Making Cross Products and Guarded Ontology Languages Compatible

Pierre Bourhis[1], **Michael Morak**[2] **and Andreas Pieris**[3]

[1]CRIStAL, CNRS & Université Lille 1, France
[2]Institute of Information Systems, TU Wien, Austria
[3]School of Informatics, University of Edinburgh, UK

pierre.bourhis@univ-lille1.fr, morak@dbai.tuwien.ac.at, apieris@inf.ed.ac.uk

Abstract

Cross products form a useful modelling tool that allows us to express natural statements such as "elephants are bigger than mice", or, more generally, to define relations that connect every instance in a relation with every instance in another relation. Despite their usefulness, cross products cannot be expressed using existing guarded ontology languages, such as description logics (DLs) and guarded existential rules. The question that comes up is whether cross products are compatible with guarded ontology languages, and, if not, whether there is a way of making them compatible. This has been already studied for DLs, while for guarded existential rules remains unanswered. Our goal is to give an answer to the above question. To this end, we focus on the guarded fragment of first-order logic (which serves as a unifying framework that subsumes many of the aforementioned ontology languages) extended with cross products, and we investigate the standard tasks of satisfiability and query answering. Interestingly, we isolate relevant fragments that are compatible with cross products.

1 Introduction

Ontology-based data access (OBDA) has recently emerged as a promising application of AI technologies in information management systems [Poggi *et al.*, 2008]. Actually, OBDA refers to the utilization of ontologies for providing a unified conceptual view of various data sources. Users can then pose their queries solely in the schema provided by the ontology, abstracting away from the specifics of the sources. In this setting, the actual user query and the ontology can be interpreted as two components of one composite query, called *ontology-mediated query* (OMQ) [Bienvenu *et al.*, 2014]. Thus, OBDA is often realized as the problem of answering OMQs.

In the OMQ setting, the ontology is a set of first-order sentences, while the query is a union of conjunctive queries. Of course, in its full generality, the problem of evaluating OMQs is very hard (in fact, undecidable) due to the very high expressiveness of first-order logic. This gave rise to a flourishing research activity in the KR community towards the discovery of ontology languages that achieve the right balance between expressivity and complexity. Consequently, a significant range of ontology languages, which vary in syntax, expressivity and complexity, has been developed. Two prominent families of languages, obtained from this extensive effort, are *description logics* (DLs) and *existential rules* (a.k.a. *Datalog*$^{\pm}$ *rules*).

Many of the existing DL-based and rule-based ontology languages guarantee good computational and model-theoretic properties by implicitly posing some restrictions on the use of quantifiers. They are essentially defined through the relativisation of quantifiers by atomic formulas, an idea that goes back to the *guarded fragment* of first-order logic, introduced with the aim of explaining and generalizing the good properties of modal logic [Andréka *et al.*, 1998]. In fact, the guarded fragment serves as a unifying framework that subsumes many DL-based and rule-based ontology languages.

Although guardedness is a well-established paradigm that gives rise to highly expressive and robust ontology languages, the relativised nature of the permitted quantifiers excludes key modeling features that are very useful for knowledge representation purposes. Prominent examples of such features are *transitivity* and *cross products*. The former allows us to state that a binary relation is transitive; for example,

$$\forall x \forall y (\mathrm{Desc}(x, y) \ \wedge \ \mathrm{Desc}(y, z) \ \rightarrow \ \mathrm{Desc}(x, z)),$$

states that the descendant relation is transitive. The latter allows us to define relations that connect every instance in a relation with every instance in another relation; for example,

$$\forall x \forall y (\mathrm{Elephant}(x) \ \wedge \ \mathrm{Mouse}(y) \ \rightarrow \ \mathrm{BiggerThan}(x, y)),$$

states that elephants are bigger than mice by defining the binary relation $\mathrm{BiggerThan}$ that connects every instance in the class $\mathrm{Elephant}$ with every instance in the class Mouse.

The question whether the feature of transitivity is compatible with the guarded fragment, or with DL/rule-based subfragments of it, has attracted considerable attention in the logic and KR communities; see, e.g., [Ganzinger *et al.*, 1999; Szwast and Tendera, 2004; Eiter *et al.*, 2009; Gottlob *et al.*, 2013; Baget *et al.*, 2015; Amarilli *et al.*, 2016] (let us stress that the list is by no means exhaustive). On the other hand, the same question for cross products has received far less attention. To the best of our knowledge, the only work that explicitly studies this question is [Rudolph *et al.*, 2008], where the *concept product* is investigated as an expressive feature for

DLs. The above cross product is actually a concept product, which, in DL syntax, is written as

$$\text{Elephant} \times \text{Mouse} \sqsubseteq \text{BiggerThan}.$$[1]

The main message of [Rudolph *et al.*, 2008] is that concept products are, in general, compatible with DLs. Notice that the above work focuses on the problem of satisfiability, that is, given an ontology, to decide whether it has at least one model. However, by exploiting techniques developed in [Rudolph *et al.*, 2008], one can easily show that expressive DLs are compatible with concept products even for OMQ evaluation purposes. To sum up, we know that satisfiability for the highly expressive DL $\mathcal{SROIQ}$ extended with concept products is decidable, and OMQ evaluation for $\mathcal{ALCHOI}$ extended with concept products is also decidable.

The goal of this work is to answer the question whether cross products are compatible with existing guarded-based ontology languages that deviate from DLs, both for satisfiability and OMQ evaluation purposes. To this end, we focus on the guarded fragment of first-order logic, which, as said above, serves as a unifying framework that subsumes many of the existing ontology languages that rely on guardedness, and we investigate the reasoning tasks of satisfiability and OMQ evaluation. Our contributions can be summarized as follows:

► After extending the guarded fragment with cross products, we focus on satisfiability. The decidability of the problem depends on the arity of the cross products, i.e., the arity of the relation defined by the cross product; e.g., the above cross product is binary. We show that the problem is undecidable, even for ternary cross products, but 2ExpTime-complete for binary cross products. Hence, the decidability/undecidability frontier lies between ternary and binary cross products.

► We then turn our attention to OMQ evaluation, and we show that the situation changes dramatically. The problem is undecidable, even if we focus on the two-variable fragment of the guarded fragment extended with binary cross products. Our main finding here is that, for OMQ evaluation purposes, cross products are not compatible with the guarded fragment.

► Given the negative result above, we investigate whether by restricting the query or the ontology language we can obtain decidability. We first restrict the query language and focus on acyclic UCQs. In this case, OMQ evaluation under the (full) guarded fragment extended with binary cross products is 2ExpTime-complete in the combined, and coNP-complete in the data complexity. We then focus on arbitrary UCQs, and show that OMQ evaluation under guarded existential rules with binary cross products is 2ExpTime-complete in the combined, and PTime-complete in the data complexity; the latter holds even for frontier-guarded existential rules, which are incomparable to the guarded fragment. The above results show that, in some relevant cases, binary cross products can be considered without paying a price in terms of complexity.

2 Preliminaries

Interpretations and Queries. Let $\mathbf{C}$, $\mathbf{N}$, and $\mathbf{V}$ be pairwise disjoint countably infinite sets of *constants*, (labeled) *nulls* and (regular) *variables*, respectively. Different constants represent different values (*unique name assumption*), while different nulls may represent the same value. A *schema* $\mathbf{S}$ is a finite set of relation symbols with associated arity; we write R/n to denote that R has arity n. A *term* is either a constant, null or variable. An *atom* over $\mathbf{S}$ is an expression $R(\bar{t})$, where $R/n \in \mathbf{S}$ and $\bar{t}$ is an n-tuple of terms. An *interpretation* over $\mathbf{S}$ is a (possibly infinite) set of atoms over $\mathbf{S}$ that contains only constants and nulls, while a *database* over $\mathbf{S}$ is a finite interpretation over $\mathbf{S}$ that contains only constants. We may call an interpretation and a database over $\mathbf{S}$ an $\mathbf{S}$-interpretation and $\mathbf{S}$-database, respectively. The *domain* of an interpretation I, denoted $dom(I)$, is the set of all terms occurring in I.

A *conjunctive query* (CQ) over $\mathbf{S}$ is a formula of the form $q(\bar{x}) = \exists \bar{y}\big(R_1(\bar{v}_1) \wedge \cdots \wedge R_m(\bar{v}_m)\big)$, where each $R_i(\bar{v}_i)$ $(1 \leq i \leq m)$ is an atom without nulls over $\mathbf{S}$, each variable mentioned in the $\bar{v}_i$'s appears either in $\bar{x}$ or $\bar{y}$, and $\bar{x}$ are the free variables of q. If $\bar{x}$ is empty, then q is a *Boolean CQ*. As usual, the evaluation of CQs is defined in terms of homomorphisms. Let I be an instance and $q(\bar{x})$ a CQ as above. A *homomorphism* from q to I is a mapping h, which is the identity on $\mathbf{C}$, from the variables that appear in q to $dom(I)$ such that $R_i(h(\bar{v}_i)) \in I$, for each $1 \leq i \leq m$. The *evaluation of $q(\bar{x})$ over I*, denoted $q(I)$, is the set of all tuples $h(\bar{x})$ of constants such that h is a homomorphism from q to I. We denote by $\mathbb{CQ}$ the class of conjunctive queries. A *union of conjunctive queries* (UCQ) over $\mathbf{S}$ is a formula of the form $q(\bar{x}) := q_1(\bar{x}) \vee \cdots \vee q_n(\bar{x})$, where each $q_i(\bar{x})$ is a CQ. The *evaluation of $q(\bar{x})$ over I*, denoted $q(I)$, is the set of tuples $\cup_{1 \leq i \leq n} q_i(I)$. We denote by $\mathbb{UCQ}$ the class of UCQs.

Ontologies. An *ontology language* $\mathbb{L}$ is a (possibly infinite) set of first-order sentences that can mention constants and variables—nulls and function symbols are not allowed. An *ontology* Σ that falls in $\mathbb{L}$, or simply $\mathbb{L}$-*ontology*, is a finite set of sentences from $\mathbb{L}$. Concrete ontology languages, which are based on the guarded fragment, are discussed in the next section. An interpretation I is a *model of an ontology* Σ, written $I \models \Sigma$, if it satisfies all its sentences, and it is a *model of a database D*, written $I \models D$, if $D \subseteq I$.

Ontology-Mediated Queries. An *ontology-mediated query* (OMQ) is a pair $Q = (\Sigma, q(\bar{x}))$, where Σ is an ontology and q a UCQ. Given a database D, the *evaluation of Q over D* is defined as $\cap_{I \models D, I \models \Sigma} \{\bar{c} \in dom(D)^{|\bar{x}|} \mid \bar{c} \in q(I)\}$, i.e., the *certain answers* to q w.r.t. D and Σ, and is denoted $Q(D)$. We write $(\mathbb{L}, \mathbb{CQ})$ for the OMQ language that consists of all OMQs of the form (Σ, q), where Σ is an $\mathbb{L}$-ontology and $q \in \mathbb{CQ}$. The language $(\mathbb{L}, \mathbb{UCQ})$ is defined analogously.

Reasoning Tasks. We concentrate on the two main reasoning tasks that involve ontologies and OMQs, that is, *satisfiability* and *query evaluation*, respectively. Satisfiability simply asks whether a given ontology admits at least one model:

PROBLEM :	$\mathsf{Sat}(\mathbb{L})$
INPUT :	An $\mathbb{L}$-ontology Σ.
QUESTION :	Does Σ has at least one model?

As is customary when studying the complexity of the evaluation problem for a query language, we consider its associ-

[1] This is actually the title of [Rudolph *et al.*, 2008], i.e., "All Elphants are Bigger than All Mice".

ated decision problem, i.e., whether a tuple of constants belongs to the evaluation of a query over a database:

PROBLEM :	$\mathsf{Eval}(\mathbb{L}, \mathbb{Q})$
INPUT :	An OMQ $Q = (\Sigma, q(\bar{x})) \in (\mathbb{L}, \mathbb{Q})$, a database D, and $\bar{c} \in dom(D)^{\lvert\bar{x}\rvert}$.
QUESTION :	Does $\bar{c} \in Q(D)$?

For the evaluation problem, we focus on the standard complexity measures: *combined complexity*, calculated by considering both the database and the query as part of the input, and *data complexity*, where the query is fixed.

3 Guarded Fragment with Cross Products

The primary goal of this work is to perform an in-depth complexity analysis of the above crucial tasks in the presence of ontologies that fall in an extended version of the guarded fragment that allows cross products. Let us first recall the guarded fragment, and then extend it with cross products.

Guarded Fragment. The *guarded fragment* of first-order logic, introduced in [Andréka *et al.*, 1998], is a collection of first-order formulas with some syntactic restrictions on quantification patterns. We write $\mathbb{GF}$ for the smallest set of formulas (over a schema $\mathbf{S}$) that (i) contains all atoms over $\mathbf{S}$;[2] (ii) is closed under $\neg$, $\wedge$, $\vee$, $\rightarrow$; and (iii) if α is an atom containing all the variables of $(\bar{x} \cup \bar{y})$, and $\varphi \in \mathbb{GF}$ with free variables in $(\bar{x} \cup \bar{y})$, then $\forall \bar{x}(\alpha \rightarrow \varphi)$ and $\exists \bar{x}(\alpha \wedge \varphi)$ belong to $\mathbb{GF}$ as well; the atom α is called the *guard* of the quantifier. $\mathsf{Sat}(\mathbb{GF})$ has been studied in [Grädel, 1999], and $\mathsf{Eval}(\mathbb{GF}, \mathbb{UCQ})$ in [Bárány *et al.*, 2014]:[3]

Proposition 1 *The following holds:*

1. $\mathsf{Sat}(\mathbb{GF})$ *is* 2ExpTime*-complete.*
2. $\mathsf{Eval}(\mathbb{GF}, \mathbb{UCQ})$ *is* 2ExpTime*-complete in the combined, and* coNP*-complete in the data complexity.*

Adding Cross Products. We proceed to extend the guarded fragment with cross products. Let us first make the notion of cross product more precise. A *k-ary cross product* over a schema $\mathbf{S}$, where $k \geq 2$, is a first-order formula φ of the form

$$\forall x_1 \ldots \forall x_n \forall y_1 \ldots \forall y_m \\ (R(x_1, \ldots, x_n) \wedge S(y_1, \ldots, y_m) \rightarrow \\ T(x_{i_1}, \ldots, x_{i_\ell}, y_{j_1}, \ldots, x_{j_p})),$$

where the following conditions hold: R/n, S/m and T/k belong to $\mathbf{S}$, $x_i \neq x_j$ $(1 \leq i < j \leq n)$, $y_i \neq y_j$ $(1 \leq i < j \leq m)$, $\{x_1, \ldots, x_n\} \cap \{y_1, \ldots, y_m\} = \varnothing$, $1 \leq i_1 < \cdots < i_\ell \leq n$, and $1 \leq j_1 < \cdots < j_p \leq m$. Let us clarify that the arity of φ is the arity of the relation T that appears in the right-hand side of "$\rightarrow$". Moreover, $n, m > 0$, i.e., the relations that appear in the left-hand side of "$\rightarrow$" are at least unary; hence, the arity of a cross product is at least two. We call φ *full* if $x_1, \ldots, x_n = x_{i_1}, \ldots, x_{i_\ell}$ and $y_1, \ldots, y_m = y_{j_1}, \ldots, x_{j_p}$, i.e., a full cross product computes the cross product of two relations without projecting any of their positions.

[2] For technical clarity we exclude equalities; however, our results hold even for the guarded fragment with equalities.

[3] The work [Bárány *et al.*, 2014] does not explicitly consider OMQs, but its results can be effortlessly transferred to OMQs.

Example 1 Given two relations $R/3$ and $S/2$, we can compute their cross product via the full 5-ary cross product

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

Now, if we need their partial cross product focussing, e.g., on the first and third position of R, and the second position of S, we can use the (partial) ternary cross product

$$\forall x \forall y \forall z \forall v \forall w (R(x, y, z) \wedge S(v, w) \rightarrow T(x, z, w)),$$

which first applies a projection operator, and then computes the cross product of the projected positions. ∎

Let $\mathbb{CP}^k$ be the set of all i-ary cross products, where $2 \leq i \leq k$. We define $\mathbb{CP}$ as the set $\cup_{k \geq 2} \mathbb{CP}^k$, that is, the set of all cross products. The guarded fragment with cross products is defined as the set of first-order formulas $\mathbb{GF}^\times = \mathbb{GF} \cup \mathbb{CP}$. We also write $\mathbb{GF}^{\times_k}$, where $k \geq 2$, for the set $\mathbb{GF} \cup \mathbb{CP}^k$.

Partial vs. Full Cross Products. At this point, one may be tempted to think that partial cross products are more powerful than full cross products since the former can compute the cross product of some projected positions of the involved relations. This is not true since the projection can be simulated using guarded formulas. In the sequel, we assume, w.l.o.g., that $\mathbb{GF}^\times$-ontologies contain only full cross products.

4 Satisfiability of $\mathbb{GF}^\times$

Having the formal definition of $\mathbb{GF}^\times$ in place, we are now ready to proceed with our reasoning tasks. In this section, we focus on $\mathsf{Sat}(\mathbb{GF}^\times)$. Although $\mathsf{Sat}(\mathbb{GF})$ is decidable [Grädel, 1999], $\mathsf{Sat}(\mathbb{GF}^\times)$ is undecidable, even for ternary cross products. The reason for this negative result is because with $\mathbb{GF}^{\times_3}$ we can state that a binary relation is transitive. However, if we focus on binary cross products, then the problem in question is decidable, and, in particular, 2ExpTime-complete. Therefore, we can safely conclude that the frontier between decidability and undecidality of $\mathsf{Sat}(\mathbb{GF}^\times)$ lies between $\mathbb{GF}^{\times_2}$ and $\mathbb{GF}^{\times_3}$. In the rest of this section we discuss the above results.

4.1 Ternary Cross Products

We start by showing the following negative result:

Theorem 2 $\mathsf{Sat}(\mathbb{GF}^{\times_3})$ *is undecidable.*

The key observation underlying the proof of the above result is that the transitivity axiom

$$\varphi_{\mathsf{tr}} = \forall x \forall y \forall z (R(x, y) \wedge R(y, z) \rightarrow R(x, z)),$$

which states that the relation R is transitive, can be simulated by a set of formulas that fall in $\mathbb{GF}^{\times_3}$. This fact allows us to reduce the satisfiability problem for the *guarded fragment with transitivity* ($\mathbb{GF}^{\mathsf{tr}}$), which is an undecidable problem [Ganzinger *et al.*, 1999; Grädel, 1999], to $\mathsf{Sat}(\mathbb{GF}^{\times_3})$.

Consider a $\mathbb{GF}^{\mathsf{tr}}$-ontology Σ over a schema $\mathbf{S}$. Let Σ' be the ontology over $\mathbf{S}' \supseteq \mathbf{S}$ obtained from Σ by replacing each transitivity axiom that has the form of φ_{tr} above with

$$\begin{aligned} \psi_1 &= \forall x (\exists y (R(x, y)) \leftrightarrow R_1(x)) \\ \psi_2 &= \forall x \forall y \forall z ((R_1(x) \wedge R(y, z)) \leftrightarrow T_R(x, y, z)) \\ \psi_3 &= \forall x \forall y \forall z ((T_R(x, y, z) \wedge R(x, y)) \rightarrow R(x, z)), \end{aligned}$$

where R_1 and T_R do not belong to $\mathbf{S}$. It is easy to show that Σ and Σ' are equisatisfiable. Moreover, although $\{\psi_1, \psi_2, \psi_3\}$ does not directly fall in $\mathbb{GF}^{\times 3}$, it can be equivalently rewritten as a set of formulas that fall in $\mathbb{GF}^{\times 3}$, and Theorem 2 follows.

Cross Product Guards. $\mathsf{Sat}(\mathbb{GF}^{\mathsf{tr}})$ becomes decidable if we focus on *transitive guards*, i.e., the transitive relations appear only in guards [Ganzinger *et al.*, 1999; Szwast and Tendera, 2004]. Hence, one may be tempted to think that by focussing on *cross product guards*, i.e., forcing the relations that appear in the right-hand side of a cross product to appear only in guards, we ensure the decidability of $\mathsf{Sat}(\mathbb{GF}^{\times 3})$. This is not the case since the construction underlying Theorem 2 does not use a cross product relation outside a guard.

4.2 Binary Cross Products

Theorem 2 confirms that even ternary cross products are not compatible with $\mathbb{GF}$. However, binary cross products can be simulated by guarded formulas, which allows us to show that:

Theorem 3 $\mathsf{Sat}(\mathbb{GF}^{\times 2})$ *is* 2ExpTime*-complete.*

The lower bound is inherited from Proposition 1. We now discuss the main ingredients of the reduction to $\mathsf{Sat}(\mathbb{GF})$.

Normalization and the Chase. A $\mathbb{GF}^{\times 2}$-ontology Σ can be rewritten in polynomial time into an ontology Σ' that contains *only one* binary cross product of the form

$$\varphi_{\mathsf{cp}} \;=\; \forall x \forall y \,(\mathrm{Term}(x) \wedge \mathrm{Term}(y) \;\rightarrow\; \mathrm{Cross}(x, y))\,,$$

and each sentence of $\Sigma' \setminus \{\varphi_{\mathsf{cp}}\}$ is either of the form

$$\forall \bar{x} \forall \bar{y} (\varphi(\bar{x}, \bar{y}) \;\rightarrow\; \bigvee_i \exists \bar{z}_i \, \psi_i(\bar{x}, \bar{z}_i)), \tag{1}$$

where φ, ψ_i are conjunctions of atoms, and φ has an atom that contains all the variables $(\bar{x} \cup \bar{y})$, or

$$\forall \bar{x} (\varphi(\bar{x}) \;\rightarrow\; \bot), \tag{2}$$

where φ has an atom that contains all the variables $\bar{x}$, and $\bot$ denotes false. Sentences of the form (1) are known as *guarded disjunctive existential rules* (or *guarded disjunctive tuple-generating dependencies*), while sentences of the form (2) are known as *guarded negative constraints*. Although sentences of the form (1) and (2) do not directly fall in $\mathbb{GF}$, they can be transformed in linear time into $\mathbb{GF}$ sentences.

The crucial property of normalized ontologies Σ is that for satisfiability purposes it suffices to focus on a *canonical set of models*, which acts as a representative of all the other models of Σ. It can be constructed via the *disjunctive chase*, a natural generalization of the well-known chase procedure for existential rules. We assume the reader is familiar with the disjunctive chase; for details, see, e.g., [Bourhis *et al.*, 2016]. We write $dchase(\Sigma)$ for the set of models of Σ constructed by the disjunctive chase—we know that Σ is satisfiable iff there exists $I \in dchase(\Sigma)$ that satisfies all the negative constraints occurring in Σ. Moreover, we can refer to the subtree of an atom in a chase interpretation, a useful notion that we heavily exploit in our analysis. Consider an interpretation $I \in dchase(\Sigma)$. Roughly speaking, the subtree of an atom $\alpha \in I$, denoted T_α, is inductively defined as follows: (i) α is the root of T_α, and (ii) if an atom β is obtained from the atoms $\beta_1, \ldots, \beta_n$ by applying a guarded rule φ, where the witness β_i of the guard atom of φ belongs to T_α, then there exists an edge from β_i to β. In what follows, we fix an ontology Σ over a schema $\mathbf{S}$ obtained after normalizing a $\mathbb{GF}^{\times 2}$ ontology.

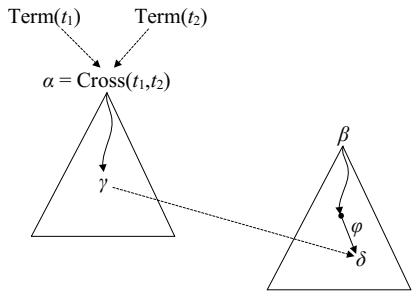

Figure 1: Violation of the guarded property.

Harmless Applications of the Cross Product. The goal is to translate Σ into an equisatisfiable $\mathbb{GF}$-ontology by simulating the applications of the cross product φ_{cp}, during the disjunctive chase, via guarded formulas. Interestingly, some of those applications are harmless, and they can easily be simulated via guarded rules. Assume that the atom $\alpha = \mathrm{Cross}(t_1, t_2)$ is obtained by applying φ_{cp}. Such an application of the cross product is *harmless* if the atoms $\mathrm{Term}(t_1)$ and $\mathrm{Term}(t_2)$, due to which φ_{cp} has been applied, have a common ancestor β that contains both t_1 and t_2. In this case, we can convert φ_{cp} into a guarded rule via a guard atom $G(x, y)$ that can be computed from β. We simply need to add to Σ the rules:

$$\forall x \forall y (G(x, y) \wedge \mathrm{Term}(x) \wedge \mathrm{Term}(y) \;\rightarrow\; \mathrm{Cross}(x, y))$$

and, for each $R/k \in \mathbf{S}$ and $1 \leq i, j \leq k$,

$$\forall x_1 \ldots \forall x_k (R(x_1, \ldots, x_k) \;\rightarrow\; G(x_i, x_j)).$$

From the above discussion, we can safely assume that whenever the cross product φ_{cp} is being applied due to $\mathrm{Term}(t_1)$ and $\mathrm{Term}(t_2)$, the latter atoms do not have a common ancestor that contains both t_1 and t_2. These are the applications of φ_{cp} that we call *dangerous*, which we discuss below.

Dangerous Applications of the Cross Product. The dangerous applications of φ_{cp} are responsible for the violation of the key model-theoretic property of the chase under guarded rules, which, for brevity, we call *guarded property*. Roughly, the guarded property states that if we focus on a certain interpretation $I \in dchase(\Sigma)$, the atoms that are involved in the application of a guarded rule φ during the construction of I, have a common ancestor in I, or, in other words, they are connected in I. This property does not hold in the presence of the cross product since a situation like the one illustrated in Figure 1 may occur. Assuming that $\alpha = \mathrm{Cross}(t_1, t_2)$, it is possible that an atom γ generated inside T_α acts as the witness of a non-guard atom during the application of a guarded rule φ, and the atom δ that belongs to T_β is generated. Notice that T_α and T_β are disjoint trees since the atoms of the form $\mathrm{Cross}(\cdot, \cdot)$, obtained via the cross product, give rise to new trees that are disconnected with the rest of I. Let us examine more carefully the reason that causes this situation.

Recall that the atoms $\mathrm{Term}(t_1)$ and $\mathrm{Term}(t_2)$ do not have a common ancestor that contains both t_1 and t_2. At the same time, it may be the case that $\mathrm{Term}(t_i)$, for $i \in \{1,2\}$, and β have a common ancestor; thus, β may contain the term t_i. Consequently, the atom γ, which belongs to T_α, may be involved in the application of the guarded rule φ that generates δ, which belongs to T_β; hence, the guarded property is violated since T_α and T_β are disjoint trees. The key observation, however, is that γ contains *only* the term t_i. Intuitively, this means that the only atoms of T_α that can affect some other disjoint tree T_β, or, they are "visible" from T_β, are atoms of the form $P(t,\ldots,t)$, where $t \in \{t_1,t_2\}$. This is the crucial observation that allows us to simulate the dangerous applications of φ_{cp} via guarded formulas.

Terms and Types. The above observation led us to introduce the notion of the type for a term in an interpretation. Given an interpretation I over a schema $\mathbf{S}$ and a term $t \in dom(I)$, the *type of* t *in* I is defined as the set of relations

$$type_I(t) \;=\; \{R \in \mathbf{S} \mid R(t,\ldots,t) \in I\}.$$

We call $I \in dchase(\Sigma)$ *type invariant* if the following holds: for each $\alpha = \mathrm{Cross}(t_1,t_2) \in I$ and $\beta = \mathrm{Cross}(t_3,t_4) \in I$ such that $type_I(t_1) = type_I(t_3)$ and $type_I(t_2) = type_I(t_4)$, T_α and T_β are isomorphic. Interestingly, if Σ is satisfiable, then there exists a type invariant $I \in dchase(\Sigma)$ that satisfies all the guarded negative constraints of Σ.[4] This fact, together with the observation that the atoms inside a tree $T_{\mathrm{Cross}(t,t')}$ that are visible from other disjoint trees depend solely on the types of t and t', brings us to the main idea underlying the transformation of Σ into an equisatisfiable $\mathbb{GF}$-ontology. The intention is to simulate a type invariant $I \in dchase(\Sigma)$ by generating, via $\mathbb{GF}$ formulas, trees that act as representatives of the isomorphic trees in I rooted at $\mathrm{Cross}(\cdot,\cdot)$ atoms.

Transformation of Σ into a $\mathbb{GF}$-ontology. We make use of the following auxiliary relations (in the sequel, $T \in 2^{\mathbf{S}}$): (i) $\mathrm{Partial}_T(t)$ means that T is a subset of the type of the term t in some chase interpretation, (ii) $\mathrm{Full}_T(t)$ means that T is the actual type of t in some chase interpretation, and (iii) $[T_1,T_2,T_3,T_4](t_1,t_2)$, where $T_1,\ldots,T_4 \in 2^{\mathbf{S}}$, $T_1 \subseteq T_2$ and $T_3 \subseteq T_4$, means that the partial type of t_1 and t_2 constructed so far is T_1 and T_3, respectively, while T_2 and T_4 is the actual type of t_1 and t_2, respectively, that eventually should be constructed. The atoms in (iii) are auxiliary atoms that are used to inductively generate the atom $[T,T,T',T'](t_1,t_2)$, which acts as the representative for the atoms $\mathrm{Cross}(t,t')$ occurring in a chase interpretation I such that $type_I(t) = T$ and $type_I(t') = T'$. We obtain Σ^+ by adding to Σ the following:

1. (**Guess the Types**) For each term, we guess its type in a certain chase interpretation via the disjunctive rule:

$$\forall x(\mathrm{Term}(x) \;\rightarrow\; \bigvee_{T \in 2^S} \mathrm{Full}_T(x)).$$

   Notice that we assume, w.l.o.g., that for each term t occurring in a chase interpretation I, $\mathrm{Term}(t)$ occurs in I; this is a consequence of our normalization procedure. We also add, for each $T \in 2^S$ and $R/k \notin T$:

$$\forall x(\mathrm{Full}_T(x) \wedge R(x,\ldots,x) \;\rightarrow\; \bot)$$

   to ensure that the guessing is correct. We also state that initially the partial type of a term is empty:

$$\forall x(\mathrm{Term}(x) \;\rightarrow\; \mathrm{Partial}_\varnothing(x)).$$

   Finally, we need to guarantee that the types are actually encoding the relations that are realized in a chase interpretation. For each $T \in 2^{\mathbf{S}}$ and relation $R/k \in T$:

$$\forall x(\mathrm{Partial}_T(x) \;\rightarrow\; R(x,\ldots,x)),$$

   and for each $T_1,T_2 \in 2^{\mathbf{S}}$ such that $T_1 \setminus T_2 = \{R/k\}$:

$$\forall x(\mathrm{Partial}_{T_2}(x) \wedge R(x,\ldots,x) \;\rightarrow\; \mathrm{Partial}_{T_1}(x)).$$

2. (**Construct the Representatives**) We generate the auxiliary atoms that eventually will lead to the representatives for Cross atoms as follows. For each $T_1,\ldots,T_4 \in 2^S$ such that $T_1 \subseteq T_2$ and $T_3 \subseteq T_4$:

$$\begin{aligned}\forall x\forall y(&\mathrm{Term}(x) \wedge \mathrm{Partial}_{T_1}(x) \wedge \mathrm{Full}_{T_2}(x) \wedge\\ &\mathrm{Term}(y) \wedge \mathrm{Partial}_{T_3}(y) \wedge \mathrm{Full}_{T_4}(y) \rightarrow\\ &\exists z\exists w([T_1,T_2,T_3,T_4](z,w) \wedge \mathrm{Cross}(z,w)\\ &\mathrm{Partial}_{T_1}(z) \wedge \mathrm{Full}_{T_2}(z) \wedge \mathrm{Partial}_{T_3}(w) \wedge \mathrm{Full}_{T_4}(w))).\end{aligned}$$

3. (**Type Propagation to Original Atoms**) Finally, we need to propagate the derived types for the representative terms back to the original terms. For each $T_1,\ldots,T_6 \in 2^{\mathbf{S}}$ such that $T_1 \subset T_5 \subseteq T_2$ and $T_3 \subset T_6 \subseteq T_4$, we add:

$$\begin{aligned}&\forall x\forall y\forall z([T_1,T_2,T_3,T_4](x,y) \wedge \mathrm{Partial}_{T_5}(x) \wedge\\ &\mathrm{Term}(z) \wedge \mathrm{Partial}_{T_1}(z) \wedge \mathrm{Full}_{T_2}(z) \;\rightarrow\; \mathrm{Partial}_{T_5}(z))\end{aligned}$$

   and

$$\begin{aligned}&\forall x\forall y\forall z([T_1,T_2,T_3,T_4](x,y) \wedge \mathrm{Partial}_{T_6}(y) \wedge\\ &\mathrm{Term}(z) \wedge \mathrm{Partial}_{T_3}(z) \wedge \mathrm{Full}_{T_4}(z) \;\rightarrow\; \mathrm{Partial}_{T_6}(z)).\end{aligned}$$

This completes the construction of Σ^+, and the desired ontology $\Sigma' = \Sigma^+ \setminus \{\varphi_{\mathsf{cp}}\}$. Let $\mathbf{S}'$ be the schema of Σ'.

Lemma 4 *It holds that (i) Σ and Σ' are equisatisfiable, (ii) Σ' can be constructed in exponential time, and (iii)* $\max\{k \mid R/k \in \mathbf{S}\} = \max\{k \mid R/k \in \mathbf{S}'\}$.

The above lemma, together with the fact that $\mathsf{Sat}(\mathbb{GF})$ is feasible in exponential time in the size of the ontology and the size of the schema, and in double-exponential time in the arity of the schema [Grädel, 1999], implies that $\mathsf{Sat}(\mathbb{GF}^{\times_2})$ is in 2ExpTime, and Theorem 3 follows.

Let us conclude by saying that in the case of bounded arity relations the exact complexity of $\mathsf{Sat}(\mathbb{GF}^{\times_2})$ is open. However, we can show that it is NExpTime-hard by exploiting the standard tiling problem for the exponential grid. This is an indication that (in the case of bounded arity) the presence of cross products increases the complexity since without cross products the problem is ExpTime-complete [Grädel, 1999].

[4] This is *not* true if unguarded negative constraints occur in Σ.

5 Evaluation of $(\mathbb{GF}^{\times}, \mathbb{UCQ})$ Queries

We now focus on OMQ evaluation. Let us first observe that $\mathsf{Eval}(\mathbb{GF}^{\times 3}, \mathbb{UCQ})$ is at least as hard as $\mathsf{Sat}(\mathbb{GF}^{\times 3})$. Indeed, given a $\mathbb{GF}^{\times 3}$-ontology Σ, assuming that $Q = (\Sigma, \exists x\, R(x))$ and $D = \{P(c)\}$, where R and P do not occur in Σ, then Σ is unsatisfiable iff $Q(D) \neq \varnothing$. This observation, together with Theorem 2, implies that $\mathsf{Eval}(\mathbb{GF}^{\times 3}, \mathbb{UCQ})$ is undecidable; in fact, the latter holds even for atomic queries. Thus, in order to obtain a positive result for OMQ evaluation, we should focus on $\mathbb{GF}^{\times 2}$-ontologies. One may expect that Eval behaves like Sat, and the frontier between decidability and undecidability lies between $(\mathbb{GF}^{\times 2}, \mathbb{UCQ})$ and $(\mathbb{GF}^{\times 3}, \mathbb{UCQ})$. Unfortunately, Eval is more complex than Sat, and we can establish a strong negative result: OMQ evaluation remains undecidable even for $(\mathbb{GF}_2^{\times 2}, \mathbb{CQ})$, where $\mathbb{GF}_2^{\times 2}$ denotes the *two-variable fragment* of $\mathbb{GF}^{\times 2}$, that is, the set of $\mathbb{GF}^{\times 2}$ sentences that can mention at most two variables. It is apparent that for OMQ evaluation purposes, $\mathbb{GF}$ and cross products are incompatible. However, we can regain decidability by restricting either the query language and focus on *acyclic UCQs*, or the ontology language and concentrate on *guarded existential rules*. The rest of this section is devoted to discuss in more details the above results.

5.1 Evaluation of $(\mathbb{GF}^{\times 2}, \mathbb{CQ})$ is Hard

We proceed to show that:

Theorem 5 $\mathsf{Eval}(\mathbb{GF}_2^{\times 2}, \mathbb{CQ})$ *is undecidable, even if we focus on unary and binary relations.*

The proof of the above result proceeds in two main steps:

1. We show that $\mathsf{Eval}(\mathbb{GF}_2^{\times 2}, \mathbb{UCQ})$ is undecidable by a reduction from the halting problem. Given a Turing machine M, we construct a database D, and an OMQ $Q = (\Sigma, q) \in (\mathbb{GF}_2^{\times 2}, \mathbb{UCQ})$, where Σ mentions only unary and binary relations, such that M halts iff $Q(D) \neq \varnothing$.
2. We then reduce $\mathsf{Eval}(\mathbb{GF}_2^{\times 2}, \mathbb{UCQ})$ to $\mathsf{Eval}(\mathbb{GF}_2^{\times 2}, \mathbb{CQ})$. It is known that such a reduction exists, which relies on the idea of encoding boolean operations via a set of database atoms; see, e.g., [Bourhis *et al.*, 2016]. However, this technique increases the arity of the underlying schema and the number of variables by one. By exploiting the binary cross products, we can devise a reduction that preserves both the arity and the number of variables.

There is an easier way to establish Theorem 5 by exploiting the undecidability of OMQ evaluation under the two-variable fragment of first-order logic ($\mathbb{FO}_2$) [Rosati, 2007]. In particular, by exploiting the binary cross product, we can define the so-called universal role, that is, the binary relation interpreted as the cross product of the domain with itself. Then, we can use the universal role to convert every formula in $\mathbb{FO}_2$ into a formula in $\mathbb{GF}_2^{\times 2}$. Nevertheless, we believe it is more insightful to provide a proof from first principles, which helps us to better understand the power of $(\mathbb{GF}_2^{\times 2}, \mathbb{CQ})$.

5.2 Acyclic Queries

A well-behaved class of UCQs is the one based on acyclicity. A UCQ is *acyclic* if each of its disjuncts is acyclic, i.e., its hypergraph is acyclic [Chekuri and Rajaraman, 2000]. Let $\mathbb{AUCQ}$ be the class of acyclic UCQs. Then:

Theorem 6 $\mathsf{Eval}(\mathbb{GF}^{\times 2}, \mathbb{AUCQ})$ *is* 2ExpTime*-complete in the combined, and* coNP*-complete in the data complexity.*

The above result relies on the fact that acyclic UCQs can be rewritten in linear time as UCQs that fall in $\mathbb{GF}$. This allows us to use the construction underlying Theorem 3 in order to reduce $\mathsf{Eval}(\mathbb{GF}^{\times 2}, \mathbb{AUCQ})$ to $\mathsf{Eval}(\mathbb{GF}, \mathbb{UCQ})$, and the desired upper bounds follow from Proposition 1. The lower bound for the combined complexity follows from the fact that $\mathsf{Sat}(\mathbb{GF})$ is 2ExpTime-hard, while the coNP-hardness is implicit in [Calvanese *et al.*, 2013].

5.3 Guarded Existential Rules

The other way to ensure decidability is by restricting the ontology language. We focus on guarded (non-disjunctive) existential rules. Recall that a rule $\forall \bar{x} \forall \bar{y}(\varphi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z}\, \psi(\bar{x}, \bar{z}))$ is guarded if φ has an atom that contains the variables $(\bar{x} \cup \bar{y})$. As said above, guarded rules do not directly fall in $\mathbb{GF}$, but they can be transformed into $\mathbb{GF}$ sentences. We write $\mathbb{GER}^{\times 2}$ for the set of all guarded rules and binary cross products.

Theorem 7 $\mathsf{Eval}(\mathbb{GER}^{\times 2}, \mathbb{UCQ})$ *is* 2ExpTime*-complete in the combined, and* PTime*-complete in the data complexity.*

The fact that guarded existential rules do not permit disjunction in the right-hand side, allows us to devise a reduction from $\mathsf{Eval}(\mathbb{GER}^{\times 2}, \mathbb{UCQ})$ to $\mathsf{Eval}(\mathbb{FGER}, \mathbb{UCQ})$, where $\mathbb{FGER}$ is the class of *frontier-guarded existential rules*, that is, rules of the form $\forall \bar{x} \forall \bar{y}(\varphi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z}\, \psi(\bar{x}, \bar{z}))$ with φ having an atom that contains all variables $\bar{x}$ (instead of $(\bar{x} \cup \bar{y})$). This reduction exploits a simplified version of the construction underlying Theorem 3. The main reason why such a simplified version exists is because the (non-disjunctive) chase under $\mathbb{GER}^{\times 2}$ constructs a single interpretation (not a set of interpretations) that is a type invariant. Having the above reduction in place, the desired upper bounds are inherited from $\mathsf{Eval}(\mathbb{FGER}, \mathbb{UCQ})$ [Baget *et al.*, 2011], while the lower bounds are inherited from $\mathsf{Eval}(\mathbb{GER}, \mathbb{CQ})$ [Calì *et al.*, 2013]. By following the same approach, we can show that Theorem 7 holds even if we consider the language $(\mathbb{FGER}^{\times 2}, \mathbb{UCQ})$.

6 Conclusions

Our main finding is that there are central guarded-based ontology languages that are compatible with binary cross products for satisfiability (e.g., the guarded fragment of first-order logic) and OMQ evaluation (e.g., (frontier-)guarded existential rules). An interesting research direction is to study query rewritabilty for the above ontology languages.

Acknowledgements

We would like to thank the anonymous reviewers for there useful comments. Bourhis is funded by CPER Nord-Pas de Calais/FEDER DATA Advanced data science and technologies 2015-2020 and the DeLTA project (ANR-16-CE40-0007). Morak is funded by the Austrian Science Fund (FWF), grant number Y698. Pieris is supported by the EPSRC Programme Grant EP/M025268/ "VADA: Value Added Data Systems – Principles and Architecture".

References

[Amarilli *et al.*, 2016] Antoine Amarilli, Michael Benedikt, Pierre Bourhis, and Michael Vanden Boom. Query answering with transitive and linear-ordered data. In *IJCAI*, pages 893–899, 2016.

[Andréka *et al.*, 1998] Hajnal Andréka, Johan van Benthem, and István Németi. Modal languages and bounded fragments of predicate logic. *J. Philosophical Logic*, 27:217–274, 1998.

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

[Baget *et al.*, 2015] Jean-François Baget, Meghyn Bienvenu, Marie-Laure Mugnier, and Swan Rocher. Combining existential rules and transitivity: Next steps. In *IJCAI*, pages 2720–2726, 2015.

[Bárány *et al.*, 2014] Vince Bárány, Georg Gottlob, and Martin Otto. Querying the guarded fragment. *Logical Methods in Computer Science*, 10(2), 2014.

[Bienvenu *et al.*, 2014] Meghyn Bienvenu, Balder ten Cate, Carsten Lutz, and Frank Wolter. Ontology-based data access: A study through disjunctive datalog, csp, and MMSNP. *ACM Trans. Database Syst.*, 39(4):33:1–33:44, 2014.

[Bourhis *et al.*, 2016] Pierre Bourhis, Marco Manna, Michael Morak, and Andreas Pieris. Guarded-based disjunctive tuple-generating dependencies. *ACM Trans. Database Syst.*, 41(4):27:1–27:45, 2016.

[Calì *et al.*, 2013] 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.

[Calvanese *et al.*, 2013] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Data complexity of query answering in description logics. *Artif. Intell.*, 195:335–360, 2013.

[Chekuri and Rajaraman, 2000] Chandra Chekuri and Anand Rajaraman. Conjunctive query containment revisited. *Theor. Comput. Sci.*, 239(2):211–229, 2000.

[Eiter *et al.*, 2009] Thomas Eiter, Carsten Lutz, Magdalena Ortiz, and Mantas Simkus. Query answering in description logics with transitive roles. In *IJCAI*, pages 759–764, 2009.

[Ganzinger *et al.*, 1999] Harald Ganzinger, Christoph Meyer, and Margus Veanes. The two-variable guarded fragment with transitive relations. In *LICS*, pages 24–34, 1999.

[Gottlob *et al.*, 2013] Georg Gottlob, Andreas Pieris, and Lidia Tendera. Querying the guarded fragment with transitivity. In *ICALP*, pages 287–298, 2013.

[Grädel, 1999] Erich Grädel. On the restraining power of guards. *J. Symb. Log.*, 64(4):1719–1742, 1999.

[Poggi *et al.*, 2008] 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.

[Rosati, 2007] Riccardo Rosati. The limits of querying ontologies. In *ICDT*, pages 164–178, 2007.

[Rudolph *et al.*, 2008] Sebastian Rudolph, Markus Krötzsch, and Pascal Hitzler. All elephants are bigger than all mice. In *DL*, 2008.

[Szwast and Tendera, 2004] Wieslaw Szwast and Lidia Tendera. The guarded fragment with transitive guards. *Ann. Pure Appl. Logic*, 128(1-3):227–276, 2004.