Dyadic Existential Rules

Georg Gottlob[1,2,†], Marco Manna[3,†] and Cinzia Marte[3,*,†]

[1]Department of Computer Science, University of Oxford, United Kingdom
[2]Faculty of Informatics, TU Wien, Austria
[3]Department of Mathematics and Computer Science, University of Calabria, Italy

Abstract
In the field of ontology-based query answering, existential rules (a.k.a. tuple-generating dependencies) form an expressive Datalog-based language to specify implicit knowledge. The presence of existential quantification in rule-heads, however, makes the main reasoning tasks undecidable. To overcome this limitation, in the last two decades, a number of classes of existential rules guaranteeing the decidability of query answering have been proposed. Unfortunately, such classes are typically based on different syntactic conditions imposing the development of different ad hoc reasoners. This paper introduces a novel general condition that allows to define, systematically, from any decidable class C of existential rules, a new class called Dyadic-C that enjoys the following properties: (i) it is decidable; (ii) it generalizes C; (iii) it keeps the same data complexity as C; and (iv) it can exploit any reasoner for query answering over C. Additionally, the paper proposes a simple and elegant syntactic condition that gives rise to the class Ward$^+$ generalizing the well-known decidable classes Shy and Ward, and being included in Dyadic-Shy.

Keywords
Existential rules, Datalog, ontology-based query answering, tuple-generating dependencies, computational complexity.

1. Introduction

In ontology-based query answering, a conjunctive query is typically evaluated over a logical theory consisting of a relational database paired with an ontology. Description Logics [1] and Existential Rules (a.k.a. tuple generating dependencies) [2] are the main languages used to specify ontologies. In particular, the latter are essentially classical datalog rules extended with existential quantified variables in rule-heads. The presence of existential quantification in the head of rules, however, makes query answering undecidable in the general case. To overcome this limitation, in the last two decades, a number of classes of existential rules —based on both semantic and syntactic conditions— that guarantee the decidability of query answering have been proposed. Concerning the semantic conditions, we recall *finite expansions sets*, *finite treewidth sets*, *finite unification sets*, and *strongly parsimonious sets* [3, 2, 4]. Each of these classes

Datalog 2.0 2022: 4th International Workshop on the Resurgence of Datalog in Academia and Industry, September 05, 2022, Genova - Nervi, Italy
*Corresponding author.
†These authors contributed equally.
georg.gottlob@cs.ox.ac.uk (G. Gottlob); manna@mat.unical.it (M. Manna); marte@mat.unical.it (C. Marte)
0000-0002-2353-5230 (G. Gottlob); 0000-0003-3323-9328 (M. Manna); 0000-0003-3920-8186 (C. Marte)

Main syntactic classes	Data Complexity	Combined Complexity
Weakly-(Fr)-Guarded [5, 6]	ExpTime-complete	2ExpTime-complete
(Fr)-Guarded [6]	PTime-complete	2ExpTime-complete
Weakly-Acyclic [7]	PTime-complete	2ExpTime-complete
Jointly-Acyclic [8]	PTime-complete	2ExpTime-complete
Datalog [9]	PTime-complete	ExpTime-complete
Shy, Ward [4, 10]	PTime-complete	ExpTime-complete
Protected [11]	PTime-complete	ExpTime-complete
Sticky-(Join) [12, 13]	AC_0	ExpTime-complete
Linear, Joinless [14, 15]	AC_0	PSpace-complete
Inclusion-Dependencies [16]	AC_0	PSpace-complete

Table 1
Computational complexity of query answering over the main concrete classes of existential rules.

encompasses a number of concrete classes based on syntactic conditions. Table 1 summarizes them and their computational complexity with respect to query answering. Unfortunately, the fact that such classes are typically based on different syntactic conditions imposes the development of different ad hoc reasoners.

This paper introduces a novel general condition that allows to define, systematically, from any class $\mathcal{C}$ of existential rules for which conjunctive query answering is decidable, a new class called Dyadic-$\mathcal{C}$ that enjoys the following properties: (i) it is decidable; (ii) it generalizes $\mathcal{C}$; (iii) it keeps the same data complexity as $\mathcal{C}$; and (iv) it can exploit any reasoner for query answering over $\mathcal{C}$. The key idea behind this new class is the existence of a *dyadic decomposition* of an ontology Σ consisting of a pair $(\Sigma_{\mathrm{HG}}, \Sigma_{\mathcal{C}})$ such that: (i) $\Sigma_{\mathrm{HG}} \cup \Sigma_{\mathcal{C}}$ is equivalent to Σ with respect to query answering; (ii) $\Sigma_{\mathcal{C}} \in \mathcal{C}$; and (iii) Σ_{HG} is a set of "head-ground" rules, which intuitively are rules generating only ground atoms when paired with $\Sigma_{\mathcal{C}}$. In analogy with the existing semantic classes, the union of all Dyadic-$\mathcal{C}$ classes are called *dyadic decomposable sets*.

Finally, the paper proposes a simple and elegant syntactic condition that gives rise to the concrete class Ward$^+$ generalizing the well-known decidable classes Shy and Ward, and being included in Dyadic-Shy.

2. Preliminaries

2.1. Basics on Relational Structures

Fix three pairwise disjoint lexicographically enumerable infinite sets Δ_C of *constants*, Δ_N of *nulls* (φ, φ_0, φ_1, ...), and Δ_V of *variables* (x, y, z, and variations thereof). Their union is denoted by Δ and its elements are called *terms*. For any integer $k \geq 0$, we may write $[k]$ for the set $\{1, ..., k\}$; in particular, as usual, if $k = 0$, then $[k] = \emptyset$. An *atom* $\underline{a}$ is an expression of the form $P(\mathbf{t})$, where $P = \mathsf{pred}(\underline{a})$ is a *(relational) predicate*, $\mathbf{t} = t_1, ..., t_k$ is a tuple of *terms* $k = \mathsf{arity}(\underline{a}) = \mathsf{arity}(P) \geq 0$ is the *arity* of both $\underline{a}$ and P, and $\underline{a}[i]$ denotes the i-th term $\mathbf{t}[i] = t_i$ of $\underline{a}$, for each $i \in [k]$. In particular, if $k = 0$, then $\mathbf{t}$ is the empty tuple and $\underline{a} = P()$. By $\mathsf{const}(\underline{a})$ (resp., $\mathsf{vars}(\underline{a})$) we denote the set of constants (resp., variables) occurring in $\underline{a}$. A *fact* is an atom that contains only constants. A *(relational) schema* $\mathbf{S}$ is a finite set of predicates, each with its own arity. The set of *positions* of $\mathbf{S}$, denoted $\mathsf{pos}(\mathbf{S})$, is defined as $\{P[i] \mid P \in \mathbf{S} \wedge 1 \leq i \leq \mathsf{arity}(P)\}$, where each $P[i]$ denotes the i-th *position* of P. A *(relational) structure* over $\mathbf{S}$ is any (possibly infinite) set of atoms using only predicates from $\mathbf{S}$. The *domain* of a structure S, denoted $\mathsf{dom}(S)$, is the set of all the terms occurring in S. An *instance* over $\mathbf{S}$ is any structure I over $\mathbf{S}$ such that $\mathsf{dom}(I) \subseteq \Delta_C \cup \Delta_N$. A *database* over $\mathbf{S}$ is any finite instance over $\mathbf{S}$ containing only facts. Consider a map $\mu : T_1 \to T_2$ where $T_1 \subseteq \Delta$ and $T_2 \subseteq \Delta$. Given a set T of terms, the *restriction* of μ with respect to T is the map $\mu|_T = \{t \mapsto \mu(t) : t \in T_1 \cap T\}$. An *extension* of μ is any map μ' between terms, denoted by $\mu' \supseteq \mu$, such that $\mu'|_{T_1} = \mu$. A *homomorphism* from a structure S_1 to a structure S_2 is any map $h : \mathsf{dom}(S_1) \to \mathsf{dom}(S_2)$ such that both the following hold: (i) if $t \in \Delta_C \cap \mathsf{dom}(S_1)$, then $h(t) = t$; and (ii) $h(S_1) = \{P(h(\mathbf{t})) : P(\mathbf{t}) \in S_1\} \subseteq S_2$.

2.2. Conjunctive Queries

A *conjunctive query* (CQ) q over a schema $\mathbf{S}$ is a (first-order) formula of the form

$$\langle \mathbf{x} \rangle \leftarrow \exists\, \mathbf{y}\ \Phi(\mathbf{x}, \mathbf{y}), \tag{1}$$

where $\mathbf{x}$ and $\mathbf{y}$ are tuples (often seen as sets) of variables such that $\mathbf{x} \cap \mathbf{y} = \emptyset$, and $\Phi(\mathbf{x}, \mathbf{y})$ is a conjunction (often seen as a set) of atoms using only predicates from $\mathbf{S}$. In particular, (i) $\mathsf{dom}(\Phi) \subseteq \mathbf{x} \cup \mathbf{y} \cup \Delta_C$, (ii) $z \in \mathbf{x} \cup \mathbf{y}$ implies that z occurs in some atom of Φ, (iii) $\mathbf{x}$ are the *output* variables of q, and (iv) $\mathbf{y}$ are the *existential* variables of q. To highlight the output variables, we may write $q(\mathbf{x})$ instead of q. The *evaluation* of q over an instance I is the set $q(I)$ of every tuple $\mathbf{t}$ of constants admitting a homomorphism $h_{\mathbf{t}}$ from $\Phi(\mathbf{x}, \mathbf{y})$ to I such that $h_{\mathbf{t}}(\mathbf{x}) = \mathbf{t}$. A *Boolean conjunctive query* (BCQ) is a CQ with no output variable, namely an expression of the form $\langle\rangle \leftarrow \exists\, \mathbf{y}\ \Phi(\mathbf{y})$. An instance I satisfies a BCQ q, denoted $I \models q$, if $q(I)$ is nonempty, namely $q(I)$ contains only the empty tuple $\langle\rangle$.

2.3. Tuple-Generating Dependencies

A *tuple-generating dependency* (TGD) σ —also known as *(existential) rule*— over a schema $\mathbf{S}$ is a (first-order) formula of the form

$$\Phi(\mathbf{x}, \mathbf{y}) \to\ \exists\, \mathbf{z}\ \Psi(\mathbf{x}, \mathbf{z}), \tag{2}$$

where $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ are pairwise disjoint tuples of variables, and both $\Phi(\mathbf{x}, \mathbf{y})$ and $\Psi(\mathbf{x}, \mathbf{z})$ are conjunctions (often seen as a sets) of atoms using only predicates from $\mathbf{S}$. In particular, (i) Φ (resp., Ψ) contains all and only the variables in $\mathbf{x} \cup \mathbf{y}$ (resp., $\mathbf{x} \cup \mathbf{z}$), (ii) constants (but not nulls) may also occur in σ, (iii) $\mathbf{x} \cup \mathbf{y}$ are the *universal* variables of σ, (iv) $\mathbf{z}$ are the *existential* variables of σ denoted by $\mathsf{var}_\exists(\sigma)$, and (v) $\mathbf{x}$ are the *frontier* variables of σ denoted by $\mathsf{front}(\sigma)$. In particular, if $\mathsf{var}_\exists(\sigma) = \emptyset$ and $|\mathsf{head}(\sigma)| = 1$, then σ is called *datalog* rule. We refer to $\mathsf{body}(\sigma) = \Phi$ and $\mathsf{head}(\sigma) = \Psi$ as the *body* and *head* of σ, respectively. With $\mathsf{hp}(\sigma)$ (resp., $\mathsf{bp}(\sigma)$) we denote the set of predicates in $\mathsf{head}(\sigma)$ (resp., $\mathsf{body}(\sigma)$). An *ontology* Σ is a set of rules. Without loss of generality, we assume that $\mathsf{vars}(\sigma_1) \cap \mathsf{vars}(\sigma_2) = \emptyset$, for each pair σ_1, σ_2 of rules in Σ. Operators $\mathsf{var}_\exists$, hp, and bp (defined for rules) naturally extend on ontologies. A class $\mathcal{C}$ of ontologies is any (typically infinite) set of TGDs fulfilling some syntactic or semantic conditions (see, for example, the classes shown in Table 1, some of which will be formally defined in the subsequent sections). In particular, Datalog is the class of ontologies containing only datalog rules. The *schema* of Σ, denoted $\mathsf{sch}(\Sigma)$, is the subset of $\mathbf{S}$ containing all and only the predicates occurring in Σ, whereas $\mathsf{arity}(\Sigma) = \max_{P \in \mathsf{sch}(\Sigma)} \mathsf{arity}(P)$. For simplicity of exposition, we write $\mathsf{pos}(\Sigma)$ instead of $\mathsf{pos}(\mathsf{sch}(\Sigma))$. An instance I satisfies a rule σ as in Equation 2, written $I \models \sigma$, if the existence of a homomorphism h from Φ to I implies the existence of a homomorphism $h' \supseteq h_{|\mathbf{x}}$ from Ψ to I. An instance I satisfies a set Σ of TGDs, written $I \models \Sigma$, if $I \models \sigma$ for each $\sigma \in \Sigma$.

2.4. Ontological Query Answering

Consider a database D and a set Σ of TGDs. A *model* of D and Σ is an instance I such that $I \supseteq D$ and $I \models \Sigma$. Let $\mathsf{mods}(D, \Sigma)$ be the set of all models of D and Σ. The *certain answers* to a CQ q w.r.t. D and Σ are defined as the set of tuples $\mathsf{cert}(q, D, \Sigma) = \bigcap_{M \in \mathsf{mods}(D,\Sigma)} q(M)$. Accordingly, for any fixed schema $\mathbf{S}$, two ontologies Σ_1 and Σ_2 over $\mathbf{S}$ are said $\mathbf{S}$-*equivalent* (in symbols $\Sigma_1 \equiv_{\mathbf{S}} \Sigma_2$) if, for each D and q over $\mathbf{S}$, it holds that $\mathsf{cert}(q, D, \Sigma_1) = \mathsf{cert}(q, D, \Sigma_2)$. The pair D and Σ satisfies a BCQ q, written $D \cup \Sigma \models q$, if $\mathsf{cert}(q, D, \Sigma) = \langle\rangle$, namely $M \models q$ for each $M \in \mathsf{mods}(D, \Sigma)$. Fix a class $\mathcal{C}$ of ontologies. The computational problem studied in this work —called $\mathsf{CQAns}(\mathcal{C})$— can be schematized as follows: *given a database D, a set $\Sigma \in \mathcal{C}$ of TGDs, a CQ $q(\mathbf{x})$, and a tuple $\mathbf{c} \in dom(D)^{|\mathbf{x}|}$, does $\mathbf{c} \in \mathsf{cert}(q, D, \Sigma)$ hold?* In what follows, we informally say that $\mathcal{C}$ is decidable whenever $\mathsf{CQAns}(\mathcal{C})$ is decidable. Note that $\mathbf{c} \in \mathsf{cert}(q, D, \Sigma)$ if, and only if, $D \cup \Sigma \models q(\mathbf{c})$, where $q(\mathbf{c})$ is the BCQ obtained from $q(\mathbf{x})$ by replacing, for each $i \in \{1, ..., |\mathbf{x}|\}$, every occurrence $\mathbf{x}[i]$ with $\mathbf{c}[i]$. Actually, the former problem is AC_0 reducible to the latter. Finally, while considering the computational complexity of $\mathsf{CQAns}(\mathcal{C})$, we recall the following convention: (i) *combined complexity* means that D, Σ, q, and $\mathbf{c}$ are given in input; and (ii) *data complexity* means that only D and $\mathbf{c}$ are given in input, whereas Σ and q are considered fixed.

2.5. The Chase Procedure

The chase procedure [17] is a tool exploited for reasoning with TGDs. Consider a database D and a set Σ of TGDs. Given an instance $I \supseteq D$, a *trigger* for I is any pair $\langle\sigma, h\rangle$, where $\sigma \in \Sigma$ is a rule as in Equation 2 and h is a homomorphism from $\mathsf{body}(\sigma)$ to I. Let $I' = I \cup h'(\mathsf{head}(\sigma))$, where $h' \supseteq h|_{\mathbf{x}}$ maps each $z \in \mathsf{var}_\exists(\sigma)$ to a "fresh" null $h'(z)$ not occurring in I such that $z_1 \neq z_2$ in $\mathsf{var}_\exists(\sigma)$ implies $h'(z_1) \neq h'(z_2)$. Such an operation which constructs I' from I is called *chase step* and denoted $\langle\sigma, h\rangle(I) = I'$. The *chase procedure* of $D \cup \Sigma$ is an exhaustive application of chase steps, starting from D, which produce a sequence $I_0 = D \subset I_1 \subset I_2 \subset \cdots \subset I_m \subset \ldots$ of instances in such a way that: (i) for each $i \geq 0$, $I_{i+1} = \langle\sigma, h\rangle(I_i)$ is a chase step obtained via some trigger $\langle\sigma, h\rangle$ for I_i; (ii) for each $i \geq 0$, if there exists a trigger $\langle\sigma, h\rangle$ for I_i, then there exists some $j > i$ such that $I_j = \langle\sigma, h\rangle(I_{j-1})$ is a chase step; and (iii) any *trigger* $\langle\sigma, h\rangle$ is used only once. We define $\mathsf{chase}(D, \Sigma) = \cup_{i \geq 0} I_i$. It is well know that $\mathsf{chase}(D, \Sigma)$ is a *universal model* of $D \cup \Sigma$, that is, for each $M \in \mathsf{mods}(D, \Sigma)$ there is a homomorphism from $\mathsf{chase}(D, \Sigma)$ to M. Hence, given a BCQ q it holds that $\mathsf{chase}(D, \Sigma) \models q \Leftrightarrow D \cup \Sigma \models q$. Finally, we recall that $\mathsf{chase}(D, \Sigma)$ can be decomposed into *levels* [12]: each atom of D has level $\gamma = 0$; an atom of $\mathsf{chase}(D, \Sigma)$ has level $\gamma + 1$ if, during its generation, the exploited trigger $\langle \sigma, h \rangle$ maps the body of σ via h to atoms whose maximum level is γ. We refer to the part of the chase up to level γ as $\mathsf{chase}^{\gamma}(D, \Sigma)$. Clearly, $\mathsf{chase}(D, \Sigma) = \cup_{\gamma > 0} \mathsf{chase}^{\gamma}(D \cup \Sigma)$.

3. Dyadic Decomposable Sets

In this section we introduce a novel general condition that allows to define, from any decidable class $\mathcal{C}$ of ontologies, a new decidable class called Dyadic-$\mathcal{C}$ enjoying desirable properties. We start with some preliminary notions. Then, we present the new notion of dyadic decomposition. Finally, we conclude with a computational analysis.

3.1. Preliminary Notions

This section fixes the basics that are needed to define dyadic decomposable sets, by providing a uniform notation for key existing notions: affected/invaded positions and attacked/protected/harmless/harmful/dangerous variables [4, 6, 18, 19]. Basically, these notions serve to separate positions in which the chase can introduce only constants from those where nulls might appear.

Definition 1. Consider an ontology Σ and a variable $z \in \mathsf{var}_{\exists}(\Sigma)$. A position $\pi \in \mathsf{pos}(\Sigma)$ is said to be *z-affected* (or *invaded* by z) if one of the following two properties holds:

1. there exists $\sigma \in \Sigma$ such that z appears in the head of σ at position π;
2. there exist $\sigma \in \Sigma$ and $x \in \mathsf{front}(\sigma)$ such that x occurs both in $\mathsf{head}(\sigma)$ at position π and in $\mathsf{body}(\sigma)$ at z-affected positions only.

Moreover, a position $\pi \in \mathsf{pos}(\Sigma)$ is *S-affected*, where $S \subseteq \mathsf{var}_{\exists}(\Sigma)$, if:

1. for each $z \in S$, π is z-affected; and
2. for each $z \in \mathsf{var}_{\exists}(\Sigma)$, if π is z-affected, then $z \in S$. □

Note that for every position π there exists a unique set S such that π is S-affected. We write $\mathsf{aff}(\pi)$ for this set S. Moreover, $\mathsf{aff}(\Sigma) = \{\pi \in \mathsf{pos}(\Sigma) \mid \mathsf{aff}(\pi) \neq \emptyset\}$, and $\mathsf{nonaff}(\Sigma) = \mathsf{pos}(\Sigma) \setminus \mathsf{aff}(\Sigma)$. We can now categorize the variables occurring in a conjunction of atoms with the following definition.

Definition 2. Given a TGD $\sigma \in \Sigma$ and a variable x in $\mathsf{body}(\sigma)$:

- if x occurs at positions $\pi_1, \ldots, \pi_n$ and $\bigcap_{i=1}^{n} \mathsf{aff}(\pi_i) = \emptyset$, then x is *harmless*,
- if x is not harmless, placed $S = \bigcap_{i=1}^{n} \mathsf{aff}(\pi_i)$, then it is *$S$-harmful*,
- if x is S-harmful and belongs to $\mathsf{front}(\sigma)$, then x is *S-dangerous*. □

Given a variable x that is S-dangerous, we write $\mathsf{dang}(x)$ for the set S. Hereinafter, the prefix S- is omitted when it is not necessary. Consider an ontology Σ. Given a rule $\sigma \in \Sigma$, we denote by $\mathsf{dang}(\sigma)$ (resp., $\mathsf{harmless}(\sigma)$ and $\mathsf{harmful}(\sigma)$) the dangerous (resp., harmless and harmful) variables in σ. These sets of variables naturally extend to the whole Σ by taking, for each of them, the union over all the rules of Σ.

3.2. Dyadic TGDs

In order to define the notion of Dyadic TGDs, we now introduce the concept of *head-ground* set of rules, being roughly “non-recursive” rules in which nulls are neither created nor propagated.

Definition 3. Consider a set Σ of TGDs. A set $\Sigma' \subseteq \Sigma$ is *head-ground* w.r.t. Σ if:

1. $\Sigma' \in \mathsf{Datalog}$;
2. each head atom of Σ' contains only harmless variables w.r.t. Σ;
3. $\mathsf{hp}(\Sigma') \cap \mathsf{bp}(\Sigma') = \emptyset$;
4. $\mathsf{hp}(\Sigma') \cap \mathsf{hp}(\Sigma \setminus \Sigma') = \emptyset$. □

The following example is given to better understand the above definition.

Example 1. Consider the next set of rules:

$$\begin{array}{lrcl} \sigma_1 : & R(x_1, y_1) & \rightarrow & \exists\, z_1, w_1\, Q(z_1, w_1) \\ \sigma_2 : & C(y_2), R(x_2, z_2) & \rightarrow & S(y_2, z_2) \\ \sigma_3 : & D(y_3, z_3), R(x_3, w_3) & \rightarrow & T(x_3, y_3) \\ \sigma_4 : & Q(x_4, y_4) & \rightarrow & \exists\, z_4 A(x_4, z_4) \\ \sigma_5 : & A(x_5, z_5), D(y_5, z_5) & \rightarrow & Q(x_5, y_5) \end{array}$$

A subset of head ground rule w.r.t. Σ is given by $\Sigma_{\mathrm{HG}} = \{\sigma_2, \sigma_3\}$. In fact, $\mathsf{harmless}(\Sigma)$ is the set $\{x_1, y_1, y_2, x_2, z_2, x_3, y_3, z_3, y_5, z_5\}$; hence, it is easy to check that (i) σ_2 and σ_3 are datalog rules; (ii) the head atoms of σ_2 and σ_3 contain only harmless variables; (iii) both predicates that appear in $\mathsf{head}(\sigma_2)$ and $\mathsf{head}(\sigma_3)$ do not occur in any body of Σ_{HG}, (iv) nor in the head of rules σ_1, σ_4 and σ_5. To the contrary, rules σ_1, σ_4 and σ_5 could not be in Σ_{HG}, since they violate Properties 2 and 3 of Definition 3. Hence, we observe that the set Σ_{HG} is maximal. □

We are now ready to formally introduce the class Dyadic-$\mathcal{C}$.

Definition 4. Consider a class $\mathcal{C}$ of TGDs, and a set Σ of TGDs. Let $\mathbf{S} = \mathsf{sch}(\Sigma)$. A pair $(\Sigma_{\mathrm{HG}}, \Sigma_{\mathcal{C}})$ of TGDs is a *dyadic decomposition* of Σ w.r.t. $\mathcal{C}$ if:

1. $\Sigma_{\mathrm{HG}} \cup \Sigma_{\mathcal{C}} \equiv_{\mathbf{S}} \Sigma$;
2. $\Sigma_{\mathcal{C}} \in \mathcal{C}$;
3. Σ_{HG} is head-ground w.r.t. $\Sigma_{\mathrm{HG}} \cup \Sigma_{\mathcal{C}}$; and
4. the head atoms of Σ_{HG} do not occur in Σ.

Dyadic-$\mathcal{C}$ is the class of all sets of TGDs that admit a dyadic decomposition w.r.t. $\mathcal{C}$. □

The union of all Dyadic-$\mathcal{C}$, with $\mathcal{C}$ being any decidable class of TGDs, forms what we call *dyadic decomposable sets*, which encompass and generalize any other existing decidable class, including those based on semantic conditions.

We now provide an example of a Dyadic-Shy ontology, where Shy [4] is a known decidable class. Before that, we recall the syntactic conditions underlying this class. A set Σ of TGDs is *shy* if, for each $\sigma \in \Sigma$ the following conditions both hold: (i) if a variable x occurs in more than one body atom, then x is harmless; (ii) for every pair of distinct dangerous variable z and w in different atoms, $\mathsf{dang}(z) \cap \mathsf{dang}(w) = \emptyset$. The class of all shy ontologies is called Shy.

Example 2. Let consider the following set Σ of TGDs:

$$\begin{array}{lrcl} \sigma_1: & R(x_1,y_1) & \rightarrow & \exists\, z_1\, T(z_1) \\ \sigma_2: & R(x_2,y_2) & \rightarrow & \exists\, z_2\, V(z_2) \\ \sigma_3: & S(x_3,y_3) & \rightarrow & \exists\, z_3\, P(z_3) \\ \sigma_4: & V(x_4) & \rightarrow & Q(x_4) \\ \sigma_5: & T(x_5), P(y_5), V(z_5), Q(z_5) & \rightarrow & U(x_5,y_5) \end{array}$$

where $\mathsf{harmless}(\Sigma) = \{x_1, y_1, x_2, y_2, x_3, y_3\}$, $\mathsf{harmful}(\Sigma) = \{x_4, x_5, y_5, z_5\}$, and $\mathsf{dang}(\Sigma) = \{x_4, x_5, y_5\}$. A dyadic decomposition of Σ w.r.t. Shy is given by $(\Sigma_{\mathrm{HG}}, \Sigma_{\mathcal{S}})$, where Σ_{HG} is:

$$\begin{array}{rcl} R(x_1,y_1) & \rightarrow & Aux_1(x_1,y_1) \\ R(x_2,y_2) & \rightarrow & Aux_2(x_2,y_2) \\ S(x_3,y_3) & \rightarrow & Aux_3(x_3,y_3) \\ V(x_4) & \rightarrow & Aux_4() \\ T(x_5), P(y_5), V(z_5), Q(z_5) & \rightarrow & Aux_5() \end{array}$$

and $\Sigma_{\mathcal{S}}$ is:

$$\begin{array}{rcl} Aux_1(x_1,y_1) & \rightarrow & \exists\, z_1\, T(z_1) \\ Aux_2(x_2,y_2) & \rightarrow & \exists\, z_2\, V(z_2) \\ Aux_3(x_3,y_3) & \rightarrow & \exists\, z_3\, P(z_3) \\ V(x_4), Aux_4() & \rightarrow & Q(x_4) \\ T(x_5), P(y_5), Aux_5() & \rightarrow & U(x_5,y_5) \end{array}$$

According to the above decomposition, the considered set Σ is Dyadic-Shy. □

Note that, in general, without any assumption on the specific class $\mathcal{C}$, we do not have any concrete means to construct a dyadic decomposition for an arbitrary Dyadic-$\mathcal{C}$ ontology. Concerning Dyadic-Shy, however, in Section 4, we define a syntactic subclass —called Ward$^+$— for which a dyadic decomposition is (easily) computable.

3.3. Decidability and Complexity

To provide an algorithm for computing the answers to a query q over a database D paired with a set $\Sigma \in$ Dyadic-$\mathcal{C}$, we are going to exploit the dyadic decomposition $(\Sigma_{\mathrm{HG}}, \Sigma_{\mathcal{C}})$ of Σ w.r.t. $\mathcal{C}$. Our idea is to reduce query answering over Dyadic-$\mathcal{C}$ to query answering over $\mathcal{C}$, the latter being decidable by assumption. To this aim, we first "complete" D by adding all the "auxiliary" ground consequences of $D \cup \Sigma_{\mathrm{HG}} \cup \Sigma_{\mathcal{C}}$, contained in the set

$$D' = \{\underline{a} \in \mathsf{chase}(D, \Sigma_{\mathrm{HG}} \cup \Sigma_{\mathcal{C}}) \mid \mathsf{pred}(\underline{a}) \in \mathsf{hp}(\Sigma_{\mathrm{HG}})\}. \quad (3)$$

Let $\mathcal{D} = D \cup D'$ be the result of this completion operation. We point out that D' actually contains only ground atoms, since the atoms generated during the chase procedure that derive from the head rules of Σ_{HG} cannot contain nulls by definition of head-ground rules. Accordingly, we evaluate the query q over $\mathcal{D} \cup \Sigma_{\mathcal{C}}$. Therefore, to show that Dyadic-$\mathcal{C}$ is decidable, it suffices to prove that D' is computable and also that $\mathsf{cert}(q, D, \Sigma) = \mathsf{cert}(q, \mathcal{D}, \Sigma_{\mathcal{C}})$ holds for any CQ q. We start by showing the correctness of our reduction.

Algorithm 1: Database Completion w.r.t. a fixed decidable class $\mathcal{C}$ of TGDs

Input: A dyadic decomposition $(\Sigma_{\mathrm{HG}}, \Sigma_{\mathcal{C}})$ of Σ w.r.t. $\mathcal{C}$
A database D
Output: The completed database $\mathcal{D}$

1 $\mathcal{D} := D$
2 $\tilde{D} := \emptyset$
3 **for** each rule of the form $\Phi(\mathbf{x}, \mathbf{y}) \rightarrow Aux_i(\mathbf{x})$ in Σ_{HG} **do**
4     $q := \langle \mathbf{x} \rangle \leftarrow \Phi(\mathbf{x}, \mathbf{y})$
5     $\tilde{D} = \tilde{D} \cup \{Aux_i(\mathbf{t}) \mid \mathbf{t} \in \mathsf{cert}(q, \mathcal{D}, \Sigma_{\mathcal{C}})\}$
6 **if** $(D \cup \tilde{D} \supset \mathcal{D})$ **then**
7     $\mathcal{D} := D \cup \tilde{D}$
8   **go to** instruction 2
9 **return** $\mathcal{D}$

Lemma 1. *Fix a decidable class $\mathcal{C}$ of TGDs. Consider a database D, a set $\Sigma \in \mathsf{Dyadic}\text{-}\mathcal{C}$ and a conjunctive query $q(\mathbf{x})$. Let $(\Sigma_{\mathrm{HG}}, \Sigma_{\mathcal{C}})$ be a dyadic decomposition of Σ w.r.t. $\mathcal{C}$ and let $\mathcal{D} = D \cup D'$, where $D' = \{\underline{a} \in \mathsf{chase}(D, \Sigma_{\mathrm{HG}} \cup \Sigma_{\mathcal{C}}) \mid \mathsf{pred}(\underline{a}) \in \mathsf{hp}(\Sigma_{\mathrm{HG}})\}$. Then, it holds that* $\mathsf{cert}(q, D, \Sigma) = \mathsf{cert}(q, \mathcal{D}, \Sigma_{\mathcal{C}})$.

Proof. Let $\mathbf{S} = \mathsf{sch}(\Sigma)$. Since, by hypothesis, $(\Sigma_{\mathrm{HG}}, \Sigma_{\mathcal{C}})$ is a dyadic decomposition of Σ, by Definition 4, it holds that $\Sigma \equiv_{\mathbf{S}} \Sigma_{\mathrm{HG}} \cup \Sigma_{\mathcal{C}}$. Hence, by definition of $\mathbf{S}$-equivalence, we have

$$\mathsf{cert}(q, D, \Sigma) = \mathsf{cert}(q, D, \Sigma_{\mathrm{HG}} \cup \Sigma_{\mathcal{C}}) \tag{4}$$

Fix any arbitrary $|\mathbf{x}|$-ary tuple $\mathbf{c}$ of constants. From Equation 4, we immediately get that $\mathbf{c} \in \mathsf{cert}(q, D, \Sigma)$ if, and only if, $\mathbf{c} \in \mathsf{cert}(q, D, \Sigma_{\mathrm{HG}} \cup \Sigma_{\mathcal{C}})$. Thus, let $q' = q(\mathbf{c})$, we now have

$$D \cup \Sigma \models q' \Leftrightarrow D \cup \Sigma_{\mathrm{HG}} \cup \Sigma_{\mathcal{C}} \models q'. \tag{5}$$

To show $\mathsf{cert}(q, D, \Sigma) \subseteq \mathsf{cert}(q, \mathcal{D}, \Sigma_{\mathcal{C}})$, it suffices to prove that $D \cup \Sigma \models q'$ implies $\mathcal{D} \cup \Sigma_{\mathcal{C}} \models q'$. Assume $D \cup \Sigma \models q'$ holds. By Equation 5, we know that $D \cup \Sigma_{\mathrm{HG}} \cup \Sigma_{\mathcal{C}} \models q'$ holds too. Hence, $\mathsf{chase}(D, \Sigma_{\mathrm{HG}} \cup \Sigma_{\mathcal{C}}) \models q'$. Since $D' \subset \mathsf{chase}(D, \Sigma_{\mathrm{HG}} \cup \Sigma_{\mathcal{C}})$, it holds that $\mathsf{chase}(D \cup D', \Sigma_{\mathrm{HG}} \cup \Sigma_{\mathcal{C}}) \models q'$. Since D' contains all the auxiliary ground consequences of Σ_{HG}, the latter becomes equivalent to $\mathsf{chase}(D \cup D', \Sigma_{\mathcal{C}}) \models q'$. Hence, $D \cup D' \cup \Sigma_{\mathcal{C}} \models q'$. Since, by hypothesis, $\mathcal{D} = D \cup D'$, we finally get that $\mathcal{D} \cup \Sigma_{\mathcal{C}} \models q'$.

To show that $\mathsf{cert}(q, D, \Sigma) \supseteq \mathsf{cert}(q, \mathcal{D}, \Sigma_{\mathcal{C}})$ it suffices to prove that if $\mathcal{D} \cup \Sigma_{\mathcal{C}} \models q'$, then $D \cup \Sigma \models q'$. Assume that $\mathcal{D} \cup \Sigma_{\mathcal{C}} \models q'$, hence $\mathsf{chase}(\mathcal{D}, \Sigma_{\mathcal{C}}) \models q'$. Since $\Sigma_{\mathcal{C}} \subseteq \Sigma_{\mathrm{HG}} \cup \Sigma_{\mathcal{C}}$, it holds that $\mathsf{chase}(\mathcal{D}, \Sigma_{\mathrm{HG}} \cup \Sigma_{\mathcal{C}}) \models q'$. By hypothesis, $D' \subseteq \mathsf{chase}(D, \Sigma_{\mathrm{HG}} \cup \Sigma_{\mathcal{C}})$; hence $\mathsf{chase}(D, \Sigma_{\mathrm{HG}} \cup \Sigma_{\mathcal{C}}) \models q'$, that is $D \cup \Sigma_{\mathrm{HG}} \cup \Sigma_{\mathcal{C}} \models q'$. Applying Equation 5, it holds that $D \cup \Sigma \models q'$, and hence the thesis. □

With Lemma 1 in place, we now design Algorithm 1 in order to iteratively construct the set $\mathcal{D} = D \cup D'$, with D' being the set defined by Equation 3. Roughly speaking, the first two instructions are required, respectively, to add D to $\mathcal{D}$ and to initialize a temporary set $\tilde{D}$ used to store ground consequences derived from Σ_{HG}. The rest of the algorithm is an iterative procedure that computes the answers (instruction 5) to the queries constructed from the rules of Σ_{HG} (instruction 4) and completes the initial database D (instruction 7) until no more auxiliary ground atoms can be produced (instruction 6). We point out that, in general, $\tilde{D} \subseteq D'$ holds; in particular, $\tilde{D} = D'$ holds in the last execution of instruction 7 or, equivalently, when the condition $D \cup \tilde{D} \supset \mathcal{D}$ examined at instruction 6 is false, since all the auxiliary ground atoms have been added to $\mathcal{D}$. We now prove that Algorithm 1 always terminates and correctly constructs $\mathcal{D}$.

Lemma 2. *Fix a decidable class $\mathcal{C}$ of TGDs. Consider a database D and a set Σ of* Dyadic-$\mathcal{C}$ *TGDs. Let $(\Sigma_{\mathrm{HG}}, \Sigma_{\mathcal{C}})$ be a dyadic decomposition of Σ, and D' be the set of ground auxiliary atoms defined as $\{\underline{a} \in \mathsf{chase}(D, \Sigma_{\mathrm{HG}} \cup \Sigma_{\mathcal{C}}) \mid \mathsf{pred}(\underline{a}) \in \mathsf{hp}(\Sigma_{\mathrm{HG}})\}$. Then, Algorithm 1 both terminates and computes $D \cup D'$.*

Proof sketch. We split the proof in two parts.

Termination. To prove the termination of Algorithm 1, it suffices to show that each instruction alone always terminates and that the overall procedure never falls into an infinite loop. First, observe that $|D'| \leq |\mathsf{hp}(\Sigma_{\mathrm{HG}})| \cdot d^{\mu}$, where $d = |\mathsf{const}(D)|$ and $\mu = \max_{P \in \mathsf{hp}(\Sigma_{\mathrm{HG}})} \mathsf{arity}(P)$. Instructions 1, 2, 4, 8 and 9 trivially terminates. Instructions 6 and 7 both terminate, since $\tilde{D} \subseteq D'$ always holds (see correctness below). Each time instruction 3 is reached, the **for**-loop simply scans the set Σ_{HG}, which is finite by definition. Concerning instruction 5, it suffices to observe that its termination relies on the termination of $\mathsf{CQAns}(\mathcal{C})$ —which is true by hypothesis— and on the fact that, for each query q, to construct the set $\{Aux_i(\mathbf{t}) \mid \mathbf{t} \in \mathsf{cert}(q, \mathcal{D}, \Sigma_{\mathcal{C}})\}$, the problem $\mathsf{CQAns}(\mathcal{C})$ must be solved at most d^{μ} times, being the maximum number of tuples $\mathbf{t}$ for which the check $\mathbf{t} \in \mathsf{cert}(q, \mathcal{D}, \Sigma_{\mathcal{C}})$ has to be performed. Since each instruction alone terminates, it remains to analyze the overall procedure. It contains two loops. The first, namely the **for**-loop at instruction 3, is not problematic; indeed, we shown that it locally terminates. The second one, namely the **go to**-loop, depends on the evaluation of the **if**-instruction, which can be executed at most $|D'|$ times. Thus, also the **go to**-loop does the same.

Correctness. We now claim that Algorithm 1 correctly completes the database. Let $\mathcal{D}$ be the output of Algorithm 1. Our claim is that $\mathcal{D} = D \cup D'$.

Inclusion 1 ($D \cup D' \subseteq \mathcal{D}$). Assume, by contradiction, that $D \cup D'$ contains some atom that does not belong to $\mathcal{D}$. This means that there exists some $j > 0$ such that both $\bar{D} = ((D \cup D') \cap \mathsf{chase}^{j-1}(D, \Sigma_{\mathrm{HG}} \cup \Sigma_{\mathcal{C}})) \subseteq \mathcal{D}$ and $((D \cup D') \cap \mathsf{chase}^{j}(D, \Sigma_{\mathrm{HG}} \cup \Sigma_{\mathcal{C}})) \setminus \mathcal{D} \neq \emptyset$ hold. Thus, there exists some $\alpha \in \mathsf{chase}^{j}(D, \Sigma_{\mathrm{HG}} \cup \Sigma_{\mathcal{C}})$ whose level is exactly j and that does not belong to $\mathcal{D}$. Let $\langle \sigma, h \rangle$ be the trigger used by the chase to generate α, where σ is of the form $\Phi(\mathbf{x}, \mathbf{y}) \to Aux(\mathbf{x})$. Clearly, h maps $\Phi(\mathbf{x}, \mathbf{y})$ to $\mathsf{chase}^{j-1}(D, \Sigma_{\mathrm{HG}} \cup \Sigma_{\mathcal{C}})$, and we also have that $\alpha = Aux(h(\mathbf{x}))$. Consider now the query $q = \langle \mathbf{x} \rangle \leftarrow \Phi(\mathbf{x}, \mathbf{y})$ constructed from σ by Algorithm 1 at instruction 4. Thus, $\mathsf{chase}^{j-1}(D, \Sigma_{\mathrm{HG}} \cup \Sigma_{\mathcal{C}}) \models q(h(\mathbf{x}))$ holds. Since $\bar{D} \subseteq \mathcal{D}$, we have that $\mathsf{chase}^{j-1}(D, \Sigma_{\mathrm{HG}} \cup \Sigma_{\mathcal{C}}) \subseteq \mathsf{chase}(\bar{D}, \Sigma_{\mathcal{C}}) \subseteq \mathsf{chase}(\mathcal{D}, \Sigma_{\mathcal{C}})$. Hence, $\mathsf{chase}(\mathcal{D}, \Sigma_{\mathcal{C}}) \models q(h(\mathbf{x}))$, namely $h(\mathbf{x}) \in \mathsf{cert}(q, \mathcal{D}, \Sigma_{\mathcal{C}})$ and, thus, $\alpha \in \mathcal{D}$, which is a contradiction.

Inclusion 2 ($\mathcal{D} \subseteq D \cup D'$). An argument analogous for the fist inclusion can be provided also for this second case. Here we assume, by contradiction, that $\mathcal{D}$ contains some atom that does not belong to $D \cup D'$. Let ℓ be the number of time instruction 7 of Algorithm 1 is executed. Let $\tilde{D}_0 = D$ and, for each $i \in [\ell]$, $\tilde{D}_i$ denote the specific $\tilde{D}$ appearing at instruction 7 the i-th time it is executed. Let $I_i = \tilde{D}_i \setminus \tilde{D}_{i-1}$, for each $i \in [\ell]$. It can be shown that there exists a sequence of chase applications leading to $\mathsf{chase}(D, \Sigma_{\mathrm{HG}} \cup \Sigma_{\mathcal{C}})$ such that, for each $i \in [\ell]$ and for each atom $\alpha \in I_i$, α is generated via such a chase sequence at a certain level (in general, different from i) being strictly greater than the level of every other atom contained in $\tilde{D}_{i-1}$. This is a contradiction since $\mathcal{D} = D \cup \tilde{D}_\ell$. □

It remains to show that Dyadic-$\mathcal{C}$ is decidable. We rely on Algorithm 1 together with Lemma 2 and Lemma 1 to state the following:

Theorem 1. *Let $\mathcal{C}$ be a decidable class of TGDs. Then,* Dyadic-$\mathcal{C}$ *is decidable.*

We point out that Algorithm 1 does not depend on the technique exploited for performing query answering over the class $\mathcal{C}$; hence, such external techniques can be used like a "black box". We can now study the complexity of $\mathsf{CQAns}(\mathcal{C})$ for an arbitrary decidable class $\mathcal{C}$.

Theorem 2. *Consider a decidable class $\mathcal{C}$ of TGDs. Assume that* $\mathsf{CQAns}(\mathcal{C})$ *is complete in data complexity for a certain complexity class $\mathbb{C}$. The following are true:*

1. *If* $\mathbb{C} \subseteq$ PTIME*, then* $\mathsf{CQAns}(\text{Dyadic-}\mathcal{C})$ *is in* PTIME *in data complexity;*
2. *If* $\mathbb{C} \supseteq$ PTIME*, then* $\mathsf{CQAns}(\text{Dyadic-}\mathcal{C})$ *is in* $\text{PTIME}^{\mathbb{C}}$ *in data complexity;*
3. *If* $\mathbb{C} \supseteq$ PTIME *is deterministic, then* $\mathsf{CQAns}(\text{Dyadic-}\mathcal{C})$ *is $\mathbb{C}$-complete in data complexity.*

Proof sketch. To prove the memberships of point 1 and 2, we rely on the complexity of Algorithm 1. In particular, let D be a database, $\Sigma \in \mathcal{C}$ an ontology, $(\Sigma_{\mathrm{HG}}, \Sigma_{\mathcal{C}})$ a dyadic decomposition of Σ, and D' the set defined in Equation 3. Moreover, let $d = |\mathsf{const}(D)|$ and $\mu = \max_{P \in \mathsf{hp}(\Sigma_{\mathrm{HG}})} \mathsf{arity}(P)$. Via Lemma 2, we shown that Algorithm 1 always terminates and it correctly constructs the completed database $\mathcal{D} = D \cup D'$. In particular, by neglecting the computational cost of the "trivial" instructions (i.e., 1–4 and 6–9), Algorithm 1 overall calls $|\Sigma_{\mathrm{HG}}| \cdot |\mathsf{hp}(\Sigma_{\mathrm{HG}})| \cdot d^{2\mu}$ times the problem $\mathsf{CQAns}(\mathcal{C})$. To reach the claimed bounds, first, we observe that $\mathcal{D}$ is polynomial in D. Moreover, since we are in data complexity, the following parameters are bounded: the maximum arity μ, the size of both Σ_{HG} and $\Sigma_{\mathcal{C}}$, as well as the size and the number of each query q constructed at instruction 4. Hence, Algorithm 1 calls polynomially many times $\mathsf{CQAns}(\mathcal{C})$. Finally, hardnesses of point 3 follow from the fact that for any deterministic class $\mathbb{C} \supseteq$ PTIME, we know that $\text{PTIME}^{\mathbb{C}} = \mathbb{C}$, and from the fact that Dyadic-$\mathcal{C}$ includes the class $\mathcal{C}$. □

A similar analysis can be performed in combined complexity. Here, however, we need further assumptions on: (i) the size of dyadic decompositions being equivalent to the ontologies of Dyadic-$\mathcal{C}$; and (ii) both the data and combined complexity of the class $\mathcal{C}$ under consideration.

4. Ward$^+$

We start this section recalling the syntactic condition of the class Ward [10, 19], useful for the comprehension of the new class Ward$^+$ that we will introduce below. We point out that

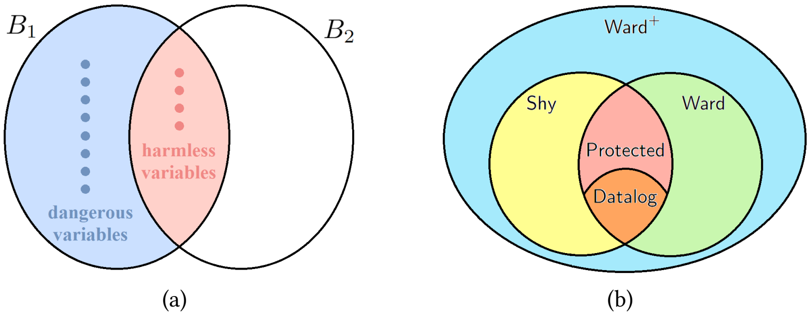

Figure 1: (a)Structure of a ward$^+$rule. (b)Syntactical relation among classes.

we state the wardedness condition according to Definition 2; hence, the class Ward presented here is actually larger than the original one. A set Σ of TGDs is *warded* if, for each $\sigma \in \Sigma$, there are no dangerous variables in $\mathsf{body}(\sigma)$, or there exists an atom $\alpha \in \mathsf{body}(\sigma)$, called a *ward*, such that:(i) all the dangerous variables in $\mathsf{body}(\sigma)$ occur in α, and (ii) each variable of $\mathsf{vars}(\alpha) \cap \mathsf{vars}(\mathsf{body}(\sigma) \setminus \{\alpha\})$ is harmless. The class of all warded ontologies is called Ward.

We now formally introduce the syntactic condition that gives rise to the class Ward$^+$ generalizing the well-known decidable classes Shy and Ward, and being included in Dyadic-Shy. Intuitively, the condition can be explained as follows. If σ is a ward$^+$rule, then $\mathsf{body}(\sigma)$ can be partitioned into two sets of atoms, B_1 and B_2, that share only harmless variables (see Figure 1(a)). Having in mind the notion of wardedness, the set B_1 can be seen as a "multi-ward" that contains all the dangerous variables and that, at the same time, satisfies the shyness conditions. The set B_2, instead, is any atoms conjunction that can share with B_1 only harmless variables. More formally, a set of ward$^+$ TGDs is defined as follows.

Definition 5. A set Σ of TGDs is ward$^+$ if, for each TGD $\sigma \in \Sigma$, there are no dangerous variables in $\mathsf{body}(\sigma)$, or there exists a partition $\{B_1, B_2\}$ of $\mathsf{body}(\sigma)$ such that:

1. B_1 contains all the dangerous variables
2. $\mathsf{vars}(B_1) \cap \mathsf{vars}(B_2)$ are harmless variables
3. for every pair of distinct dangerous variable z and w in different atoms, $\mathsf{dang}(z) \cap \mathsf{dang}(w) = \emptyset$
4. for every pair of distinct atoms $\underline{a}, \underline{b} \in B_1$, $\mathsf{vars}(\underline{a}) \cap \mathsf{vars}(\underline{b})$ are harmless variables.

We write Ward$^+$ for the class of all finite ward$^+$ sets of TGDs. □

Below we propose an example of an ontology that is in Ward$^+$and an example of one that does not belong to Ward$^+$, respectively.

Example 3. Consider the ontology Σ of Example 2. It easy to see that $\sigma_1, \sigma_2, \sigma_3$ and σ_4 are trivially ward$^+$ rules w.r.t. Σ, since they are rules with one single body atom, which cannot violate any conditions of Definition 5. Let us focus on rule σ_5. Since $\mathsf{dang}(\sigma_5) = \{x_5, y_5\}$, $\mathsf{harmful}(\sigma_5) = \{z_5\}$ and $\mathsf{harmless}(\sigma_5) = \emptyset$, there exists a partition of $\mathsf{body}(\sigma_5)$ into two set B_1, B_2, that satisfies Definition 5, where, $B_1 = \{T(x_5), P(y_5)\}$ and $B_2 = \{V(z_5), Q(z_5)\}$. Hence, $\Sigma \in$ Ward$^+$. □

Example 4. Let Σ be the following set of TGDs:

$$\begin{array}{lrcl} \sigma_1: & P(x_1) & \rightarrow & \exists\, y_1\, S(y_1) \\ \sigma_2: & S(x_2) & \rightarrow & \exists\, y_2, z_2\, R(y_2, x_2, z_2) \\ \sigma_3: & R(x_3, y_3, z_3), S(y_3) & \rightarrow & T(x_3, y_3, z_3) \end{array}$$

We pay particular attention to rule σ_3, since as previously explained, rules σ_1 and σ_2, that have only one body atom, do not go against the definition of ward$^+$ rule. Here, Condition 4 of Definition 5 is violated, since there is a join on variable y_3 that appears at positions $R[2]$ and $S[1]$, both y_1-affected. □

Now we show that Ward$^+$ strictly includes both Shy and Ward. According to Definition 5, the class Ward trivially coincides with the class Ward$^+$ if $|B_1| = 1$; thus, Ward $\subseteq$ Ward$^+$. On the other hand, if $|B_2| = \emptyset$, we have that Shy $\subseteq$ Ward$^+$, since the “multi-ward" satisfies the shyness conditions. We show that the latter relations are strict inclusions presenting a set of TGDs that belongs to Ward$^+$, but it is not both in Shy and Ward.

Example 5. Let Σ be the ontology introduced in Example 2. It is easy to see that $\sigma_1, \sigma_2, \sigma_3$ and σ_4 are both shy and warded rules w.r.t. Σ, since they are rules with one single body atom, which cannot violate any condition of the classes under consideration. However, rule $\sigma_5 \notin$ Ward, since the dangerous variables x_5 and y_5 are not contained in a single ward, and $\sigma_5 \notin$ Shy, since there is a join on the variable z_5 that is z_2-harmful. Hence, $\Sigma \in$ Ward$^+$(see Example 3), but $\Sigma \notin$ Shy and $\Sigma \notin$ Ward. □

Accordingly, it follows the next result.

Theorem 3. Ward$^+$ $\supset$ Shy $\cup$ Ward.

To show that Ward$^+$$\subset$ Dyadic-Shy, we have to prove the existence of a dyadic decomposition $(\Sigma_{\mathrm{HG}}, \Sigma_{\mathcal{S}})$ for every set Σ of ward$^+$ TGDs. Intuitively, the construction of a dyadic decomposition for a ward$^+$ set of TGDs takes advantage of the structure of a ward$^+$ rule. Indeed, by definition, a ward$^+$ rule can be always partitioned into two sets B_1 and B_2 of atoms, where B_1 is a conjunction of atoms that satisfies the shyness conditions, and B_2 is any atom conjunction. Roughly speaking, starting from a rule $\sigma \in$ Ward$^+$, a rule in Σ_{HG} has the same body of σ, while the head contains a “fresh" atom in which are propagated only harmless variables; this last atom, with the set B_1, is used to construct the body of a rule in $\Sigma_{\mathcal{S}}$, while the head contains the same atoms of head(σ) (see Example 2). Now, we formally prove the existence of a dyadic decomposition for a ward$^+$ set Σ of TGDs with respect to Shy.

Theorem 4. *For every* $\Sigma \in$ Ward$^+$, *there is a dyadic decomposition* $(\Sigma_{\mathrm{HG}}, \Sigma_{\mathcal{S}})$ *of* Σ *w.r.t.* Shy.

Proof. Let Σ be a set of ward$^+$ TGDs. To show the existence of a dyadic decomposition $(\Sigma_{\mathrm{HG}}, \Sigma_{\mathcal{S}})$ of Σ w.r.t. Shy, we propose the following procedure. Consider a ward$^+$ rule

$$\sigma:\ \Phi(\mathbf{x}, \mathbf{y}, \mathbf{z}), \Psi(\mathbf{z}, \mathbf{u}) \rightarrow \exists\, \mathbf{w}\ \Xi(\mathbf{x}, \mathbf{w}, \mathbf{z}),$$

where $\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{u}$ are pairwise disjoint, $\Phi(\mathbf{x}, \mathbf{y}, \mathbf{z})$, $\Psi(\mathbf{z}, \mathbf{u})$ and $\Xi(\mathbf{x}, \mathbf{w}, \mathbf{z})$ are conjunctions of atoms such that $\Phi(\mathbf{x}, \mathbf{y}, \mathbf{z}) = B_1$ and $\Psi(\mathbf{z}, \mathbf{u}) = B_2$ (according to Definition 5). Moreover, dang$(\sigma) = \{\mathbf{x}\}$, harmless$(\sigma) = \{\mathbf{z}\}$ and harmful$(\sigma) = \{\mathbf{x}, \mathbf{u}, \mathbf{y}\}$. Let $m_\sigma = |\text{head}(\sigma)|$, then we produce $m_\sigma + 2$ rules $\rho'(\sigma)$, $\rho''_0(\sigma)$, ..., $\rho''_{m_\sigma}(\sigma)$ such that:

$$
\begin{array}{rrcl}
\rho'(\sigma): & \Phi(\mathbf{x},\mathbf{y},\mathbf{z}),\Psi(\mathbf{z},\mathbf{u}) & \rightarrow & Aux'_\sigma(\mathbf{z}) \\
\rho''_0(\sigma): & \Phi(\mathbf{x},\mathbf{y},\mathbf{z}), Aux'_\sigma(\mathbf{z}) & \rightarrow & \exists\, \mathbf{w}\; Aux''_\sigma(\mathbf{x},\mathbf{w},\mathbf{z}) \\
\rho''_1(\sigma): & Aux''_\sigma(\mathbf{x},\mathbf{w},\mathbf{z}) & \rightarrow & \underline{a}_1(\mathbf{v}_1) \\
\vdots & & & \\
\rho''_{m_\sigma}(\sigma): & Aux''_\sigma(\mathbf{x},\mathbf{w},\mathbf{z}) & \rightarrow & \underline{a}_{m_\sigma}(\mathbf{v}_{m_\sigma})
\end{array}
$$

where $\mathbf{v}_i \subseteq \{\mathbf{x},\mathbf{w},\mathbf{z}\}$ for each $i \in \{1,\ldots,m_\sigma\}$, $\{\underline{a}_1(\mathbf{v}_1),\ldots,\underline{a}_{m_\sigma}(\mathbf{v}_{m_\sigma})\} = \Xi(\mathbf{x},\mathbf{w},\mathbf{z})$ and $Aux'_\sigma,\ Aux''_\sigma$ are fresh auxiliary predicates.

Now, we prove that $(\Sigma_{\mathrm{HG}}, \Sigma_{\mathcal{S}})$ is a dyadic decomposition for any ward$^+$ set of TGDs w.r.t. Shy, where

$$\Sigma_{\mathrm{HG}} = \bigcup_{\sigma\in\Sigma} \rho'(\sigma) \quad \text{and} \quad \Sigma_{\mathcal{S}} = \bigcup_{\sigma\in\Sigma\,\wedge\,0\leq j\leq m_\sigma} \rho''_j(\sigma).$$

According to Definition 4, the pair $(\Sigma_{\mathrm{HG}}, \Sigma_{\mathcal{S}})$ has to satisfies four properties. Property 4 is trivially fulfilled: head predicates of Σ_{HG} do not occur in Σ, since, by construction, Aux'_σ is a fresh auxiliary predicate introduced for each $\sigma \in \Sigma$. Property 3 states that the set Σ_{HG} is head-ground w.r.t. $\Sigma_{\mathrm{HG}} \cup \Sigma_{\mathcal{S}}$. This is true since, by construction, we have that: for each $\sigma \in \Sigma$, $\rho'(\sigma)$ is a datalog rule; $\mathsf{hp}(\Sigma_{\mathrm{HG}}) = \{Aux'_\sigma : \sigma \in \Sigma\}$, where each Aux'_σ is a predicate that does not occur neither in any body of Σ_{HG} nor in any head of $\Sigma_{\mathcal{S}}$ (i.e., $\mathsf{hp}(\Sigma') \cap \mathsf{bp}(\Sigma') = \emptyset$, and $\mathsf{hp}(\Sigma') \cap \mathsf{hp}(\Sigma \setminus \Sigma') = \emptyset$), and it contains only harmless variable. Now, we have to prove that Property 2 holds, i.e., $\Sigma_{\mathcal{S}} \in \mathsf{Shy}$. This is ensured by the fact that rule $\rho''_0(\sigma)$ is made by joining the set B_1 of σ (that has to satisfy the shyness conditions by definition), and the atom Aux'_σ, which contains only harmless variables, and hence, cannot violate any of the shyness conditions; moreover, each rule $\rho''_j(\sigma)$, for $j = 1,\ldots,m$, is linear, and therefore is a shy rule. Finally, Property 1, that is $\Sigma_{\mathrm{HG}} \cup \Sigma_{\mathcal{S}} \equiv_{\mathsf{sch}(\Sigma)} \Sigma$, follows by construction. □

Finally —by combining Theorem 2, Theorem 4, the fact that CQAns is PTime-complete (resp., ExpTime-complete) for both Shy and Ward in data (resp., combined) complexity, and the fact that Ward$^+$ admits dyidic decompositions of polynomial size— we can state the following result.

Theorem 5. *The following are true:*

1. Ward$^+ \subseteq$ Dyadic-Shy*;*
2. CQAns *is* PTime*-complete in data complexity over* Ward$^+$, Dyadic-Shy *and* Dyadic-Ward*;*
3. CQAns *is* ExpTime*-complete in combined complexity over* Ward$^+$.

5. Conclusion

Dyadic decomposable sets form a novel decidable class of TGDs that encompasses and generalizes all the existing (syntactic and semantic) decidable classes of TGDs. In the near feature, it would be interesting to identify more syntactic Dyadic-$\mathcal{C}$ fragments —such as Ward$^+$with respect to Dyadic-Shy— for which a dyadic decomposition can be easily computed. Moreover, our plan is to implement Algorithm 1 to perform query answering by exploiting existing reasoners.

References

[1] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, P. F. Patel-Schneider (Eds.), The Description Logic Handbook: Theory, Implementation, and Applications, Cambridge University Press, 2003.

[2] J. Baget, M. Leclère, M. Mugnier, E. Salvat, On rules with existential variables: Walking the decidability line, Artif. Intell. 175 (2011) 1620–1654.

[3] J. Baget, M. Leclère, M. Mugnier, E. Salvat, Extending decidable cases for rules with existential variables, in: IJCAI, 2009, pp. 677–682.

[4] N. Leone, M. Manna, G. Terracina, P. Veltri, Fast query answering over existential rules, ACM Trans. Comput. Log. 20 (2019) 12:1–12:48.

[5] J.-F. Baget, M. Leclère, M.-L. Mugnier, Walking the decidability line for rules with existential variables., KR 10 (2010) 466–476.

[6] A. Calì, G. Gottlob, M. Kifer, Taming the infinite chase: Query answering under expressive relational constraints, J. Artif. Intell. Res. 48 (2013) 115–174.

[7] R. Fagin, P. G. Kolaitis, R. J. Miller, L. Popa, Data exchange: semantics and query answering, Theoretical Computer Science 336 (2005) 89–124.

[8] M. Krötzsch, S. Rudolph, Extending decidable existential rules by joining acyclicity and guardedness, in: IJCAI, IJCAI/AAAI, 2011, pp. 963–968.

[9] S. Ceri, G. Gottlob, L. Tanca, What you always wanted to know about datalog (and never dared to ask), IEEE Trans. Knowl. Data Eng. 1 (1989) 146–166.

[10] G. Gottlob, A. Pieris, Beyond SPARQL under OWL 2 QL entailment regime: Rules to the rescue, in: IJCAI, AAAI Press, 2015, pp. 2999–3007.

[11] T. Baldazzi, L. Bellomarini, M. Favorito, E. Sallinger, On the relationship between shy and warded datalog+/-, CoRR abs/2202.06285 (2022).

[12] A. Calì, G. Gottlob, A. Pieris, Advanced processing for ontological queries, Proceedings of the VLDB Endowment 3 (2010) 554–565.

[13] A. Calì, G. Gottlob, A. Pieris, Query answering under non-guarded rules in datalog+/-, in: RR, volume 6333 of *LNCS*, Springer, 2010, pp. 1–17.

[14] A. Calì, G. Gottlob, T. Lukasiewicz, A general datalog-based framework for tractable query answering over ontologies, J. Web Semant. 14 (2012) 57–83.

[15] T. Gogacz, J. Marcinkowski, Converging to the chase - A tool for finite controllability, volume 83, 2017, pp. 180–206.

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

[17] A. Deutsch, A. Nash, J. B. Remmel, The chase revisited, in: PODS, ACM, 2008, pp. 149–158.

[18] M. Krötzsch, S. Rudolph, Extending decidable existential rules by joining acyclicity and guardedness, in: IJCAI, IJCAI/AAAI, 2011, pp. 963–968.

[19] G. Berger, G. Gottlob, A. Pieris, E. Sallinger, The space-efficient core of vadalog, volume 47, 2022, pp. 1:1–1:46.