Then y is a constant of $\mathsf{Ancestors}_{\Sigma_Q}(F)$. □

In consequence, the "$\Leftarrow$"-direction of Lemma 4 also holds, and Point 1 of Theorem 2 is proven.

B.2 Proof of Point 2 of Theorem 2

In order to prove Point 2 of Theorem 2 let us just notice that we can simply reuse the proof of Point 1 from the previous section.

Clearly, if $\digamma$ does not stop then T_2 is Σ_D, Σ_Q-hom-conservative over T_1 so it is also Σ_D, Σ_Q-CQ-conservative. We need to notice that if $\digamma$ stops then T_2 is not Σ_D, Σ_Q-CQ-conservative over T_1. To this end, it will be enough to find a *finite* subinstance Q of $\mathsf{chase}_{T_2}(\emptyset)$ which cannot be homomorphically embedded in $\mathsf{chase}_{T_1}(\emptyset)$.

So suppose $\digamma$ stops. Then there is an n such that $\digamma^n(2) = 1$. Let $\kappa = \langle[p_1, \ldots, p_n], [t_1, \ldots, t_n]\rangle$ with $p_1 = 2$, $t_n = 1$, and $p_{i+1} = \digamma(p_i)$ as well as $t_i = p_{i+1}$ for $1 \le i < n$. Let $m \in \mathbb{N}$ be any natural number bigger than the longest Thisbe or Pyramus path in River_κ that does not visit an eternity.

Let $\mathcal{C}_m$ be the fragment of the chase $\mathcal{C}$ resulting from m applications of existential TGDs in T_{myth}, (and then using the datalog projections)[5]. Let also $\mathsf{River}_\kappa \cup_G \mathcal{C}_m$ be the union of the two instances, with the only fact of the Encounter relation in River_κ identified with the only Encounter fact in $\mathcal{C}_m$.

We can use the arguments that were used in the proof of the "$\Leftarrow$"-direction in the previous section to show that $\mathsf{River}_\kappa \cup_G \mathcal{C}_m$ is the Q we need.

B.3 Proof of Point 3 of Theorem 2

We use the same schema Σ_Q as before, except that Encounter is not in Σ_Q. We will define Σ_D so that $\Sigma_Q \subseteq \Sigma_D$. Also, Start is now in Σ_D. And for each rule $\mathcal{R}$ from T_{rec} of the form

$$P(\bar{x}, \bar{y}) \to \exists \bar{z}\, Q(\bar{y}, \bar{z})$$

[4]With the obvious exception of the single fact of the relation Start in $\mathsf{chase}_{T_1}(\emptyset)$, which has no parent.

[5]The instance from Figure 1 can be seen as $\mathcal{C}_4$.

Σ_D contains a new relation symbol $\mathcal{S}_\mathcal{R}$ of arity $|\bar{x}|+|\bar{y}|+|\bar{z}|$ in Σ_D.

We now define a set of TGDs T_0 whose triviality we are interested in. T_0 contains all rules from T_{myth} as well as the new rule

$$\mathsf{End}(\dagger, b, b'), \mathsf{Pyramus}(b', b), \mathsf{Thisbe}(b', b) \rightarrow \mathsf{Encounter}(b, b').$$

Finally, for each rule $\mathcal{R}$ from T_{rec}, as above, T_0 contains the following rule:

$$(\clubsuit)\ \ \mathcal{S}_\mathcal{R}(\bar{x},\bar{y},\bar{z}), P(\bar{x},\bar{y}), \mathsf{chase}_{T_{\mathsf{proj}}}(\{P(\bar{x},\bar{y})\}) \ \rightarrow\ Q(\bar{y},\bar{z}).$$

Notice that this rule is indeed guarded. This is because $\mathsf{chase}_{T_{\mathsf{proj}}}(\{P(\bar{x},\bar{y})\})$ is finite[6] and contains only variables from $\bar{x}\cup\bar{y}$.

Notice that the only Σ_Q-facts created when chasing with T_0 are the ones created by T_{myth} and, in order for them to be created, some Encounter fact $\mathsf{Encounter}(c, c')$ must be produced first. It is now easy to see that:

Lemma 16. *Let D be a Σ_D-database and suppose that for each fact $F = \mathsf{Encounter}(c, c')$ in $\mathsf{chase}_{T_0}(D)$, there exists a database-preserving Σ_Q-homomorphism h from $T_{\mathsf{myth}}(\{F\})$ to D, such that $h(c) = c$ and $h(c') = c'$. Then there exists a database-preserving Σ_Q-homomorphism from $\mathsf{chase}_{T_{\mathsf{myth}}}(D)$ to D.*

To establish Point 3 of Theorem 2, we need to prove the following.

Lemma 17. *$\digamma$ does not stop if and only if T is Σ_D, Σ_Q-trivial.*

So assume first that $\digamma$ stops. Then there is an n such that $\digamma^n(2) = 1$. Let $\kappa = \langle [p_1,\ldots,p_n],[t_1,\ldots,t_n]\rangle$ with $p_1 = 2$, $t_n = 1$, and $p_{i+1} = \digamma(p_i)$ as well as $t_i = p_{i+1}$ for $1 \le i < n$. We need to produce a database D such that $\mathsf{chase}_{T_0}(D)$ has no database-preserving Σ_Q-homomorphism to D.

We know, from Lemma 14(ii) that there exists a fact $F \in \mathsf{chase}_{T_1}(\emptyset)$ (of the relation End) such that $\mathsf{Ancestors}_{\Sigma_Q}(F) = \mathsf{River}_\kappa$. Let D be the database that consists of the following:

- all facts of $\mathsf{Ancestors}_{\Sigma_Q}(F)$, with the exception of the Encounter fact, which is not in Σ_D (so we almost have the entire River_κ in D, just the Encounter fact is missing);
- the (only) fact of the relation Start from $\mathsf{chase}_{T_1}(\emptyset)$;
- for any two facts $G = P(\bar{a},\bar{a'})$ and $G' = Q(\bar{a'},\bar{a''})$ from $\mathsf{Ancestors}(F)$ such that $G \rightarrow G'$ and G' was created from G using the rule $\mathcal{R}$ of T_{rec}, the fact $\mathcal{S}_\mathcal{R}(\bar{a},\bar{a'},\bar{a''})$.

Now let us try to imagine what facts will be produced by $\mathsf{chase}_{T_0}(D)$. There is the Start fact in D, and it will match with the P of one of the rules of the form ($\clubsuit$) in T_0, producing a next fact in $\mathsf{Ancestors}(F)$, which will again match with the P of some other ($\clubsuit$) rule, and so on. All the facts of $\mathsf{Ancestors}(F)$ will be produced in this way, with F as the last one. But F is a fact that uses the relation End, so we have a rule in T_0 that will now use F to produce the missing Encounter fact of River_κ. At this point T_{myth} will fire, producing the chase $\mathcal{C}$, which (by Observation 1, since κ is correct) will not have a database-preserving Σ_Q-homomorphism to D.

For the converse direction, assume that $\digamma$ does not stop and let D be any Σ_D-database.

The next lemma says that if any Encounter fact is created in $\mathsf{chase}_{T_0}(D)$ then D must contain (a homomorphic image of) an entire partially correct river:

Lemma 18.

(i) If $F \in \mathsf{chase}_{T_0}(D)$ is a $\Sigma_\digamma$-fact, then there is a fact $F_0 \in \mathsf{chase}_{T_1}(\emptyset)$ of the same relation such that there exists a homomorphism h from $\mathsf{Ancestors}(F_0) \cup \mathsf{Ancestors}_{\Sigma_Q}(F_0)$ to $\mathsf{chase}_{T_0}(D)$ with $h(F_0) = F$.

(ii) If $G \in \mathsf{chase}_{T_0}(D)$ is a fact of the relation Encounter, then there exists a partially correct river River_κ and a homomorphism h from River_κ to $\mathsf{chase}_{T_0}(D)$ such that $h(G_0) = G$, where G_0 is the Encounter fact of River_κ.

Proof. (i) By induction of the number of steps of the chase needed to create F. If it is zero, meaning that $F \in D$, then F must be a fact of the relation Start (which is the only relation from $\Sigma_\digamma$ which is also in Σ_D), and the claim holds true. If it is greater than zero, then F was created by chase using one of the ($\clubsuit$) rules. This required the fact P in the body of the rule to be created first. Now apply the induction hypothesis to P.

(ii) G can only be created in $\mathsf{chase}_{T_0}(D)$ from some fact F of relation End. Now apply Claim (i) to this F. ☐

Now recall that, in order to finish the proof, we only need to prove that there is a database-preserving Σ_Q-homomorphism from $\mathsf{chase}_{T_0}(D)$ to D. By Lemma 16, it is enough to show that for each fact $G = \mathsf{Encounter}(c, c')$ in $\mathsf{chase}_{T_0}(D)$, there exists a Σ_Q-homomorphism h from $T_{\mathsf{myth}}(\{G\})$ to D, such that $h(c) = c$ and $h(c') = c'$. But this easily follows from Point (ii) of Lemma 18, from the assumption that $\digamma$ does not stop (and hence no partially correct river is correct) and from Observation 1.

C Proofs for Section 5

For the proofs in this section, we need some knowledge about the structure of the chase of a database with a set of linear TGDs.

Let T be a set of linear TGDs and I an instance. With every fact $\alpha \in \mathsf{adom}(\mathsf{chase}_T(I))$, we associate a unique fact $\mathsf{src}(\alpha) \in I$ that α was 'derived from', as follows:

- if $\alpha \in I$, then $\mathsf{src}(\alpha) = \alpha$;
- if α was introduced by applying a TGD from T, mapping the body of T to a fact $\beta \in \mathsf{chase}_T(I)$, then $\mathsf{src}(\alpha) = \mathsf{src}(\beta)$.

We further associate, with every fact $\alpha \in \mathsf{adom}(I)$, the subinstance $\mathsf{chase}_T(I)|^{\downarrow}_{\alpha}$ of $\mathsf{chase}_T(I)$ that consists of all facts β with $\mathsf{src}(\beta) = \alpha$. One should think of $\mathsf{chase}_T(I)|^{\downarrow}_{\alpha}$ as the 'tree-like instance' that the chase of I with T generates 'below α'. The following lemma essentially says that the

[6] The set of all facts which can be produced from $P(\bar{x},\bar{y})$ by projections from T_{proj}.

shape of $\mathsf{chase}_T(I)|^{\downarrow}_{\alpha}$ only depends on α, but not on any other facts in I.

Lemma 19. *Let I be an instance, T a set of linear TGDs, and $\alpha \in I$. Then there is a homomorphism from $\mathsf{chase}_T(I)|^{\downarrow}_{\alpha}$ to $\mathsf{chase}_T(\{\alpha\})$ that is the identity on all constants in α.*

To prove Lemma 19, one considers a chase sequence $I_0, I_1, \dots$ a for I with T and shows by induction on i that for all $i \geq 0$, there is a homorphism h from $I_i|^{\downarrow}_{\alpha}$ to $\mathsf{chase}_T(\{\alpha\})$ that is the identity on all constants in α. This is done by replicating the application of the TGD that generated I_i from I_{i-1} in $\mathsf{chase}_T(\{\alpha\})$. The homomorphism obtained in the limit is as desired. Details are omitted.

Lemma 5. *Let T be a set of linear TGDs and Σ_D, Σ_Q schemas. Then T is Σ_D, Σ_Q-trivial iff $\mathsf{chase}_T(D) \rightarrow_{\Sigma_Q} D$ for all singleton Σ_D-databases D.*

Proof. Since hom triviality and CQ triviality are equivalent, we may choose to work with hom triviality. The 'only if' direction is immediate, so concentrate on the (contrapositive of the) 'if' direction.

Assume that T is not Σ_D, Σ_Q-trivial. Then there is a Σ_D-database D such that $\mathsf{chase}_T(D) \not\rightarrow_{\Sigma_Q} D$. If D is empty, then we are done. To establish Point 2 it suffices to show that otherwise, there is a fact $\alpha \in D$ such that $\mathsf{chase}_T(\{\alpha\}) \not\rightarrow_{\Sigma_Q} \{\alpha\}$.

Assume to the contrary that there is no such $\alpha \in D$. Then for every $\alpha \in D$, there is a Σ_Q-homomorphism h_α from $\mathsf{chase}_T(\{\alpha\})$ to $\{\alpha\}$ that is the identity on all constants in α. By Lemma 19, there is a database-preserving homomorphism g_α from $\mathsf{chase}_T(D)|^{\downarrow}_{\alpha}$ to $\mathsf{chase}_T(\{\alpha\})$. Then $h = \bigcup_{\alpha \in D} h_\alpha \circ g_\alpha$ is a database-preserving Σ_Q-homomorphism from $\mathsf{chase}_T(D)$ to D, in contradiction to $\mathsf{chase}_T(D) \not\rightarrow_{\Sigma_Q} D$. □

Lemma 6. *Let T be a set of linear TGDs and D a singleton database. Then $\mathsf{chase}_T(D) \not\rightarrow_{\Sigma_Q} D$ implies that there is a connected database $C \subseteq \mathsf{chase}_T(D)$ that contains at most two facts and such that $C \not\rightarrow_{\Sigma_Q} D$.*

Proof. Let $D = \{R(\bar{c})\}$. If $\mathsf{chase}_T(D)$ contains a fact $S(\bar{d})$ with $R \neq S \in \Sigma_Q$, then we may choose $C = \{S(\bar{d})\}$. Thus assume that the only relation symbol from Σ_Q that occurs in $\mathsf{chase}_T(D)$ is R. Assume that for every connected database $C \subseteq \mathsf{chase}_T(D)$ that contains at most two facts, $C \rightarrow_{\Sigma_Q} D$. We show that $\mathsf{chase}_T(D) \rightarrow_{\Sigma_Q} D$, that is, we have to construct a database-preserving Σ_Q-homomorphism h from I to D.

Since we can clearly ignore facts in I that use a relation symbol from outside of Σ_Q, we only need to consider facts that use the relation symbol R. For each fact $\alpha = R(\bar{d}) \in I$, we have $\{\alpha\} \rightarrow_{\Sigma_Q} D$ and thus find a database-preserving homomorphism h_α from $\{\alpha\}$ to D. We set $h = \bigcup_{\alpha \in I} h_\alpha$. To show that h is the desired database-preserving Σ_Q-homomorphism h from I to D, it suffices to show that h is a function, that is, if $\alpha = R(\bar{d}) \in I$, $\beta = R(\bar{e}) \in I$, and $\bar{d}$ and $\bar{e}$ share a constant c, then $h_\alpha(c) = h_\beta(c)$. We know that $\{\alpha, \beta\} \rightarrow_{\Sigma_Q} D$. Take a witnessing homomorphism $h_{\alpha,\beta}$. Since every R-fact homomorphically maps *in at most one way* into the single R-fact in D, the restriction of $h_{\alpha,\beta}$ to the variables in $\bar{d}$ is identical to h_α, and likewise for h_β and the variables in $\bar{e}$. Consequently, $h_\alpha(c) = h_\beta(c)$ as desired. □

D Proofs for Section 6: Model Theory

In this section, we provide the proofs of all model-theoretic results from Section 6, that is, Theorem 6, Theorem 7, Theorem 9, and Lemma 9. The complexity upper bounds in Theorems 5 and 10 follow from the automata constructions in the subsequent Section F. We start with giving some auxiliary results.

We first make explicit the structure of the chase for the case that T is a set of frontier-one TGDs, in terms of tree-like databases. Note that when a frontier-one TGD is applicable to a tuple $\bar{c}$, then $\bar{c}$ is in fact a single constant. With every $c \in \mathsf{adom}(\mathsf{chase}_T(I))$, we associate a unique constant $\mathsf{src}(c) \in \mathsf{adom}(I)$ that c was 'derived from', as follows:

- $\mathsf{src}(c) = c$ for all $c \in \mathsf{adom}(I)$;
- if c is a null that was introduced by applying a TGD from T at d then $\mathsf{src}(c) = \mathsf{src}(d)$.

We further associate, with every $c \in \mathsf{adom}(I)$, the subinstance $\mathsf{chase}_T(I)|^{\downarrow}_{a}$ of $\mathsf{chase}_T(I)$ that is the restriction of I to constants $\{d \in \mathsf{adom}(\mathsf{chase}_T(I)) \mid \mathsf{src}(d) = c\}$.

Lemma 20. *Let T be a set of frontier-one TGDs of head width ℓ. Then for every $c \in \mathsf{adom}(I)$, $\mathsf{chase}_T(I)|^{\downarrow}_{c}$ is a rooted tree-like instance of width at most ℓ.*

Informally, we can think of $\mathsf{chase}_T(I)$ as I with rooted tree-like instances of width at most ℓ attached to each constant. We next define the unraveling of a database D into a rooted tree-like instance U of width $k \geq 1$. A *k-sequence* takes the form

$$v = S_0, c_0, S_1, c_1, S_2, \dots, S_{n-1}, c_{n-1}, S_n,$$

where each $S_i \subseteq \mathsf{adom}(D)$ satisfies $|S_i| \leq k$ and $c_i \in S_i \cap S_{i+1}$ for $0 \leq i < n$. The empty k-sequence is denoted by ε. For every $c \in \mathsf{adom}(D)$, reserve a countably infinite set of fresh constants that we refer to as *copies* of c.

Now let (V, E) be the infinite directed tree with V the set of all k-sequences and E the prefix order on V. We choose a database $B(v)$ for every $v = S_0 \cdots S_n \in V$, proceeding by induction on n:

1. $B(\varepsilon) = \emptyset$;
2. if $v = S_0$, then $B(v)$ is obtained from $D|_{S_0}$ by replacing every constant c with a fresh copy of c;
3. if $v = S_0 c_0 \cdots c_{n-1} S_n$ with $n > 0$, then $B(v)$ is obtained from $D|_{S_0}$ by replacing
   - c_{n-1} with the copy of c_{n-1} used in $B(v')$ where $v' = S_0 \cdots S_{n-1}$ is the predecessor of v in (V, E);
   - every constant $c \neq c_{n-1}$with a fresh copy of c.

Set $\mathcal{T} = (V, E, B)$ and $U = I_{\mathcal{T}}$. It is easy to see that the 'uncopying' map is a homomorphism from U to D.

We next observe some properties of unraveled databases.

Lemma 21. *Let D be a database and U its k-unraveling, $k \geq 1$, and let T be a set of frontier-one TGDs with body width bounded by k. Then for every $c \in \mathsf{adom}(D)$ and copy c' of c in U, there is a homomorphism h from $\mathsf{chase}_T(D)|_c^\downarrow$ to $\mathsf{chase}_T(U)$ with $h(c) = c'$.*

Proof. Let D, U, k, and T be as in the lemma. Let $I_0, I_1, \dots$ be a chase sequence for D with T. The definition of src extends to the instances $I_0, I_1, \dots$ in an obvious way and thus it is also clear what we mean by $I_i|_c^\downarrow$, for $i \geq 0$ and $c \in \mathsf{adom}(D)$.

For all $i \geq 0$, $c \in \mathsf{adom}(D)$, and copies c' of c in U, we construct homomorphisms $h_{i,c,c'}$ from $I_i|_c^\downarrow$ to $\mathsf{chase}_T(U)$ with $h_{i,c,c'}(c) = c'$. Clearly, this suffices to prove the lemma because we obtain the desired homomorphism h in the limit.

The construction of the homomorphisms $h_{i,c,c'}$ proceeds by induction on i. The induction start is trivial as we may simply set $h_{i,c,c'}(c) = c'$ for every $c \in \mathsf{adom}(D)$ and copy c' of c in U. Now assume that I_{i+1} was obtained from I_i by applying a TGD $\vartheta = \phi(x, \bar{y}) \to \exists \bar{z}\, \psi(x, \bar{z})$ from T at some $d \in \mathsf{adom}(I_i)$. Let $\mathsf{src}(d) = c$. Then $I_{i+1}|_e^\downarrow = I_i|_e^\downarrow$ for all $e \in \mathsf{adom}(D) \setminus \{c\}$, and thus the only homomorphisms that we need to take care of are $h_{i+1,c,c'}$ with c' a copy of c in U. Take any such c'.

To apply ϑ at d, there must be a homomorphism g from ϕ to I_i with $g(x) = d$. Let $d' = h_{i,c,c'}(d)$. We argue that there is also a homomorphism g' from ϕ to $\mathsf{chase}_T(U)$ with $g'(x) = d'$. Let $S = (\mathsf{ran}(g) \cap \mathsf{adom}(D))$. By construction of U and since k is not smaller than then number of variables in ϕ, we find an $S' \subseteq \mathsf{adom}(U)$ and an isomorphism ι from $D|_S$ to $U|_{S'}$ such that $\iota(c) = c'$ if $c \in S$. Moreover, the non-reflexive[7] non-unary facts in $D|_S$ are identical to those in $I_i|_S$ because applying a frontier-one TGD can never add such facts. It follows that we can assemble the desired homomorphism g' from ι and the homomorphisms $h_{i,e,e'}$ with $e \in S$ and $\iota(e) = e'$.

We have just shown that ϑ is applicable in $\mathsf{chase}_T(U)$ at d' or there is (already) a homomorphism $\widehat{g}$ from ψ to $\mathsf{chase}_T(U)$ with $\widehat{g}(x) = d'$. In either case, we can extend $h_{i,c,c'}$ to the desired homomorphism $h_{i+1,c,c'}$ from $I_{i+1}|_c^\downarrow$ to $\mathsf{chase}_T(U)$ with $h_{i,c,c'}(c) = c'$ in an obvious way. □

Theorem 6. *Let T_1 and T_2 be sets of frontier-one TGDs, and Σ_D and Σ_Q schemas. Let k be the body width of T_1. Then the following are equivalent:*

1. *$T_1 \models^{hom}_{\Sigma_D,\Sigma_Q} T_2$;*
2. *$\mathsf{chase}_{T_2}(D) \to_{\Sigma_Q} \mathsf{chase}_{T_1}(D)$, for all tree-like Σ_D-databases D of width at most k.*

Proof. The "only if"-direction is immediate from the definition of hom-conservativity.

For "if", assume that $T_1 \not\models^{\text{hom}}_{\Sigma_D,\Sigma_Q} T_2$. Then there is a Σ_D-database D such that $\mathsf{chase}_{T_2}(D) \not\to_{\Sigma_Q} \mathsf{chase}_{T_1}(D)$. It suffices to show that $\mathsf{chase}_{T_2}(U) \not\to_{\Sigma_Q} \mathsf{chase}_{T_1}(U)$, U the unraveling of D of width k. We prove the contrapositive.

[7] A fact $R(c_1, \dots, c_n)$ is *reflexive* if $c_1 = \cdots = c_n$.

Thus assume that $\mathsf{chase}_{T_2}(U) \to_{\Sigma_Q} \mathsf{chase}_{T_1}(U)$. We have to show that $\mathsf{chase}_{T_2}(D) \to_{\Sigma_Q} \mathsf{chase}_{T_1}(D)$. We start with noting that, since T_2 is a set of frontier-one TGDs,

$$\mathsf{chase}_{T_2}(D) = D \cup \bigcup_{c \in \mathsf{adom}(D')} \mathsf{chase}_{T_2}(D)|_c^\downarrow.$$

As a consequence, it suffices to prove that $\mathsf{chase}_{T_2}(D)|_c^\downarrow \to_{\Sigma_Q} \mathsf{chase}_{T_1}(D)$ for all $c \in \mathsf{adom}(D)$.

Let $c \in \mathsf{adom}(D)$ and choose any copy c' of c in U. By Lemma 21, there is a homomorphism h from $\mathsf{chase}_{T_2}(D)|_c^\downarrow$ to $\mathsf{chase}_{T_2}(U)$ with $h(c) = c'$. Together with $\mathsf{chase}_{T_2}(U) \to_{\Sigma_Q} \mathsf{chase}_{T_1}(U)$, this implies that there is a Σ_Q-homomorphism h' from $\mathsf{chase}_{T_2}(D)|_c^\downarrow$ to $\mathsf{chase}_{T_1}(U)$ with $h'(c) = c'$. By construction of U, there is a homomorphism from U to D that maps c' to c. It is easy to extend this homomorphism to a homomorphism from $\mathsf{chase}_{T_1}(U)$ to $\mathsf{chase}_{T_1}(D)$ by following the application of chase rules. Thus $\mathsf{chase}_{T_2}(D)|_c^\downarrow \to_{\Sigma_Q} \mathsf{chase}_{T_1}(D)$, as desired.

It remains to show that there is a finite $U' \subseteq U$ with $\mathsf{chase}_{T_2}(U') \not\to_{\Sigma_Q} \mathsf{chase}_{T_1}(U')$. Since the TGDs in T_2 are frontier-one TGDs, an easy analysis of the chase procedure shows that $\mathsf{chase}_{T_2}(U) \not\to_{\Sigma_Q} \mathsf{chase}_{T_1}(U)$ implies that, for some $c \in \mathsf{adom}(U)$, $\mathsf{chase}_{T_2}(U)|_c^\downarrow \not\to_{\Sigma_Q} \mathsf{chase}_{T_1}(U)$. It follows from Lemma 7 that $\mathsf{chase}_{T_2}(U')|_c^\downarrow \not\to_{\Sigma_Q} \mathsf{chase}_{T_1}(U)$ for any $U' \subseteq U$ with $\mathsf{tp}_{T_2}(U, c) = \mathsf{tp}_{T_2}(U', c)$. By compactness of first-order logic, there is a finite such U', as required. □

Theorem 7. *Let T_1 and T_2 be sets of frontier-one TGDs, and Σ_D and Σ_Q schemas. Let k be the body width of T_1. Then the following are equivalent:*

1. *$T_1 \models^{CQ}_{\Sigma_D,\Sigma_Q} T_2$;*
2. *$q_{T_2}(D) \subseteq q_{T_1}(D)$, for all tree-like Σ_D-databases D of width at most k and connected Σ_Q-CQs q of arity 0 or 1.*

Proof. The "only if"-direction is immediate from the definition of CQ-conservativity.

For "if", assume that $T_1 \not\models^{\text{CQ}}_{\Sigma_D,\Sigma_Q} T_2$. Then there is a Σ_D-database D and a Σ_Q-CQ $q(\bar{x})$ such that $q_{T_2}(D) \not\subseteq q_{T_1}(D)$. We first manipulate q so that it has arity 0 or 1.

We may assume w.l.o.g. that q is connected because if it is not, then $p_{T_2}(D) \not\subseteq p_{T_1}(D)$ for some maximal connected component p of q and we can replace q by p. Choose some $\bar{c} \in q_{T_2}(D) \setminus q_{T_1}(D)$ and let h be a homomorphism from q to $\mathsf{chase}_{T_2}(D)$ such that $h(\bar{x}) = \bar{c}$. Let $q'(\bar{x}')$ be obtained from $q(\bar{x})$ in the following way:

- identify all variables $x_1, x_2 \in \mathsf{var}(q)$ in q such that $h(x_1) = h(x_2)$;
- if $h(y) \in \mathsf{adom}(D)$ for some quantified variable y in q, then make y an answer variable.

It is easy to see that $q'_{T_2}(D) \not\subseteq q'_{T_1}(D)$ and in fact $\bar{c}' := h(\bar{x}') \in q'_{T_2}(D) \setminus q'_{T_1}(D)$. Also note that h is an injective homomorphism from q' to $\mathsf{chase}_{T_2}(D)$.

Let $C = \{c \in \mathsf{adom}(D) \mid \exists x \in \mathsf{var}(q') : \mathsf{src}(h(x)) = c\}$. For $c \in C$, let $q^c(\bar{x}^c)$ denote the restriction of q to those variables x such that $\mathsf{src}(h(x)) = c$. The arity of each q^c is 0 or 1 because h maps all answers variables in q^c to c and thus all such answer variables have been identified during the construction of q'. By the following claim, we thus obtain a CQ q of the required form by choosing one of the queries q^c.

Claim. There is a $c \in C$ such that $q^c_{T_2}(D) \not\subseteq q^c_{T_1}(D)$.

To prove the claim, assume to the contrary that $q^c_{T_2}(D) \subseteq q^c_{T_1}(D)$ for all $c \in C$. Let $c \in C$. Since h is a homomorphism from q^c to $\mathsf{chase}_{T_2}(D)$, $h(\bar{x}^c) \in q^c_{T_2}(D)$ and thus $h(\bar{x}^c) \in q^c_{T_1}(D)$. Consequently, there is a homomorphism h_c from q^c to $\mathsf{chase}_{T_1}(D)$ with $h_c(\bar{x}^c) = h(\bar{x}^c)$. Set $h' = \bigcup_{c \in C} h_c$ and note that h' is functional since the queries q^c do not share any variables (this is because h is an injective homomorphism from q' to $\mathsf{chase}_{T_2}(D)|^{\downarrow}_c$ and by construction of the queries q^c). By construction of h', we have $h'(\bar{x}) = h(\bar{x}) = \bar{c}$. It thus remains to argue that h' is a homomorphism from q' to $\mathsf{chase}_{T_1}(D)$ as this contradicts $\bar{c} \notin q_{T_1}(D)$. Let $R(\bar{z})$ be an atom in q'. First assume that there is a $c \in C$ such that $\mathsf{src}(h(z)) = c$ for all $z \in \bar{z}$. Then $h'(\bar{z}) = h_c(\bar{z})$ and thus $R(h(\bar{z})) \in \mathsf{chase}_{T_1}(D)$ by definition of h'. Now assume that there are z_1, z_2 in $\bar{z}$ with $\mathsf{src}(h(z_1)) \neq \mathsf{src}(h(z_2))$. Since the TGDs in T are frontier-one, an easy analysis of the chase shows that this implies $R(h(\bar{z})) \in D$, that is, the fact $R(h(\bar{z}))$ was in the original database as no such fact is ever added by the chase. Thus $h(z) \in \mathsf{adom}(D)$ for all variables z in $\bar{z}$. By construction of q', it follows that $\bar{z}$ consists only of answer variables. This implies $h'(\bar{z}) = h(\bar{z})$ by definition of h', and thus $R(h'(\bar{z})) \in D \subseteq \mathsf{chase}_{T_1}(D)$.

At this point, we know that q is connected and of arity 0 or 1. We next argue that the database D can be replaced by its k-unraveling U. We concentrate on the case that q is unary. The Boolean case is very similar. We have to show that there is some $c' \in \mathsf{adom}(U)$ such that $c' \in q_{T_2}(U)$, but $c' \notin q_{T_1}(U)$.

By choice of q, there is a $c \in \mathsf{adom}(D)$ and a homomorphism h from $q(x)$ to $\mathsf{chase}_{T_2}(D)|^{\downarrow}_c$ such that $h(x) = c$. Choose any copy c' of c in U. By Lemma 21, there is a homomorphism g from $\mathsf{chase}_{T_2}(D)|^{\downarrow}_c$ to $\mathsf{chase}_{T_2}(U)$ with $g(c) = c'$. Composing h and g, we obtain a homomorphism h' from $q(x)$ to $\mathsf{chase}_{T_2}(U)$ with $h'(x) = c'$ and thus $c' \in q_{T_2}(U)$. It remains to show that $c' \notin q_{T_1}(U)$. But this follows from the facts that $c \notin q_{T_1}(D)$ and that there is a homomorphism from U to D that maps c' to c, which easily extends to a homomorphism from $\mathsf{chase}_{T_1}(U)$ to $\mathsf{chase}_{T_1}(D)$.

We may now finish the proof by argueing that for some finite $U' \subseteq U$, we have $q_{T_2}(U') \not\subseteq q_{T_1}(U')$. This, however, is a direct consequence of the compactness of first-order logic. □

For $c \in \mathsf{adom}(I)$ and $i \geq 0$, we use $I|^c_i$ to denote the restriction of I to the constants that are reachable from c in the Gaifman graph of I on a path of length at most i. Note that when we chase a finite database with a set of frontier-one TGDs, then the resulting instance has finite degree. This fails when frontier-one TGDs are replaced with guarded TGDs.

Lemma 22. *Let I_1, I_2 be instances of finite degree with I_1 Σ-connected, for a schema Σ. If there are $a_0 \in \mathsf{adom}(I_1)$ and $b_0 \in \mathsf{adom}(I_2)$ such that for each $i \geq 0$ there is a database-preserving Σ-homomorphism h_i from $I_1|^{a_0}_i$ to I_2 with $h_i(a_0) = b_0$, then $I_1 \rightarrow_\Sigma I_2$.*

Proof. We are going to construct a database-preserving Σ-homomorphism h from I_1 to I_2 step by step, obtaining in the limit a homomorphism that shows $I_1 \rightarrow_\Sigma I_2$. We will take care that, at all times, the domain of h is finite and

(∗) there is a sequence $h_0, h_1, \ldots$ with h_i a database-preserving Σ-homomorphism from $I_1|^{a_0}_i$ to I_2 such that whenever $h(c)$ is already defined, then $h_i(c) = h(c)$ for all $i \geq 0$.

Start with setting $h(a_0) = b_0$. The original sequence of homomorphisms $h_0, h_1, \ldots$ from the lemma witnesses (∗). Now consider the set Λ that consists of all constants $c \in \mathsf{adom}(I_1)$ with $h(c)$ is undefined and such that there is a $d \in \mathsf{adom}(I_1)$ with $h(d)$ defined and that co-occurs with c in some Σ-fact in I_1. Since the domain of h is finite and I_1 has finite degree, Λ is finite. By (∗) and since I_2 has finite degree, for each $c \in \Lambda$, there are only finitely many c' such that $h_i(c) = c'$ for some i. Thus, there must be a function $\delta : \Lambda \rightarrow \mathsf{adom}(I_2)$ such that, for infinitely many i, we have $h_i(c) = \delta(c)$ for all $c \in \Lambda$. Extend h accordingly, that is, set $h(c) = \delta(c)$ for all $c \in \Lambda$. Clearly, the sequence $h_0, h_1, \ldots$ from (∗) before the extension is no longer sufficient to witness (∗) after the extension. We fix this by skipping homomorphisms that do not respect δ, that is, define a new sequence $h'_0, h'_1, \ldots$ by using as h'_i the restriction of h_j to the domain of $I_1|^{a_0}_i$ where $j \geq i$ is smallest such that $h_j(c) = \delta(d)$ for all $c \in \Lambda$. This finishes the construction. The lemma follows from the fact that, due to the Σ-connectedness of I_1, every element is eventually reached. Note that h is database-preserving since all the homomorphisms in the original sequence $h_0, h_1, \ldots$ are. □

Theorem 9. *Let T_1 and T_2 be sets of frontier-one TGDs, Σ_D and Σ_Q schemas, and k the body width of T_1. Then $T_1 \models^{CQ}_{\Sigma_D,\Sigma_Q} T_2$ iff for all tree-like Σ_D-databases D of width at most k, the following holds:*

1. $\mathsf{chase}_{T_2}(D)|^{\mathsf{con}}_{\Sigma_Q} \rightarrow_{\Sigma_Q} \mathsf{chase}_{T_1}(D)$*;*
2. *for all maximally Σ_Q-connected components I of* $\mathsf{chase}_{T_2}(D) \setminus \mathsf{chase}_{T_2}(D)|^{\mathsf{con}}_{\Sigma_Q}$*, one of the following holds:*
   (a) $I \rightarrow_{\Sigma_Q} \mathsf{chase}_{T_1}(D)$*;*
   (b) $I \rightarrow^{\lim}_{\Sigma_Q} \mathsf{chase}_{T_1}(D)|^{\downarrow}_c$ *for some* $c \in \mathsf{adom}(D)$.

Proof. "⇐". Assume that $T_1 \not\models^{\mathsf{CQ}}_{\Sigma_D,\Sigma_Q} T_2$. By Theorem 7 there is a tree-like Σ_D-databases D of width at most k and a connected Σ_Q-CQ q of arity 0 or 1 such that $q_{T_2}(D) \not\subseteq q_{T_1}(D)$.

First assume that $q(x)$ is of arity 1. Then there is a constant $c \in \mathsf{adom}(D)$ such that $a \in q_{T_2}(D) \setminus q_{T_1}(D)$. Take a homomorphism h from q to $\mathsf{chase}_{T_2}(D)$ such that $h(x) = a$. Since q is connected and uses only symbols from Σ_Q, h is actually a homomorphism from q to $\mathsf{chase}_{T_2}(D)|^{\mathsf{con}}_{\Sigma_Q}$.

We show that $\mathsf{chase}_{T_2}(D)|^{\mathsf{con}}_{\Sigma_Q} \not\to_{\Sigma_Q} \mathsf{chase}_{T_1}(D)$ and thus Point 1 of Theorem 9 is violated. Assume to the contrary that $\mathsf{chase}_{T_2}(D)|^{\mathsf{con}}_{\Sigma_Q} \to_{\Sigma_Q} \mathsf{chase}_{T_1}(D)$ and take a witnessing Σ_Q-homomorphism g. Then $g \circ h$ is a homomorphism from q to $\mathsf{chase}_{T_1}(D)$ that maps x to a, implying $a \in q_{T_1}(D)$ and thus a contradiction.

Now assume that $q(\emptyset)$ is of arity 0 and take a homomorphism from q to $\mathsf{chase}_{T_2}(D)$. If the range of h falls within $\mathsf{chase}_{T_2}(D)|^{\mathsf{con}}_{\Sigma_Q}$, then we can argue as above that Point 1 of Theorem 9 is violated. Thus assume that this is not the case. Then the connectedness of q implies that the range of h does not overlap with $\mathsf{chase}_{T_2}(D)|^{\mathsf{con}}_{\Sigma_Q}$. Moreover, there must be a maximally Σ_Q-connected component I of $\mathsf{chase}_{T_2}(D) \setminus \mathsf{chase}_{T_2}(D)|^{\mathsf{con}}_{\Sigma_Q}$ such that the range of h falls within I. We may again argue as above to show that $I \not\to^n_{\Sigma_Q} \mathsf{chase}_{T_1}(D)$, with n the number of variables in q, as otherwise we find a homomorphism from q to $\mathsf{chase}_{T_1}(D)$. This implies $I \not\to_{\Sigma_Q} \mathsf{chase}_{T_1}(D)$ and $I \not\to^{\mathsf{lim}}_{\Sigma_Q} \mathsf{chase}_{T_1}(D,c)|^{\downarrow}_c$ for all $c \in \mathsf{adom}(D)$. Thus Point 2 of Theorem 9 is violated.

"⇒". Assume that $T_1 \models^{\mathsf{CQ}}_{\Sigma_D,\Sigma_Q} T_2$ and let D be a tree-like Σ_D-database of width at most k. We have to show that Points 1 and 2 of Theorem 9 hold.

We start with Point 1. By Theorem 8, $T_1 \models^{\mathsf{CQ}}_{\Sigma_D,\Sigma_Q} T_2$ implies $\mathsf{chase}_{T_2}(D) \to^{\mathsf{lim}}_{\Sigma_Q} \mathsf{chase}_{T_1}(D)$. Let $I_1,\dots,I_k$ be the maximally connected components of $\mathsf{chase}_{T_2}(D)|^{\mathsf{con}}_{\Sigma_Q}$. It suffices to show that $I_i \to_{\Sigma_Q} \mathsf{chase}_{T_1}(D)$ for $1 \leq i \leq k$. Fix such an i. By definition of $\mathsf{chase}_{T_2}(D)|^{\mathsf{con}}_{\Sigma_Q}$, I_i must contain some constant $c \in \mathsf{adom}(D)$. Since $\mathsf{chase}_{T_2}(D) \to^{\mathsf{lim}}_{\Sigma_Q} \mathsf{chase}_{T_1}(D)$, we find a sequence $h_0, h_1, \dots$ where h_ℓ is a database-preserving Σ_Q-homomorphism from $I_i|^c_\ell$ to $\mathsf{chase}_{T_1}(D)$. In particular, $h_\ell(c) = c$ for all ℓ. Thus, Lemma 22 yields $I_i \to_{\Sigma_Q} \mathsf{chase}_{T_1}(D)$. In summary, as required we obtain $\mathsf{chase}_{T_2}(D)|^{\mathsf{con}}_{\Sigma_Q} \to_{\Sigma_Q} \mathsf{chase}_{T_1}(D)$.

Now for Point 2. Let I be a maximally Σ_Q-connected component of $\mathsf{chase}_{T_2}(D) \setminus \mathsf{chase}_{T_2}(D)|^{\mathsf{con}}_{\Sigma_Q}$. Theorem 8 yields $I \to^{\mathsf{lim}}_{\Sigma_Q} \mathsf{chase}_{T_1}(D)$. Consequently, we find a sequence $h_0, h_1, \dots$ where h_i is a Σ_Q-homomorphism from $I|^{c_0}_i$ to $\mathsf{chase}_{T_1}(D)$, for some $c_0 \in \mathsf{adom}(I)$. We distinguish two cases.

First assume that there is a $d_0 \in \mathsf{adom}(\mathsf{chase}_{T_1}(D))$ such that $h_i(c_0) = d_0$ for infinitely many i. Construct a new sequence $h'_0, h'_1, \dots$ with h'_i a Σ_Q-homomorphism from $I|^{c_0}_i$ to $\mathsf{chase}_{T_1}(D)$ by skipping homomorphisms that do not map c_0 to d_0, that is, h'_i is the restriction of h_j to the domain of $I|^{c_0}_i$ where $j \geq i$ is smallest such that $h_j(c_0) = d_0$. Lemma 22 yields $I \to_{\Sigma_Q} \mathsf{chase}_{T_1}(D)$ and thus Point 2a is satisfied.

Otherwise, there is no $d_0 \in \mathsf{adom}(\mathsf{chase}_{T_1}(D))$ such that $h_i(c_0) = d_0$ for infinitely many i. We can assume that there is an $a_0 \in \mathsf{adom}(D)$ such that $\mathsf{src}(h_i(c_0)) = a_0$ for all i; in fact, there must be an a_0 such that $\mathsf{src}(h_i(c_0)) = a_0$ for infinitely many i and we can again skip homomorphisms to achieve this for all i. For brevity, let $J = \mathsf{chase}_{T_1}(D)|^{\downarrow}_{a_0}$. By Lemma 20, J is tree-like of width ℓ, where ℓ is the head width of T_1. Thus, there is a rooted instance tree $\mathcal{T} = (V, E, B)$ of width k that is finitely branching and satisfies $I_{\mathcal{T}} = J$. Since there is no $d_0 \in \mathsf{adom}(\mathsf{chase}_{T_1}(D))$ such that $h_i(c_0) = d_0$ for infinitely many i, it follows that for all $i, n \geq 0$ we must find a $j \geq i$ such that $h_j(c_0)$ is a domain element whose distance from a_0 in the Gaifman graph of J exceeds n. Based on this observation, we construct a sequence of homomorphisms $h'_0, h'_1, \dots$ as follows. For all $i \geq 0$, let h'_i be the restriction of h'_j to the domain of $I|^{c_0}_i$ where $j \geq i$ is smallest such that the distance of $h_j(c_0)$ from a_0 exceeds i. Note that each h'_i is a Σ_Q-homomorphism from $I|^{c_0}_i$ to J. Since I is connected, it is not hard to verify that this implies $I \to^{\mathsf{lim}}_{\Sigma_Q} J$. Thus Point 2b is satisfied. □

The proof of the following lemma is somewhat technical. We recommend to read it with Example 1 in mind.

Lemma 9. *Let I be a countable Σ-connected instance such that $\mathsf{adom}(I)$ contains only nulls. Then $I \to^{\mathsf{lim}}_{\Sigma} \mathsf{chase}_{T_1}(\widehat{t})$ iff there is an $\widehat{I} \in \mathcal{R}(T_1, \widehat{t})$ with $I \to \widehat{I}$.*

Proof. (⇐) Assume that $I \to_\Sigma \widehat{I}$ for some $\widehat{I} \in \mathcal{R}(T_1, \widehat{t})$, that is, $I \to_\Sigma I_{\mathcal{T}}$ for some $\widehat{t}$-proper T_1-labeled instance tree $\mathcal{T} = (V, E, B, \mu)$. To show that $I \to^{\mathsf{lim}}_{\Sigma} \mathsf{chase}_{T_1}(\widehat{t})$, it clearly suffices to prove that $I_{\mathcal{T}} \to^{\mathsf{lim}} \mathsf{chase}_{T_1}(\widehat{t})$.

Let $n \geq 1$ and I' an induced subinstance of $I_{\mathcal{T}}$ with $|\mathsf{adom}(I')| \leq n$. We have to show that $I' \to \mathsf{chase}_{T_1}(\widehat{t})$. Let V' be the minimal subset of V such that $v \in V'$ whenever $\mathsf{adom}(B(v)) \cap \mathsf{adom}(I') \neq \emptyset$ and for $E' = E \cap (V' \times V')$, the graph (V', E') is connected (and thus a tree). Let $\mathcal{T}' = (V', E', B')$ with B' the restriction of B to V'. It is enough to prove that $I_{\mathcal{T}'} \to \mathsf{chase}_{T_1}(\widehat{t})$.

We start to construct a homomorphism h from $I_{\mathcal{T}'}$ to $\mathsf{chase}_{T_1}(\widehat{t})$ as follows. Let v be the root of (V', E') or a non-root such that $\mathsf{adom}(B(v)) \cap \mathsf{adom}(B(v')) = \emptyset$, v' the predecessor of v. We know from Condition 1 of properness that $B(v)$ has the form $\{\mathsf{true}(c_0)\}$ and $\mu(c_0) = \widehat{t}$ or there is a TGD ϑ in T_1 such that $B(v)$ is isomorphic to the head of ϑ and $\widehat{t}, T_1 \models q_{(B(v),\mu_v)}$. In both cases, we find a homomorphism h_v from $D_{(B(v),\mu_v)}$ to $\mathsf{chase}_{T_1}(\widehat{t})$. The initial h is the union of all the homomorphisms h_v.

We now extend h in a step-wise fashion. Let $(v, v') \in E'$ such that h already covers $\mathsf{adom}(B'(v))$, but not $\mathsf{adom}(B'(v'))$. Then h is a homomorphism from $D_{(B(v),\mu_v)}$ to $\mathsf{chase}_{T_1}(\widehat{t})$. Since h does not yet cover $\mathsf{adom}(B'(v'))$, $\mathsf{adom}(B(v)) \cap \mathsf{adom}(B(v')) \neq \emptyset$. By Condition 2 of properness and because $\mathsf{chase}_{T_1}(\widehat{t})$ is a model of T_1, we can extend h to $\mathsf{adom}(D_{(B(v'),\mu_{v'})})$.

(⇒) Let I be a countable Σ-connected instance such that $I \to^{\mathsf{lim}}_{\Sigma} \mathsf{chase}_{T_1}(\widehat{t})$, and let $\alpha_0, \alpha_1, \dots$ be a (finite or infinite) enumeration of the non-unary facts in I (we assume that there is at least one such fact). Consider the (finite or infinite) sequence of instances

$$I_0 \subseteq I_1 \subseteq \cdots$$

with I_i the restriction of I to the constants in $\{\alpha_0, \dots, \alpha_i\}$. Since I is Σ-connected, we may clearly choose $\alpha_0, \alpha_1, \dots$ so that I_i is Σ-connected for all $i \geq 0$. Once more since I

is Σ-connected (and thus there are no isolated unary facts), $\bigcup_{i \geq 0} I_i = I$.

Since $I \rightarrow^{\mathsf{lim}}_{\Sigma} \mathsf{chase}_{T_1}(\widehat{t})$ there is a sequence

$$h_0, h_1, \ldots$$

with h_i a homomorphism from I_i to $\mathsf{chase}_{T_1}(\widehat{t})$ for all $i \geq 0$. We have to identify a $\widehat{t}$-proper T_1-labeled instance tree $\widehat{\mathcal{T}}$ with $I \rightarrow I_{\widehat{\mathcal{T}}}$. We do this by identifying a sequence

$$\mathcal{T}_0, \mathcal{T}_1, \ldots \text{ with } \mathcal{T}_i = (V_i, E_i, B_i, \mu_i)$$

of finite $\widehat{t}$-proper T_1-labeled instance trees that are monotonically growing in the sense that for all $i \geq 0$, $V_i \subseteq V_{i+1}$, $E_i \subseteq E_{i+1}$, and $B_i(v) = B_{i+1}(v)$ as well as $\mu_i(v) = \mu_{i+1}(v)$ for all $v \in V_i$. The desired instance tree $\widehat{\mathcal{T}}$ is then obtained in the limit. In particular, each $\mathcal{T}_i$ is constructed such that $I_i \rightarrow I_{\mathcal{T}_i}$ and along with the construction of the trees $\mathcal{T}_i$ we construct a sequence of homomorphisms

$$g_0, g_1, \ldots$$

witnessing this. Also this sequence is monotonically growing in the sense that $g_0 \subseteq g_1 \subseteq \cdots$ and we obtain the desired homomorphism g from I to $I_{\widehat{\mathcal{T}}}$ in the limit.

As a 'guide' for the construction of the two sequences $\mathcal{T}_0, \mathcal{T}_1, \ldots$ and $g_0, g_1, \ldots$, we use the homomorphisms h_i from I_i to $\mathsf{chase}_{T_1}(\widehat{t})$. During the process, we also uniformize the sequence $h_0, h_1, \ldots$ be removing 'unsuitable' homomorphisms from it, similar to what has been done in the proof of Lemma 22. For the construction, we make more precise the synchronization between the sequence $\mathcal{T}_0, \mathcal{T}_1, \ldots$ and the sequence $h_0, h_1, \ldots$. As described after the definition of T_1-labeled instance trees, the construction of $\mathsf{chase}_{T_1}(\widehat{t})$ gives rise to a ($\widehat{t}$-proper) T_1-labeled instance tree $\mathcal{T} = (V, E, B, \mu)$ such that $I_{\mathcal{T}} = \mathsf{chase}_{T_1}(\widehat{t})$. Moreover, the width of $\mathcal{T}$ is bounded by the head width of TGDs in T_1. An *embedding* of a T_1-labeled instance tree $\mathcal{T}_i$ into $\mathcal{T}$ is a pair of mappings $f : V_i \rightarrow V, \iota : \mathsf{adom}(I_{\mathcal{T}_i}) \rightarrow \mathsf{adom}(I_{\mathcal{T}})$ such that f is an injective homomorphism from (V_i, E_i) to (V, E) and ι satisfies the following conditions:

1. for every $v \in V_i$, the restriction of ι to $\mathsf{adom}(B_i(v))$ is an isomorphism from $B_i(v)$ to $B(f(v))$;
2. for every $c \in \mathsf{adom}(I_{\mathcal{T}_i})$, we have $\mu_i(c) = \mu(\iota(c))$.

We remark for further use that actually

3. ι is an isomorphism from $I_{\mathcal{T}_i}$ to $\bigcup_{u \in \mathsf{ran}} B(u)$.

This is easy to show using the injectivity of f and the definition of instance trees.

Now, along with the sequences $\mathcal{T}_0, \mathcal{T}_1, \ldots$ and $g_0, g_1, \ldots$, we also construct embeddings

$$f_{i,j}, \iota_{i,j} \text{ with } 0 \leq i \leq j$$

where each $f_{i,j}, \iota_{i,j}$ is an embedding of $\mathcal{T}_i$ into $\mathcal{T}$. Note that there are infinitely many embeddings $f_{i,j}, \iota_{i,j}$ for each i instead of only a single one. The reason is that these embeddings achieve a synchronization of each $\mathcal{T}_i$ with the entire sequence $h_i, h_{i+1}, \ldots$ in the sense that, for $0 \leq i \leq j$, we shall take care that

(†) $h_j(c) = \iota_{i,j} \circ g_i(c)$ for all $c \in \mathsf{adom}(I_i)$.

Informally, (†) states that all homomorphisms h_j, $j \geq i$, map I_i into $\mathsf{chase}_{T_1}(\widehat{t})$ in the same way as g_i maps I_i into $\mathcal{T}_i$.

Now for the actual construction. To define $\mathcal{T}_0$ and g_0, choose for every $k \geq 0$ a node $u_k \in V$ from $\mathcal{T}$ such that $h_k(I_0) \subseteq B(u_k)$. Such a u_k must exist since all constants in I_0 co-occur in a single fact in I_0. Consider the sequence

$$(B_0, \lambda_0, \bar{d}_0), (B_1, \lambda_1, \bar{d}_1), \ldots$$

with $(B_i, \lambda_i, \bar{d}_i) = (B(u_k), \mu(u_k), h_k(\bar{c}))$. Since the width of $\mathcal{T}$ is bounded, there are only finitely many isomorphism types of these triples, where $(B_i, \lambda_i, \bar{d}_i)$ and $(B_j, \lambda_j, \bar{d}_k)$ are isomorphic if there is an isomorphism ι from B_i to B_j with $\iota(\bar{d}_i) = \bar{d}_j$ and $\lambda_i(c) = \lambda_j \circ \iota(c)$ for all $c \in \mathsf{adom}(B_i)$. Thus we may choose an isomorphism type that occurs infinitely often. We skip all homomorphisms h_i such that $(B_i, \lambda_i, \bar{d}_i)$ is not of that type, that is, we replace each h_i with h_j where $j \geq i$ is minimal such that $(B_j, \lambda_j, \bar{d}_j)$ is of the chosen isomorphism type. Now define $\mathcal{T}_0$ by taking

$$V_0 = \{v_0\}, \quad E_0 = \emptyset, \quad B_0(v_0) = B_0, \quad \mu_0 = \lambda_0,$$

and set $g_0(c) = h_0(c)$ for all constants $c \in \mathsf{adom}(I_0)$ and $f_{0,j}(v_0) = u_k$ for all $j \geq 0$. As $\iota_{0,j}$, we use the isomorphism that witness that $(B_0, \lambda_0, \bar{d}_0)$ and $(B_j, \lambda_j, \bar{d}_j)$ have the same isomorphism type. Based on this choice, it can be verified that (†) is satisfied.

For the inductive step, assume that we have already constructed $\mathcal{T}_i$, g_i, as well as the embeddings $f_{i,j}, \iota_{i,j}$ for all $j \geq i$. We obtain $\mathcal{T}_{i+1}, g_{i+1}$ from $\mathcal{T}_i, g_i$ by starting with $\mathcal{T}_{i+1} = \mathcal{T}_i$ and $g_{i+1} = g_i$ and then extending as follows.

By construction of I_{i+1}, there is a non-unary fact $R(\bar{c}) \in I_{i+1}$ such that I_{i+1} is the restriction of I to $\mathsf{adom}(I_i) \cup \bar{c}$. For every $k > i$, there is thus a $u_k \in V$ with $R(h_k(\bar{c})) \in B(u)$. In fact, each u_k is unique by definition of instance trees. By skipping homomorphisms from $h_{i+1}, h_{i+2}, \ldots$ along with the associated functions $f_{i,i+1}, f_{i,i+2}, \ldots$ and isomorphisms $\iota_{i,i+1}, \iota_{i,i+2}, \ldots$, we can achieve that one of the following two cases applies:

1. u_k is in the range of $f_{i,k}$ for all $k > i$, or
2. u_k is not in the range of $f_{i,k}$ for all $k > i$.

In Case 1, since V_i is finite we can once more skip homomorphisms and achieve that there is some $v \in V_i$ such that $f_{i,k}(v) = u_k$, for all $k > i$. For every $k > i$, by Property 3 of embeddings we may define

$$\bar{d}_k = \iota^-_{i,k}(h_k(\bar{c})).$$

The choice of v and Property 1 of embeddings yield $\bar{d}_k \subseteq \mathsf{adom}(B_i(v))$ for all $k > i$. Since $\mathsf{adom}(B_i(v))$ is finite, there are only finitely many possible choices for the $\bar{d}_k$. By skipping homomorphism, we may thus achieve that they are all identical. Extend g_{i+1} by setting $g_{i+1}(c) = \iota^-_{i,k}(h_k(c))$ for some (equivalently: all) $k > i$, for every $c \in \bar{c}$ such that $g_{i+1}(c)$ is not yet defined. Define $f_{i+1,j} = f_{i,j}$ and $\iota_{i+1,j} = \iota_{i,j}$ for all $j > i$. One may verify that (†) is satisfied.

We argue that, as required, g_{i+1} is a homomorphism from I_{i+1} to $I_{\mathcal{T}_{i+1}}$. Take any fact $S(\bar{e}) \in I_{i+1}$. The definition of

g_{i+1} and (†) yield $g_{i+1}(c) = \iota^-_{i,i+1}(h_{i+1}(c))$ for all $c \in \bar{e}$. It remains to note that h_{i+1} is a homomorphism from I_{i+1} to $\mathsf{chase}_{T_1}(\widehat{t})$, thus $S(h_{i+1}) \in \mathsf{chase}_{T_1}(\widehat{t})$, and $\iota^-_{i,i+1}$ is an isomorphism.

We consider now Case 2, that is, u_k is not in the range of $f_{i,k}$, for all $k > i$. Now, observe that since I_{i+1} is connected, there is some $d \in \bar{c} \cap \mathsf{adom}(I_i)$, and thus $g_i(d)$ is already defined. Choose $v \in V_i$ with $g_i(d) \in \mathsf{adom}(B(v))$ and consider the sequence

$$w_k = f_{i,k}(v), \quad k > i.$$

Since u_k is not in the range of $f_{i,k}$, we have $u_k \neq w_k$, for all k. However, $h_k(d) \in \mathsf{adom}(B(w_k)) \cap \mathsf{adom}(B(u_k))$. Due to Property 1 from the definition of instance trees, we can skip homomorphisms to reach one of the following situations:

(a) u_k is the predecessor of w_k, for all $k > i$,
(b) u_k is a successor of w_k, for all $k > i$, or
(c) u_k, w_k are siblings, for all $k > i$.

In all cases, we start as follows. Similarly to the induction start, consider the sequence

$$(B_{i+1}, \lambda_{i+1}, \bar{d}_{i+1}), (B_{i+2}, \lambda_{i+2}, \bar{d}_{i+2}), \ldots$$

with $(B_k, \lambda_k, \bar{d}_k) = (B(u_k), \mu(u_k), h_k(\bar{c}))$, for all $k > i$. Since the width of $\mathcal{T}$ is bounded, there are only finitely many isomorphism types of these triples. Thus we may choose an isomorphism type that occurs infinitely often. By skipping homomorphisms, we can achieve that all $(B_k, \lambda_k, \bar{d}_k)$ are of the same type. We further select a triple $(B, \lambda, \bar{d})$ that is of the same isomorphism type to be used as a bag in the tree $\mathcal{T}_{i+1}$. We make this choice such that $g_i(d) \in \mathsf{adom}(B)$ and $\mathsf{adom}(B) \setminus \{g_i(d)\}$ consists only of fresh constants, that is, constants not used in $I_{\mathcal{T}_i}$. Define τ_k to be an isomorphism that witnesses that $(B, \lambda, \bar{d})$ and $(B_k, \lambda_k, \bar{d}_k)$ have the same isomorphism type, for all $k > i$.

We now extend $\mathcal{T}_{i+1}, g_{i+1}$ distinguishing Cases (a)–(c).

In Case (a), let us first argue that v is the root of $\mathcal{T}_i$. If not, then $f_{i,k}$ maps the predecessor v' to u_k, for all $k > i$, in contradiction to the fact that u_k is not in the range of $f_{i,k}$. Then, add a predecessor v' of v to $\mathcal{T}_{i+1}$, set

$$B_{i+1}(v') = B, \quad \mu_{i+1} = \mu_{i+1} \cup \lambda,$$

and set, for all $c \in \mathsf{adom}(I_{i+1})$ such that $g_{i+1}(c)$ is not yet defined, $g_{i+1}(c) = \iota^-_{i+1,k}(h_k(c))$, for some (equivalently: all) $k > i$. It can be verified that setting, for all $j > i$,

$$f_{i+1,j} = f_{i,j} \cup \{(v', u_j)\}, \text{ and}$$
$$\iota_{i+1,j} = \iota_{i,j} \cup \tau_j$$

witnesses (†) for $i + 1$.

In Case (b), we do exactly the same as in Case (a) with v' a fresh *successor* of v (instead of predecessor).

In Case (c), we make a final case distinction. If v has a predecessor v_0 in $\mathcal{T}_i$, then proceed exactly as in Case (a), but make v' a fresh successor of v_0. Otherwise, let u'_k be the predecessor of u_k, for all $k > i$, and consider the sequence

$$(B'_{i+1}, \lambda'_{i+1}, d_{i+1}), (B'_{i+2}, \lambda'_{i+2}, d_{i+2}), \ldots$$

with $(B'_k, \lambda'_k, d_k) = (B(u'_k), \mu(u'_k), h_k(d))$, for all $k > i$ (recall that we fixed d in the beginning of Case 2). We can again skip homomorphisms and achieve that all the (B'_k, λ'_k, d_k) are of the same isomorphism type. We further select a triple (B', λ', d') that is of the same isomorphism type to be used as a bag in the tree $\mathcal{T}_{i+1}$. We make this choice such that $g_i(d) \in \mathsf{adom}(B')$ and $\mathsf{adom}(B') \setminus \{g_i(d)\}$ consists only of fresh constants, that is, constants not used in $I_{\mathcal{T}_i}$. Define τ'_k to be an isomorphism that witnesses that (B', λ', d') and (B'_k, λ'_k, d_k) have the same isomorphism type, for all $k > i$. Then, add a predecessor v' of v and a fresh successor v'' of v' to $\mathcal{T}_{i+1}$, set

$$B_{i+1}(v') = B', \quad B_{i+1}(v'') = B, \quad \mu_{i+1} = \mu_{i+1} \cup \lambda \cup \lambda',$$

and set, for all $c \in \mathsf{adom}(I_{i+1})$ such that $g_{i+1}(c)$ is not yet defined, $g_{i+1}(c) = \iota^-_{i+1,k}(h_k(c))$, for some (equivalently: all) $k > i$. It can be verified that setting, for all $j > i$,

$$f_{i+1,j} = f_{i,j} \cup \{(v', u'_j), (v'', u_j)\}, \text{ and}$$
$$\iota_{i+1,j} = \iota_{i,j} \cup \tau_j \cup \tau'_j$$

witnesses (†) for $i + 1$.

This finishes the construction of the sequences $\mathcal{T}_0, \mathcal{T}_1, \ldots$ and $g_0, g_1, \ldots$. Recall that both the $\mathcal{T}_i$ and the g_i are monotonically growing and that we are interested in the limits $\widehat{\mathcal{T}}$ and g of the sequences, that is,

$$\widehat{\mathcal{T}} = \left(\bigcup_{i \geq 0} V_i, \bigcup_{i \geq 0} E_i, \bigcup_{i \geq 0} B_i, \bigcup_{i \geq 0} \mu_i \right)$$

and

$$g = \bigcup_{i \geq 0} g_i.$$

Since each g_i is a homomorphism from I_i to $I_{\mathcal{T}_i}$, it is clear that g is a homomorphism from I to $I_{\widehat{\mathcal{T}}}$. Moreover, $\widehat{\mathcal{T}}$ is $\widehat{t}$-proper since each $\mathcal{T}_i$ is. □

E Proofs for Section 6: Decision Procedures

We prove the decidability results from Section 6 using the characterizations provided in that section and tree automata. More precisely, to prove the 3EXPTIME upper bounds for hom-conservativity and CQ-conservativity in Theorem 5, we show how to construct, given sets T_1, T_2 of frontier-one TGDs and signatures Σ_D and Σ_Q, a tree automaton $\mathfrak{A}$ such that $L(\mathfrak{A}) \neq \emptyset$ iff $T_1 \not\models^{\text{hom}}_{\Sigma_D,\Sigma_Q} T_2$ resp. $T_1 \not\models^{\text{CQ}}_{\Sigma_D,\Sigma_Q} T_2$. The use of tree automata is sanctioned by the characterizations of hom-conservativity and CQ-conservativity in terms of tree-shaped witnesses provided by Theorem 6 and Theorem 9.

We start with giving the necessary details on tree automata.

E.1 Tree Automata

A *tree* is a non-empty (and potentially infinite) set of words $W \subseteq (\mathbb{N} \setminus 0)^*$ closed under prefixes. We assume that trees are finitely branching, that is, for every $w \in W$, the set $\{i > 0 \mid w \cdot i \in W\}$ is finite. For $w \in (\mathbb{N} \setminus 0)^*$, set $w \cdot 0 := w$. For $w = n_0 n_1 \cdots n_k$, $k > 0$, set $w \cdot -1 := n_0 \cdots n_{k-1}$, and call w a *successor* of $w \cdot -1$ and $w \cdot -1$ a *predecessor* of w. For an alphabet Θ, a Θ-*labeled tree* is a pair (W, L) with W a tree and $L : W \rightarrow \Theta$ a node labeling function.

A *two-way alternating tree automaton (2ATA)* is a tuple $\mathfrak{A} = (Q, \Theta, q_0, \delta, \Theta)$ where Q is a finite set of *states*, Θ is the *input alphabet*, $q_0 \in Q$ is the *initial state*, δ is a *transition function*, and $\Theta : Q \rightarrow \mathbb{N}$ is a *priority function*. The transition function δ maps every state q and input letter $a \in \Theta$ to a positive Boolean formula $\delta(q, a)$ over the truth constants true and false and *transition atoms* of the form q, $\Diamond^- q$, $\Box^- q$, $\Diamond q$ and $\Box q$. A transition q expresses that a copy of $\mathfrak{A}$ is sent to the current node in state q; $\Diamond^- q$ means that a copy is sent in state q to the predecessor node, which is required to exist; $\Box^- q$ means the same except that the predecessor node is not required to exist; $\Diamond q$ means that a copy of q is sent to some successor and $\Box q$ means that a copy of q is sent to all successors. The semantics of 2ATA is given in terms of runs as usual.

Let (W, L) be a Θ-labeled tree and $\mathfrak{A} = (Q, \Theta, q_0, \delta, \Omega)$ a 2ATA. A *run of* $\mathfrak{A}$ *over* (W, L) is a $W \times Q$-labeled tree (W_r, r) such that $\varepsilon \in W_r$, $r(\varepsilon) = (\varepsilon, q_0)$, and for all $y \in W_r$ with $r(y) = (x, q)$ and $\delta(q, V(x)) = \theta$, there is an assignment v of truth values to the transition atoms in θ such that v satisfies θ and:

- if $v(q') = 1$, then $r(y') = (x, q')$ for some successor y' of y in W_r;
- if $v(\Diamond^- q') = 1$, then $x \neq \varepsilon$ and $r(y') = (x \cdot -1, q')$ for some successor y' of y in W_r;
- if $v(\Box^- q') = 1$, then $x = \varepsilon$ or $r(y') = (x \cdot -1, q')$ for some successor y' of y in W_r;
- if $v(\Diamond q') = 1$, then there is some j and a successor y' of y in W_r with $r(y') = (x \cdot j, q')$;
- if $v(\Box q') = 1$, then for all successors x' of x, there is a successor y' of y in W_r with $r(y') = (x', q')$.

Let $\gamma = i_0 i_1 \cdots$ be an infinite path in W_r and denote, for all $j \geq 0$, with q_j the state such that $r(i_j) = (x, q_j)$. The path γ is *accepting* if the largest number m such that $\Omega(q_j) = m$ for infinitely many j is even. A run (W_r, r) is accepting, if all infinite paths in W_r are accepting. $\mathfrak{A}$ accepts a tree if $\mathfrak{A}$ has an accepting run over it. We use $L(\mathfrak{A})$ to denote the set of Θ-labeled trees accepted by $\mathfrak{A}$.

It is not hard to show that 2ATA are closed under intersection and that the intersection automaton can be constructed in polynomial time, see for example (Comon et al. 2007). The *emptiness problem* for 2ATA means to decide, given a 2ATA $\mathfrak{A}$, whether $L(\mathfrak{A}) = \emptyset$. Emptiness of 2ATA can be solved in time single exponential in the number of states and the maximal priority, and polynomial in all other components. This was proved for 2ATAs on ranked trees in (Vardi 1998) and it was shown in (Jung et al. 2020) that the result carries over to the particular version of 2ATAs used here, which run on trees of arbitrary finite degree.

E.2 Upper Bound for Hom-Conservativity

To decide hom-conservativity via Theorem 6 it suffices to devise a 2ATA $\mathfrak{A}$ such that

$(*_{\mathfrak{A}})$ $\mathfrak{A}$ accepts all tree-like instances I of width $\max(k, \ell)$ that are models of T_1 and some tree-like Σ_Q-databases D of width k such that $\mathsf{chase}_{T_2}(D) \not\rightarrow_{\Sigma_Q} I$, where k and ℓ are the body and head width of T_1.

However, 2ATAs cannot run directly on tree-like databases or instances because the potential labels of the underlying trees (the bags) may use any number of constants and do not constitute a finite alphabet. We therefore use an appropriate encoding of tree-like databases that reuses constants so that we can make do with finitely many constants overall, similar to what has been done, for example, in (Grädel and Walukiewicz 1999).

Encoding of tree-like instances. Let $m = \max(k, \ell)$ with k the body width and ℓ the head width of T_1. Fix a set Δ of $2m$ constants and define Θ_0 to be the set of all Σ-databases B with $\mathsf{adom}(B) \subseteq \Delta$ and $|\mathsf{adom}(B)| \leq m$, where Σ is the union of Σ_D and $\mathsf{sig}(T_1)$, that is, all relation symbols that occur in T_1.

Let (W, L) be a Θ_0-labeled tree. For convenience, we use B_w to refer to the database $L(w)$ at node w. For a constant $c \in \Delta$, we say that $v, w \in W$ are *c-equivalent* if $c \in \mathsf{adom}(B_u)$ for all u on the unique shortest path from v to w. Informally, this means that c represents the same constant in B_v and in B_w. In case that $c \in \mathsf{adom}(B_w)$, we use $[w]_c$ to denote the set of all v that are c-equivalent to w. We call (W, L) *well-formed* if it satisfies the following counterparts of Conditions 1 and 2 of instance trees:

$1'$. for every $w \in W$ and every $c \in \mathsf{adom}(B_w)$, the restriction of W to $[w]_c$ is a tree of depth at most 1;

$2'$. for every $w \in W$ and successor v of w, $\mathsf{adom}(B_w) \cap \mathsf{adom}(B_v)$ contains at most one constant.

Each well-formed Θ_0-labeled tree (W, L) *represents* a Σ-instance tree $\mathcal{T}_{W,L} = (V, E, B)$ as follows. The underlying tree (V, E) is the tree (described by) W. The active domain of $I_{\mathcal{T}_{W,L}}$ is the set of all equivalence classes $[w]_c$ with $w \in W$ and $c \in \mathsf{adom}(B_w)$ and the labeling B is defined by taking

$$R([w]_{c_1}, \ldots, [w]_{c_k}) \in B(w) \quad \text{iff} \quad R(c_1, \ldots, c_k) \in B_w,$$

for all $w \in W$ and $c \in \mathsf{adom}(B_w)$. As a shorthand, we use $I_{W,L}$ to denote the instance $I_{\mathcal{T}_{W,L}}$.

Conversely, for every Σ_D-instance I such that $I = I_\mathcal{T}$ for a instance tree $\mathcal{T} = (V, E, B)$ of width m, we can find a Θ_0-labeled tree (W, L) that represents I in the sense that $I_{W,L}$ is isomorphic to I. Since Δ is of size $2m$, it is possible to select a mapping $\pi : \mathsf{adom}(D) \rightarrow \Delta$ such that for each $(v, w) \in E$ and each $d, e \in \mathsf{adom}(B(w)) \cup \mathsf{adom}(B(v))$, we have $\pi(d) = \pi(e)$ iff $d = e$. Define the Θ_0-labeled tree (W, L) by setting $W = (V, E)$, and for all $w \in W$, B_w to the image of $B(w)$ under π. Clearly, (W, L) satisfies the desired properties.

Automata Constructions We construct a 2ATA $\mathfrak{A}$ that satisfies $(*_{\mathfrak{A}})$, for given $T_1, T_2, \Sigma_D, \Sigma_Q$. We may assume without loss of generality that all symbols from Σ_D and Σ_Q occur in T_1. The desired 2ATA runs over Θ-labeled trees with $\Theta = \Theta_0 \times \Theta_0 \times \Theta_1$ where Θ_0 is defined as above, and

Θ_1 is the set of all mappings $\mu : \Delta' \to \mathsf{TP}(T_2)$ for some $\Delta' \subseteq \Delta$ with $|\Delta'| \leq m$. Intuitively, the first component will represent a Σ_D-database D, the second component will represent a model I of T_1 and D, and the last component will represent the T_2 chase of D, restricted to $\mathsf{adom}(D)$.

For a Θ-labeled tree (W, L), we set $L(w) = (L_0(w), L_1(w), L_2(w))$ for all $w \in W$ and thus may use $L_i(w)$ to refer to the i-th component of the label of w, for $i \in \{0, 1, 2\}$. For the sake of readability, we may use μ_w to denote $L_2(w)$. A Θ-labeled tree (W, L) is called *well-typed* if, for all $w \in W$:

1. the domain of μ_w is $\mathsf{adom}(L_0(w))$ and
2. for every successor v of w and every $c \in \mathsf{adom}(L_0(w)) \cap \mathsf{adom}(L_0(v))$, we have $\mu_w(c) = \mu_v(c)$.

The desired 2ATA $\mathfrak{A}$ is constructed as the intersection of the five 2ATAs $\mathfrak{A}_0, \mathfrak{A}_1, \mathfrak{A}_2, \mathfrak{A}_3, \mathfrak{A}_4$ provided by the following lemma.

Lemma 23. *There are 2ATAs $\mathfrak{A}_0, \mathfrak{A}_1, \mathfrak{A}_2, \mathfrak{A}_3, \mathfrak{A}_4$ such that:*

- *$\mathfrak{A}_0$ accepts (W, L) iff it is well-typed and (W, L_0) and (W, L_1) are well-formed;*
- *$\mathfrak{A}_1$ accepts (W, L) iff I_{W,L_0} is a Σ_D-database of width k;*
- *$\mathfrak{A}_2$ accepts (W, L) iff I_{W,L_1} is a model of I_{W,L_0} and T_1;*
- *$\mathfrak{A}_3$ accepts (W, L) iff for every $w \in W$ and every $c \in \mathsf{adom}(L_0(w))$,*

$$\mu_w(c) = \mathsf{tp}_{T_2}(\mathsf{chase}_{T_2}(I_{W,L_0}), [w]_c).$$

- *$\mathfrak{A}_4$ accepts (W, L) iff $\mathsf{chase}_{T_2}(I_{W,L_1}) \not\to I_{W,L_1}$.*

The number of states of

- *$\mathfrak{A}_0$ is exponential in $||T_1||$ (and independent of T_2);*
- *$\mathfrak{A}_1$ does not depend on the input;*
- *$\mathfrak{A}_2$ is exponential in $||T_1||$ (and independent of T_2);*
- *$\mathfrak{A}_3$ is exponential in $||T_2||$ (and independent of T_1);*
- *$\mathfrak{A}_4$ is double exponential in $||T_2||$ (and independent of T_1).*

All automata can be constructed in time triple exponential in $||T_1|| + ||T_2||$ and have maximum priority one.

It can be verified that $\mathcal{A}$ satisfies $(*_{\mathfrak{A}})$ and thus $L(\mathfrak{A}) \neq \emptyset$ iff $T_1 \not\models^{\text{hom}}_{\Sigma_D,\Sigma_Q} T_2$. The rest of this section is devoted to proving Lemma 23.

Automaton $\mathfrak{A}_0$. This automaton is straightforward to construct.

Automaton $\mathfrak{A}_1$. This automaton simply verifies that all databases $L_0(w)$ use only symbols from Σ_D and at most k constants, and that on every path there are only finitely many non-empty databases. Constantly many states suffice for this purpose.

Automaton $\mathfrak{A}_2$. First note that I_{W,L_1} is a model of I_{W,L_0} iff $L_0(w)$ is a subset of $L_1(w)$, for every $w \in W$. This check can easily be done by a 2ATA with constantly many states. In order to verify that I_{W,L_1} is a model of T_1, it is essential to realize that the employed encoding allows a 2ATA to do the following:

(†) given some $w \in W$ and $c \in \mathsf{adom}(L_1(w))$, and a unary CQ $q(x)$, verify that there is a homomorphism h from q to I_{W,L_1} with $h(x) = [w]_c$.

Since (parts of) 2ATAs can easily be complemented by dualization, they are also able to verify that there is no such homomorphism. The 2ATA $\mathfrak{A}_2$ may thus visit all $w \in W$ and all $c \in \mathsf{adom}(L_1(w))$ and verify that, for every TGD $\phi(x, \bar{y}) \to \exists \bar{z}\, \psi(x, \bar{z})$ in T_1, there is no homomorphism h from $q_\phi(x)$ to I_{W,L_1} with $h(x) = [w]_c$ or there is a homomorphism g from $q_\psi(x)$ to I_{W,L_1} with $g(x) = [w]_c$.

Informally, a 2ATA can achieve (†) by memorizing (in its state) a CQ p for which it still has to check the existence of a homomorphism, plus the target constant of the free variable of p (if any). If the automaton visits a given node $w \in W$ in such a state, it guesses the variables $y_1, \dots, y_n$ that the homomorphism will map to $\mathsf{adom}(L_1(w))$ and also the corresponding homomorphism targets $e_1, \dots, e_n \in \mathsf{adom}(L_1(w))$. It verifies that the guess indeed give rise to a partial homomorphism to database $L_1(w)$ and proceeds with the parts of p that have not been mapped to the current database $L_1(w)$.

To formalize this idea, we use *instantiated CQs* in which all answer variables are replaced with constants, writing $q(\bar{c})$ to indicate that $\bar{c}$ are precisely the constants that occur in q and that all variables are quantified. We will mostly drop the word 'instantiated' and only speak of CQs.

Let $q(\bar{c})$ be an (instantiated) CQ. A Δ-*splitting* of $q(\bar{c})$ is obtained by first replacing any number of variables in $q(\bar{c})$ with constants[8] from Δ and then partitioning the (atoms of the) resulting CQ into CQs $q_0(\bar{c}_0), q_1(\bar{c}_1), \dots, q_n(\bar{c}_n)$ such that:

1. q_0 has no quantified variables;
2. for all $i > 0$, $\bar{c}_i$ is empty or a single constant from $\bar{c}_0$;
3. for all $j > i > 0$, q_i and q_j share no variables.

For a set T of frontier-one TGDs, the Δ-*closure* $\mathsf{cls}(T, \Delta)$ *of* T is the smallest set of CQs such that:

- For every TGD $\phi(x, \bar{y}) \to \exists \bar{z}\, \psi(x, \bar{z}) \in T$ and every $c \in \Delta$, the CQs $q_\phi(c)$ and $q_\psi(c)$ are contained in $\mathsf{cls}(T, \Delta)$;
- if $q(\bar{c}) \in \mathsf{cls}(T, \Delta)$ and $q_0(\bar{c}_0), q_1(\bar{c}_1), \dots, q_n(\bar{c}_n)$ is a Δ-splitting of $q(\bar{c})$, then $q_1(\bar{c}_1), \dots, q_k(\bar{c}_k) \in \mathsf{cls}(T, \Delta)$.

It is important to note that:

Lemma 24. *The cardinality of* $\mathsf{cls}(T, \Delta)$ *is bounded by* $|T| \cdot 2^m \cdot (|\Delta| + 1)^m$*, with m the maximum of body and head width of T.*

Proof. Every query in $\mathsf{cls}(T, \Delta)$ can be obtained by starting with a Boolean CQ $\exists x\, q_\psi(x, \bar{z})$ or $\exists x\, q_\phi(x, \bar{y})$ for some TGD $\phi(x, \bar{y}) \to \exists \bar{z}\, \psi(x, \bar{z}) \in T$, restricting it to some subset of its variables, and then possibly replacing any number of variables with constants from Δ. □

We can now describe the 2ATA achieving (†) more formally. It uses all members of $\mathsf{cls}(T_1, \Delta)$ as states. If it visits $w \in W$ in state $q(\bar{c})$, it non-deterministically chooses a $\mathsf{adom}(L_1(w))$-splitting $q_0(\bar{c}_0), q_1(\bar{c}_1), \dots, q_n(\bar{c}_n)$ of $q(\bar{c})$, verifies that $q_0(\bar{c})$ (viewed as a database) is contained in $L_1(w)$ and, for each i with $1 \leq i \leq n$:

[8] Different variables may be replaced with the same constant.

- if $q_i(\bar{c}_i)$ is unary and $\bar{c}_i = c$, then the 2ATA sends a copy in state $q_i(\bar{c}_i)$ to some $v \in [w]_c$;
- if $q_i(\bar{c}_i)$ is Boolean, then the 2ATA sends a copy in state $q_i(\bar{c}_i)$ to some $v \in W$.

Using the priorities we can make sure that the process terminates, that is, at some point the splitting takes the form of $q_0(\bar{c}_0) = q(\bar{c})$. Using Lemma 24 we can verify that $\mathfrak{A}_2$ uses exponentially many states.

Automaton $\mathfrak{A}_3$. Before we can describe the idea, we need to establish some necessary preliminaries. A TGD $\phi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z}\, \psi(\bar{x}, \bar{z})$ is *full* if $\bar{z}$ is empty. A *monadic datalog program* is a set of full frontier-one TGDs. In such programs, however, we also admit nullary relation symbols and empty frontiers.

Let T be a set of frontier-one TGDs. We construct from T a monadic datalog program T' as follows. Recall that all CQs in $q \in \mathsf{bodyCQ}(T)$ are Boolean or unary. For every CQ $q(\bar{x}) \in \mathsf{bodyCQ}(T)$, introduce a relation symbol $A_{q(\bar{x})}$ of arity $|\bar{x}|$. With $\mathsf{bodyCQ}^+(T)$, we denote the set of (Boolean or unary) CQs that can be obtained from a CQ $q \in \mathsf{bodyCQ}(T)$ by adding any number of atoms $A_{p(\bar{x})}(\bar{y})$ with $p(\bar{x}) \in \mathsf{bodyCQ}(T)$ and $\bar{y}$ a tuple of variables from q of length $|\bar{x}|$.

Given a CQ $q \in \mathsf{bodyCQ}^+(T)$, we denote with $q^\downarrow$ the CQ obtained from q by replacing every nullary atom $A_{p(\bar{x})}$ with a copy $p'(\bar{x})$ of $p(\bar{x})$ that uses only fresh variable names. In addition, if $\bar{x} = x$ is non-empty, the copy of x in $p'(\bar{x})$ is identified with x. Now, T' consists of all rules

(i) $q(\bar{x}) \rightarrow A_{p(\bar{x})}(\bar{x})$ such that $q(\bar{x}) \in \mathsf{bodyCQ}^+(T)$ and $p(\bar{x}) \in \mathsf{bodyCQ}(T)$ have the same arity and $D_{q^\downarrow}, T \models p(\bar{x})$,

(ii) $q(x) \rightarrow A_{p(\bar{x})}(\bar{x})$ such that $p(\bar{x}) \in \mathsf{bodyCQ}(T)$ and q is a conjunction of nullary atoms $A_{p'}$ and unary atoms $A_{p'}(x)$, $p' \in \mathsf{bodyCQ}(T)$, such that $D_{q^\downarrow}, T \models p(\bar{x})$.

Lemma 25. *Let T be a set of frontier-one TGDs and T' the corresponding monadic datalog program. Then, for every database D, $q(\bar{x}) \in \mathsf{bodyCQ}(T)$, and every $\bar{c} \in \mathsf{adom}(D)^{|\bar{x}|}$,*

$$D, T \models q(\bar{c}) \quad \textit{iff} \quad D, T' \models A_{q(\bar{x})}(\bar{c}).$$

Proof. For the "if"-direction, suppose that $D, T \not\models q(\bar{c})$, for some database D, $q(\bar{x}) \in \mathsf{bodyCQ}(T)$, and $\bar{c} \in \mathsf{adom}(D)^{|\bar{x}|}$, that is, $\bar{c} \notin q(\mathsf{chase}_T(D))$. Obtain an instance I from $\mathsf{chase}_T(D)$ by interpreting the fresh symbols $A_{p(\bar{y})}$ in the expected way, that is, for all $p(\bar{y}) \in \mathsf{bodyCQ}(T)$, and $\bar{d} \in \mathsf{adom}(\mathsf{chase}_T(D))^{|\bar{y}|}$, we have:

$$A_{p(\bar{y})}(\bar{d}) \in I \quad \text{iff} \quad \bar{d} \in p(\mathsf{chase}_T(D)).$$

It is readily verified that I is a model of D and T'. But since $\bar{c} \notin q(\mathsf{chase}_T(D))$, we have $A_{q(\bar{x})}(\bar{c}) \notin I$ and thus $D, T' \not\models A_{q(\bar{x})}(\bar{c})$.

For the "only if"-direction, let $D, T' \not\models A_{q(\bar{x})}(\bar{c})$, for some database D, $q(\bar{x}) \in \mathsf{bodyCQ}(T)$, and $\bar{c} \in \mathsf{adom}(D)^{|\bar{x}|}$, that is, $A_{q(\bar{x})}(\bar{c}) \notin I$ for some model I of D and T'. We associate with every $d \in \mathsf{adom}(I)$ a T-type t_d by taking:

$$\begin{aligned} t_d = \; & \{p(x) \in \mathsf{bodyCQ}(T) \mid A_{p(x)}(d) \in I\} \cup \\ & \{p() \in \mathsf{bodyCQ}(T) \mid A_{p()} \in I\}. \end{aligned}$$

Now obtain an instance I' from I by adding, for every $d \in \mathsf{adom}(I)$ a disjoint copy of $\mathsf{chase}_T(t_d)$ identifying its root with d. It remains to show that I' is a model of T and that $\bar{c} \notin q(I')$. For both statements, we will need the following auxiliary claim.

Claim 1. For all $p(\bar{x}) \in \mathsf{bodyCQ}(T)$ and all $\bar{d} \in \mathsf{adom}(I)^{|\bar{x}|}$, we have

$$\bar{d} \in p(I') \quad \text{iff} \quad A_{p(x)}(\bar{d}) \in I.$$

Proof of Claim 1. The "if"-direction is immediate from the construction of I', so we concentrate on "only if". The proof is by induction on the number of variables in p.

Let $\bar{d} \in p(I')$, that is, there is a homomorphism h from p to I' with $h(\bar{x}) = \bar{d}$. Let us define $h^\downarrow(x) = e$ in case $h(x)$ is in the copy of $\mathsf{chase}_T(t_e)$, for all variables x in p. Set $H^\downarrow = \{h^\downarrow(x) \mid x \text{ variable in } p\}$. We decompose p guided by h into queries p_0, p_e with $e \in H^\downarrow$ as follows:

- For every $e \in H^\downarrow$, p_e is the restriction of p to all variables y in p with $h^\downarrow(y) = e$.
- p_0 consists of the remaining atoms and has answer variable x in case p has an answer variable x.

Note that all obtained queries are contained in $\mathsf{bodyCQ}(T)$. We distinguish three cases. Observe that Case 1 applies if p contains at most one variable and thus establishes the induction base.

Case 1: $p_e = p$, for some $e \in H^\downarrow$. If p is Boolean, then there is a homomorphism g from p to $\mathsf{chase}_T(t_e)$. If p has answer variable x, then $\bar{d} = e$ and $g(x)$ is the constant corresponding to the free variable of t_e. Thus, $t_e, T \models p(\bar{x})$ and we find in T' the rule $\widehat{q}(x) \rightarrow A_{p(\bar{x})}(\bar{x})$ where $\widehat{q}$ is the conjunction of all $A_{p'(x)}(x)$ with $A_{p'(x)}(e) \in I$ and all $A_{p'(x)} \in I$. By definition of $\widehat{q}$, $e \in \widehat{q}(I)$ and hence $A_{p(\bar{x})}(\bar{d}) \in I$.

Case 2: $p_0 = p$. Then h witnesses that $\bar{d} \in p(I)$. Since trivially $D_{p^\downarrow}, T \models p(\bar{x})$, T' contains the rule $p(\bar{x}) \rightarrow A_{p(\bar{x})}(\bar{x})$. This implies that $A_{p(\bar{x})}(\bar{d}) \in I$.

Case 3: Otherwise. Then all obtained queries p_0, p_e have less variables than p. We obtain a CQ $\widehat{p}$ from p_0 by doing the following, for every $e \in H^\downarrow$:

- if p_0 and p_e do not share any variable, then add the nullary atom A_{p_e}, and
- if p_0 and p_e share a variable, then pick such a variable x_e, make it an answer variable in p_e, and add the unary atom $A_{p_e(x_e)}(x_e)$.

Observe that h witnesses that $\bar{d} \in \widehat{p}(I)$ since $\bar{d} \in p_0(I)$ and, for all $e \in H^\downarrow$, we have:

- If $\widehat{p}$ contains A_{p_e}, then h witnesses that $() \in p_e(I')$, and thus, by induction, $A_{p_e} \in I$.

- If $\widehat{p}$ contains $A_{p_e(x_e)}(x_e)$, then h witnesses that $h(x_e) \in p_e(I')$, and thus, by induction, $A_{p_e(x_e)}(h(x_e)) \in I$.

Moreover, it should be clear that $D_{\widehat{p}^\downarrow}, T \models p(\bar{x})$, and thus we find the rule $\widehat{p}(\bar{x}) \rightarrow A_{p(\bar{x})}(\bar{x})$ in T', hence $A_{p(\bar{x})}(\bar{d}) \in I$.

This finishes the proof of Claim 1. Claim 1 immediately implies that $\vec{c} \notin q(I')$ since $A_{q(\bar{x})}(\bar{c}) \notin I$. It remains to verify the following.

Claim 2. I' is a model of T.

Proof of Claim 2. To see that I' is a model of T let $\vartheta = \phi(x, \bar{y}) \rightarrow \exists \bar{z}\, \psi(x, \bar{z})$ be a TGD in T and suppose that $d \in q_\phi(I')$, that is, there is a homomorphism h from q_ϕ to I' with $h(x) = d$. We show that $d \in q_\psi(I')$. We distinguish cases.

Case 1: $d \in \mathsf{adom}(I)$. In this case, Claim 1 implies that $A_{q_\phi}(d) \in I$, and hence $q_\phi(x) \in t_d$. Since $\mathsf{chase}_T(t_d)$ is a model of both $q_\phi(x)$ and ϑ, we have that $x \in q_\psi(\mathsf{chase}_T(t_d))$. The construction of I' ensures that $x \in q_\psi(I')$.

Case 2: $d \notin \mathsf{adom}(I)$. Then d is in the copy of $\mathsf{chase}_T(t_e)$, for some $e \in \mathsf{adom}(I)$. We obtain CQs q_1, q_2 from q_ϕ as follows:

- q_1 is the restriction of q_ϕ to all variables y such that $h(y)$ is in the copy of $\mathsf{chase}_T(t_e)$.
- q_2 is obtained by starting from the remaining atoms and then identifying all variables shared with q_1. (Note that every such variable y satisfies $h(y) = e$.) If there is none such variable, then q_2 is Boolean. Otherwise, the variable obtained in the identification process is the answer variable.

The homomorphism h witnesses that $e \in q_2(I')$ if q_2 is unary and $() \in q_2(I')$ otherwise. If q_2 is unary, Claim 1 yields that $A_{q_2(y)}(e) \in I$ and thus $q_2(x) \in t_e$. Otherwise, Claim 1 yields that $A_{q_2} \in I$ and thus $q_2 \in t_e$. By definition of $\mathsf{chase}_T(t_e)$ we find a homomorphism g from q_2 to $\mathsf{chase}_T(t_e)$ that maps the answer variable of q_2 (if any) to the constant corresponding to the free variable of t_e. Let h' be the copy of h that maps q_1 to $\mathsf{chase}_T(t_e)$ (instead of the copy of $\mathsf{chase}_T(t_e)$ in I'). But then $g \cup h'$ is a homomorphism from q_ϕ to $\mathsf{chase}_T(t_e)$, and hence $d' \in q_\phi(\mathsf{chase}_T(t_e))$ where d' is the copy of d in $\mathsf{chase}_T(t_e)$. Since $\mathsf{chase}_T(t_e)$ is a model of ϑ, we have $d' \in q_\psi(\mathsf{chase}_T(t_e))$ and thus $d \in q_\psi(I')$. This finishes the proof of Claim 2. □

Lemma 26. *Let T be a set of frontier-one TGDs of body width k. Then, T' consists of:*

- *at most exponentially many rules of type (i), and*
- *at most double exponentially many rules of type (ii).*

Moreover, rules of type (i) have at most k variables and rules of type (ii) have only one variable. T' can be computed in time triple exponential in $||T||$.

Proof. First note that there are at most exponentially many queries in $\mathsf{bodyCQ}^+(T)$. Indeed, by construction, there are only exponentially many queries in $\mathsf{bodyCQ}(T)$, and each query has at most k variables, k the body width of T. These at most k variables are now labeled with the fresh concept names $A_{\psi(\bar{x})}$ with $\psi(\bar{x})$ in $\mathsf{bodyCQ}(T)$. It follows that there are at most exponentially many queries in $\mathsf{bodyCQ}^+(T)$. Overall, there are at most exponentially many candidates for rules of type (i) and at most double exponentially many candidates for rules of type (ii) in T'.

Also note that, for each query $q(\bar{x})$ that can occur in a rule body in (i) or (ii), the query $q^\downarrow$ is a T-type, and thus of size exponential in $||T||$. Moreover, note that the checks $D_{q^\downarrow}, T \models p(\bar{x})$ that have to be made in oder to decide whether a candidate rule is included in T' are instances of query evaluation w.r.t. frontier-one TGDs. Since query evaluation w.r.t. frontier-one TGDs is 2EXPTIME-complete (Baget et al. 2011), all these checks can be made in triple exponential time. □

We are now in a position to describe the automaton $\mathfrak{A}_3$. Let T_2' be the monadic datalog program obtained from T_2. The automaton uses T_2' to verify the correctness of the labeling μ_w by visiting every node $w \in W$ and doing the following for every $c \in \mathsf{adom}(L_0(w))$ and every $q(\bar{x}) \in \mathsf{bodyCQ}(T_2)$:

1. if $q(x) \in \mu_w(c)$ is unary, then verify that $I_{W,L_0}, T_2 \models \mathcal{A}_{q(x)}([w]_c)$;
2. if $q \in \mu_w(c)$ is Boolean, then verify that $I_{W,L_0}, T_2 \models \mathcal{A}_q$;
3. if $q(x) \notin \mu_w(c)$ is unary, then verify that $I_{W,L_0}, T_2 \not\models \mathcal{A}_{q(x)}([w]_c)$;
4. if $q \notin \mu_w(c)$ is Boolean, then verify that $I_{W,L_0}, T_2 \not\models \mathcal{A}_q$.

By Lemma 25, the automaton may use T_2' in place of T_2. For Points 1 and 2, the automaton guesses a *derivation*, as commonly used to define the semantics of datalog; for details, we refer to (Abiteboul, Hull, and Vianu 1995). For Points 3 and 4, it needs to verify that there is no derivation, which is easy by dualizing the subautomaton for Points 1 and 2. We thus concentrate on Points 1 and 2.

To verify that $I_{W,L_0}, T_2' \models A_{q(x)}([w]_c)$ (resp., $I_{W,L_0}, T_2' \models A_q$), the automaton non-deterministically chooses a derivation of $A_{q(\bar{x})}([w]_c)$ (resp., A_q) in I_{W,L_0} under T_2'. For doing so, it uses states from $\mathsf{cls}(T_2'', \Delta)$ where T_2'' is the fragment of T_2' consisting only of the rules of type (i) and where cls defined as in the description of $\mathfrak{A}_2$. It starts in state $A_{q(\bar{x})}(c)$ (resp., A_q). (Recall that in world w, the element $[w]_c$ of I_{W,L_0} is represented by constant c.)

Intuitively, if the automaton visits $w \in W$ in a state $q(\bar{c})$, then this represents the obligation to find a derivation for $q(\widehat{c})$ in I_{W,L_0} under T_2', where $\widehat{c} = [w]_c$ if $\bar{c} = c$ consists of a single constant and $\widehat{c}$ is empty otherwise. We distinguish cases depending on the shape of $q(\bar{c})$.

Case (1) If $q(\bar{c})$ is of shape $A_{p(\bar{x})}(\bar{c})$, then the automaton non-deterministically does one of the following:

- non-deterministically choose a rule $q'(\bar{x}) \rightarrow A_{p(\bar{x})}(\bar{x})$ of type (i) in T_2' and proceed in state $q'(\bar{c})$, or
- non-deterministically choose a rule $q'(x) \rightarrow A_{p(\bar{x})}(\bar{x})$ of type (ii) in T_2' and:
  - if $\bar{c} = c$ is a single constant, then (using alternation) the automaton proceeds in states $A_{p'}(c)$, for all unary atoms $A_{p'}(x)$ that occur in q', and in $A_{p'}$, for all nullary atoms $A_{p'}$ that occur in q';

- if $\bar{c}$ is empty, the automaton navigates (non-deterministically) to some $w \in W$, picks a constant $c \in \mathsf{adom}(L_0(w))$ and proceeds as in the previous item (again using alternation).

Case (2) If $q(\bar{c})$ is not of shape $A_{p(\bar{x})}(\bar{c})$, then the automaton non-deterministically chooses an $\mathsf{adom}(B(w))$-splitting $q_0(\bar{c}_0), q_1(\bar{c}_1), \ldots, q_n(\bar{c}_n)$ of $q(\hat{c})$. It then obtains $q_0'(\bar{c}_0)$ from $q_0(\bar{c}_0)$ by dropping all atoms of the form $A_{p(x)}(x)$ and A_p and proceeds to verify that $q_0'(\bar{c})$ (viewed as a database) is contained in $L_0(w)$. Additionally, for each i with $1 \leq i \leq n$:

- if $q_i(\bar{c}_i)$ is unary with $\bar{c}_i = c$, then the 2ATA sends a copy in state $q_i(\bar{c}_i)$ to some $v \in [w]_c$;
- if $q_i(\bar{c}_i)$ is Boolean, then the 2ATA sends a copy in state $q_i(\bar{c}_i)$ to some $v \in W$.

Finally, the dropped atoms are processed as follows.

- if $A_p(c)$ is a unary atom in $q_0(\bar{c}_0)$, the automaton sends a copy in state $A_p(c)$ to w;
- if A_p is a Boolean atom in $q_0(\bar{c}_0)$, the automaton sends a copy in state A_p to w.

Using the priorities, we can make sure that the process terminates. Combining Lemma 24 and Lemma 26, one can verify that $\mathsf{cls}(T_2'', \Delta)$ (recall that T_2'' is the subset of T_2' consisting only of rules of type (i)) contains exponentially many queries and thus $\mathfrak{A}_3$ uses at most exponentially many states. By Lemma 26, $\mathfrak{A}_3$ can be computed in triple exponential time.

Automaton $\mathfrak{A}_4$. To construct automaton $\mathfrak{A}_4$, first note that $\mathsf{chase}_{T_2}(I_{W,L_1}) \not\to I_{W,L_1}$ if for some $w \in W$ and some $c \in \mathsf{adom}(L_0(w))$, there is no Σ_Q-homomorphism h from $\mathsf{chase}_{T_2}(\mu_w(c))$ to I_{W,L_1} with $h(x) = [w]_c$. It thus suffices to check the latter.

For convenience, we concentrate on the complement and build an automaton that is capable of verifying that, given $w \in W$ and $c \in \mathsf{adom}(L_0(w))$,

(†) there is a Σ_Q-homomorphism h from $\mathsf{chase}_{T_2}(\mu_w(c))$ to I_{W,L_1} with $h(x) = [w]_c$.

The automaton $\mathfrak{A}_4$ then non-deterministically guesses a $w \in W$ and a $c \in \mathsf{adom}(L_0(w))$ and uses the complement/dualization of the automaton that verifies (†).

We rely on the representation of $\mathsf{chase}_{T_2}(\mu_w(c))$ as a (rooted!) $\mu_w(c)$-proper T_2-labeled instance tree $\mathcal{T} = (V, E, B, \mu)$, see the discussion that preceeds Lemma 9. It is important to realize that the instance $B(v)$ at some node $v \in V$ together with the labeling μ_v of $\mathsf{adom}(B(v))$ with T_2-types completely determine the successors v' of v and their labeling $B(v')$ and $\mu_{v'}$. More precisely, the type $\mu(c)$ of a constant $c \in \mathsf{adom}(B(v))$ determines all successors v' of v that have c in their domain $\mathsf{adom}(B(v'))$. Moreover, v has a successor with label $B(v'), \mu_{v'}$ iff $\mu_v(c), T_2 \models q^c_{(B(v'),\mu_{v'})}(x)$ (c.f. Condition 2 of properness). Since query evaluation w.r.t. frontier-one TGDs is 2EXPTIME-complete (Baget et al. 2011) and the size of the input is exponential in $||T||$, this check is possible in triple exponential time. Hence, all possible successors can be computed in triple exponential time.

For achieving (†), the automaton proceeds as follows. It memorizes (in its states) the database $B(v)$ at the current node $v \in V$ of $\mathcal{T}$ and the type labeling μ_v. It then guesses a partial Σ_Q-homomorphism from $B(v)$ to the currently visited node $w \in W$. Each variable that is mapped to the current state gives rise to successors v' of v with associated $B(v')$ and $\mu_{v'}$ labelings, and the automaton spawns copies of itself that generate these successor (as states), moves to neighboring nodes in the input tree, and proceeds there. As in the encoding of $\mathcal{T}$ as a labeled tree, it remaps the constants in the instances $B(\cdot)$ to ensure that only finitely many states are used; this is possible since every instance $B(v)$ is isomorphic to the head of some TGD in T_2.

More formally, the automaton uses as states pairs $\langle q(\bar{c}), \mu \rangle$ where:

- $q(\bar{c})$ is an element of $\mathsf{cls}(T_2, \Delta)$, and
- μ assigns a T_2-type to every variable in $q(\bar{c})$.

When the automaton visits a node $w \in W$ in state $\langle q(\bar{c}), \mu \rangle$, this represents the obligation to verify that there is a Σ_Q-homomorphism h from q to I_{W,L_1} such that:

- for every constant $c \in \bar{c}$, $h(c) = [w]_c$, and
- for every variable x in q, there is a Σ_Q-homomorphism g from $\mathsf{chase}_{T_2}(\mu(x))$ to I_{W,L_1} with $g(x) = h(x)$.

For doing so, the automaton non-deterministically chooses an $\mathsf{adom}(B(w))$-splitting $q_0(\bar{c}_0), \ldots, q_n(\bar{c}_n)$ of $q(\bar{c})$ and proceeds as follows:

- it verifies that the Σ_Q-restriction of $q_0(\bar{c}_0)$ is a subset of $L_1(w)$;
- for every i with $1 \leq i \leq n$, we let μ_i be the restriction of μ to the variables in q_i, then
  - if $q_i(\bar{c}_i)$ is unary with $\bar{c}_i = c$, then the 2ATA sends a copy in state $\langle q_i(\bar{c}_i), \mu_i \rangle$ to some $v \in [w]_c$.
  - if $q_i(\bar{c}_i)$ is Boolean, then the 2ATA sends a copy in state $\langle q_i(\bar{c}_i), \mu_i \rangle$ to some $v \in W$.
- for every variable x in q that was replaced by a constant d in the splitting, consider any node v in $\mathcal{T}$ and any $e \in \mathsf{adom}(B(v))$ with $\mu_v(e) = \mu(x)$,[9] and all successors v' of v with $\mathsf{adom}(B(v)) \cap \mathsf{adom}(B(v')) \subseteq \{e\}$. Let q' be $B(v')$ viewed as a CQ which is Boolean with e viewed as the answer variable if $e \in \mathsf{adom}(B(v'))$ and Boolean otherwise. Further let $\mu' = \mu_{v'}$. The automaton does the following:
  - if q' is unary, then it sends a copy in state $\langle q'(d), \mu' \rangle$ to some $w' \in [w]_d$;
  - if q' is Boolean, then it sends a copy in state $\langle q', \mu' \rangle$ to some $w' \in W$.

Overall, one can verify that the number of states is at most double exponential in $||T_2||$. There are doubly exponentially many types and, by Lemma 24, the size of $\mathsf{cls}(T_2, \Delta)$ is bounded exponentially in the size of $||T_2||$. Since all queries in $\mathsf{cls}(T_2, \Delta)$ have at most $||T_2||$ variables, the triple exponential bound follows. As argued, the automaton can be computed in time triple exponential in $||T_2||$.

[9] Choosing different v and e leads to exactly the same result provided that $\mu_v(e) = \mu(x)$.

E.3 Upper Bounds for CQ-Conservativity

We actually work with a refinement of the characterization given in Theorem 9; its proof is based on Lemma 8. The formulation of this refinement is somewhat more technical than the formulation of Theorem 9, and in fact we decided to go in these two steps for didactic reasons.

Theorem 11. *Let T_1 and T_2 be sets of frontier-one TGDs, Σ_D and Σ_Q schemas, k the body width of T_1, and ℓ the head width of T_1. Then $T_1 \models^{CQ}_{\Sigma_D,\Sigma_Q} T_2$ iff for all tree-like Σ_D-databases D of width at most k and all tree-like models I of T_1 and D of width $\max(k,\ell)$, the following holds:*

1. $\mathsf{chase}_{T_2}(D)|^{\mathsf{con}}_{\Sigma_Q} \to_{\Sigma_Q} I$;
2. *for every labeled Σ_Q-head fragment $A = (F,\mu)$ of T_2 with $\mathsf{chase}_{T_2}(D) \models q_A$, one of the following holds:*
   (a) $\mathsf{chase}_{T_2}(D_A)|^{\mathsf{con}}_{\Sigma_Q} \to_{\Sigma_Q} I$;
   (b) $\mathsf{chase}_{T_2}(D_A)|^{\mathsf{con}}_{\Sigma_Q} \to^{\mathsf{lim}}_{\Sigma_Q} \mathsf{chase}_{T_1}(\mathsf{tp}_{T_1}(I,c))$ *for some* $c \in \mathsf{adom}(D)$.

Proof. It suffices to show that for every tree-like Σ_D-database D of width k, Conditions 1 and 2 of Theorem 9 are satisfied if and only if for all tree-like models I of T_1 and D of width $\max(k,\ell)$, Conditions 1 and 2 above are satisfied.

First assume that for all tree-like models I of T_1 and D of width $\max(k,\ell)$, Conditions 1 and 2 above are satisfied. Since $\mathsf{chase}_{T_1}(D)$ is such a model, Condition 1 of Theorem 9 is also satisfied. Now for Condition 2. By Lemma 8, for all maximally Σ_Q-connected components J of $\mathsf{chase}_{T_2}(D) \setminus \mathsf{chase}_{T_2}(D)|^{\mathsf{con}}_{\Sigma_Q}$, Condition 2(a) or 2(b) above is satisfied when $\mathsf{chase}_{T_2}(D_A)|^{\mathsf{con}}_{\Sigma_Q}$ is replaced by J. In the former case, also Condition 2(a) of Theorem 9 is satisfied. In the latter case, it follows that $J \to^{\mathsf{lim}}_{\Sigma_Q} \mathsf{chase}_{T_1}(D)$. One can then show exactly as in the proof of Theorem 9 that either Condition 2(a) or 2(b) of that theorem is satisfied.

Conversely, suppose that Conditions 1 and 2 of Theorem 9 are satisfied for D. Since $\mathsf{chase}_{T_1}(D) \to I$ for every model I of T_1 and D, Condition 1 of Theorem 9 implies that for all tree-like models I of T_1 and D of width $\max(k,\ell)$, Condition 1 above is satisfied. It remains to argue that Condition 2 above is satisfied. Assume to the contrary that is is not. Then there is some tree-like model I of T_1 and D of width $\max(k,\ell)$ and some labeled Σ_Q-head fragment $A = (F,\mu)$ of T_2 such that both 2(a) and 2(b) above are violated. Since $\mathsf{chase}_{T_1}(D) \to I$, these conditions are still violated when I is replaced by $\mathsf{chase}_{T_1}(D)$.

We distinguish the following cases:

- $\mathsf{chase}_{T_2}(D_A)|^{\mathsf{con}}_{\Sigma_Q} \not\to \mathsf{chase}_{T_2}(D)|^{\mathsf{con}}_{\Sigma_Q}$.

  Then $\mathsf{chase}_{T_2}(D_A)|^{\mathsf{con}}_{\Sigma_Q}$ is an induced subinstance of a maximally Σ_Q-connected component I of $\mathsf{chase}_{T_2}(D) \setminus \mathsf{chase}_{T_2}(D)|^{\mathsf{con}}_{\Sigma_Q}$. Thus, $\mathsf{chase}_{T_2}(D_A)|^{\mathsf{con}}_{\Sigma_Q} \not\to_{\Sigma_Q} \mathsf{chase}_{T_1}(D)$ implies $I \not\to_{\Sigma_Q} \mathsf{chase}_{T_1}(D)$ and Condition 2(a) of Theorem 9 is not satisfied. Moreover, $\mathsf{chase}_{T_2}(D_A)|^{\mathsf{con}}_{\Sigma_Q} \not\to^{\mathsf{lim}}_{\Sigma_Q} \mathsf{chase}_{T_1}(\mathsf{tp}_{T_1}(\mathsf{chase}_{T_1}(D),c))$ for all $c \in \mathsf{adom}(D)$ implies $I \not\to^{\mathsf{lim}}_{\Sigma_Q} \mathsf{chase}_{T_1}(D)|^{\downarrow}_{c}$, for all $c \in \mathsf{adom}(D)$. This is because $\mathsf{chase}_{T_2}(D_A)|^{\mathsf{con}}_{\Sigma_Q}$ is a subinstance of I and, due to Lemma 7, $\mathsf{chase}_{T_1}(D)|^{\downarrow}_{c}$ is a subinstance of $\mathsf{chase}_{T_1}(\mathsf{tp}_{T_1}(\mathsf{chase}_{T_1}(D),c))$. Thus, both Condition 2(a) and 2(b) of Theorem 9 are violated, a contradiction.

- $\mathsf{chase}_{T_2}(D_A)|^{\mathsf{con}}_{\Sigma_Q} \to \mathsf{chase}_{T_2}(D)|^{\mathsf{con}}_{\Sigma_Q}$.

  Then $\mathsf{chase}_{T_2}(D_A)|^{\mathsf{con}}_{\Sigma_Q} \not\to_{\Sigma_Q} \mathsf{chase}_{T_1}(D)$ implies $\mathsf{chase}_{T_2}(D)|^{\mathsf{con}}_{\Sigma_Q} \not\to_{\Sigma_Q} \mathsf{chase}_{T_1}(D)$. Hence, Condition 1 of Theorem 9 is not satisfied, which is again a contradiction. □

Let $T_1, T_2, \Sigma_D, \Sigma_Q$ be given. We may again assume without loss of generality that all symbols from Σ_D and Σ_Q occur in T_1. Let k and ℓ be the body and head width of T_1. It suffices to devise a 2ATA $\mathfrak{B}$ such that

($*_{\mathfrak{B}}$) $\mathfrak{B}$ accepts all tree-like instances I of width $\max(k,\ell)$ that are a model of T_1 and of some tree-like Σ_Q-database D of width k such that Condition 1 and Condition 2 of Theorem 11 are violated.

In order to represent tree-like instances of bounded width as the input to 2ATAs, we use exactly the same encoding of infinite instances as for hom-conservativity, and in fact, the constructed automata run over the same alphabet $\Theta = \Theta_0 \times \Theta_0 \times \Theta_1$. Recall that an input tree over this alphabet represents a Σ_D-database D in the first component, a model I of T_1 and D in the second component, and the chase of D with T_2, restricted to $\mathsf{adom}(D)$, in the last component.

The desired 2ATA $\mathfrak{B}$ is constructed as the intersection of 2ATAs $\mathfrak{B}_0, \mathfrak{B}_1, \mathfrak{B}_2, \mathfrak{B}_3$, and $\mathfrak{B}$' where $\mathfrak{B}$' in turn is the union of 2ATAs $\mathfrak{B}_4$ and $\mathfrak{B}_5$, all of them provided by the following lemma.

Lemma 27. *There are 2ATAs $\mathfrak{B}_0, \mathfrak{B}_1, \mathfrak{B}_2, \mathfrak{B}_3, \mathfrak{B}_4$ such that:*

- *$\mathfrak{B}_0$ accepts (W,L) iff it is well-typed and (W,L_0) and (W,L_1) are well-formed;*
- *$\mathfrak{B}_1$ accepts (W,L) iff I_{W,L_0} is a Σ_D-database of width k;*
- *$\mathfrak{B}_2$ accepts (W,L) iff I_{W,L_1} is a model of I_{W,L_0} and T_1;*
- *$\mathfrak{B}_3$ accepts (W,L) iff for every $w \in W$ and every $c \in \mathsf{adom}(L_0(w))$,*

$$\mu_w(c) = \mathsf{tp}_{T_2}(\mathsf{chase}_{T_2}(I_{W,L_0}), [w]_c).$$

- *$\mathfrak{B}_4$ accepts (W,L) iff Condition 1 of Theorem 11 is violated; with 'I' replaced with 'I_{W,L_1}' is violated;*
- *$\mathfrak{B}_5$ accepts (W,L) iff Condition 2 of Theorem 11 with 'I' and 'D' replaced with 'I_{W,L_1}' and 'I_{W,L_0}', respectively, is violated;*

The number of states

- *of $\mathfrak{B}_0$ is exponential in $||T_1||$ (and independent of T_2);*
- *of $\mathfrak{B}_1$ does not depend on the input;*
- *of $\mathfrak{B}_2$ is exponential in $||T_1||$ (and independent of T_2);*
- *of $\mathfrak{B}_3$ is exponential in $||T_2||$ (and independent of T_1);*
- *of $\mathfrak{B}_4$ is exponential in $||T_2||$ (and independent of T_1);*
- *of $\mathfrak{B}_5$ is double exponential in both $||T_1||$ and $||T_2||$.*

All automata can be constructed in time triple exponential in $||T_1|| + ||T_2||$ and have maximum priority one.

It can be verified that $\mathfrak{B}$ satisfies $(*_{\mathfrak{B}})$ and thus $L(\mathfrak{B}) \neq \emptyset$ iff $T_1 \not\models^{\mathsf{CQ}}_{\Sigma_D,\Sigma_Q} T_2$. The rest of this section is devoted to proving Lemma 27. Automata $\mathfrak{B}_0, \mathfrak{B}_1, \mathfrak{B}_2, \mathfrak{B}_3$ are exactly as $\mathfrak{A}_0, \mathfrak{A}_1, \mathfrak{A}_2, \mathfrak{A}_3$ in Lemma 23, so we concentrate on $\mathfrak{B}_4$ and $\mathfrak{B}_5$.

Automaton $\mathfrak{B}_4$**.** The task of $\mathfrak{B}_4$ is to verify $\mathsf{chase}_{T_2}(I_{W,L_0})|^{\mathsf{con}}_{\Sigma_Q} \not\to_{\Sigma_Q} I_{W,L_1}$. Note that this is very similar to what is achieved by automaton $\mathfrak{A}_4$ from Lemma 23, which verifies that $\mathsf{chase}_{T_2}(I_{W,L_0}) \not\to_{\Sigma_Q} I_{W,L_1}$. In fact, it can be solved using essentially the same construction and thus has the same size and can be computed in the same time as $\mathfrak{A}_4$. More precisely, the gist of the construction of $\mathfrak{A}_4$ is to find an automaton that verifies, given $w \in W$ and $c \in \mathsf{adom}(L_0(w))$, that there is a Σ_Q-homomorphism h from $\mathsf{chase}_{T_2}(\mu_w(c))$ to I_{W,L_1} with $h(x) = [w]_c$. This is done by constructing $\mathsf{chase}_{T_2}(\mu_w(c))$ 'in the states'. $\mathfrak{B}_4$ does exactly the same, but disregards Σ_Q-disconnected parts of $\mathsf{chase}_{T_2}(\mu_w(c))$.

Automaton $\mathfrak{B}_5$**.** The task of $\mathfrak{B}_5$ is to verify that for all Σ_Q-labeled head fragments $A = (F, \mu)$ of T_2 such that $\mathsf{chase}_{T_2}(I_{W,L_0}) \models q_A$, the following hold:

1. $\mathsf{chase}_{T_2}(D_A)|^{\mathsf{con}}_{\Sigma_Q} \not\to_{\Sigma_Q} I_{W,L_1}$;
2. $\mathsf{chase}_{T_2}(D_A)|^{\mathsf{con}}_{\Sigma_Q} \not\to^{\mathsf{lim}}_{\Sigma_Q} \mathsf{chase}_{T_1}(\mathsf{tp}_{T_1}(I_{W,L_0}, [w]_c))$ for all $[w]_c \in \mathsf{adom}(I_{W,L_0})$.

Note that the condition $\mathsf{chase}_{T_2}(I_{W,L_0}) \models q_A$ is satisfied iff for some $[w]_c \in \mathsf{adom}(I_{W,L_0})$, the type $t = \mathsf{tp}_{T_2}(\mathsf{chase}_{T_2}(I_{W,L_0}, [w]_c))$ satisfies $t, T_2 \models q_A$. Since we are considering the intersection with $\mathfrak{B}_3$, we can assume that $\mu_w(c) = \mathsf{tp}_{T_2}(\mathsf{chase}_{T_2}(I_{W,L_0}, [w]_c))$ and thus, the latter condition is satisfied iff $\mu_w(c), T_2 \models q_A$ for some $w \in W$ and $c \in \mathsf{adom}(L_0(w))$.

Thus, the automaton can identify all relevant labeled Σ_Q-head fragments $A = (F, \mu)$ of T_2 by visiting all $w \in W$, all $c \in \mathsf{adom}(L_0(w))$, and testing for each whether $\mu_w(c), T_2 \models q_A$ is satisfied. The result of all possible such tests can be computed in time triple exponential in $||T_2||$ already during the construction of $\mathfrak{B}_5$, since query evaluation w.r.t. frontier-one TGDs is 2EXPTIME-complete (Baget et al. 2011).

If the test $\mu_w(c), T_2 \models q_A$ is succesfull, the automaton has to verify Points 1 and 2 above for $\mathsf{chase}_{T_2}(D_A)|^{\mathsf{con}}_{\Sigma_Q}$. There is once more a lot of similarity between Point 1 and what is achieved by automaton $\mathfrak{A}_4$ from Lemma 23. Constructing an automaton that verifies Point 1 is thus another variation of the construction of $\mathfrak{A}_4$, the main difference being that instead of chasing a single type we chase D_A with T_2 in the states of the automaton. In particular, the automaton starts in state $\langle q_F, \mu \rangle$ (note that $q_F \in \mathsf{cls}(T_2, \Delta)$).

For Point 2, we invoke Theorem 10 for every labeled Σ_Q-head fragment $A = (F, \mu)$ of T_2 identified above. The automaton memorizes A in its states and visits (again) all $w \in W$, and all $c \in \mathsf{adom}(I_{W,L_0})$ in order to verify that

$$\mathsf{chase}_{T_2}(D_A)|^{\mathsf{con}}_{\Sigma_Q} \not\to^{\mathsf{lim}}_{\Sigma_Q} \mathsf{chase}_{T_1}(\mathsf{tp}_{T_1}(I_{W,L_1}, [w]_c)).$$

Recall that these tests have been precomputed via Theorem 10. Hence, all the automaton has to do at this point is to guess[10] the T_1-type t of $[w]_c$ in I_{W,L_1}, verify that it is the correct type using the monadic datalog rewriting T_1' of T_1 as in automaton $\mathfrak{A}_3$ of Lemma 23, and lookup the result of $\mathsf{chase}_{T_2}(D_A)|^{\mathsf{con}}_{\Sigma_Q} \to^{\mathsf{lim}}_{\Sigma_Q} \mathsf{chase}_{T_1}(t)$ in the precomputated table.

Overall, the resulting automaton is of size double exponential in both $||T_2||$ (for Point 1) and $||T_1||$ (for guessing a T_1-type and verifying it in Point 2). It can be computed in triple exponential time. In particular, the computation of the lookup table for Point 2 is possible in triple exponential time, by Theorem 10.

E.4 Proof of Theorem 10

We prove Theorem 10 via the characterization of bounded homomorphisms in terms of standard (unbounded) homomorphisms given by Lemma 9. Let two sets of frontier-one TGDs T_1, T_2, a schema Σ, a labeled Σ-head fragment $A = (D, \mu)$ of T_2, and a T_1-type $\widehat{t}$ be given. It suffices to devise a 2ATA $\mathfrak{C}$ such that:

$(*_{\mathfrak{C}})$ $\mathfrak{C}$ accepts all encodings of $\widehat{t}$-proper T_1-labeled instance trees $\mathcal{T}$ of width m such that $\mathsf{chase}_{T_2}(D_A)|^{\mathsf{con}}_{\Sigma} \to I_{\mathcal{T}}$.

Here we need to work with possibly non-rooted instance trees, and thus we slightly modify our encoding of instance trees as input to the 2ATA. The input alphabet is $\Theta' = \Theta_0 \times \{0,1\} \times \Theta_1'$ where Θ_0 is defined as above and Θ_1' is the set of all mappings $\mu : \Delta' \to \mathsf{TP}(T_1)$ for some $\Delta' \subseteq \Delta$ with $|\Delta'| \leq m$. Note that, in contrast to the alphabet Θ_1 employed before, here we use T_1-types in place of T_2-types. For a Θ'-labeled tree (W, L) and $w \in W$ with $L(w) = (B, i, \mu)$, we use $L_0(w)$ to denote B, i_w to denote i, and μ_w to denote μ. Our aim is that every Θ'-labeled tree (W, L) represents a T_1-labeled instance tree $\mathcal{T} = (V, E, B, \mu)$ where the (V, E, B)-part is represented by (W, L_0) as before and the μ-part is represented by (W, μ_w).

The additional labeling with the 0/1-marker i_w is necessary because T_1-labeled instance trees need not have a root and thus may contain an infinite predecessor path. This path will be represented as a *downward* path in the (rooted!) Θ'-labeled trees, but marked with a 1-marker for identification purposes.

A Θ'-labeled tree (W, L) is *well-typed* if, for all $w \in W$, the domain of μ_w is $\mathsf{adom}(L(w))$, and for all successors v of w, and all $d \in \mathsf{adom}(L_0(w)) \cap \mathsf{adom}(L_0(v))$, we have $\mu_w(d) = \mu_v(d)$. It is *well-formed* if (W, L_0) satisfies the two conditions of well-formedness for Θ_0-labeled trees from the automata constructions above plus the following additional condition:

- there is a finite or infinite (and non-empty) path $\Pi = w_0, w_1, w_2, \ldots$ in W that starts at the root such that all nodes $w \in W$ with $i_w = 1$ lie on this path.

Every well-typed and well-formed Θ'-labeled tree (W, L) gives rise to a Σ-instance tree (V, E, B) and an associated instance $I_{W,L}$ as follows.

- the set of nodes V is W;
- the set of edges E is defined as follows:

[10] Recall that the T_1-type is not represented in the input.

– if w' is a successor of w and $i_{w'} = 0$, then $(w, w') \in E$;
– if w' is a successor of w and $i_{w'} = 1$, then (both w, w' lie on the path Π) and $(w', w) \in E$.

That is, the successor relation on the path Π becomes the predecessor relation; the remaining successor relations stay the same. The definition of the labeling B and consequently also of $I_{W,L}$ is exactly as in the preceeding encoding. Setting $\mu = \bigcup_{w \in W} \mu_w$, this extends to a T_1-labeled instance tree $\mathcal{T}_{W,L} = (V, E, B, \mu)$.

Conversely, for every T_1-labeled instance tree $\mathcal{T} = (V, E, B, \mu)$ of width at most m, we can find a Θ'-labeled tree (W, L) that represents $\mathcal{T}$ in the sense that $\mathcal{T}_{W,L}$ is isomorphic to $\mathcal{T}$. Since Δ is of size $2m$, it is possible to select a mapping $\pi : \mathsf{adom}(I_\mathcal{T}) \to \Delta$ such that for each edge $(v, w) \in E$ and all constants $c, c' \in \mathsf{adom}(B(w)) \cup \mathsf{adom}(B(v))$, we have $\pi(c) = \pi(c')$ iff $c = c'$. Define the Θ'-labeled tree (W, L) as follows:

- If (V, E) has a root, then $W = (V, E)$. Otherwise, there is an infinite path $v_0, v_1, \ldots$ in (V, E) such that v_{i+1} is a predecessor of v_i, for all $i \geq 0$. We make this path the infinite *successor* path Π starting from v_0 (and leave all other successor relations untouched).
- For all $w \in W$, $L(w) = (B(w), 0, \mu_w)$ if $w \notin \Pi$ and $B_w = (B(w), 1, \mu_w)$, if $w \in \Pi$.

Clearly, (W, L) satisfies the desired properties.

The automaton $\mathfrak{C}$ is the intersection of the three 2ATAs $\mathfrak{C}_0, \mathfrak{C}_1, \mathfrak{C}_2$ provided by the following lemma.

Lemma 28. *There are 2ATAs $\mathfrak{C}_0, \mathfrak{C}_1, \mathfrak{C}_2$ such that:*

– $\mathfrak{C}_0$ *accepts* (W, L) *iff* (W, L) *is well-typed and well-formed;*
– $\mathfrak{C}_1$ *accepts* (W, L) *iff the T_1-labeled instance tree $\mathcal{T}_{W,L}$ is $\widehat{t}$-proper;*
– $\mathfrak{C}_2$ *accepts* (W, L) *iff* $\mathsf{chase}_{T_2}(D_2)|^{\mathsf{con}}_{\Sigma} \to I_{W,L}$.

The number of states of

– $\mathfrak{C}_0$ *is exponential in $||T_1||$ (and independent of T_2);*
– $\mathfrak{C}_1$ *is linear in $||T_1||$ (and independent of T_2);*
– $\mathfrak{C}_2$ *is double exponential in $||T_2||$ (and independent of T_1).*

All automata can be constructed in time triple exponential in $||T_1|| + ||T_2||$ and have maximum priority one.

It can be verified that $\mathfrak{C}$ satisfies $(*_\mathfrak{C})$, and thus $L(\mathfrak{C}) \neq \emptyset$ iff there is some $\widehat{t}$-proper T_1-labeled instance tree $\mathcal{T}$ with $\mathsf{chase}_{T_2}(D_A)|^{\mathsf{con}}_{\Sigma} \to I_\mathcal{T}$. The automaton $\mathfrak{C}_0$ is straightforward.

Automaton $\mathfrak{C}_1$. The automaton simply visits every node $w \in W$ in the input tree (W, L) and verifies locally at each node that Conditions 1 and 2 of properness are satisfied. For Condition 1, we have to check whether the labeling $L_1(w)$ of the current node w satisfies Condition 1 of Properness. Condition 1(a) is a simple lookup and for Condition 1(b), one has to decide (during the construction of the automaton) whether $\widehat{t}, T_1 \models q_{(B(v),\mu_v)}$ for all possible labelings $(B(v), \mu(v))$. This is possible in time triple exponential in $||T_1||$, since query evaluation w.r.t. frontier-one TGDs is 2ExpTime-complete (Baget et al. 2011). For Condition 2 of properness, the automaton needs to memorize the constant c (if any) that is shared between neighboring nodes in W. The condition $\mu_u(c), T_1 \models q^c_{(B(w),\mu_v)}(x)$ that is part of Condition 2 of properness can then be checked, again in triple exponential time in $||T_1||$. Thus, $\mathfrak{C}_1$ can be computed in triple exponential time.

Automaton $\mathfrak{C}_2$. The check $\mathsf{chase}_{T_2}(D_A)|^{\mathsf{con}}_{\Sigma} \to I_{W,L}$ is similar to what $\mathfrak{A}_4$ in Lemma 23 achieves and, in fact, exactly what the sub-automaton for Point 1 of $\mathfrak{B}_5$ in Lemma 27 achieves. We repeat it here for the sake of convenience. The 2ATA $\mathfrak{C}_2$ behaves exactly as $\mathfrak{A}_4$, but starts in state $\langle q_F, \mu \rangle$. Recall that $A = (F, \mu)$, that q_F is F viewed as Boolean CQ, and that $q_F \in \mathsf{cls}(T_2, \Delta)$ and so $\langle q_F, \mu \rangle$ is a state in $\mathfrak{A}_4$.