Polynomial Combined Rewritings for Existential Rules

Georg Gottlob[1] and **Marco Manna**[2] and **Andreas Pieris**[1]

[1]Department of Computer Science, University of Oxford, UK
[2]Department of Mathematics and Computer Science, University of Calabria, Italy

{georg.gottlob,andreas.pieris}@cs.ox.ac.uk
manna@mat.unical.it

Abstract

We consider the scenario of ontology-based data access where a conjunctive query is evaluated against a database enriched with intensional knowledge via an ontology. It is generally accepted that true scalability of query answering in this setting can only be achieved by using standard relational database management systems (RDBMSs). An approach to query answering that enables the use of RDBMSs is the so-called polynomial combined approach. We investigate this approach for the main guarded- and sticky-based classes of existential rules, and we highlight the assumptions on the underlying schema which are sufficient for the polynomial combined first-order rewritability of those classes. To the best of our knowledge, this is the first work which explicitly studies the polynomial combined approach for existential rules.

1 Introduction

An ontology is an explicit specification of a conceptualization of an area of interest. One of the main applications of ontologies nowadays is in *ontology-based data access (OBDA)* (Poggi *et al.* 2008), where they are used to enrich the extensional data with intensional knowledge. In this setting, Description Logics (DLs) and rule-based formalisms such as existential rules are popular ontology languages, while conjunctive queries (CQs) are used as a vital querying tool. Therefore, efficient approaches to CQ answering over such languages are of great importance.

It is widely accepted that true scalability in OBDA can only be achieved by exploiting standard relational database management systems (RDBMSs), which provide mature and efficient technology for answering queries. A significant step forward in this direction was the introduction of the DL-Lite family of DLs (Calvanese *et al.* 2007; Poggi *et al.* 2008) — the logical underpinning of the OWL 2 QL profile of OWL 2 — and the proposal of the first practical approach to OBDA via query rewriting. Most of the languages of the DL-Lite family have been designed with the aim of being *first-order rewritable*, namely a TBox $\mathcal{T}$ can be incorporated together with a given CQ q into a first-order query $q_{\mathcal{T}}$ such that, for every ABox $\mathcal{A}$, $q_{\mathcal{T}}$ evaluated over $\mathcal{A}$ yields exactly the same result as q evaluated against $\mathcal{A}$ and $\mathcal{T}$. Clearly, $q_{\mathcal{T}}$ can be translated into a standard SQL query, and then submitted to the RDBMS holding $\mathcal{A}$ (as a relational database), where it is evaluated and optimized in the usual way.

Although first-order rewritability is a most desirable property, it suffers from two serious shortcomings: *(i)* rewriting algorithms, in general, generate from a reasonably sized CQ an exponentially sized SQL query (and even for simple cases such as OWL 2 QL this blowup is unavoidable (Kikot *et al.* 2012)), which can be prohibitive for efficient execution by an RDBMS, and *(ii)* it applies only to lightweight ontology languages for which the data complexity of CQ answering is at most LOGSPACE, and thus useful formalisms with PTIME-hard data complexity are excluded a priori.

A more refined approach to query answering, which overcomes the inherent limitations of first-order rewritability, is the *polynomial combined approach* (Lutz et al. 2009), that allows the encoding of the consequences of the ontology in the given database (or ABox). This approach applies to languages which are *polynomially combined first-order rewritable*, i.e., for every CQ q, database D, and ontology Σ, it is possible to rewrite in *polynomial time (i)* q and Σ into a first-order query q_{Σ}, and *(ii)* D and Σ into a database D_{Σ}, in such a way that the answer to q_{Σ} over D_{Σ} is the same as the answer to q over D and Σ.

The polynomial combined approach has been successfully applied to the DL $\mathcal{ELH}^{dr}_{\bot}$ (Lutz *et al.* 2009), i.e., the extension of $\mathcal{EL}$ with the inconsistent concept, role inclusions, and domain and range restrictions, and also to DL-Lite$^{\mathcal{N}}_{horn}$ (Kontchakov *et al.* 2010; 2011), one the most commonly used DL-Lite formalisms without role inclusions. Notice that DL-Lite$^{(\mathcal{HN})}_{horn}$, the extension DL-Lite$^{\mathcal{N}}_{horn}$ with role hierarchies, has been also considered in (Kontchakov *et al.* 2010) but, while the new database can be constructed in polynomial time, the rewritten query is of exponential size in the number of roles occurring in the ontology.

Although the polynomial combined approach has been already applied to several DLs, this is not the case for existential (a.k.a. Datalog$^{\exists}$) rules of the form $\forall \mathbf{X}\, \varphi(\mathbf{X}) \rightarrow \exists \mathbf{Z} p(\mathbf{X}, \mathbf{Z})$. Notice that the investigation of expressive fragments of existential rules that can be employed as an alternative way to represent ontologies is currently a field of intense research (Baget *et al.* 2011a; 2011b; Krötzsch and Rudolph 2011; Calì *et al.* 2012a; 2012b; 2013; Leone *et al.* 2012). The only known cases of existential rules which are polyno-

Size	Arity	L	G	S	T
∞	∞	?	$\times$	$[\times]$	$\times$
∞	$d \geqslant 2$	$\checkmark$	$[\times]$	$\checkmark^\star$	$[\times]$
∞	$d \leqslant 1$	$\checkmark$	$\checkmark^\star$	$\checkmark^\star$	$\checkmark^\star$
$c \geqslant 1$	∞	?	$\times^\star$	$[\times]^\star$	$\times^\star$
$c \geqslant 1$	$d \geqslant 0$	$\checkmark$	$\checkmark^\star$	$\checkmark$	$\checkmark^\star$

Table 1: Polynomial combined first-order rewritablity of (L)inear, (G)uarded, (S)ticky and (T)ame rules. The symbol ∞ (resp., $c \geqslant i$ and $d \geqslant i$) asserts that the parameter under consideration is unbounded (resp., bounded by the constants c and d). The expression $d \leqslant 1$ implies that the arity is either 0 or 1. The symbol ? refers to an open problem. The symbol $\checkmark$(resp., $\times$) refers to a positive (resp., negative) case. The symbol $[\times]$ refers to a negative case, assuming that PSPACE $\neq$ EXPTIME. Our results are marked with $\star$.

mially combined first-order rewritable are the class of *linear* rules when the arity of the schema is bounded, and the class of *sticky* rules when both the size and the arity of the schema are bounded. Those results are implicit in (Gottlob and Schwentick 2012); however, the goal of that paper was not the investigation of the polynomial combined approach for linear and sticky rules, but the construction of a non-recursive Datalog rewriting of polynomial size.

To the best of our knowledge, this is the first work which explicitly studies the combined approach for existential rules. This problem is considered, by the KR community, as an interesting research direction that must be investigated (Kontchakov *et al.* 2011). Towards this direction, we explore guarded- and sticky-based classes of existential rules. Both guardedness and stickiness are well-accepted paradigms. On the one hand, guarded rules form a robust language which captures important DLs such as DL-Lite and $\mathcal{EL}$ (Calì *et al.* 2012a) — a rule is *guarded* if it has a body-atom, called guard, which contains all the body-variables. On the other hand, sticky rules allow for joins in rule-bodies which are expressible only via non-guarded rules, and they are able to capture well-known data modeling constructs such as inclusion and multivalued dependencies (Calì *et al.* 2012b). Recently, a class called *tame*, which combines guardedness and stickiness by taming the interaction between the sticky rules and the guard atoms occurring in guarded rules, has been proposed (Gottlob *et al.* 2013).

It is well-known that the problem of evaluating a first-order query over a database is feasible in polynomial space. Therefore, the polynomial combined approach is applicable only to ontology languages for which the combined complexity of CQ answering is at most PSPACE (assuming that PSPACE $\neq$ EXPTIME). Since CQ answering under guarded and tame rules is 2EXPTIME-hard, and under sticky rules is EXPTIME-hard, it is interesting to investigate which assumptions on the underlying schema are sufficient for the polynomial combined first-order rewritability of the above classes. The techniques employed for $\mathcal{ELH}^{dr}_{\bot}$ and DL-Lite$^{\mathcal{N}}_{horn}$, although are quite insightful, leverage specificities of DLs, such as the limit to unary and binary predicates and the special form of DL axioms, so that it is not clear how they can be extended to general rule-based languages. Thus, we had to come up with novel techniques beyond the state of the art.

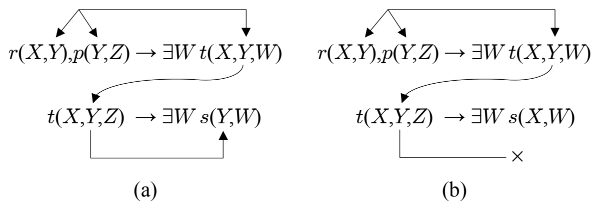
Our contributions can be summarized as follows:

- We first show that guarded rules are, in general, not polynomially combined first-order rewritable, even if the underlying schema contains a single predicate. The same holds even if we concentrate on unary and binary predicates (unless PSPACE = EXPTIME). Guarded rules are polynomially combined first-order rewritable if we consider only unary predicates, or when the size and the arity of the underlying schema are bounded by a constant.
- We show that sticky rules are, in general, not polynomially combined first-order rewritable, even if the underlying schema contains a single predicate (unless PSPACE = EXPTIME). Sticky rules are polynomially combined first-order rewritable if we consider schemas of bounded arity.
- Finally, we show that the results for guarded rules described above can be extended to tame rules.

In summary, we form an (almost) complete picture, depicted in Table 1, of the polynomial combined first-order rewritability for the main guarded- and sticky-based classes of existential rules. Notice that the case of linear rules, even when the schema contains a single predicate, is still open. We conjecture that linear rules are polynomially combined first-order rewritable, and we give some evidence for this in Section 7. At this point, let us clarify that guarded and tame rules are not first-order rewritable since the data complexity of CQ answering under those classes is PTIME-hard, even when the size and the arity of the underlying schema are bounded. Moreover, sticky rules are strictly more expressive than OWL 2 QL, and thus they are not polynomially first-order rewritable, even if the arity is fixed. Hence, our results enlarge the family of ontology languages which are suitable for OBDA purposes. Full proofs can be downloaded from *https://www.mat.unical.it/datalog-exists/pub/gmp14.pdf.*

2 Preliminaries

Technical Definitions. We define the following pairwise disjoint (infinite countable) sets: a set $\mathbf{C}$ of *constants*, a set $\mathbf{N}$ of *labeled nulls*, and a set $\mathbf{V}$ of regular *variables*. A *term* t is a constant, null, or variable. An *atom* has the form $p(t_1, \ldots, t_n)$, where p is an n-ary predicate, and $t_1, \ldots, t_n$ are terms. Let $arity(p)$ be the arity of the predicate p. For an atom $\underline{a}$, we denote $dom(\underline{a})$ and $var(\underline{a})$ the set of its terms and the set of its variables, respectively; these extend to sets of atoms. Conjunctions of atoms are often identified with the sets of their atoms. An *instance* I is a (possibly infinite) set of atoms $p(\mathbf{t})$, where $\mathbf{t}$ is a tuple of constants and nulls. Given a set of predicates $\mathcal{R}$, let $I_{|\mathcal{R}} = \{p(\mathbf{t}) \in I \mid p \in \mathcal{R}\}$, i.e., the restriction of I on $\mathcal{R}$. A *database* D is a finite instance such that $dom(D) \subset \mathbf{C}$. Given a set T of terms, we denote by $base(T, \mathcal{R})$ the set of atoms that can be formed using terms of T and predicates of $\mathcal{R}$. A *homomorphism* is a substitution $h : \mathbf{C} \cup \mathbf{N} \cup \mathbf{V} \rightarrow \mathbf{C} \cup \mathbf{N} \cup \mathbf{V}$ which is the identity on $\mathbf{C}$. Two sets of atoms A, A' are *homomorphically equivalent*, written $A \longleftrightarrow A'$, if there exist homomorphisms h and h' such that $h(A) \subseteq A'$ and $h'(A') \subseteq A$. We assume the reader is familiar with *conjunctive queries (CQs)*. We denote $body(q)$ the body of a query q. The answer to q over



$r(X,Y),p(Y,Z) \rightarrow \exists W\, t(X,Y,W)$
$t(X,Y,Z) \rightarrow \exists W\, s(Y,W)$
(a)
$r(X,Y),p(Y,Z) \rightarrow \exists W\, t(X,Y,W)$
$t(X,Y,Z) \rightarrow \exists W\, s(X,W)$
×
(b)

Figure 1: The Sticky Property.

an instance I is denoted $q(I)$. A Boolean CQ (BCQ) q has a positive answer over I, denoted $I \models q$, if $() \in q(I)$.

A *Datalog*$^{\exists}$ *rule* (or simply *rule*) σ is a constant-free first-order formula of the form $\forall \mathbf{X} \forall \mathbf{Y}\, \varphi(\mathbf{X},\mathbf{Y}) \rightarrow \exists \mathbf{Z}\, p(\mathbf{X},\mathbf{Z})$, where $\mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z} \subset \mathbf{V}$, $\varphi(\mathbf{X},\mathbf{Y})$ is a conjunction of atoms, and $p(\mathbf{X},\mathbf{Z})$ is a single atom; $\varphi(\mathbf{X},\mathbf{Y})$ is the *body* of σ, denoted $body(\sigma)$, while $p(\mathbf{X},\mathbf{Z})$ is the *head* of σ, denoted $head(\sigma)$. Given a set Σ of rules, $sch(\Sigma)$ is the set of predicates occurring in Σ, and $arity(\Sigma)$ is defined as $\max_{p \in sch(\Sigma)}\{arity(p)\}$. For brevity, we will omit the universal quantifiers in front of rules. An instance I satisfies σ, written $I \models \sigma$, if the following holds: whenever there exists a homomorphism h such that $h(\varphi(\mathbf{X},\mathbf{Y})) \subseteq I$, then there exists $h' \supseteq h_{|\mathbf{X}}$, where $h_{|\mathbf{X}}$ is the restriction of h on $\mathbf{X}$, such that $h'(p(\mathbf{X},\mathbf{Z})) \in I$; I satisfies a set Σ of rules, denoted $I \models \Sigma$, if I satisfies each σ of Σ. The *models* of a database D and a set Σ of rules, denoted $mods(D,\Sigma)$, is the set of instances $\{I \mid I \supseteq D \text{ and } I \models \Sigma\}$. The *answer* to a CQ q w.r.t. D and Σ is defined as the set of tuples $ans(q,D,\Sigma) = \bigcap_{I \in mods(D,\Sigma)}\{\mathbf{t} \mid \mathbf{t} \in q(I)\}$. The answer to a BCQ q is *positive*, denoted $D \cup \Sigma \models q$, if $ans(q,D,\Sigma) \neq \varnothing$. The problem of *CQ answering* is defined as follows: given a CQ q, a database D, a set Σ of rules, and a tuple of constants $\mathbf{t}$, decide whether $\mathbf{t} \in ans(q,D,\Sigma)$. In case q is a BCQ, the above problem is called *BCQ answering*. These decision problems are LOGSPACE-equivalent (implicit in (Chandra and Merlin 1977)), and we thus focus only on BCQ answering. For query answering purposes, we can assume, w.l.o.g., that both the BCQ q and the database D contain only predicates of $sch(\Sigma)$.

We are going to employ the *chase procedure*, which works on an instance through the so-called *chase rule* defined as follows. Consider an instance I, and a rule σ as defined above. Then, σ is *applicable* to I if there exists a homomorphism h such that $h(\varphi(\mathbf{X},\mathbf{Y})) \subseteq I$. Let $I' = I \cup \{h'(p(\mathbf{X},\mathbf{Z}))\}$, where $h' \supseteq h_{|\mathbf{X}}$ is such that $h'(Z)$ is a "fresh" null not occurring in I, for each $Z \in \mathbf{Z}$. The result of applying σ to I with h is I', and we write $I\langle\sigma,h\rangle I'$; in fact, $I\langle\sigma,h\rangle I'$ defines a single *chase step*. The chase algorithm for a database D and a set Σ of rules consists of an exhaustive application of chase steps in a fair fashion, which leads to a (possibly infinite) model of D and Σ, denoted $chase(D,\Sigma)$; for more details see, e.g., (Calì *et al.* 2012b). The instance $chase(D,\Sigma)$ is a *universal model* of D and Σ, i.e., for each $I \in mods(D,\Sigma)$, there exists a homomorphism that maps $chase(D,\Sigma)$ to I, and thus $D \cup \Sigma \models q$ iff $chase(D,\Sigma) \models q$, for each BCQ q (implicit in (Fagin *et al.* 2005)).

Fragments of Datalog$^{\exists}$. A rule σ is called *guarded* if there exists $\underline{a} \in body(\sigma)$ which contains all the variables occurring in $body(\sigma)$ (Calì *et al.* 2013). Conventionally, the leftmost such an atom is the *guard* of σ, denoted $guard(\sigma)$, while the non-guard atoms are called the *side* atoms of σ. The class of guarded Datalog$^{\exists}$, denoted GUARDED, is defined as the family of all possible sets of guarded rules. Given a set $\Sigma \in$ GUARDED, we denote by $sch(\Sigma)^{-}$ the predicates of Σ which occur in at least one side atom in Σ; clearly, $sch(\Sigma)^{-} \subseteq sch(\Sigma)$. Guarded rules with one body-atom are called *linear* (Calì *et al.* 2012a), and the corresponding class is denoted LINEAR.

Sticky sets of rules have been proposed in (Calì *et al.* 2012b). The central property of stickiness is that, during the chase, terms which are associated with body-variables that appear more than once (i.e., join variables) always are propagated (or "stick") to the inferred atoms. This is illustrated in Figure 1; the first case is sticky, while the second is not. The associated class is denoted STICKY.

Towards the identification of even more expressive decidable fragments of Datalog$^{\exists}$, a condition, called *tameness*, which allows the consolidation of guardedness with stickiness has been recently proposed (Gottlob *et al.* 2013). In particular, a set Σ of rules is called *tame* if it can be partitioned into $\{\Sigma[g],\Sigma[s]\}$, where $\Sigma[g]$ is a set of guarded rules and $\Sigma[s]$ is a sticky set of rules, and in addition, none of the rules of $\Sigma[s]$ "feeds" the guard atom of a guarded rule during the construction of the chase. In other words, whenever a guarded rule σ is applied with homomorphism h, then its guard must be mapped by h into an atom obtained from a guarded rule. The corresponding class is denoted TAME.

Fix a class $\mathbb{C} \in \{$GUARDED, LINEAR, STICKY, TAME$\}$. Given two integer constants $c > 0$ and $d \geqslant 0$, we define $\mathbb{C}[c,d] = \{\Sigma \in \mathbb{C} \mid |sch(\Sigma)| \leqslant c \text{ and } arity(\Sigma) \leqslant d\}$ as the subclass of $\mathbb{C}$ where the size and the maximum arity of the underlying schema are bounded by c and d, respectively. We define the subclass of $\mathbb{C}$ where the size of the underlying schema is bounded by c as $\mathbb{C}[c,\infty] = \bigcup_{j \geqslant 0} \mathbb{C}[c,j]$, and the subclass of $\mathbb{C}$ where the maximum arity of the underlying schema is bounded by d as $\mathbb{C}[\infty,d] = \bigcup_{i>0} \mathbb{C}[i,d]$.

3 The Polynomial Combined Approach

The formal definition of polynomial combined first-order rewritability, introduced in (Lutz *et al.* 2009), which is at the basis of the polynomial combined approach, is as follows:

Definition 1 *A Datalog*$^{\exists}$ *class* $\mathbb{C}$ *is* combined first-order rewritable *if, for every BCQ q, for every database D, and for every* $\Sigma \in \mathbb{C}$*, it is possible to rewrite:*

- *q and Σ, independently of D, into a first-order query q_{Σ},*
- *D and Σ, independently of q, into a database D_{Σ},*

such that $D \cup \Sigma \models q$ *iff* $D_{\Sigma} \models q_{\Sigma}$*. If* q_{Σ} *and* D_{Σ} *are constructible in polynomial time, then* $\mathbb{C}$ *is* polynomially combined (P-combined) first-order rewritable.

Combined Reducibility. To show that a Datalog$^{\exists}$ class is P-combined first-order rewritable, one may exploit the fact that existing formalisms already enjoy this property. Towards this direction, we introduce the notion of polynomial combined reducibility:

Definition 2 *Consider two Datalog$^\exists$ classes $\mathbb{C}$ and $\mathbb{C}'$. We say that $\mathbb{C}$ is* combined reducible *to $\mathbb{C}'$ if, for every BCQ q, database D, and $\Sigma \in \mathbb{C}$, we can construct:*

- *a BCQ q' independently of D,*
- *a database D' independently of q, and*
- *a set $\Sigma' \in \mathbb{C}'$ independently of q and D,*

such that $D \cup \Sigma \models q$ iff $D' \cup \Sigma' \models q'$. If q', D' and Σ' are constructible in polynomial time, then $\mathbb{C}$ is polynomially combined (P-combined) reducible *to $\mathbb{C}'$, and the employed reduction is called P-combined.*

Obviously, the set of P-combined first-order rewritable Datalog$^\exists$ classes is closed under P-combined reductions:

Theorem 3 *Consider two Datalog$^\exists$ classes $\mathbb{C}, \mathbb{C}'$. If $\mathbb{C}$ is P-combined reducible to $\mathbb{C}'$, and $\mathbb{C}'$ is P-combined first-order rewritable, then also $\mathbb{C}$ is P-combined first-order rewritable.*

The above result suggests the following: to show that a Datalog$^\exists$ class $\mathbb{C}$ is P-combined first-order rewritable, it suffices to reduce query answering under $\mathbb{C}$ to query answering under some P-combined first-order rewritable formalism via a P-combined reduction.

Polynomial Witness Property. A semantic property which is sufficient for P-combined first-order rewritability is the *polynomial witness property*, introduced in (Gottlob and Schwentick 2012). Roughly, a Datalog$^\exists$ class $\mathbb{C}$ enjoys this property if, whenever a query q is entailed by a database D and a set $\Sigma \in \mathbb{C}$, then q is already entailed by a finite part of $chase(D, \Sigma)$ (the witness) of polynomial size in q and Σ.

Definition 4 *A Datalog$^\exists$ class $\mathbb{C}$ enjoys the* polynomial witness property (PWP) *if there exists a polynomial f such that, for every BCQ q, database D, and $\Sigma \in \mathbb{C}$, $D \cup \Sigma \models q$ implies the existence of a chase sequence $I_i\langle h_i, \sigma_i\rangle I_{i+1}$ of D w.r.t. Σ, for $0 \leqslant i < f(|q|, |\Sigma|)$, such that $I_{f(|q|,|\Sigma|)} \models q$.*

The following key result is implicit in (Gottlob and Schwentick 2012):

Proposition 5 *If a Datalog$^\exists$ class $\mathbb{C}$ enjoys the PWP, then $\mathbb{C}$ is P-combined first-order rewritable.*

4 Guarded Datalog$^\exists$

By making different assumptions on the underlying schema, we trace the frontier between those sets of guarded rules which are P-combined first-order rewritable and those which are not. On the one hand, we show that guarded Datalog$^\exists$ is not P-combined first-order rewritable even if the underlying schema contains a single predicate, or even if we focus on unary and binary predicates. On the other hand, guarded Datalog$^\exists$ is P-combined first-order rewritable if we consider only unary predicates, or if the size and the arity of the underlying schema are bounded by an integer constant; the latter forms one of the main results of this paper.

It is well-known that query answering under guarded rules is 2EXPTIME-hard (Calì *et al.* 2013). Since the evaluation of a first-order query is feasible in polynomial space, and also PSPACE $\subsetneq$ 2EXPTIME, we get that GUARDED is not P-combined first-order rewritable. Interestingly, the same argument applies when the underlying schema contains only one predicate. More precisely, query answering under GUARDED$[1, \infty]$ is already 2EXPTIME-hard. This can be shown by a reduction from query answering under arbitrary guarded rules; let us explain the reduction by means of a simple example. Consider the following instance of query answering under guarded Datalog$^\exists$:

$$\begin{array}{rcl} D & = & \{p(a), s(a,b,c)\} \\ \Sigma & = & \{p(X), s(X,Y,Z) \rightarrow \exists W\, r(X,W)\} \\ q & = & \exists X \exists Y\, p(X), r(X,Y). \end{array}$$

The idea is to simulate the predicates p, r and s using a single predicate sim of higher arity; in fact, the atom $p(a)$ can be encoded as $sim(p, a, \diamond_1, \diamond_2, p, r, s, \diamond_1, \diamond_2)$, where $\diamond_1, \diamond_2$ are "dummy" constants not occurring in D, while $s(a, b, c)$ can be encoded as $sim(s, a, b, c, p, r, s, \diamond_1, \diamond_2)$. Then, the rule of Σ will be rewritten as

$$\begin{array}{r} sim(P_1, X, V_1, V_2, \mathbf{T}), sim(P_3, X, Y, Z, \mathbf{T}) \\ \rightarrow \exists W\, sim(P_2, X, W, V_2, \mathbf{T}), \end{array}$$

where $\mathbf{T} = (P_1, P_2, P_3, V_1, V_2)$, while the query q will be rewritten as

$$\exists X \exists Y \exists \mathbf{T}\, sim(P_1, X, V_1, V_2, \mathbf{T}), sim(P_2, X, Y, V_2, \mathbf{T}).$$

From the above discussion, we get the following result:

Theorem 6 GUARDED$[1, \infty]$ *is not P-combined first-order rewritable.*

The above result follows also from Theorem 3 and the fact that the reduction from multiple predicates to a single predicate is P-combined. Let us now consider the bounded arity case. It is known that query answering under the description logic $\mathcal{ELI}$, i.e., $\mathcal{EL}$ extended with inverse roles, is EXPTIME-hard. Since each $\mathcal{ELI}$-TBox $\mathcal{T}$ can be translated into a set $\Sigma_\mathcal{T}$ of guarded rules, where $sch(\Sigma_\mathcal{T})$ contains only unary and binary predicates, the next result follows:

Theorem 7 *If* PSPACE $\neq$ EXPTIME, *then* GUARDED$[\infty, 2]$ *is not P-combined first-order rewritable.*

It is believed that PSPACE $\subsetneq$ EXPTIME, and therefore it is unlikely for GUARDED$[\infty, 2]$ to be P-combined first-order rewritable. However, if we concentrate on unary predicates, then it is possible to establish a positive result. In particular, a set $\Sigma \in$ GUARDED$[\infty, 1]$ may contain only rules of the form $p_1(X), \ldots, p_n(X) \rightarrow \underline{a}$, for $n \geqslant 1$, where $\underline{a}$ is either $p(X)$ or $\exists Z\, p(Z)$. This fact allows us to show that GUARDED$[\infty, 1]$ enjoys the PWP. Hence, by Proposition 5, we immediately get the following result:

Theorem 8 GUARDED$[\infty, 1]$ *is P-combined first-order rewritable.*

Let us now investigate whether guarded Datalog$^\exists$ is P-combined first-order rewritable if we bound both the size and the arity of the schema. Notice that query answering under GUARDED$[c, d]$ is NP-complete (Calì *et al.* 2013), and thus the answer to this question is likely to be affirmative; the rest of this section is devoted to show that this is the case. Towards this direction, we exploit the following result established in (Gottlob and Schwentick 2012):

Proposition 9 LINEAR$[\infty, d]$ *enjoys the PWP.*

The above result, together with Proposition 5, implies that $\mathsf{LINEAR}[\infty, d]$ is P-combined first-order rewritable. Therefore, to establish our desired result, by Theorem 3, it suffices to show that $\mathsf{GUARDED}[c, d]$ is P-combined reducible to $\mathsf{LINEAR}[\infty, d]$. A key notion employed in the studies of query answering under guarded rules is the *type* of an atom. Given a database D and a set $\Sigma \in \mathsf{GUARDED}$, the type of an atom $\underline{a} \in \mathit{chase}(D, \Sigma)$ is defined as $\mathit{type}(\underline{a}, D, \Sigma) = \{\underline{b} \in \mathit{chase}(D, \Sigma) \mid \mathit{dom}(\underline{b}) \subseteq \mathit{dom}(\underline{a})\}$. Intuitively, whenever $\underline{a}$ is associated with the guard atom of a rule σ during a chase step application, then the side atoms of σ are necessarily associated with atoms of $\mathit{type}(\underline{a}, D, \Sigma)$. The abstract idea underline our P-combined reduction of query answering under $\mathsf{GUARDED}$ to query answering under LINEAR is as follows: given a guarded rule σ, encode the shape of the type τ of $\mathit{guard}(\sigma)$ in a predicate $[\tau]$, and then replace σ with a linear rule of the form $[\tau](\mathbf{X}, \mathbf{Y}) \rightarrow \exists Z\, p(\mathbf{X}, Z)$. Let us now give some extra terminology which is needed for the definition of the reduction.

4.1 Technical Definitions and Notation

Fix two sets $\mathcal{R}, \mathcal{R}'$ of predicates, where $\mathcal{R}' \subseteq \mathcal{R}$. A *guarded type* for a predicate $p \in \mathcal{R}$ w.r.t. $\mathcal{R}'$ is defined as a pair $(\underline{a}, S)$, where $\underline{a}$ is an atom of the form $p(t_1, \ldots, t_n)$ with $t_i \in \{1, \ldots, \mathit{arity}(p)\}$, and $S \subseteq \mathit{base}(\mathit{dom}(\underline{a}), \mathcal{R}') \setminus \{\underline{a}\}$. Let $\mathit{gtypes}(p, \mathcal{R}')$ be the set of all possible guarded types for p w.r.t. $\mathcal{R}'$. We define the guarded types of $\mathcal{R}$ w.r.t. $\mathcal{R}'$ as the set $\mathit{gtypes}(\mathcal{R}, \mathcal{R}') = \{\mathit{gtypes}(p, \mathcal{R}')\}_{p \in \mathcal{R}}$. Given $\tau = (\underline{a}, S) \in \mathit{gtypes}(\mathcal{R}, \mathcal{R}')$, we refer to $\underline{a}$ by $\mathit{guard}(\tau)$ and to $(\{\underline{a}\} \cup S)$ as $\mathit{atoms}(\tau)$, respectively. The arity of τ, denoted $\mathit{arity}(\tau)$, is the maximum integer occurring in $\mathit{guard}(\tau)$.

Example 1 Let $\mathcal{R} = \{p, s, r\}$ and $\mathcal{R}' = \{s, r\}$, where p is a ternary predicate, while s and r are binary predicates. A possible guarded type τ for p w.r.t. $\mathcal{R}'$ is the pair

$$(p(1,1,2), \{s(1,2), s(1,1), r(2,1), r(2,2)\}),$$

with $\mathit{arity}(\tau) = 2$. ■

Roughly speaking, a guarded type describes the equality type of a guard-atom and a set of further side atoms which are "covered" by the guard. The integer arguments occurring in a guarded type, which can be seen as constants, are mere indicators of equalities among arguments. It is important to clarify that their actual values have no semantic meaning beyond this. For normalization reasons, we adopt the convention that the integers appearing in $\mathit{guard}(\tau)$, for some guarded type τ, will always start by 1, and will be consecutive, except for repeated integers. Thus, when constructing new guarded types from existing ones, we will have to rename the arguments of a newly constructed set of atoms in order to follow the above convention. To this purpose, we use a renaming function ρ whose formal definition is obvious and it is omitted; for example, given $\tau = (\underline{a}, S)$, where $\underline{a} = p(2,2,4,1)$ and $S = \{s(1,2), s(4,1)\}$, $\rho(\mathit{atoms}(\tau)) = \{p(1,1,2,3), s(3,1), s(2,3)\}$.

Given a guarded type $\tau = (\underline{a}, S)$, and a tuple of terms $\mathbf{t} = (t_1, \ldots, t_n)$, where $n = \mathit{arity}(\tau)$, the *instantiation* of τ over $\mathbf{t}$, denoted $\tau(\mathbf{t})$, is the set of atoms $\{h_{\mathbf{t}}(\underline{b}) \mid \underline{b} \in \mathit{atoms}(\tau)\}$, where $h_{\mathbf{t}} = \{i \rightarrow t_i\}_{i \in [n]}$; for brevity, $[n] = \{1, \ldots, n\}$.

Example 2 Let τ be the guarded type for p given in Example 1. Then, $\tau((a,b))$ is the pair

$$\{p(a,a,b), s(a,b), s(a,a), r(b,a), r(b,b)\}$$

with $h_{(a,b)} = \{1 \rightarrow a, 2 \rightarrow b\}$. ■

The *projection* of $\tau = (\underline{a}, S)$ over $I \subseteq [\mathit{arity}(\tau)]$ is the set of atoms $\Pi_I(\tau) = \{\underline{b} \in \mathit{atoms}(\tau) \mid \mathit{dom}(\underline{b}) \subseteq I\}$. E.g., given $\tau = (p(1,2,3), \{r(1), s(2)\})$, then $\Pi_{\{1,2\}}(\tau) = \{r(1), s(2)\}$. Having the above notions in place, we are now ready to show that $\mathsf{GUARDED}[c, d]$ is P-combined reducible to $\mathsf{LINEAR}[\infty, d]$.

4.2 A Reduction from GUARDED to LINEAR

Let us first give a general reduction GL of query answering under guarded to query answering under linear Datalog$^{\exists}$. Fix a database D and a set $\Sigma_0 \in \mathsf{GUARDED}$. We are going to construct a database $D^\star$ and a set $\Sigma^\star \in \mathsf{LINEAR}$ such that, for every BCQ q, $D \cup \Sigma_0 \models q$ iff $D^\star \cup \Sigma^\star \models q$.

Step 1: Normalization. The first step is to normalize Σ_0 in such a way that each rule contains at most one existentially quantified variable which occurs at the last position of the head-atom. More precisely, for a rule $\sigma \in \Sigma_0$, if σ is already in normal form, then $\mathsf{N}(\sigma) = \{\sigma\}$; otherwise, assuming that $\mathit{head}(\sigma) = \underline{a}$, $\mathbf{X} = \mathit{var}(\mathit{body}(\sigma)) \cap \mathit{var}(\mathit{head}(\sigma))$, and $Z_1, \ldots, Z_m$ are the existentially quantified variables of σ, let $\mathsf{N}(\sigma)$ be the set of rules

$$\begin{array}{rcl} \mathit{body}(\sigma) & \rightarrow & \exists Z_1\, p_\sigma^1(\mathbf{X}, Z_1) \\ p_\sigma^1(\mathbf{X}, Z_1) & \rightarrow & \exists Z_2\, p_\sigma^2(\mathbf{X}, Z_1, Z_2) \\ & \cdots & \\ p_\sigma^{m-1}(\mathbf{X}, Z_1, \ldots, Z_{m-1}) & \rightarrow & \exists Z_m\, p_\sigma^m(\mathbf{X}, Z_1, \ldots, Z_m) \\ p_\sigma^m(\mathbf{X}, Z_1, \ldots, Z_m) & \rightarrow & \underline{a}, \end{array}$$

where p_σ^i is an $(|\mathbf{X}|+i)$-ary auxiliary predicate not occurring in $\mathit{sch}(\Sigma_0)$, for each $i \in [m]$. Let $\Sigma = \bigcup_{\sigma \in \Sigma_0} \mathsf{N}(\sigma)$. □

A key tool that we shall use in our reduction is the *completion* of a (finite) instance I w.r.t. Σ, i.e., the process of enriching I with all the possible atoms that can be entailed from I and Σ without altering the domain of I. Formally, the result of such a process, denoted $\mathit{complete}(I, \Sigma)$, is the set of atoms $\{\underline{a} \mid \mathit{dom}(a) \subseteq \mathit{dom}(I) \text{ and } (I \cup \Sigma) \models \underline{a}\}$. It is important to say that the atom entailment problem under guarded rules is decidable (Calì *et al.* 2013), and thus the (finite) set $\mathit{complete}(I, \Sigma)$ can be explicitly computed. Henceforth, a guarded type for a predicate $p \in \mathit{sch}(\Sigma)$ is always w.r.t. $\mathit{sch}(\Sigma)^-$, and we will simply write $\mathit{gtypes}(p)$ for the set $\mathit{gtypes}(p, \mathit{sch}(\Sigma)^-)$. Moreover, we will refer to $\mathit{gtypes}(\mathit{sch}(\Sigma), \mathit{sch}(\Sigma)^-)$ as $\mathit{gtypes}(\Sigma)$.

Step 2: Database Construction. We define

$$D^\star \;=\; \left\{ [\tau](\mathbf{t}^1) \;\middle|\; \begin{array}{l} p(\mathbf{t}) \in D, \\ \tau \in \mathit{gtypes}(p), \\ \mathit{guard}(\tau) \simeq p(\mathbf{t}), \\ \tau(\mathbf{t}^1) \subseteq \mathit{complete}(D, \Sigma) \end{array} \right\},$$

where $[\tau]$, for some guarded type τ, is a new $\mathit{arity}(\tau)$-ary predicate not occurring in $\mathit{sch}(\Sigma)$, $\mathit{guard}(\tau) \simeq p(\mathbf{t})$ means that $guard(\tau)$ and $p(\mathbf{t})$ have the same equality type (e.g., $p(1,2,1,3,2) \simeq p(t_1,t_2,t_1,t_3,t_2)$), and $\mathbf{t}^1$ is obtained from $\mathbf{t}$ by keeping only the first occurrence of each term occurring in $\mathbf{t}$ (e.g., if $\mathbf{t} = (t_1,t_2,t_1,t_3,t_2)$, then $\mathbf{t}^1 = (t_1,t_2,t_3)$). Intuitively, an atom $[\tau](\mathbf{t}^1)$, occurring in $D^\star$ due to an atom $p(\mathbf{t}) \in D$, encodes a possible subset of $chase(D,\Sigma)$ which contains the atoms that may act as side atoms whenever $p(\mathbf{t})$ plays the role of the guard in a chase application. □

Example 3 Assume that $D = \{p(a,a), p(a,b), s(b)\}$ and $\Sigma = \{p(X,Y), s(Y) \rightarrow s(X)\}$. For the atom $p(a,a) \in D$, an atom $[\tau](a)$ will be added in $D^\star$, where τ is a guarded type of the form $(p(1,1),S)$. Observe that the only atom, apart from $p(a,a)$, which is entailed by D and Σ, and contains only the term a, is $s(a)$; therefore, S is either empty or is the set $\{s(1)\}$. Consequently, for $p(a,a)$, the atoms $[(p(1,1),\varnothing)](a)$, $[(p(1,1),\{s(1)\})](a)$ will be added in $D^\star$. Consider now the atom $p(a,b) \in D$. It is easy to verify that the set of atoms $\bigcup_{S \in \mathcal{P}(\{p(1,1),s(1),s(2)\})}\{[(p(1,2),S)]\}$, where $\mathcal{P}(X)$ denotes the powerset of X, will be added in $D^\star$. Finally, for $s(b) \in D$, only the atom $[(s(1),\varnothing)](b)$ will be added in $D^\star$, since no other atom than $s(b)$, which contains only the term b, is entailed by D and Σ. ■

Step 3: Rules Construction. The set $\Sigma^\star$ consists of the following two groups of rules:

1. The *type generator*, denoted $\Sigma^\star_G$, which is responsible for generating new guarded types from existing ones, and
2. The *unfolder*, denoted $\Sigma^\star_U$, that is responsible for unfolding a derived guarded type τ, i.e., to explicitly construct the atoms over the original schema $sch(\Sigma_0)$ which are encoded in τ so that they can be queried directly.

Let us now define formally $\Sigma^\star_G$ and $\Sigma^\star_U$; let $w = arity(\Sigma)$.

Step 3.1: Type Generator. For each $\sigma \in \Sigma$ of the form

$$body(\sigma) \rightarrow \exists Z\, p(X_1,\ldots,X_n,Z),$$

and for each guarded type $\tau \in gtypes(\Sigma)$ for which there exists a homomorphism h such that $h(body(\sigma)) \subseteq atoms(\tau)$ and $h(guard(\sigma)) = guard(\tau)$, in $\Sigma^\star_G$ there exists the rule:

$$[\tau](V_1,\ldots,V_m) \rightarrow \exists Z\,[\tau'](W_1,\ldots,W_k,Z),$$

where $m = arity(\tau)$, and

- $\{V_1,\ldots,V_m\} \subseteq var(body(\sigma))$, and $h(V_i) = i$, for each $i \in [m]$,
- $\{W_\ell\}_{\ell\in[k]} \subseteq \{X_\ell\}_{\ell\in[n]}$ and $(h(W_1),\ldots,h(W_k)) = (h(X_1),\ldots,h(X_n))^1$,
- $\tau' = (\rho(\underline{a}), (complete(\rho(\{\underline{a}\} \cup S),\Sigma) \setminus \{\rho(\underline{a})\})_{|sch(\Sigma)^-})$, where $\underline{a}$ is the atom $p(h(X_1),\ldots,h(X_n),w+1)$ and S is the set of atoms $\Pi_{\{h(X_1),\ldots,h(X_n)\}}(\tau)$.

No other rules occur in $\Sigma^\star_G$. □

Example 4 Assume that Σ consists of the rules

$$\sigma_1 \;:\; p(X,Y,X,Z,W), s(X,Z) \rightarrow \exists U\, r(Z,Y,X,U)$$
$$\sigma_2 \;:\; r(X,X,Y,Z) \rightarrow t(X,Z).$$

Let us focus our attention on σ_1. A possible guarded type τ for the predicate of the guard of σ_1, that is, the predicate p, for which there exists a homomorphism that maps $body(\sigma_1)$ to $atoms(\tau)$, and $guard(\sigma_1)$ to $guard(\tau)$ is:

$$\tau = (p(1,2,1,2,3), \{s(1,1), s(1,2)\}).$$

Therefore, in $\Sigma^\star_G$ we have the linear rule:

$$[\tau](X,Y,W) \rightarrow \exists U\,[\tau'](Y,X,U),$$

where $\tau' = (r(1,1,2,3), \{s(2,2), s(2,1), t(1,3)\})$. ■

Step 3.2: Unfolder. For each $\tau \in gtypes(\Sigma_0)$, if $guard(\tau)$ is of the form $p(t_1,\ldots,t_n)$, in $\Sigma^\star_U$ there exists the rule:

$$[\tau](V_1,\ldots,V_m) \rightarrow p(W_1,\ldots,W_n),$$

where

- $\{V_1,\ldots,V_m\} \subset \mathbf{V}$ and $(V_1,\ldots,V_m) = (W_1,\ldots,W_n)^1$,
- $t_i = t_j$ implies $W_i = W_j$, for each $i,j \in [n]$.

No other rules occur in $\Sigma^\star_U$. □

Example 5 Consider the guarded type

$$\tau = (p(1,2,1,2,3), \{s(1,1), s(1,2)\}).$$

Due to τ, in $\Sigma^\star_U$ we have the linear rule:

$$[\tau](X,Y,Z) \rightarrow p(X,Y,X,Y,Z).$$

Notice that, for any type of the form $(p(1,2,1,2,3),S)$, i.e., that has the same guard as τ, the rule added to $\Sigma^\star_U$ is the same as the one above. ■

We proceed to show that GL is indeed a reduction of query answering under guarded Datalog$^\exists$ to query answering under linear Datalog$^\exists$. To this aim, we exploit the central property of the type of an atom. In particular, the subtree of the *guarded chase forest* of D and Σ — the forest obtained from $chase(D,\Sigma)$ by keeping only the guards and their children — rooted at $\underline{a}$ is determined by $type(\underline{a},D,\Sigma)$ (modulo renaming of nulls) (Calì *et al.* 2013). This implies that, given an atom $\underline{a} = p(t_1,\ldots,t_n,z)$, where z is a null value invented in $\underline{a}$, we can construct $type(a,D,\Sigma)$ from $\underline{a}$ and the restriction of $type(a,D,\Sigma)$ to $\{t_1,\ldots,t_n\}$. This justifies the definition of the type generator, where we complete the known part of the type, denoted S, of a generated atom $\underline{a}$ by calling the procedure $complete$ using as a database the instance $(\{\underline{a}\} \cup S)$.

The above property allows us to show two useful lemmas. The first one asserts that, for each $\underline{a} \in chase(D,\Sigma)$, GL encodes $type(\underline{a},D,\Sigma)$ in a complete way. The function $\gamma_{\underline{a}}$ renames the arguments of an atom $\underline{a} \in chase(D,\Sigma)$ into integers. For example, given $\underline{a} = p(a,z_1,a,b)$, then $\gamma_{\underline{a}}(\underline{a}) = p(1,2,1,3)$; the definition of $\gamma_{\underline{a}}$ is obvious and it is omitted. Let $\tau_{\underline{a}} = (\gamma_{\underline{a}}(\underline{a}), \gamma_{\underline{a}}(type(\underline{a},D,\Sigma) \setminus \{\underline{a}\}))$.

Lemma 10 *There is a homomorphism h such that, $p(\mathbf{t}) \in chase(D,\Sigma)$ implies $[\tau_{p(\mathbf{t})}](h(\mathbf{t}^1)) \in chase(D^\star,\Sigma^\star_G)$.*

The second technical lemma states that the reduction GL encodes in a sound way the various types that can be realized in the instance constructed by the chase:

Lemma 11 *There is a homomorphism h such that, $[\tau](\mathbf{t}) \in chase(D^\star, \Sigma_G^\star)$ implies $h(\tau(\mathbf{t})) \subseteq chase(D, \Sigma)$.*

By exploiting the above technical lemmas, we can show that $chase(D, \Sigma_0)$ and $chase(D^\star, \Sigma^\star)$, after eliminating the atoms with predicates not occurring in $sch(\Sigma_0)$, are homomorphically equivalent:

Proposition 12 *It holds that,*

$$chase(D, \Sigma_0) \longleftrightarrow chase(D^\star, \Sigma^\star)_{|sch(\Sigma_0)}.$$

Proof (sketch). By construction, there exists a homomorphism μ that maps $chase(D, \Sigma_0)$ to $chase(D, \Sigma)_{|sch(\Sigma_0)}$. Due to $\Sigma_U^\star$, we get that h_1, provided by Lemma 10, maps $chase(D, \Sigma)_{|sch(\Sigma_0)}$ to $chase(D^\star, \Sigma^\star)_{|sch(\Sigma_0)}$. Therefore,

$$h_1(\mu(chase(D, \Sigma_0))) \subseteq chase(D^\star, \Sigma^\star)_{|sch(\Sigma_0)}.$$

Conversely, let h_2 be the homomorphism as in Lemma 11. Since $chase(D^\star, \Sigma^\star)_{|sch(\Sigma_0)} \subseteq \cup_{[\tau](\mathbf{t}) \in chase(D^\star, \Sigma_G^\star)} \tau(\mathbf{t})$, h_2 maps $chase(D^\star, \Sigma^\star)_{|sch(\Sigma_0)}$ to $chase(D, \Sigma)_{|sch(\Sigma_0)}$. Clearly, by construction, there exists a homomorphism μ' that maps $chase(D, \Sigma)_{|sch(\Sigma_0)}$ to $chase(D, \Sigma_0)$. Thus,

$$\mu'(h_2(chase(D^\star, \Sigma^\star)_{|sch(\Sigma_0)})) \subseteq chase(D, \Sigma_0).$$

Consequently, due to $(h_1 \circ \mu)$ and $(\mu' \circ h_2)$, we get that $chase(D, \Sigma_0) \longleftrightarrow chase(D^\star, \Sigma^\star)_{|sch(\Sigma_0)}$. □

Using Proposition 12, we can show that, for every BCQ q, $chase(D, \Sigma_0) \models q$ iff $chase(D^\star, \Sigma^\star) \models q$, since q contains only predicates of $sch(\Sigma_0)$, and the next result follows:

Theorem 13 *For every BCQ q, $D \cup \Sigma_0 \models q$ iff $D^\star \cup \Sigma^\star \models q$.*

The question that comes up is whether GL is a P-combined reduction. Recall that we are interested in sets of guarded rules where the size and the maximum arity of the underlying schema are bounded by a constant. In this case, as we show below, indeed GL is P-combined.

4.3 The Reduction GL is P-combined

By construction, $D^\star$ depends only on D and Σ_0, while $\Sigma^\star$ depends only on Σ_0; hence, GL is a combined reduction. Thus, to establish the desired result, it remains to show that, if $\Sigma_0 \in \mathsf{GUARDED}[c, d]$, then $D^\star$ and $\Sigma^\star$ are constructible in polynomial time. Via a combinatorial analysis, we can establish some upper bounds on the number of all possible guarded types, and also on the size of a guarded type τ, denoted $|\tau|$, and defined as $|atoms(\tau)|$.

Lemma 14 *It holds that,*

1. *$|gtypes(p)| \leqslant arity(p)^{arity(p)} \cdot 2^{|sch(\Sigma)^-| \cdot arity(p)^w}$, for each $p \in sch(\Sigma)$,*
2. *$|gtypes(\Sigma)| \leqslant |sch(\Sigma)| \cdot w^w \cdot 2^{|sch(\Sigma)^-| \cdot w^w}$, and*
3. *$|\tau| \leqslant |sch(\Sigma)| \cdot w^w$, for each $\tau \in gtypes(\Sigma)$.*

Henceforth, we assume that $\Sigma_0 \in \mathsf{GUARDED}[c, d]$. Observe that the maximum number of existentially quantified variables that can appear in a rule of Σ_0 is $arity(\Sigma_0)$. Therefore, $|\Sigma| \in \mathcal{O}(|\Sigma_0|)$ and $|sch(\Sigma)| \in \mathcal{O}(|\Sigma_0|)$. Since the auxiliary predicates introduced during the normalization procedure do not appear in side atoms, we get that $sch(\Sigma)^- \subseteq sch(\Sigma_0)$, and thus $|sch(\Sigma)^-| \leqslant c$. It is important to say that $arity(\Sigma) = arity(\Sigma_0) \leqslant d$.

To construct $D^\star$ we need, for each atom $p(\mathbf{t}) \in D$, and for each $\tau \in gtypes(p)$ such that $guard(\tau) \simeq p(\mathbf{t})$, to check whether $D \cup \Sigma \models \underline{a}$, for each $\underline{a} \in \tau(\mathbf{t}^1)$. Importantly, the latter (atom entailment) check is feasible in polynomial time; this follows from the fact that atomic query answering under $\mathsf{GUARDED}[c, d]$ can be performed in polynomial time in D and Σ (Calì *et al.* 2013). In the worst case, we need to apply this polynomial check $\sum_{p(\mathbf{t}) \in D} \sum_{\tau \in gtypes(p)} |\tau|$ times. By Lemma 14, and since $|sch(\Sigma)^-| \leqslant c$ and $arity(p) \leqslant d$, for each $p \in sch(\Sigma)$, eventually the crucial entailment check must be performed $|D| \cdot |sch(\Sigma)| \cdot 2^{c \cdot d^d} \cdot d^{2d} \in \mathcal{O}(|D| \cdot |\Sigma_0|)$ times, and hence $D^\star$ is constructible in polynomial time.

We now consider the construction of $\Sigma^\star$. By exploiting Lemma 14, one can show that the maximum number of rules that can appear in $\Sigma_G^\star$ is $\mathcal{O}(|\Sigma_0|^2)$. The construction of each linear rule (apart from some easy steps) amounts to compute the set of atoms $complete(I, \Sigma)_{|sch(\Sigma)^-}$, where I is an instance of size bounded by $|\tau|$ with τ be the guarded type under consideration. This requires to call a polynomial entailment check $\mathcal{O}(|\Sigma_0|^d)$ times, and hence $\Sigma_G^\star$ is constructible in polynomial time. To construct the unfolder $\Sigma_U^\star$ we actually need to construct $|gtypes(\Sigma_0)|$ linear rules, each of which can be constructed in time $\mathcal{O}(1)$. Thus, by Lemma 14, the construction of $\Sigma_U^\star$ can be carried out in time $\mathcal{O}(1)$.

From the above analysis, we conclude that $D^\star$ and $\Sigma^\star$ are constructible in polynomial time, and the next result follows:

Proposition 15 $\mathsf{GUARDED}[c, d]$ *is P-combined reducible to* $\mathsf{LINEAR}[\infty, d]$.

By combining Theorem 3 with Propositions 5, 9 and 15, the main result of this section follows:

Theorem 16 $\mathsf{GUARDED}[c, d]$ *is P-combined first-order rewritable.*

5 Tame Datalog$^\exists$

In this section, we investigate the question whether the class of tame rules is P-combined first-order rewritable whenever the size and the arity of the schema are bounded. One may claim that the technique proposed for guarded rules can be directly applied on tame rules in order to obtain the desired result. More precisely, one can suggest the following: it can be shown that $\mathsf{TAME}[c, d]$ is P-combined reducible to a class that is P-combined first-order rewritable — which in turn implies that $\mathsf{TAME}[c, d]$ is P-combined first-order rewritable — by exploiting GL and rewriting the given database D into $D^\star$, and the guarded part $\Sigma[g]$ of the given tame set $\Sigma \in \mathsf{TAME}[c, d]$ into $\Sigma[g]^\star$, while keeping the sticky part $\Sigma[s]$ of Σ untouched. Although this approach is quite promising, there are two non-trivial difficulties:

- $(\Sigma[g]^\star \cup \Sigma[s])$ is sticky, and it is not known whether $\mathsf{STICKY}[\infty, d]$ is P-combined first-order rewritable, and
- GL is not complete on tame rules since a side atom of a guarded rule may depend on the sticky part.

In what follows, we explore the combined approach for sticky rules, and we show that $\mathsf{STICKY}[\infty, d]$ is P-combined first-order rewritable. We also reveal the key reason of the incompleteness of GL, and we briefly explain how it can be adapted in order to regain completeness.

5.1 Stickiness and the Combined Approach

Query answering under sticky sets of rules is EXPTIME-complete (Calì *et al.* 2012b). Interestingly, the problem remains EXPTIME-hard even if the underlying schema contains only one predicate. This can be established by employing the same construction given in the previous section to show an analogous result for guarded rules. Thus, we get the following impossibility result:

Theorem 17 *If* PSPACE $\neq$ EXPTIME, *then* $\mathsf{STICKY}[1,\infty]$ *is not P-combined first-order rewritable.*

Let us now show that $\mathsf{STICKY}[\infty,d]$ is P-combined first-order rewritable by showing first that it enjoys the polynomial witness property. Consider a BCQ q, a database D, and a set $\Sigma \in \mathsf{STICKY}[\infty,d]$. Assume that $D \cup \Sigma \models q$. This implies that there exists a homomorphism h such that $h(body(q)) = H \subseteq chase(D,\Sigma)$. Consider now the proof of H, i.e., the (finite) part $P \subseteq chase(D,\Sigma)$ due to which H is generated. Because of stickiness, apart from the terms occurring in H, all the other terms in P do not participate in a join operation during the construction of P, and therefore can be seen as "don't care" terms. Given an atom $\underline{a} \in P$, let $\underline{a}_\star$ be the atom obtained from $\underline{a}$ by replacing the "don't care" terms, i.e., the terms of $dom(\underline{a}) \setminus dom(H)$, with $\star$. It is possible to show that a witness P' of q can always be constructed such that, for every $\underline{a}, \underline{a}' \in P'$, $\underline{a} \neq \underline{a}'$ implies $\underline{a}_\star \neq \underline{a}'_\star$. Since $|P'| \leqslant n \cdot (|dom(H)|+1)^d$, where $n = |sch(\Sigma)|$, and $|dom(H)| \leqslant |q| \cdot d$, we conclude that a witness of size $\mathcal{O}(n \cdot |q|^d)$ always exists, and the next result follows:

Lemma 18 $\mathsf{STICKY}[\infty,d]$ *enjoys the PWP.*

The above result improves an existing one asserting that $\mathsf{STICKY}[c,d]$ enjoys the PWP (Gottlob and Schwentick 2012). By Proposition 5 and Lemma 18 we get that:

Theorem 19 $\mathsf{STICKY}[\infty,d]$ *is P-combined first-order rewritable.*

5.2 A Reduction from TAME to STICKY

We now concentrate on the reduction GL. We first show that it is not complete on tame rules. Let $D = \{p(a,b), p(b,c)\}$, and Σ be the tame set of rules consisting of:

$$\begin{array}{lcl}\sigma_1 & : & p(X,Y) \rightarrow \exists Z\, r(Y,Z)\\ \sigma_2 & : & r(X,Y), s(Y,X) \rightarrow \exists Z\, r(Y,Z)\\ \sigma_3 & : & r(X,Y), p(X,Z) \rightarrow s(Y,X),\end{array}$$

where the guarded part $\Sigma[g] = \{\sigma_1, \sigma_2\}$ and the sticky part $\Sigma[s] = \{\sigma_3\}$. The chase of D w.r.t. Σ is

$$C = \{p(a,b), p(b,c), r(b,z_1), r(c,z_2), s(z_1,b), r(z_1,z_3)\}.$$

Let us now construct $D^\star$ and $\Sigma[g]^\star$. It is easy to verify that

$$D^\star = \{[(p(1,2),\varnothing)](a,b), [(p(1,2),\varnothing)](b,c)\}.$$

Due to σ_1 and the guarded type $(p(1,2),\varnothing)$ for p, in $\Sigma[g]^\star$ we have the linear rule

$$[(p(1,2),\varnothing)](X,Y) \rightarrow \exists Z\, [(r(1,2),\varnothing)](Y,Z).$$

Moreover, due to σ_2 and the type $(r(1,2),\{s(2,1)\})$ for r, in $\Sigma[g]^\star$ we have the linear rule

$$[(r(1,2),\{s(2,1)\})](X,Y) \rightarrow \exists Z\, [(r(1,2),\varnothing)](Y,Z).$$

Notice that several other rules occur in $\Sigma[g]^\star$. However, for this particular example they are meaningless, and therefore they are omitted for brevity; e.g., rules that must occur in $\Sigma[g]^\star$ due to σ_1 and a type of the form $(p(1,1),S)$ will never be triggered. In addition, in $\Sigma[g]^\star$ we have the linear rules

$$\begin{array}{l}[(p(1,2),\varnothing)](X,Y) \rightarrow \exists Z\, p(X,Y)\\ [(r(1,2),\varnothing)](X,Y) \rightarrow \exists Z\, r(X,Y)\\ [(r(1,2),\{s(2,1)\})](X,Y) \rightarrow \exists Z\, r(X,Y).\end{array}$$

Observe that $chase(D^\star, \Sigma[g]^\star \cup \Sigma[s])_{|sch(\Sigma)}$ is

$$C^\star = \{p(a,b), p(b,c), r(b,z_1^\star), r(c,z_2^\star), s(z_1^\star,b)\}.$$

The fact that there exists a homomorphism that maps $C^\star$ to C implies that, for every BCQ q, $C^\star \models q$ implies $C \models q$. Actually, the reduction GL is sound even if we consider tame rules. However, it is not also complete: consider the BCQ $q = \exists X \exists Y \exists Z\, r(X,Y), r(Y,Z)$, and observe that $C \models q$ but $C^\star \not\models q$. The reason why $C^\star \not\models q$ is because GL failed to encode the fact that an atom $r(z_1^\star, z_3^\star)$ must also be generated during the construction of $chase(D^\star, \Sigma[g]^\star \cup \Sigma[s])$. More precisely, the crucial linear rule σ of the form

$$[(p(1,2),\varnothing)](X,Y) \rightarrow \exists Z\, [(r(1,2),\{s(2,1)\})](Y,Z)$$

is missing from $\Sigma[g]^\star$. Because of the above rule, the atom $[(r(1,2),\{s(2,1)\}](b,z_1^\star)$ will be obtained during the construction of the chase, which in turn will trigger the rule

$$[(r(1,2),\{s(2,1)\})](X,Y) \rightarrow \exists Z\, [(r(1,2),\varnothing)](Y,Z)$$

and the atom $[(r(1,2),\varnothing)](z_1^\star, z_3^\star)$ will be generated. Then, due to the rule

$$[(r(1,2),\varnothing)](X,Y) \rightarrow \exists Z\, r(X,Y)$$

the desired atom $r(z_1^\star, z_3^\star)$ will be eventually obtained. Thus, in order to reveal the real reason why GL is not complete on tame rules, we need to understand why the crucial rule σ does not occur in $\Sigma[g]^\star$. Clearly, σ should exists in $\Sigma[g]^\star$ due to σ_1 and a guarded type τ of the form $(p(1,2),S)$, where $S \subseteq base(\{1,2\}, sch(\Sigma))$. However,

$$\begin{array}{rcl}\tau' & = & (r(1,2), complete(\{r(1,2)\} \cup \Pi_{\{2\}}(\tau), \Sigma))\\ & = & (r(1,2),\varnothing),\end{array}$$

even if $S = base(\{1,2\}, sch(\Sigma))$, and therefore there is no way to obtain the head atom $\exists Z\, [(r(1,2),\{s(1,2)\})](Y,Z)$. This is because, in order to obtain $s(2,1)$ during the completion of $(\{r(1,2)\} \cup \Pi_{\{2\}}(\tau))$ w.r.t. Σ, an atom of the form $p(1,X)$ is needed. However, such an atom does not occur in $complete(\{r(1,2)\} \cup \Pi_{\{2\}}(\tau), \Sigma)$, and this is exactly the source of incompleteness. It is important to observe that the atom $p(1,X)$ is needed to trigger the rule σ_3, i.e., the sticky rule of Σ, and since X is not propagated in the inferred atom, by stickiness, we can safely conclude that such a term X will never participate in a join operation during the completion process. Hence, what is important is not the actual value of X but the existence of a witness for X.

From the above discussion, we conclude that the way to regain completeness is to add in $side(\tau)$ a witness atom $p(1,\star)$ for $p(1,X)$, and let

$$\tau' = (r(1,2), complete(\{r(1,2)\} \cup \Pi_{\{2,\star\}}(\tau), \Sigma)),$$

and also add an additional unfold rule of the form

$$[(r(1,2), \{s(2,1), p(1,\star)\})](X,Y) \rightarrow \exists Z\, p(X,Z)$$

in order to explicitly unfold the witness atom $p(1,\star)$ by inventing a "fresh" null for each occurrence of $\star$. Notice that in this particular example, the above rule is not needed since $p(1,\star)$ is a witness of the database atom $p(b,c)$, and thus it is not necessary to regenerate it. However, in general, such unfolding rules are crucial.

The above approach can be generalized in order to obtain a combined reduction, denoted TS, of query answering under tame sets of rules to query answering under sticky sets of rules. Let us clarify that $(\Sigma[g]^\star \cup \Sigma[s])$ is indeed sticky since, by construction, for each $\sigma \in \Sigma[g]^\star$, every variable in $body(\sigma)$ occurs only once, and thus there is no way to violate stickiness after the merging of $\Sigma[g]^\star$ and $\Sigma[s]$. Moreover, if the size and arity of the underlying schema is fixed, then TS is P-combined, and the next result follows:

Proposition 20 TAME$[c,d]$ *is P-combined reducible to* STICKY$[\infty,d]$.

From Theorem 3, Theorem 19 and Proposition 20, we immediately get the main result of this section:

Theorem 21 TAME$[c,d]$ *is P-combined first-order rewritable.*

6 Linear Datalog$^\exists$: The Missing Tile

It is open whether Linear Datalog$^\exists$ is P-combined first-order rewritable, and we are currently investigating this problem. The purpose of this section is to show that the solution to the above open problem cannot benefit from existing tools, and therefore novel techniques must be devised.

By exploiting the reduction in the proof of Theorem 6, we can show that LINEAR and LINEAR$[1,\infty]$ coincide w.r.t. P-combined first-order rewritability. Therefore, in order to establish the desired result, it suffices to focus our attention on sets of linear rules which contain exactly one predicate. Since query answering under LINEAR is PSPACE-hard, the same problem under LINEAR$[1,\infty]$ is also PSPACE-hard. Thus, in view of the fact that query answering under the (currently) known Datalog$^\exists$ languages which are P-combined first-order rewritable — i.e., the languages considered in this paper — is in NP, the P-combined reducibility approach cannot be applied (unless NP = PSPACE).

Unfortunately, as shown below, the existing techniques of establishing that LINEAR$[\infty,d]$ is P-combined first-order rewritable, which are based on the PWP, are also not useful, or, at least, immediately applicable. We define the class SUCC $= \{\Sigma_n\}_{n>0}$, where Σ_n is the set of linear rules

$$\{num(Z,O,\mathbf{B}_{n-i},Z,O^{i-1}) \rightarrow num(Z,O,\mathbf{B}_{n-i},O,Z^{i-1})\}_{i\in[n]}$$

with $\mathbf{B}_k = B_1,\ldots,B_k$. Roughly speaking, Σ_n simulates the successor operator on m-digit binary numbers, where $m = n-2$. The binary number $b_1b_2\ldots b_m$ is encoded as $num(Z,O,b_1,\ldots,b_m)$, where Z,O are auxiliary variables that allow us to have access to the bits 0 and 1, respectively. Given the atomic query $q = num(0,1,1,\ldots,1)$ and the database $D = \{num(0,1,0,\ldots,0)\}$, we need to apply $\mathcal{O}(2^m)$ times the chase step in order to entail q. Consequently, SUCC does not enjoy the PWP. Since SUCC $\subset$ LINEAR$[1,\infty]$, the next result follows:

Lemma 22 LINEAR$[1,\infty]$ *does not enjoy the PWP.*

One may think that Lemma 22 implies that LINEAR$[1,\infty]$ is not P-combined first-order rewritable. However, the PWP is not a necessary condition for the P-combined first-order rewritability. This is established by showing first that a class for which query answering is PTIME-hard does not enjoy the PWP. Since there exist PTIME-hard classes which are P-combined first-order rewritable, e.g., GUARDED$[c,d]$, the next result follows:

Lemma 23 *There exists a Datalog$^\exists$ class which does not enjoy the PWP, but it is P-combined first-order rewritable.*

All the known formalisms which are P-combined first-order rewritable, enjoy a property close to the PWP which we call *weakly PWP (wPWP)*. Roughly, this property is defined as the PWP, with the difference that the polynomial size witness may depend on the given database. Clearly, SUCC, and thus LINEAR$[1,\infty]$, does not enjoy the wPWP. However, it is possible to show that SUCC is P-combined first-order rewritable, and thus obtaining the first example which does not ensure a witness of polynomial size, but it is P-combined first-order rewritable.

From the above discussion, we conclude that LINEAR, although does not guarantee witnesses of polynomial size, it is likely to be P-combined first-order rewritable. However, the existing tools are not powerful enough for establishing such a result, and novel techniques must be developed.

7 Conclusion

We considered the polynomial combined approach to query answering under expressive classes of existential rules. To the best of our knowledge, this work is the first attempt to explicitly apply the combined approach to existential rules. We established that guarded and tame rules are polynomially combined first-order rewritable, whenever the size and the arity of the underlying schema are bounded. The same holds also for sticky rules, even if the size of the underlying schema is not bounded. Notice that our techniques immediately show that guarded and tame rules are combined first-order rewritable. The results of this work are, for the moment, of theoretical nature and we do not claim that they will directly lead to better practical algorithms. A smart implementation and an evaluation of the obtained rewritings will be the subject of future research. We are also planning to investigate the notion of *output-polynomial* combined first-order rewritability, i.e., when the rewriting is of polynomial size but not necessarily constructible in polynomial time.

References

Jean-François Baget, Michel Leclère, Marie-Laure Mugnier, and Eric Salvat. On rules with existential variables: Walking the decidability line. *Artif. Intell.*, 175(9-10):1620–1654, 2011.

Jean-François Baget, Marie-Laure Mugnier, Sebastian Rudolph, and Michaël Thomazo. Walking the complexity lines for generalized guarded existential rules. In *Proc. of IJCAI*, pages 712–717, 2011.

Andrea Calì, Georg Gottlob, and Thomas Lukasiewicz. A general Datalog-based framework for tractable query answering over ontologies. *J. Web Sem.*, 14:57–83, 2012.

Andrea Calì, Georg Gottlob, and Andreas Pieris. Towards more expressive ontology languages: The query answering problem. *Artif. Intell.*, 193:87–128, 2012.

Andrea Calì, Georg Gottlob, and Michael Kifer. Taming the infinite chase: Query answering under expressive relational constraints. *J. Artif. Intell. Res.*, 48:115–174, 2013.

Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *J. Autom. Reasoning*, 39(3):385–429, 2007.

Ashok K. Chandra and Philip M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *Proc. of STOCS*, pages 77–90, 1977.

Ronald Fagin, Phokion G. Kolaitis, Renée J. Miller, and Lucian Popa. Data exchange: Semantics and query answering. *Theor. Comput. Sci.*, 336(1):89–124, 2005.

Georg Gottlob and Thomas Schwentick. Rewriting ontological queries into small nonrecursive datalog programs. In *Proc. of KR*, 2012.

Georg Gottlob, Marco Manna, and Andreas Pieris. Combining decidability paradigms for existential rules. *TPLP*, 13(4-5):877–892, 2013.

Stanislav Kikot, Roman Kontchakov, Vladimir V. Podolskii, and Michael Zakharyaschev. Exponential lower bounds and separation for query rewriting. In *Proc. of ICALP*, pages 263–274, 2012.

Roman Kontchakov, Carsten Lutz, David Toman, Frank Wolter, and Michael Zakharyaschev. The combined approach to query answering in DL-Lite. In *Proc. of KR*, 2010.

Roman Kontchakov, Carsten Lutz, David Toman, Frank Wolter, and Michael Zakharyaschev. The combined approach to ontology-based data access. In *Proc. of IJCAI*, pages 2656–2661, 2011.

Markus Krötzsch and Sebastian Rudolph. Extending decidable existential rules by joining acyclicity and guardedness. In *Proc. of IJCAI*, pages 963–968, 2011.

Nicola Leone, Marco Manna, Giorgio Terracina, and Pierfrancesco Veltri. Efficiently computable Datalog$^\exists$ programs. In *Proc. of KR*, 2012.

Carsten Lutz, David Toman, and Frank Wolter. Conjunctive query answering in the description logic $\mathcal{EL}$ using a relational database system. In *Proc. of IJCAI*, pages 2070–2075, 2009.

Antonella Poggi, Domenico Lembo, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. Linking data to ontologies. *J. Data Semantics*, 10:133–173, 2008.

A Proof of Theorem 6

The proof is by reduction from query answering under arbitrary guarded rules. Consider a database D, a set Σ of guarded Datalog$^\exists$ rules, and a BCQ q. We are going to construct a database D', a set Σ' of guarded rules where $|sch(\Sigma')| = 1$, and a BCQ q' such that $D \cup \Sigma \models q$ iff $D' \cup \Sigma' \models q'$. Let us first define two translation functions τ_1 and τ_2. In what follows, we assume that $sch(\Sigma) = \{p_1, \ldots, p_n\}$ and $m = arity(\Sigma)$. Given an atom $\underline{a} = p_i(t_1, \ldots, t_k)$, where each $t_i \in (\mathbf{C} \cup \mathbf{V})$:

1. let $\tau_1(\underline{a}) = sim(p_i, t_1, \ldots, t_k, \diamond_{k+1}, \ldots, \diamond_m, p_1, \ldots, p_n, \diamond_1, \ldots, \diamond_m)$, where sim is an $(n+2m+1)$-ary predicate not occurring in $sch(\Sigma)$, $\{p_1, \ldots, p_n, \diamond_1, \ldots, \diamond_m\} \subset \mathbf{C}$, $\{p_1, \ldots, p_n, \diamond_1, \ldots, \diamond_m\} \cap dom(D) = \varnothing$, and $|\{p_1, \ldots, p_n, \diamond_1, \ldots, \diamond_m\}| = n + m$; and
2. let $\tau_2(\underline{a}) = sim(P_i, t_1, \ldots, t_k, V_{k+1}, \ldots, V_m, P_1, \ldots, P_n, V_1, \ldots, V_m)$, where $\{P_1, \ldots, P_n, V_1, \ldots, V_m\} \subset \mathbf{V}$, $\{P_1, \ldots, P_n, V_1, \ldots, V_m\} \cap var(\Sigma) = \varnothing$, and $|\{P_1, \ldots, P_n, V_1, \ldots, V_m\}| = n + m$.

Having the above translation functions in place, we are now ready to construct D', Σ' and q' as follows:

1. $D' = \{\tau_1(\underline{a}) \mid \underline{a} \in D\}$;
2. Assuming that q is of the form $\exists \mathbf{X}\, \underline{a}_1, \ldots, \underline{a}_k$, let q' be the BCQ $\exists \mathbf{X}\, \tau_2(\underline{a}_1), \ldots, \tau_2(\underline{a}_k)$; and
3. for each $\sigma \in \Sigma$ of the form $\underline{b}_1, \ldots, \underline{b}_k \rightarrow \exists \mathbf{Z}\, \underline{a}$, in Σ' the rule $\tau_2(\underline{b}_1), \ldots, \tau_2(\underline{b}_k) \rightarrow \exists \mathbf{Z}\, \tau_2(\underline{a})$ exists.

Is is not difficult to see that indeed $D \cup \Sigma \models q$ iff $D' \cup \Sigma' \models q'$. Since $|sch(\Sigma')| = 1$ the claim follows.

B Proof of Theorem 8

To show that $\mathsf{GUARDED}[\infty, 1]$ is P-combined first-order rewritable, we show that this class enjoys the PWP, but before, we need to introduce some useful notation. The *chase relation* (Calì *et al.* 2012b) of a database D and a set Σ of TGDs is a binary relation on atoms, denoted by $CR[D, \Sigma]$, which mimics all the chase derivations of the chase. More precisely, $CR[D, \Sigma]$ is the maximum subset of $chase(D, \Sigma) \times chase(D, \Sigma)$ such that $\langle \underline{a}, \underline{b} \rangle \in CR[D, \Sigma]$ implies that $\underline{b}$ is obtained from $\underline{a}$ via a chase step $I\langle \sigma, h \rangle I'$ where $I' \setminus I = \{b\}$ and $\underline{a} \in h(body(\sigma))$. The transitive closure of $CR[D, \Sigma]$ is denoted by $CR^+[D, \Sigma]$.

Consider a BCQ q, a database D, and a set $\Sigma \in \mathsf{GUARDED}[\infty, 1]$. Assume that $D \cup \Sigma \models q$. This implies that there is a homomorphism h such that $h(body(q)) = chase(D, \Sigma)$. Consider now the proof of $H = h(body(q))$, namely the set $P = H \cup \{\underline{a} \in chase(D, \Sigma) \mid$ there exits $\underline{b} \in H$ such that $\langle \underline{a}, \underline{b} \rangle \in CR^+[D, \Sigma]\}$, which represents the finite part $P \subseteq chase(D, \Sigma)$ due to which H is generated. (Note that P is not necessarily a superset of D.) Once we now P, we can identify from the chase steps that have been applied to construct $chase(D, \Sigma)$ a sequence $S = I_0\langle \sigma_1, h_1 \rangle I_1 \langle \sigma_2, h_2 \rangle I_2 \ldots I_n$ of chase steps such that $I_0 = P \cap D$, $I_n = P$, and $I_i \subseteq P$, for each $i \in \{1, \ldots, n\}$.

Let $\Sigma_\exists$ denote the subset of Σ collecting all the rules that contain existentially quantified variables. Moreover, let $\kappa = (|\Sigma_\exists| + |q|) \cdot |sch(\Sigma)|$. Starting from S, it is possible to construct a sequence $S' = I'_0\langle \sigma'_1, h'_1 \rangle I'_1 \langle \sigma'_2, h'_2 \rangle I'_2 \ldots I'_m$ of chase steps such that: (i) each rule of $\Sigma_\exists$ is applied at most once; (ii) $m \leqslant \kappa + |\Sigma_\exists|$; (iii) $|I'_m| \leqslant \kappa$; (iv) $dom(I'_0) \leqslant |q|$, which implies that $|I'_0| \leqslant |q| \cdot |sch(\Sigma)|$. The construction mimics the one presented in the proof of Lemma 18.

C Proof of Lemma 10

In the proof of Lemma 10 (and also in the proof of Lemma 11 given in the next section) we implicitly use the central property of the type of an atom. In particular, the subtree of the *guarded chase forest* of D and Σ — the forest obtained from $chase(D, \Sigma)$ by keeping only the guards and their children — rooted at $\underline{a}$ is determined by $type(\underline{a}, D, \Sigma)$ (modulo renaming of nulls) (Calì *et al.* 2013, Theorem 5.16)[1]. This result implies that, given an atom $\underline{a} = p(t_1, \ldots, t_n, z)$, where z is a null value invented in $\underline{a}$, we can construct $type(a, D, \Sigma)$ from $\underline{a}$ and the restriction of $type(a, D, \Sigma)$ to $\{t_1, \ldots, t_n\}$.

Let us now give the proof of Lemma 10. We proceed by induction on the number of chase steps needed to generate $p(\mathbf{t})$. Let $chase^{[k]}(D, \Sigma)$ be the initial part of the chase obtained after k applications of the chase step. We first show that, for each $k \geqslant 0$, there exists h_k such that $p(\mathbf{t}) \in chase^{[k]}(D, \Sigma)$ implies $[\tau_{p(\mathbf{t})}](h_k(\mathbf{t}^1)) \in chase(D^\star, \Sigma^\star_G)$.

Base Step: Clearly, $chase^{[0]}(D, \Sigma) = D$. Consider an arbitrary atom $p(\mathbf{t}) \in D$. By definition of $D^\star$, the atom $[\tau_{p(\mathbf{t})}](\mathbf{t}^1)$ belongs to $D^\star$. Since $D^\star \subseteq chase(D^\star, \Sigma^\star_G)$, the claim follows with h_0 be the identity on $dom(D)$.

Inductive Step: Let $\underline{a} = r(\mathbf{v})$ be the atom obtained during the k-th application of the chase step by applying the rule $\sigma \in \Sigma$. Clearly, there exists a homomorphism μ that maps $body(\sigma)$ to $chase^{[k-1]}(D, \Sigma)$ and $\mu'(head(\sigma)) = \underline{a}$, where $\mu' \supseteq \mu_{|\mathbf{X}}$ with $\mathbf{X} = var(body(\sigma)) \cap var(head(\sigma))$. Assume that $\mu(guard(\sigma))$ is an atom of the form $p(\mathbf{t})$. By induction hypothesis, $[\tau_{p(\mathbf{t})}](h_{k-1}(\mathbf{t}^1)) \in chase(D^\star, \Sigma^\star_G)$. Observe that $(\gamma \circ \mu)$ maps $body(\sigma)$ to $side(\tau_{p(\mathbf{t})})$ and $guard(\sigma)$ to $guard(\tau_{p(\mathbf{t})})$. Therefore, by definition of $\Sigma^\star_G$, a rule σ' of the form $[\tau_{p(\mathbf{t})}](\mathbf{X}) \rightarrow \exists Z\, [\tau'](\mathbf{Y}, Z)$ occurs in $\Sigma^\star_G$; for the formal definition of σ', see the construction of $\Sigma^\star_G$. Clearly, there exists a homomorphism λ that maps $body(\sigma')$ to $[\tau_{p(\mathbf{t})}](h_{k-1}(\mathbf{t}^1))$. Thus, an atom

[1]Notice that this result is stated in (Calì *et al.* 2013) for the more general class of weakly-guarded Datalog$^\exists$ where the notion of the cloud is needed. However, in the case of guarded Datalog$^\exists$ the notion of the cloud and the notion of the type coincide.

$\lambda'([\tau'](\mathbf{Y},Z)) = [\tau'](\mathbf{u})$ occurs in $chase(D^\star, \Sigma_G^\star)$, where $\lambda' \supseteq \lambda_{|\mathbf{X}\cap\mathbf{Y}}$. Assuming that Z' is the existentially quantified variable of σ, we define $h_k = h_{k-1} \cup \{\mu'(Z') \to \lambda'(Z)\}$; if σ has no existentially quantified variable, then $h_k = h_{k-1}$. It is obvious that h_k is well-defined since the symbol $\mu'(Z')$ does not occur in the domain of h_{k-1}. It is not difficult to verify that $[\tau'](\mathbf{u}) = [\tau_{r(\mathbf{v})}](h_k(\mathbf{v}^1))$.

Consequently, the claim follows with $h = \cup_{i=0}^{\infty} h_i$.

D Proof of Lemma 11

The proof is by induction on the number of chase step applications needed to generate the atom $[\tau](\mathbf{t})$. We show that, for each $k \geqslant 0$, there exists a homomorphism h_k such that $[\tau](\mathbf{t}) \in chase^{[k]}(D^\star, \Sigma_G^\star)$ implies $h_k(\tau(\mathbf{t})) \subseteq chase(D, \Sigma)$.

Base Step: Clearly, $chase^{[0]}(D^\star, \Sigma_G^\star) = D^\star$. Consider an arbitrary atom $[\tau](\mathbf{t}) \in D^\star$. By definition of $D^\star$, $[\tau](\mathbf{t}) \in D^+$. Since $D^+ = base(dom(D), sch(\Sigma)) \cap chase(D, \Sigma)$, the claim follows with h_0 be the identity on $dom(D)$.

Inductive Step: Assume that $[\tau](\mathbf{t})$ is obtained during the k-th application of the chase step by applying the TGD $\sigma \in \Sigma_G^\star$. Clearly, there exists a homomorphism μ that maps $body(\sigma)$ to $chase^{[k-1]}(D^\star, \Sigma_G^\star)$ and $\mu'(head(\sigma)) = [\tau](\mathbf{t})$, where $\mu' \supseteq \mu_{|\mathbf{X}}$ with $\mathbf{X} = var(body(\sigma)) \cap var(head(\sigma))$; let $\mu(body(\sigma)) = [\tau'](\mathbf{t}')$. By construction, there exists a TGD $\sigma' \in \Sigma$ of the form $\varphi(\mathbf{X}, \mathbf{Y}) \to \exists Z\, p(\mathbf{X}, Z)$ and a homomorphism λ such that $\lambda(body(\sigma')) \subseteq side(\tau')$ and $\lambda(guard(\sigma')) \subseteq guard(\tau')$. Thus, σ' is applicable with homomorphism $\nu = h_{k-1} \circ \gamma_{\mathbf{t}'} \circ \lambda$ during the construction of $chase(D, \Sigma)$, and the atom $\nu'(head(\sigma))$, where $\nu' \supseteq \nu_{|\mathbf{X}}$ is obtained. Assuming that Z' is the existentially quantified variable of σ, we define $h_k = h_{k-1} \cup \{\mu'(Z') \to \nu'(Z)\}$; if σ has no existentially quantified variable, then $h_k = h_{k-1}$. Since $\mu'(Z')$ does not occur in the domain of h_{k-1}, h_k is a well-defined substitution. It is not difficult to verify that $h_k(\tau(\mathbf{t})) \subseteq chase(D, \Sigma)$.

The claim follows with $h = \cup_{i=0}^{\infty} h_i$.

E Proof of Lemma 14

Fix a predicate $p \in sch(\Sigma)$. The maximum number of atoms of the form $p(t_1, \ldots, t_n)$, where $t_i \in [arity(p)]$, is $arity(p)^{arity(p)}$, i.e., the maximum number of $arity(p)$-tuples that can be constructed using $arity(p)$ symbols. Consider such an atom $\underline{a}$. Since the cardinality of the powerset of $base(dom(\underline{a}), sch(\Sigma)^-)$ is $2^{|sch(\Sigma)^-| \cdot |dom(\underline{a})|^w} \leqslant 2^{|sch(\Sigma)^-| \cdot arity(p)^w}$, we get that

$$|gtypes(p)| \leqslant arity(p)^{arity(p)} \cdot 2^{|sch(\Sigma)^-| \cdot arity(p)^w},$$

and (1) follows.

Clearly,

$$\begin{array}{ll} & |gtypes(\Sigma)| \\ \leqslant & \sum_{p \in sch(\Sigma)} |gtypes(p)| \\ \leqslant & \sum_{p \in sch(\Sigma)} arity(p)^{arity(p)} \cdot 2^{|sch(\Sigma)^-| \cdot arity(p)^w} \\ \leqslant & |sch(\Sigma)| \cdot w^w \cdot 2^{|sch(\Sigma)^-| \cdot w^w}, \end{array}$$

and (2) follows.

Finally, given a type $\tau \in gtypes(\Sigma)$, $|\tau|$ is bounded by the number of atoms that can be formed using predicates of $sch(\Sigma)$ and $arity(\tau)$ symbols, namely, $|sch(\Sigma)| \cdot arity(\tau)^w \leqslant |sch(\Sigma)| \cdot w^w$, and (3) follows.

F Proof of Proposition 15

We assume that $\Sigma_0 \in \mathsf{GUARDED}[c, d]$. Observe that the maximum number of existentially quantified variables that can appear in a rule of Σ_0 is $arity(\Sigma_0)$. Therefore, $|\Sigma| \leqslant arity(\Sigma_0) \cdot |\Sigma_0| \leqslant d \cdot |\Sigma_0| \in \mathcal{O}(|\Sigma_0|)$. Moreover, $|sch(\Sigma)| \leqslant |sch(\Sigma_0)| + arity(\Sigma_0) \cdot |\Sigma_0| \leqslant c + d \cdot |\Sigma_0| \in \mathcal{O}(|\Sigma_0|)$. Since the auxiliary predicates introduced during the normalization procedure do not appear in side atoms, we get that $sch(\Sigma)^- = sch(\Sigma_0)$, and thus $|sch(\Sigma)^-| \leqslant c$. It is important to say that $arity(\Sigma) = arity(\Sigma_0) \leqslant d$.

Let us now concentrate on $D^\star$. To construct $D^\star$ we need, for each atom $p(\mathbf{t}) \in D$, and for each $\tau \in gtypes(p)$ such that $guard(\tau) \simeq p(\mathbf{t})$, to check whether $D \cup \Sigma \models \underline{a}$, for each $\underline{a} \in \tau(\mathbf{t}^1)$. Importantly, the latter (atom entailment) check is feasible in polynomial time; this follows from the fact that atomic query answering under $\mathsf{GUARDED}[c, d]$ can be performed in polynomial time in $|dom(D)|$ and $|\Sigma| \in \mathcal{O}(|\Sigma_0|)$ (Calì *et al.* 2013). In the worst case, we need to apply this polynomial check $\sum_{p(\mathbf{t}) \in D} \sum_{\tau \in gtypes(p)} |\tau|$ times. By Lemma 14, and since $|sch(\Sigma)^-| \leqslant c$ and $arity(p) \leqslant d$, for each $p \in sch(\Sigma)$, eventually the crucial entailment check must be performed $|D| \cdot |sch(\Sigma)| \cdot 2^{c \cdot d^d} \cdot c \cdot d^{2d} \in \mathcal{O}(|D| \cdot |\Sigma_0|)$ times, and hence $D^\star$ is constructible in polynomial time.

We now consider the construction of $\Sigma^\star$. To construct the type generator $\Sigma_G^\star$ we need, for each rule $\sigma \in \Sigma$, and for each $\tau \in gtypes(\Sigma)$ where there exists a homomorphism h that maps $body(\sigma)$ to $atoms(\tau)$ and $guard(\sigma)$ to $guard(\tau)$, to construct a linear rule. In the worst case, we need to construct $|\Sigma| \cdot |gtypes(\Sigma)|$ linear rules. Hence, by Lemma 14, the maximum number of rules that must be constructed is $|\Sigma| \cdot |sch(\Sigma)| \cdot d^d \cdot 2^{c \cdot d^d} \in \mathcal{O}((|\Sigma_0|)^2)$. The construction of such a linear rule (apart from some easy steps) amounts to compute the set of atoms $complete(I, \Sigma)_{|sch(\Sigma)^-}$, where I is an instance of size bounded by $|\tau|$ with τ be the guarded type under consideration. In the worst case, to compute $complete(I, \Sigma)_{|sch(\Sigma)^-}$, we need to check if $I \cup \Sigma \models \underline{b}$, for each $\underline{b} \in base(dom(atoms(\tau)), sch(\Sigma)^-)$. Thus, the entailment check, which is feasible in polynomial time (Calì *et al.* 2013), is called $|base(dom(atoms(\tau)), sch(\Sigma)^-)| \leqslant |sch(\Sigma)^-| \cdot (|dom(atoms(\tau))|)^d \leqslant c \cdot (|dom(atoms(\tau))|)^d$ times. Observe that $|dom(atoms(\tau))| \leqslant d \cdot |\tau|$; thus, by Lemma 14, is bounded by $|sch(\Sigma)| \cdot d^{d+1}$. Therefore, the entailment check is called $c \cdot (|sch(\Sigma)| \cdot d^{d+1})^d \in \mathcal{O}((|\Sigma_0|)^d)$ times. Hence, $\Sigma^\star_G$ is constructible in polynomial time.

To construct the unfolder $\Sigma^\star_U$ we actually need to construct $|gtypes(\Sigma_0)|$ linear rules, each of which can be constructed in time $\mathcal{O}(1)$. Thus, by Lemma 14, the construction of $\Sigma^\star_U$ can be carried out in time $\mathcal{O}(1)$.

From the above analysis, we conclude that both $D^\star$ and $\Sigma^\star$ are constructible in polynomial time, and the claim follows.

G Proof of Lemma 18

Before proving that $\mathsf{STICKY}[\infty, d]$ enjoys the PWP, we need to introduce some useful notation. The *chase relation* (Calì *et al.* 2012b) of a database D and a set Σ of TGDs is a binary relation on atoms, denoted by $CR[D, \Sigma]$, which mimics all the chase derivations of the chase. More precisely, $CR[D, \Sigma]$ is the maximum subset of $chase(D, \Sigma) \times chase(D, \Sigma)$ such that $\langle \underline{a}, \underline{b} \rangle \in CR[D, \Sigma]$ implies that $\underline{b}$ is obtained from $\underline{a}$ via a chase step $I\langle \sigma, h \rangle I'$ where $I' \setminus I = \{b\}$ and $\underline{a} \in h(body(\sigma))$. The transitive closure of $CR[D, \Sigma]$ is denoted by $CR^+[D, \Sigma]$.

Consider a BCQ q, a database D, and a set $\Sigma \in \mathsf{STICKY}[\infty, d]$. Assume that $D \cup \Sigma \models q$. This implies that there is a homomorphism h such that $h(body(q)) = chase(D, \Sigma)$. Consider now the proof of $H = h(body(q))$, namely the set $P = H \cup \{\underline{a} \in chase(D, \Sigma) \mid \text{there exits } \underline{b} \in H \text{ such that } \langle \underline{a}, \underline{b} \rangle \in CR^+[D, \Sigma]\}$, which represents the finite part $P \subseteq chase(D, \Sigma)$ due to which H is generated. (Note that P is not necessarily a superset of D.) Once we now P, we can identify from the chase steps that have been applied to construct $chase(D, \Sigma)$ a sequence $S = I_0\langle \sigma_1, h_1 \rangle I_1 \langle \sigma_2, h_2 \rangle I_2 \ldots I_n$ of chase steps such that $I_0 = P \cap D$, $I_n = P$, and $I_i \subseteq P$, for each $i \in \{1, \ldots, n\}$.

Let $F = dom(P) \setminus dom(H)$. Given an atom $\underline{a} \in P$, we denote by $\underline{a}_\star$ the atom obtained from $\underline{a}$ by replacing each term of F with $\star$. Because of stickiness, all the terms of F do not participate in a join operation during the construction of P, and therefore can be seen as "don't care" terms. More specifically, for each chase step $I\langle \sigma, h \rangle I'$ such that $I' \setminus I \subseteq P$ and for each variable $X \in dom(body(\sigma))$, $h(X) \in F$ implies that X appears only once in $body(\sigma)$. This observation allows us to construct from S a new chase sequence S' of at most $\kappa = |sch(\Sigma)| \cdot (|q| \cdot d + 1)^d$ chase steps, which is a minimal proof of H.

We proceed by induction on the number $\ell \leqslant n$ of chase steps of S. In particular, we show how to construct a chase sequence $S' = I'_0\langle \sigma'_1, h'_1 \rangle I'_1 \langle \sigma'_2, h'_2 \rangle I'_2 \ldots I'_m$ with $m \leqslant \kappa$, and a function $f : \{0, \ldots, n\} \to \{0, \ldots, m\}$ such that:

1. $|I'_0| \leqslant \kappa$;
2. $I'_0 \subseteq I_0 \subseteq D$;
3. for each $i, j \in \{0, \ldots, \ell\}$, $j > i$ implies $f(j) \geqslant f(i)$;
4. for each $i \in \{0, \ldots, \ell\}$ and for each $\underline{a} \in I_i$, there exists $\underline{a}' \in I'_{f(i)}$ such that $\underline{a}_\star = \underline{a}'_\star$.

In fact, $I'_m \supseteq H$ will be the polynomial witness of q and we conclude that a witness of size $\mathcal{O}(|sch(\Sigma)| \cdot |q|^d)$ always exists.

Base Step: Consider the case of $\ell = 0$. Let $f(0) = 0$ and I'_0 be any maximal subset of I_0 such that, for each $\underline{a}, \underline{a}' \in I'_0$, $\underline{a} \neq \underline{a}'$ implies $\underline{a}_\star \neq \underline{a}'_\star$. Conditions 2–4 are satisfied by construction. Regarding condition 1, we observe that that maximum number of atoms occurring in I'_0 is at most $|sch(\Sigma)| \cdot (|dom(H)| + 1)^d$, which is bounded by κ.

Inductive Step: Let us assume that the above four conditions hold for some $\ell < n$. We now show how to proceed in the construction of S' and f in such a way that these conditions also hold at step $\ell + 1$. Let $\{\underline{a}\} = I_{\ell+1} \setminus I_\ell$. If $I'_{f(\ell)}$ contains an atom $\underline{a}'$ such that $\underline{a}'_\star = \underline{a}_\star$, then we define $f(\ell + 1) = f(\ell)$ and we do not modify S', namely we do not define any chase step from $I'_{f(\ell)}$ to $I'_{f(\ell)+1}$. Conversely, we define $f(\ell + 1) = f(\ell) + 1$ and we modify S' by adding the chase step $I'_{f(\ell)}\langle \sigma_{\ell+1}, h \rangle I'_{f(\ell)+1}$, where h is such that: $h(body(\sigma_{\ell+1})) \subseteq I'_{f(\ell)}$, $h|_{dom(H)} = h_{\ell+1}|_{dom(H)}$ and, for each atom $\underline{c} \in body(\sigma_{\ell+1})$, $h(\underline{c})_\star = h_{\ell+1}(\underline{c})_\star$. Of course, $h(head(\sigma_{\ell+1}))_\star = \underline{a}_\star$ holds.

H Proof of Proposition 20

Fix a database D and a set $\Sigma_0 \in \mathsf{TAME}[c, d]$. We are going to construct a database D^* and a set $\Sigma^* \in \mathsf{STICKY}[\infty, d]$ such that, for every BCQ q, $D \cup \Sigma_0 \models q$ iff $D^* \cup \Sigma^* \models q$. Let us call this new reduction TS.

Step 1: Normalization. The first step is to normalize Σ_0 in such a way that each rule contains at most one existentially quantified variable which occurs at the last position of the head-atom. Let $\Sigma = \bigcup_{\sigma \in \Sigma_0} \mathsf{N}(\sigma)$, where $\mathsf{N}(\sigma)$ is the normalization function defined in the GL reduction. $\square$

Given a set $\Sigma \in \mathsf{TAME}$, we denote by $sch(\Sigma)^-$ the predicates of Σ which occur in at least one side atom in $\Sigma[g]$; clearly $sch(\Sigma)^- \subseteq sch(\Sigma)$, but it is not necessarily true that $sch(\Sigma)^- \subseteq sch(\Sigma[g])$ since some side atoms can only be generated by rules of $\Sigma[s]$. Henceforth, a guarded type for a predicate $p \in sch(\Sigma[g])$ is always w.r.t. $sch(\Sigma)^-$, and we will simply write $gtypes(p)$ for the set $gtypes(p, sch(\Sigma)^-)$. Moreover, we will refer to $gtypes(sch(\Sigma[g]), sch(\Sigma)^-)$ as $gtypes(\Sigma[g])$.

Step 2: Database Construction. We define

$$D^\star \quad = \quad D \cup \left\{ [\tau](\mathbf{t}^1) \;\middle|\; \begin{array}{l} p(\mathbf{t}) \in D, \\ p \in \mathit{sch}(\Sigma[g]), \\ \tau \in \mathit{gtypes}(p), \\ \mathit{guard}(\tau) \simeq p(\mathbf{t}), \\ \tau(\mathbf{t}^1) \subseteq \mathit{complete}(D,\Sigma) \end{array} \right\},$$

where $[\tau]$, for some guarded type τ, is a new $\mathit{arity}(\tau)$-ary predicate not occurring in $\mathit{sch}(\Sigma)$, $\mathit{guard}(\tau) \simeq p(\mathbf{t})$ means that $\mathit{guard}(\tau)$ and $p(\mathbf{t})$ have the same equality type, and $\mathbf{t}^1$ is obtained from $\mathbf{t}$ by keeping only the first occurrence of each term occurring in $\mathbf{t}$. It is important to say that the atom entailment problem under tame rules is decidable (Gottlob *et al.* 2013), and thus the (finite) set $\mathit{complete}(D,\Sigma)$ can be explicitly computed. $\square$

Step 3: Rules Construction. The set $\Sigma^\star$ consists of the following three groups of rules:

1. The *type generator*, denoted $\Sigma^\star_G$, which is responsible for generating new guarded types from existing ones,
2. The *unfolder*, denoted $\Sigma^\star_U$, that is responsible for unfolding a derived guarded type τ, i.e., to explicitly construct the atoms over the original schema $\mathit{sch}(\Sigma_0)$ which are encoded in τ so that they can be queried directly, and
3. The *sticky part* $\Sigma^\star_S$ which incorporates $\Sigma[s]$ without any modification.

Let us now define formally $\Sigma^\star_G$ and $\Sigma^\star_U$; let $w = \mathit{arity}(\Sigma)$.

Step 3.1: Type Generator. For each $\sigma \in \Sigma[g]$ of the form

$$\mathit{body}(\sigma) \;\rightarrow\; \exists Z\, p(X_1,\ldots,X_n,Z),$$

and for each guarded type $\tau \in \mathit{gtypes}(\Sigma[g])$ for which there exists a homomorphism h such that $h(\mathit{body}(\sigma)) \subseteq \mathit{atoms}(\tau)$ and $h(\mathit{guard}(\sigma)) = \mathit{guard}(\tau)$, in $\Sigma^\star_G$ there exists the rule:

$$[\tau](V_1,\ldots,V_m) \;\rightarrow\; \exists Z\,[\tau'](W_1,\ldots,W_k,Z),$$

where $m = \mathit{arity}(\tau)$, and

- $\{V_1,\ldots,V_m\} \subseteq \mathit{var}(\mathit{body}(\sigma))$, and $h(V_i) = i$, for each $i \in [m]$,
- $\{W_\ell\}_{\ell\in[k]} \subseteq \{X_\ell\}_{\ell\in[n]}$ and $(h(W_1),\ldots,h(W_k)) = (h(X_1),\ldots,h(X_n))^1$,
- $\tau' = (\rho(\underline{a}), (\mathit{complete}(\rho(\{\underline{a}\} \cup S),\Sigma) \setminus \{\rho(\underline{a})\})_{|\mathit{sch}(\Sigma)^-})$, where $\underline{a}$ is the atom $p(h(X_1),\ldots,h(X_n),w+1)$ and S is the set of atoms $\Pi_{\{h(X_1),\ldots,h(X_n)\}}(\tau)$.

No other rules occur in $\Sigma^\star_G$. Note that, also in this case, to compute the completion, we use the atom entailment problem under tame rules (Gottlob *et al.* 2013). $\square$

Step 3.2: Unfolder. For each $\tau \in \mathit{gtypes}(\Sigma_0[g])$, if $\mathit{guard}(\tau)$ is of the form $p(t_1,\ldots,t_n)$, in $\Sigma^\star_U$ there exists the rule:

$$[\tau](V_1,\ldots,V_m) \;\rightarrow\; p(W_1,\ldots,W_n),$$

where

- $\{V_1,\ldots,V_m\} \subset \mathbf{V}$ and $(V_1,\ldots,V_m) = (W_1,\ldots,W_n)^1$,
- $t_i = t_j$ implies $W_i = W_j$, for each $i,j \in [n]$.

No other rules occur in $\Sigma^\star_U$. $\square$

Finally, to prove Proposition 20, only minor adaptations to the proofs of Lemma 10, Lemma 11, Proposition 12, Lemma 14, and Proposition 15 are required.

I Proof of Lemma 23

Since there exist classes for which query answering is PTIME-hard and at the same time they are P-combined first-order rewritable (e.g., $\mathsf{GUARDED}[c,d]$), it suffices to show that a PTIME-hard language does not enjoy the PWP. To this aim, we are going to show that the PWP implies that query answering is feasible in AC_0 in data complexity, and the claim follows since $\mathrm{AC}_0 \subsetneq$ PTIME. Consider a set Σ that falls in a Datalog$^\exists$ class which enjoys the PWP, and a BCQ q. As shown in (Gottlob and Schwentick 2012), a non-recursive Datalog rewriting q_Σ can be constructed such that $D \cup \Sigma \models q$ iff $D \models q_\Sigma$, for every database D where $D \supseteq \{\mathit{zero}(0), \mathit{one}(1)\} \cup \{\mathit{neq}(a,b) \mid a,b \in \mathit{dom}(D) \text{ and } a \neq b\}$. In other words, for every database D, q_Σ evaluated over D yields exactly the same result as q evaluated against D and Σ, as long as D contains two unary predicates *zero* and *one* which give access to the binary values 0 and 1, respectively, and also contains a binary predicate *neq* which stores all the pairs of distinct constants occurring in D.

From the above result, given a database D, a set Σ which falls in a class that enjoys the PWP, and a BCQ q, we get an AC_0 decision procedure (when the query and the set of rules are fixed) for the problem whether $D \cup \Sigma \models q$. More precisely, from q and Σ we can construct a first-order query $FO_{q,\Sigma}$ independently from D since the non-recursive Datalog query q_Σ can be equivalently rewritten as a first-order query. We also enrich D with the set of atoms $\{zero(0), one(1)\} \cup \{neq(a,b) \mid a,b \in dom(D) \text{ and } a \neq b\}$ in order to obtain D^+. Obviously, the addition of $zero(0)$ and $one(1)$ can be done in constant time, while the generation of $\{neq(a,b) \mid a,b \in dom(D) \text{ and } a \neq b\}$ can be done by evaluating a fixed first-order query over D which is feasible in AC_0. Finally, we evaluate $FO_{q,\Sigma}$ over D^+ which is feasible in AC_0 in data complexity, and the claim follows.

J SUCC is P-combined First-order Rewritable

Given a BCQ $q() \leftarrow \psi(W_1, \ldots, W_m)$, a database D and a set $\Sigma \in \mathsf{SUCC}$, we show how to rewrite in polynomial time q and Σ into a first-order query q_Σ in such a way that $D \cup \Sigma \models q$ iff $D \models q_\Sigma$. In fact, since our construction does not rewrite D, we prove that SUCC is not only P-combined first-order rewritable, but that it is even polynomial first-order rewritable. Before showing how to construct q_Σ, we need to introduce some straightforward notation.

Consider two (not necessarily distinct) constants $c_1, c_2 \in \mathbf{C}$, and two tuples $\mathbf{u}$ and $\mathbf{z}$ of constants from $\{c_1, c_2\}$. Assuming that c_1 precedes (or is equal to) c_2, we would like to write a first-order expression for characterizing all the pairs $\mathbf{u}, \mathbf{z}$ where $\mathbf{u}$ lexicographically precedes (or is equal to) $\mathbf{z}$ w.r.t. c_1 and c_2. More specifically, given a tuple of terms $(X, Y, \mathbf{U}, \mathbf{Z})$ where $\mathbf{U} = U_1, \ldots, U_n$ and $\mathbf{Z} = Z_1, \ldots, Z_n$ are tuples of constants and variables, we define the following first-order expression:

$$precEq(X, Y, \mathbf{U}, \mathbf{Z}) = \left(\bigwedge_{j \in [n]} (U_j = Z_j) \right) \vee \left(\bigvee_{i \in [n-1]} \left(U_{i+1} = X \ \wedge \ Z_{i+1} = Y \ \wedge \ \bigwedge_{j \in [i]} (U_j = Z_j) \right) \right).$$

Intuitively, if $X = 0$, $Y = 1$, $\mathbf{U} = (0,0,0,1,1)$ and $\mathbf{Z} = (0,0,1,0,1)$, then $precEq(X, Y, \mathbf{U}, \mathbf{Z})$ is true and we say that $\mathbf{U}$ precedes $\mathbf{Z}$ w.r.t. X and Y. In this example, for $i = 2$, we have a disjunct in the above formula that evaluates to true.

Consider an atom $\underline{a}$ of the form $num(X, Y, Z_1, \ldots, Z_n)$, where $(X, Y, Z_1, \ldots, Z_n)$ is a tuple of terms (possibly with repetitions). We define the rewriting of $\underline{a}$, denoted by $rew(\underline{a})$, as the following first-order formula:

$$rew(\underline{a}) = \exists U_1 \ldots \exists U_n \bigvee_{i \in \{0,\ldots,n\}} \left(\psi_1^i \ \wedge \ \psi_2^i \ \wedge \ \psi_3^i \ \wedge \ \psi_4^i \ \wedge \ precEq(X, Y, U_1, \ldots, U_{n-i}, Z_{i+1}, \ldots, Z_n) \right)$$

where

- ψ_1^i is $p(X, Y, Z_1, \ldots, Z_i, U_1, \ldots, U_{n-i})$ if $i \geqslant 1$, or $p(X, Y, U_1, \ldots, U_n)$ otherwise;
- ψ_2^i is $(Z_i \neq X) \wedge (Z_i \neq Y)$ if $i \geqslant 1$, or $\top$ otherwise;
- ψ_3^i is $\bigwedge_{j \in [n-i]} (U_j = X \vee U_j = Y)$;
- ψ_4^i is $\bigwedge_{j \in \{i+1,\ldots,n\}} (Z_j = X \vee Z_j = Y)$;

Finally, we have:

$$q_\Sigma = \exists W_1 \ldots \exists W_m \bigwedge_{\underline{a} \in body(q)} rew(\underline{a})$$

Of course, q_Σ can be constructed in polynomial time.