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 Wroclaw, 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$^\ominus$ (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 $\mathcal{E}\mathcal{L}\mathcal{L}$ 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 $\Sigma_D$, and a query schema $\Sigma_Q$, decide whether $T_2$ is a $\Sigma_D, \Sigma_Q$-CQ-conservative extension of $T_1$, that is, whether for all $\Sigma_D$-databases $D$ and conjunctive queries (CQ) $q(\bar{x})$ in schema $\Sigma_Q$, 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 $\Sigma_D, \Sigma_Q$-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 $\Sigma_D$ and queries in schema $\Sigma_Q$. CQ-conservative extensions have been studied for various DLs and are decidable for many members of the DL-Lite and $\mathcal{E}\mathcal{L}\mathcal{L}$ 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 $\Sigma_D, \Sigma_Q$-hom-conservative extension which asks whether for every $\Sigma_D$-database, there is a $\Sigma_Q$-homomorphism$^1$ from the chase $\text{chaser}_{T_1}(D)$ of $D$ with $T_2$ to $\text{chaser}_{T_1}(D)$ 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 $\Sigma_D, \Sigma_Q$-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 $\Sigma_D$-databases as far as conjunctive queries and homomorphisms over schema $\Sigma_Q$ are concerned. We observe that $\Sigma_D, \Sigma_Q$-CQ-triviality and $\Sigma_D, \Sigma_Q$-hom-triviality coincide even for unrestricted TGDs, and thus we only speak of $\Sigma_D, \Sigma_Q$-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 $\Sigma_Q$. 
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 $3\mathsf{ExpTime}$ upper bounds, while $2\mathsf{ExpTime}$ lower bounds can be imported from the DL $\mathcal{EFIL}$, a fragment of frontier-one TGDs (Gutiérrez-Basulto, Jung, and Sabelleko 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 from the DL-Lite and Relational Databases. Fix countably infinite and pairwise $\Sigma$-nulls $\alpha$ and $\beta$ a tuple of variables of length $\alpha(R)$. We refer to the variables in $\alpha$ as the answer variables of $\alpha$ and denote a CQ with $q(\alpha)$ to emphasize that it has answer variables $\alpha$. The arity of $q$ is the length $|\alpha|$ of $\alpha$, and $q$ is Boolean if it is of arity 0.

Every CQ $q(\alpha)$ gives rise to a database $D_q$, known as the canonical database of $q$, by viewing variables as constants and atoms as facts. A $\Sigma$-homomorphism $h$ from $\alpha$ to an instance $I$ is a $\Sigma$-homomorphism from $D_q$ to $I$. A tuple $\bar{c}$ in $\text{adom}(I)$ is an answer to $q$ on $I$ if there is a homomorphism $h$ from $\alpha$ to $I$ with $h(\alpha) = \bar{c}$. The evaluation of $q(\alpha)$ 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.

**Related Work.** We already mentioned the work on DLs from the DL-Lite and $\mathcal{EFIL}$ families (Konev et al. 2011; Jung et al. 2020). For description logics such as $\mathcal{ALC}$ that support negation and disjunction, QA- 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 QA-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 $\mathcal{C}$ and $\mathcal{N}$ and variables $V$. We refer to the constants in $\mathcal{N}$ as nulls. A schema $\Sigma$ is a set of relation symbols $R$ with associated arity $\alpha(R) \geq 1$. A $\Sigma$-fact is an expression of the form $R(\bar{c})$ with $R \in \Sigma$ and $\bar{c}$ is an $\alpha(R)$-tuple of constants from $\mathcal{C} \cup \mathcal{N}$. A $\Sigma$-instance is a possibly infinite set of $\Sigma$-facts, and a $\Sigma$-database is a finite $\Sigma$-instance that uses only constants from $\mathcal{C}$. We write $\text{adom}(I)$ for the set of constants from $\mathcal{C} \cup \mathcal{N}$ used in instance $I$. For an instance $I$ and a schema $\Sigma$, $I_{|\Sigma}$ denotes the restriction of $I$ to $\Sigma$, that is, the set of all facts in $I$ that use a relation symbol from $\Sigma$. We say that $I$ is connected (resp., $\Sigma$-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 $\Sigma$, a $\Sigma$-homomorphism from instance $I$ to instance $J$ is a function $h : \text{adom}(I) \to \text{adom}(J)$ 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 $\mathcal{C}$ (but not necessarily from $\mathcal{N}$) and write $I \to J$ if there is a database-preserving $\Sigma$-homomorphism from $I$ to $J$.

**Conjunctive Queries.** A conjunctive query (CQ) over a schema $\Sigma$ takes the form $\exists \bar{c} \phi(\bar{x}, \bar{y})$ where $\bar{x}$ and $\bar{y}$ are tuples of variables from $V$, $\phi$ is a set of atoms $R(\bar{z})$ with $R \in \mathcal{E}$ and $\bar{z}$ a tuple of variables of length $\alpha(R)$. 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 $\Sigma$-homomorphism $h$ from $q$ to an instance $I$ is a $\Sigma$-homomorphism from $D_q$ to $I$. A tuple $\bar{c}$ in $\text{adom}(I)$ 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) $\vartheta$ is a first-order sentence $\forall \bar{x} \forall \bar{y} \big( \phi(\bar{x}, \bar{y}) \to \exists \exists \psi(\bar{x}, \bar{z}) \big)$ such that $q_\vartheta = \exists \exists \psi(\bar{x}, \bar{z})$ and $q_\psi = \exists \exists \psi(\bar{x}, \bar{z})$ are CQs. We call $\psi$ the body and head of $\vartheta$. The body may be the empty conjunction, that is, logical truth. The variables in $\bar{x}$ are the frontier variables. We may write $\vartheta$ as $\phi(\bar{x}, \bar{y}) \to \exists \exists \psi(\bar{x}, \bar{z})$. An instance $I$ satisfies $\vartheta$, denoted $I \models \vartheta$, if $q_\vartheta(I) \subseteq q_\psi(I)$. It satisfies a set of TGDs $T$ if $I \models \forall \vartheta$ for each $\vartheta \in T$. We then also say that $I$ is a model of $T$.

A TGD $\vartheta$ 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 $\alpha$ 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 $\phi(\bar{x}, \bar{y}) \to \exists \exists \psi(\bar{x}, \bar{z}) \in T$ is applicable at a tuple $\bar{c}$ of constants in $I$ if $\phi(\bar{c}, \bar{e}) \subseteq I$ for some $\bar{e}$ 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 $I \cup \{\psi(\bar{c}, \bar{c}^\prime)\}$ where $\bar{c}^\prime$ is the tuple obtained from $\bar{c}$ 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 $I_0, I_1, \ldots$ 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 chase$_T(I)$ to denote the result of an arbitrary, but fixed chase sequence for $I$ with $T$ and call 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 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 chase$_T(I)$ to $J$ that is the identity on adom($I$).

Note that if $T$ is a set of frontier-one TGDs, then for any database $D$ the instance chase$_T(D)$ can be obtained from $D$ by ‘glueing’ a (potentially infinite) instance onto each constant $c \in$ adom($D$). We denote this instance with chase$_T(D)|^i$. 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 $\bar{c} \in$ 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 chase$_T(D)$ with $h(\bar{c}) = \bar{c}$. 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 $\Sigma_D, \Sigma_Q$ be schemas called the data schema and query schema. Then

- $T_2$ is $\Sigma_D, \Sigma_Q$-hom-conservative over $T_1$, written $T_1 \models_{\Sigma_D, \Sigma_Q} \Sigma_Q$-hom chase$_{T_2}(D)$ to chase$_{T_1}(D)$ for all $\Sigma_D$-databases $D$;
- $T_2$ is $\Sigma_D, \Sigma_Q$-CQ-conservative over $T_1$, written $T_1 \models_{\Sigma_D, \Sigma_Q} \Sigma_Q$-c chase$_{T_2}(D)$ to chase$_{T_1}(D)$ for all $\Sigma_D$-databases $D$ and all CQs $q$ over schema $\Sigma_Q$.
- $T_1$ is $\Sigma_D, \Sigma_Q$-hom-trivial if $T_1 \models_{\Sigma_D, \Sigma_Q} \Sigma_Q$-hom-conservative over the empty set of TGDs, and likewise for $\Sigma_D, \Sigma_Q$-CQ-triviality.

It is easy to see that logical entailment $T_1 \models T_2$ implies $T_1 \models_{\Sigma_D, \Sigma_Q} T_2$ for all schemas $\Sigma_D$ and $\Sigma_Q$, and that $\Sigma_D, \Sigma_Q$-hom-conservativity implies $\Sigma_D, \Sigma_Q$-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:

- $T_1 = \{ A(x) \rightarrow \exists y S(x, y), B(y), B(x) \rightarrow \exists y R(x, y), B(y) \}$
- $T_2 = \{ A(x) \rightarrow \exists y S(x, y), B(y), B(x) \rightarrow \exists y R(x, y), B(y) \}$.  

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

However, $\Sigma_D, \Sigma_Q$-hom-conservativity is equivalent to $\Sigma_D, \Sigma_Q$-CQ-conservativity with infinitary CQs. We refrain from making this precise and instead consider the converse, that is, $\Sigma_D, \Sigma_Q$-CQ-conservativity is equivalent to $\Sigma_D, \Sigma_Q$-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 $\Sigma$ be a schema. We write $I_1 \rightarrow I_2$ if for every induced subinstance $I$ of $I_1$ with $|\text{adom}(I)| \leq n$, there is a database-preserving $\Sigma$-homomorphism from $I$ to $I_2$. We further write $I_1 \rightarrow I_2$ if $I_1 \rightarrow I_2$ for all $n \geq 1$.

**Theorem 1.** Let $T_1$ and $T_2$ be sets of TGDs and $\Sigma_D, \Sigma_Q$ schemas. Then $T_1 \models_{\Sigma_D, \Sigma_Q} T_2$ iff chase$_{T_2}(D)$ to chase$_{T_1}(D)$.

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

**Lemma 2.** Let $T_1, T_2$ be sets of TGDs and $\Sigma_D, \Sigma_Q$ schemas. Then $T_1$ and $T_2$ are $\Sigma_D, \Sigma_Q$-hom-trivial if and only if they are $\Sigma_D, \Sigma_Q$-CQ-trivial.

Because of Lemma 2, we from now on disregard $\Sigma_D, \Sigma_Q$-CQ-triviality and refer to $\Sigma_D, \Sigma_Q$-hom-triviality simply as $\Sigma_D, \Sigma_Q$-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 $\Sigma_D, \Sigma_Q$, whether $T_2$ is $\Sigma_D, \Sigma_Q$-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 $\Sigma$ be a schema. If $I_1 \rightarrow I_2$, then $I_1 \rightarrow I_2$.

We sketch the proof of Lemma 3, details are in the appendix. If $I_1 \rightarrow I_2$, then we find database-preserving $\Sigma$-homomorphisms $h_1, h_2, \ldots$ from finite subinstances $J_1 \subseteq J_2 \subseteq \ldots$ of $I_1$ to $I_2$ such that $I_1 = \bigcup_{i \geq 1} J_i$.

If $h_1, h_2, \ldots$ 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 $\Sigma$-homomorphism that witnesses $I_1 \rightarrow \Sigma I_2$. If this is not the case, however, we can still manipulate $h_1, h_2, \ldots$ into a compatible sequence $g_1, g_2, \ldots$ 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
3. triviality for guarded TGDs.

1. hom-conservativity for linear TGDs;

Attaining Point 3 requires a non-trivial modification of the Theorem 2. The following problems are undecidable:

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

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. CO-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 $T_{\text{myth}}$. It comprises three TGDs:

- $\text{Encounter}(p, t) \rightarrow \exists p', c, t' \ M(p, p', c, t', t)$
- $M(p, p', c, t', t) \rightarrow \exists p', c', t'' \ M(p', p'', c', t''', t')$
- $M(p, p', c, t', t) \rightarrow \text{Pyramus}(p, p'), \text{Thisbe}(t, t'),$
  \hspace{1cm} \text{Channel}(c, p'), \text{Channel}(c, t').$

Now consider the database $D = \{\text{Encounter}(c_0, c'_0)\}$. The instance chase$_{T_{\text{myth}}}(D)$, 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 $T_{\text{myth}}$ are linear. As $\Sigma_Q$, we will use the set of relation symbols in $T_{\text{myth}}$ except $M$, plus a unary relation symbol $\text{Mouth}$. We advise the reader to not worry about the schema $\Sigma_D$ at this point (it will actually be empty).

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

- 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 $\text{Thisbe}(a, e_1)$ for each non-bridge constant $a$ on each of the Thisbe-paths and there is $\text{Pyramus}(a, e_2)$ for each non-bridge constant $a$ on each of the Pyramus-paths. There are also $\text{Thisbe}(a, e_2)$ and $\text{Pyramus}(a, e_1)$ for each bridge constant $a$. In addition (and not in Figure 2), there are $\text{Pyramus}(e_1, e_i)$ and $\text{Thisbe}(e_i, e_1)$ for $i \in \{1, 2\}$.
- For each worldly constant $a$, there is $\text{Channel}(c, a)$. Moreover, there are facts $\text{Channel}(e_i, e_i)$, for $i \in \{1, 2\}$. These facts are not shown in Figure 2.
- There are $\text{Encounter}(b_{n-1}, b_n)$ and $\text{Mouth}(b_0)$.

It is easy to see that chase$_{T_{\text{myth}}}(\text{River}_\kappa)$ is obtained from River$_\kappa$ by adding a copy of the instance shown in Figure 1, gluing the Encounter fact to the Encounter fact in River$_\kappa$ (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 $\Sigma_Q$-homomorphism from chase$_{T_{\text{myth}}}(\text{River}_\kappa)$ to River$_\kappa$ 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 $\gamma, \alpha_0, \beta_0, \ldots, \alpha_{n-1}, \beta_{n-1}$ be positive integers such that $\beta_k/\gamma$ and $\beta_k | \alpha_k$ for $0 \leq k < \gamma$ where as usual ‘|’ denotes divisibility without remainder. For a positive integer $n$, define $F(n)$ by setting $F(n) = n\alpha_k/\beta_k$ for $k = n \mod \gamma$. Thus, the remainder of $n$ when dividing by $\gamma$ determines the pair $(\alpha_k, \beta_k)$ used to compute the value $F(n)$. Note that due to the two divisibility conditions, the range of $F$ contains only positive integers.

The function $F$ is called the Conway function defined by $\gamma, \alpha_0, \beta_0, \ldots, \alpha_{n-1}, \beta_{n-1}$. We say that $F$ stops if there exists an $n \in \mathbb{N}$ such that $F(n(2)) = 1$, where $F(n)$ is $F$ 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 $\gamma, \alpha_0, \beta_0, \ldots, \alpha_{n-1}, \beta_{n-1}$ stops.

Take a sequence $\gamma, \alpha_0, \beta_0, \ldots, \alpha_{n-1}, \beta_{n-1}$ defining a Conway function $F$. 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 $\Sigma_D, \Sigma_Q$ such that $F$ does not stop if and only if $T_2 = \Sigma_D, \Sigma_Q$-hom-conservative over $T_1$ if and only if $T_2 = \Sigma_D, \Sigma_Q$-CQ-conservative over $T_1$. We assume without loss of generality that $F(1) = 1$ and $F(2) = 3$.

4.3 The Reduction

We say that $\kappa = [p_1, \ldots, p_n, [t_1, \ldots, t_n]]$ (or $\text{River}_x$) is

- locally correct if the following conditions hold:
  1. $p_1 = 2$ and $p_n = 1$;
  2. $F(p_i) = t_i$ for $1 \leq i < n$;

- correct if it is locally correct and $t_i = p_{i+1}$ for $1 \leq i < n$.

The database $\text{River}_x$ 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 $F(1) = 1$).

Clearly, $F$ does not stop if and only if every locally correct $\text{River}_x$ is incorrect, and by Observation 1 this is the case if and only if for each locally correct sequence $\kappa$ there exists a database-preserving $\Sigma_Q$-homomorphism from $\text{chase}_T_{\text{proj}}(\text{River}_x)$ to $\text{River}_x$.

Now the plan is as follows. Take $\Sigma_D = \emptyset$. We define $T_1$ such that $\text{chase}_{T_1}(\emptyset)$ is the ‘disjoint union’ of all locally correct databases $\text{River}_x$. Our $T_2$ will be the union of $T_1$ and $T_{\text{proj}}$. 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 $T_{\text{rec}}$ and $T_{\text{proj}}$. As intended, $T_1$ generates the union of all locally correct databases $\text{River}_x$. 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 $T_{\text{rec}}$ will not mention symbols from $\Sigma_Q$. They instead use a schema $\Sigma_F$ that consists of high arity relation symbols used as construction aids. We later use $T_{\text{proj}}$ to relate these symbols to those in $\Sigma_Q$. More precisely, $\Sigma_F$ contains relation symbols $\text{Start}$ of arity 8, $\text{Bridge}$ of arity 4, $\text{WH}_k^1$ (for WorkHorse) of arity $\alpha_k + \beta_k + 5$ for $0 \leq k < \gamma$, and $\text{BH}_k$ (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 $v_1, c_1, c_2$. 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:

$$\Rightarrow \exists b_0, x_1, y_1, z_1, \beta_1 \text{ Start}(\dagger, b_0, x_1, y_1, z_1, b_1).$$

Later, $T_{\text{proj}}$ will generate a Pyramus-path from $b_1$ via $x_1$ to $b_0$ and a Thisbe-path from $b_1$ via $y_1$ and $y_2$ 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 $F(2) = 3$. We need to know that $b_1$ is a bridge:

$$\text{Start}(\dagger, b_0, x_1, y_1, z_1, b_1) \rightarrow \text{Bridge}(\dagger, b_1).$$

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

$\text{Bridge}(\dagger, b) \rightarrow \exists x_1, \ldots, x_{\beta_k}, y_1, \ldots, y_{\alpha_k}$

$\text{WH}_k^1(\dagger, b, x_1, \ldots, x_{\beta_k}, y_1, \ldots, y_{\alpha_k})$

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

$\text{WH}_k(\dagger, x_0, x_1, \ldots, x_{\beta_k}, y_0, y_1, \ldots, y_{\alpha_k})$

$\rightarrow \exists z_1, \ldots, z_{\beta_k}, u_1, \ldots, u_{\alpha_k}$

$\text{WH}_k^{\dagger, \gamma}(\dagger, z_1, \ldots, z_{\beta_k}, y_0, y_1, \ldots, y_{\alpha_k})$

$\text{WH}_k(\dagger, c_0, c_1, \ldots, x_{\beta_k}, y_0, y_1, \ldots, y_{\alpha_k})$ promises to generate, via $T_{\text{proj}}$, a Pyramus-path of length $\beta_k$ from $x_{\beta_k}$ to $y_0$ and a Thisbe-path of length $\alpha_k$ from $y_{\alpha_k}$ to $y_0$. The above two rules thus patiently produce Pyramus- and Thisbe-paths that lead to $b$. The superscript $\dagger$ remembers how many Pyramus-edges have been produced since the last bridge, modulo $\gamma$, and the subscript $-k$ 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 $n \mod \gamma = k$.

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

$\text{WH}_k^{\gamma, \dagger}(\dagger, x_{\beta_k}, 1, \ldots, z_{\beta_k} - 1, b, y_{\alpha_k}, u_1, \ldots, u_{\alpha_k} - 1, b) \rightarrow \text{Bridge}(\dagger, b).$

In the first rule above, relation $\text{WH}_k^{\gamma, \dagger}$ indicates that we have seen $m$ Pyramus-edges, for some $m$ with $m \mod \gamma = k - \beta_k$, and that $\text{BH}_k$ will generate $\beta_k$ 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 $F(n) = n\alpha_k/\beta_k$. Thus, Point 2 of local correctness is satisfied.

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

$\text{Bridge}(\dagger, b) \rightarrow \exists b' \text{ End}(\dagger, b, b')$
This will generate direct Pyramus- and Thisbe-edges from $b'$ to $b$ (recall that $f(1) = 1$).

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

$$\text{Start}(\dagger, b_0, x_1, y_1, y_2, b_1) \rightarrow$$

$$\text{Mouth}(b_0),$$

$$\text{Pyramus}(x_1, b_0), \text{Pyramus}(b_1, x_1), \text{Pyramus}(x_2, e_2),$$

$$\text{Thisbe}(y_1, b_0), \text{Thisbe}(y_2, y_1), \text{Thisbe}(b_1, y_2),$$

$$\text{Thisbe}(y_1, e_1), \text{Thisbe}(y_2, e_1),$$

$$\text{Channel}(c, x_1), \text{Channel}(c, y_1), \text{Channel}(c, y_2),$$

$$\text{Channel}(c_1, e_1), \text{Pyramus}(e_1, e_1), \text{Thisbe}(e_1, e_1)$$

$$\text{Channel}(e_2, e_2), \text{Pyramus}(e_2, e_2), \text{Thisbe}(e_2, e_2).$$

The rules for $WH_k^i$ are then as expected:

$$WH_k^i(\dagger, x_0, x_1, \ldots, x_{\beta_k}, y_0, y_1, \ldots, y_{\alpha_k}) \rightarrow$$

$$\text{Pyramus}(x_1, x_0), \ldots, \text{Pyramus}(x_{\beta_k}, x_{\beta_k-1}),$$

$$\text{Pyramus}(x_1, e_2), \ldots, \text{Pyramus}(x_{\beta_k}, e_2),$$

$$\text{Thisbe}(y_1, y_0), \ldots, \text{Thisbe}(y_{\alpha_k}, y_{\alpha_k-1}),$$

$$\text{Thisbe}(y_1, e_1), \ldots, \text{Thisbe}(y_{\alpha_k}, e_1),$$

$$\text{Channel}(c, x_1), \ldots, \text{Channel}(c, x_{\beta_k}),$$

$$\text{Channel}(c_1, y_1), \ldots, \text{Channel}(c, y_{\alpha_k}).$$

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

$$\text{Bridge}(\dagger, b) \rightarrow \text{Channel}(c, b), \text{Pyramus}(b, e_1), \text{Thisbe}(b, e_2)$$

$$\text{End}(\dagger, b, b') \rightarrow \text{Pyramus}(b', b), \text{Thisbe}(b', b),$$

$$\text{Encounter}(b, b').$$

In the appendix, we show that:

**Lemma 4.** $F$ does not stop iff $T_2 = T_1 \cup T_{myth}$ is $\Sigma_Q, \Sigma_D$-hom-conservative over $T_1 = T_{rec} \cup T_{proj}$.

This establishes Point 1 of Theorem 2. For the “if” direction, one shows that if $\text{chase}_{T_2}(0) \rightarrow_{\Sigma_Q} \text{chase}_F(0)$, then every locally correct river is incorrect, and thus $F$ stops. Since rivers may be long, but are finite, it actually suffices that $\text{chase}_{T_2}(0) \rightarrow_{\Sigma_{Q}} \text{lim}_0 \text{chase}_T(0)$ for $F$ 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 $F$ stops if and only if there is a locally correct River$_a$ that is correct, and that River$_a$ is correct if there is no database-preserving $\Sigma_Q$-homomorphism from $\text{chase}_{T_{myth}}(\text{River}_a)$ to River$_a$. We use the database $D$ to guess a River$_a$ 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 Encounter fact of that river. Details are in the appendix.

### 5. Triviality for Linear TGDs

We show that for linear TGDs, $\Sigma_D, \Sigma_Q$-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 description logic DL-Lite (Konev et al. 2011).

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

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

**Lemma 6.** Let $T$ be a set of linear TGDs and $D$ a singleton database. Then $\text{chase}_T(D) \not{\rightarrow}_{\Sigma_Q} D$ implies that there is a connected database $C \subseteq \text{chase}_T(D)$ 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 $\Sigma_D$ and $\Sigma_Q$, all we have to do is iterate over all singleton $\Sigma_D$-databases $D$ and over all $C \subseteq \text{chase}_T(D)$ 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

$$q_D \rightarrow \exists x \exists y \exists z R(x, y), R(y, z)$$

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 $\Sigma_D$-database $\{R(c, c')\}$. Based on this, it is easy to verify that $T'$ is $\Sigma_D, \Sigma_Q$-trivial iff $D, T \not{\models} \exists x A(x)$. 

200
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 C\(\mathcal{L}\)T, 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 \(\Sigma\)-instance tree is a triple \(T = (V, E, B)\) with \((V, E)\) a directed tree and \(B\) a function that assigns a \(\Sigma\)-database \(B(v)\) to every \(v \in V\) such that the following conditions hold:

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

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

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 \(\Sigma_D\) and \(\Sigma_Q\) schemas. Let \(k\) be the body width of \(T_1\). Then the following are equivalent:

1. \(T_1 \models_{\Sigma_D, \Sigma_Q} T_2\);
2. \(\text{chase}_{T_2}(D) \rightarrow_{\Sigma_Q} \text{chase}_{T_1}(D)\), for all tree-like \(\Sigma_D\)-databases \(D\) of width at most \(k\).

The “1 \(\Rightarrow\) 2”-direction is a direct consequence of the definition of hom-conservativity. For the “2 \(\Rightarrow\) 1”-direction, let \(D\) be a \(\Sigma_D\)-database witnessing \(T_1 \models_{\Sigma_D, \Sigma_Q} T_2\), that is, \(\text{chase}_{T_2}(D) \rightarrow_{\Sigma_Q} \text{chase}_{T_1}(D)\). We show in the appendix that the unraveling of \(U\) of \(D\) into a (potentially infinite) tree-like \(\Sigma_D\)-instance of width \(k\) also satisfies \(\text{chase}_{T_2}(U) \rightarrow_{\Sigma_Q} \text{chase}_{T_1}(U)\). Compactness then yields a finite subset \(U'\) of \(U\) that still satisfies \(\text{chase}_{T_2}(U') \rightarrow_{\Sigma_Q} \text{chase}_{T_1}(U')\). We show in the appendix how Theorem 6 can be used to reduce \(\Sigma_D\), \(\Sigma_Q\)-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, \(\Sigma_D\) and \(\Sigma_Q\) schemas, \(k\) the body width of \(T_1\), and \(\ell\) 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 \(\text{chase}_{T_1}(D) \rightarrow_{\Sigma_Q} I_0\). If such an \(I_0\) is found, then \(\text{chase}_{T_2}(D) \rightarrow_{\Sigma_Q} \text{chase}_{T_1}(D)\) because \(\text{chase}_{T_1}(D) \rightarrow I_0\). The converse is also true since \(\text{chase}_{T_1}(D)\) is tree-like of width \(\max\{k, \ell\}\). In fact, the instance \(\text{chase}_{T_1}(D)\) that the chase generates below each \(c \in \text{adom}(D)\) (see Section 2) is tree-like of width \(\ell\).

Since our homomorphisms are database-preserving and \(T_2\) is a set of frontier-one TGDs, \(\text{chase}_{T_2}(D) \rightarrow_{\Sigma_Q} I_0\) if and only if there is a \(c \in \text{adom}(D)\) such that \(\text{chase}_{T_2}(D) \rightarrow_{\Sigma_Q} I_0\). 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 bodyCQ(\(T\)) to denote the set of unary or Boolean CQs that can be obtained by starting with the Boolean CQ \(\exists y \exists z \phi(y, z)\) with \(\phi(y, z)\) 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 \(\ell \subseteq \text{bodyCQ}(T)\) such that for some instance \(I\) that is a model of \(T\) and some \(c \in \text{adom}(I)\).

1. \(q(x) \in \ell\) if \(c \in q(I)\) for all unary \(q(x) \in \text{bodyCQ}(T)\) and \(\ell\).
2. \(c \in \ell\) if \(q \in \ell \models I \models q\) for all Boolean \(q \in \text{bodyCQ}(T)\).

We then also use \(\text{tp}_T(I, c)\) to denote \(t\). We assume that every type contains the additional formula \(\text{true}(x)\) (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. TP(\(T\)) is the set of all T-types. Note that the number of types is double exponential in \(|T|\).

The type \(\text{tp}_T(\text{chase}_{T_1}(D), c)\) 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 \(c \in \text{adom}(I)\). Then \(\text{chase}_{T_1}(I) \models_{\ell} c\) and \(\text{chase}_{T}(I) \models_{\ell} c\) are homomorphically equivalent, where \(I\) is obtained from \(\text{tp}_T(\text{chase}_{T_1}(I), c)\) by replacing the free variable \(x\) with \(c\).

The proof of Lemma 7 is straightforward by reproducing chase steps from the construction of \(\text{chase}_{T_1}(I)\) in \(\text{chase}_{T}(I)\) and vice versa. Details are omitted.

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

6.2 Deciding CQ-Conservativity

We start with showing that, also for deciding CQ-$\Sigma$-conservative over $\Sigma_D$ and $\Sigma_Q$ schemas. Let $k$ be the body width of $T$. Then the following are equivalent:

1. $T_1 \models_{\Sigma_D, \Sigma_Q} T_2$;
2. $\forall_{T_2}(D) \subseteq \forall_{T_1}(D)$, for all tree-like $\Sigma_D$-databases $D$ of width at most $k$ and connected $\forall_{\Sigma_Q}$-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, $\Sigma_D$ and $\Sigma_Q$ schemas, and $k$ the body width of $T_1$. Then the following are equivalent:

1. $T_1 \models_{\Sigma_D, \Sigma_Q} T_2$;
2. $\forall_{T_2}(D) \rightarrow_{\Sigma_Q} \forall_{T_1}(D)$, for all tree-like $\Sigma_D$-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 $\Sigma_Q$-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 $\text{chase}_T(D)|^{\Sigma_Q}$ we denote the union of all maximally $\Sigma$-connected components of $\text{chase}_T(D)$ that contain at least one constant from $\text{adom}(D)$.

Theorem 9. Let $T_1$ and $T_2$ be sets of frontier-one TGDs, $\Sigma_D$ and $\Sigma_Q$ schemas, and $k$ the body width of $T_1$. Then $T_1 \models_{\Sigma_D, \Sigma_Q} T_2$ if and only if all tree-like $\Sigma_D$-databases $D$ of width at most $k$, the following holds:

1. $\forall_{T_2}(D) \rightarrow_{\Sigma_Q} \forall_{T_1}(D)$;
2. for all maximally $\Sigma_Q$-connected components $I$ of $\forall_{T_2}(D) \rightarrow_{\Sigma_Q} \forall_{T_1}(D)$, one of the following holds:
   (a) $I \not\rightarrow_{\Sigma_Q} \forall_{T_1}(D)$;
   (b) $I \rightarrow_{\Sigma_Q} \forall_{T_1}(D)$, for some $c \in \text{adom}(D)$.

The subsequent example illustrates the theorem.

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

For Point 2, observe that $\text{chase}_{T_2}(D) \rightarrow_{\Sigma_Q} \forall_{T_1}(D)$ contains only one maximally $\Sigma_Q$-connected component, which is of the form

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

Moreover, $I \not\rightarrow_{\Sigma_Q} \forall_{T_1}(D)$ and thus Point 2(b) is satisfied. Point 2(a) is not satisfied since $I \not\rightarrow_{\Sigma_Q} \forall_{T_1}(D)$.

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 homomorphism limits. 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 $\forall_{T_1}(D)$ 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 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 $\mu: \text{adom}(D) \rightarrow \text{TP}(T)$. We associate $A$ with a database $D_A$ that is obtained by starting with $D$ and then adding, for each $c \in \text{adom}(D)$, a disjoint copy $c'$ of the type $\mu(c)$ viewed as a database and glueing the copy of $c$ 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 $\Sigma$-head fragment of $T_2$ is a $T_2$-labeled database $(F, \mu)$ such that $F$ can be obtained by choosing a TGD $\phi(x, y) = \exists z \psi(x, z) \in T_2$ and taking a maximally $\Sigma$-connected component of $\psi$ 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 $\Sigma_Q$-connected component of $\forall_{T_2}(D) \rightarrow_{\Sigma_Q} \forall_{T_1}(D)$ as in Point 2 of Theorem 9. Then for some labeled $\Sigma_Q$-head fragment $A = (F, \mu)$ of $T_2$.

1. $\forall_{T_2}(D) \models_{\Sigma_Q} q_A$ and
2. $I$ is homomorphically equivalent to $\forall_{T_2}(D) \models_{\Sigma_Q} q_A$.

Clearly, the number of labeled $\Sigma$-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 $I \rightarrow_{\Sigma_Q} \forall_{T_1}(D)$ in Point 2(b) of Theorem 9 does not depend on $D$: there is at most one such check for every labeled $\Sigma$-head fragment of $T_2$ and every $T_1$-type. We can do all these checks in a preprocessing step, before starting to build 2ATA for CQ-conservativity that implement the characterization provided by Theorem 9. Whenever the 2ATA needs to carry out a check $I \rightarrow_{\Sigma_Q}$
chase_{T_j}(D)\downarrow T_j$ 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 $I \rightarrow \exists Q \operatorname{chase}_{T_j}(D)\downarrow T_j$ 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 $\Sigma$, a labeled $\Sigma$-head fragment $A = (D, \mu)$ for $T_2$, and a $T_1$-type $\acute{t}$, it can be decided in time triple exponential in $|T_1| + |T_2|$ whether $\operatorname{chase}_{T_j}(D)\downarrow \exists Q \operatorname{chase}_{T_j}(\acute{t})$.

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 $T_1$, $T_2$, $\Sigma$, $A = (D, \mu)$, and $\acute{t}$ 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 $q_A^c$, for every $c \in \operatorname{dom}(D)$, 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 chase_{T_j}(\acute{t}) with homomorphisms into a class of instances $\mathcal{R}(T_1, \acute{t})$ whose disjoint union should be viewed as a relaxation of chase_{T_j}(\acute{t}). In particular, this relaxation admits a homomorphism limit to chase_{T_j}(\acute{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.\footnote{Neither in the directed nor in the undirected sense, which is equivalent if every node has at most one incoming edge.} 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 $T = (V, E, B, \mu)$ with $T' = (V, E, B)$ an instance tree (based on a directed pseudo-tree) and $\mu : \operatorname{dom}(I_{T'}) \rightarrow TP(T_1)$ a function that assigns a $T_1$-type to every element in $I_{T'}$. For $v \in V$, we use $\mu_v$ to denote the restriction of $\mu$ to $\operatorname{dom}(B(v))$. Moreover, we set $I_T = I_{T'}$. We say that $T$ is $\acute{t}$-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 $\{\text{true}(c_0)\}$, and $\mu(c_0) = \acute{t}$;
   (b) there is a TGD $\vartheta$ in $T_1$ such that $B(v)$ is isomorphic to the head of $\vartheta$ and $T_1 \models q(B(v), \mu_v)$;

2. for every $(u, v) \in E$ such that $B(u) \cap B(v)$ contains a (single) constant $c$, we have $\mu_c(c), T_1 \models q'(B(v), \mu_v)(x)$.

That is: the constant $x$ from the type $\mu_c(c)$ viewed as a database is an answer to the unary CQ $q'(B(v), \mu_v)$ w.r.t. $T_1$.

The announced class $\mathcal{R}(T_1, \acute{t})$ consists of all instances $I$ such that $I = I_T$ for some $\acute{t}$-proper $T_1$-labeled instance tree $T$. It is easy to see that chase_{T_j}(\acute{t}) \in \mathcal{R}(T_1, \acute{t})$ as there is a $\acute{t}$-proper $T_1$-labeled instance tree $T$ such that $I_T = \operatorname{chase}_{T_j}(\acute{t})$. However, there are also instances $I \in \mathcal{R}(T_1, \acute{t})$ that do not admit a homomorphism to chase(\acute{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 $\acute{t} = \operatorname{tp}_{T_1}(\operatorname{chase}_{T_1}(D), c)$, that is, $\acute{t} = \{A(x), \exists x A(x), \exists x B(x)\}$. Then $I \nvdash \operatorname{chase}_{T_2}(\acute{t})$.

However, we find a $\acute{t}$-proper $T_1$-labeled instance tree $T = (V, E, B, \mu)$ such that $I = I_T$.

We may construct $T$ 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 $T$ is $\acute{t}$-proper and satisfies $I_T = I$. 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 chase_{T_j}(\acute{t}) with instances from $\mathcal{R}(T_1, \acute{t})$, we may also replace homomorphism limits with homomorphisms.

**Lemma 9.** Let $I$ be a countable $\Sigma$-connected instance such that $\operatorname{dom}(I)$ contains only nulls. Then $I \rightarrow \exists \Sigma \operatorname{chase}_{T_j}(\acute{t})$ iff there is an $\acute{t} \in \mathcal{R}(T_1, \acute{t})$ with $I = \acute{t} \rightarrow \hat{I}$.

In the proof of Lemma 9, the laborious direction is ‘only if’, where one assumes that $I \rightarrow \exists \Sigma \operatorname{chase}_{T_j}(\acute{t})$ 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 $\operatorname{chase}_{T_j}(\acute{t})$, $i \geq 1$, to identify the desired instance $I \in \mathcal{R}(T_1, \acute{t})$. 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 $I \in \mathcal{R}(T_1, \acute{t})$ that admits a $\Sigma$-homomorphism from $\operatorname{chase}_{T_j}(D)\downarrow \exists Q \operatorname{chase}_{T_j}(\acute{t})$.

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 $3\text{ExpTime}$-complete. Note that in the description logic $\mathcal{ELI}$, they are $2\text{ExpTime}$-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 (Cali, 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


