Existential Rule Languages with Finite Chase: Complexity and Expressiveness

Heng Zhang[1] and **Yan Zhang**[1] and **Jia-Huai You**[2]
[1]School of Computing, Engineering and Mathematics, University of Western Sydney, Penrith, NSW 2751, Australia
[2]Department of Computing Science, University of Alberta, Edmonton, AB T6G 2E8, Canada

Abstract

Finite chase, or alternatively chase termination, is an important condition to ensure the decidability of existential rule languages. In the past few years, a number of rule languages with finite chase have been studied. In this work, we propose a novel approach for classifying the rule languages with finite chase. Using this approach, a family of decidable rule languages, which extend the existing languages with the finite chase property, are naturally defined. We then study the complexity of these languages. Although all of them are tractable for data complexity, we show that their combined complexity can be arbitrarily high. Furthermore, we prove that all the rule languages with finite chase that extend the weakly acyclic language are of the same expressiveness as the weakly acyclic one, while rule languages with higher combined complexity are in general more succinct than those with lower combined complexity.

Introduction

It has been shown that existential rule languages, also called Datalog$^{\pm}$, have prominent applications in ontological reasoning, knowledge representation, and databases, in which query answering is a primary yet challenging problem; see e.g., (Calì et al. 2010; Baget et al. 2011a). Under an existential rule language, queries are answered against a logical theory consisting of an input database and a finite set of existential rules, while a chase procedure is usually used. Specifically, given an input database D, a finite set Σ of existential rules, and a query q, we want to decide whether $D \cup \Sigma \models q$. Applying the chase procedure, the problem is equivalent to deciding whether $\mathsf{chase}(D, \Sigma) \models q$. Through a chase procedure, fresh *nulls* have to be introduced for each application of the existential rules, and hence, potential cyclic applications of these rules may lead the chase procedure not to terminate, i.e., $\mathsf{chase}(D, \Sigma)$ is infinite. Therefore, the problem of query answering under existential rule languages is in general undecidable (Beeri and Vardi 1981).

There have been a considerable number of works on identifying decidable classes with respect to query answering. Basically, two major approaches have provided a landscape on this study: One is to focus on some restricted fragments of existential rule languages such that the underlying chase procedure, though non-terminating in general, still enjoys some kind of *finite representability property*, so that the problem of query answering is decidable under this setting. This paradigm includes, e.g., *guarded rules* (Calì, Gottlob, and Lukasiewicz 2012), *greedy bounded treewidth sets* (Baget et al. 2011b), *sticky sets* (Calì, Gottlob, and Pieris 2012), and *Shy programs* (Leone et al. 2012). The other approach is to identify a certain *acyclicity* condition under which each existential rule can only be finitely applied so that the chase procedure always terminates. There have been many recent studies on this paradigm. Our work presented in this paper is along this line. Below, let us provide a brief summary of recent works under this approach.

In their milestone paper, Fagin et al. (2005) formulated a concept called *weak acyclicity* (WA) as a sufficient condition to ensure the chase termination for existential rules. This concept was then extended to a number of notions, such as *stratification* (Deutsch, Nash, and Remmel 2008), *super-weak acyclicity* (Marnette 2009), *local stratification* (Greco, Spezzano, and Trubitsyna 2011), *joint acyclicity* (Krötzsch and Rudolph 2011), *model-faithful acyclicity* (MFA) and *model-summarising acyclicity* (MSA) (Grau et al. 2013), and some dependency relations by (Baget et al. 2014). Among these, MFA is known to define the largest rule class. In addition, many ontologies in various domains turn out to be in the MFA class, as evidenced in (Grau et al. 2013).

It has been observed that almost all of the existential rule languages defined based on the notion of acyclicity or its variations have PTIME-complete data complexity and 2-EXPTIME-complete combined complexity. The uniformity on data complexity is in fact due to an interesting result proved in (Marnette 2009), which states that every rule language with finite Skolem chase is in PTIME for data complexity. A natural question then arises: Does this uniformity hold for combined complexity? Moreover, what is the expressiveness of existing rule languages with finite chase? Please note that the uniformity on data complexity does not imply the uniformity on expressiveness as data complexity only captures the hardest case of a language. Recently, there have been two interesting related works that studied the expressiveness of existential rules (Arenas, Gottlob, and Pieris 2014; Gottlob, Rudolph, and Simkus 2014). But both of them only focus on guarded language or its variations.

In this paper, we study the complexity, expressiveness and succinctness for existential rule languages with finite chase. Our contributions are summarized as follows:

1. A novel approach for classifying the existential rule languages with finite Skolem chase is proposed by restricting the use of existential variables in the Skolem chase. Under this approach, a family of interesting decidable rule languages, called *bounded languages*, are naturally defined. All of the existing rule languages with finite chase, e.g., the MFA class, are contained in these languages.
2. For every nonnegative integer k, the combined complexity of Boolean query answering for k-exponentially bounded language is proved to be $(k+2)$-EXPTIME-complete, and the membership problem of k-exponentially bounded language is proved to be in $(k+2)$-EXPTIME. Furthermore, for other bounded languages, the corresponding upper bounds of the complexity are also obtained.
3. All the languages with finite Skolem chase that extend the WA class are proved to be of the same expressiveness as WA, while languages with higher combined complexity are in general more succinct than those with lower combined complexity. On ordered databases, WA is shown to capture all existential rule sets whose universal models are computable in PTIME, even if they have no finite chase.

The results presented in this paper not only generalize some of the existing works, such as the two acyclicity notions of MFA and MSA proposed in (Grau et al. 2013), more importantly, they provide a global landscape for characterizing the existential rule languages with finite Skolem chase.

The rest of this paper is organized as follows. Section 2 provides necessary preliminaries. Section 3 defines a family of existential rule languages with finite Skolem chase called *bounded classes*, and presents some interesting properties of this family of languages. Section 4 then focuses on the complexity issues for bounded classes of languages, while section 5 explores the expressiveness and succinctness of these bounded classes of languages in details. Finally, section 6 concludes this paper with some remarks. Due to space limitation, proofs of some results are presented in an extended version of this paper, see (Zhang, Zhang, and You 2014).

Preliminaries

Databases and Queries. As usual, we assume (i) an infinite set Δ of *constants*, (ii) an infinite set Δ_n of *(labelled) nulls*, and (iii) an infinite set Δ_v of *variables*. A *relational schema* $\mathcal{R}$ consists of a finite set of *relation symbols*, each of which is armed with a natural number, its *arity*. *Terms* are either constants or variables. Every *atomic formula* (or *atom*) has the form $\mathsf{R}(\boldsymbol{t})$ where R is a relation symbol and $\boldsymbol{t}$ a tuple of terms of a proper length. *Ground terms* are terms involving no variable, and *facts* are atoms built from ground terms.

Given a relational schema $\mathcal{R}$, an *instance* (*database*) *over* $\mathcal{R}$, or simply $\mathcal{R}$-*instance* ($\mathcal{R}$-*database*), is a (finite) set of facts involving only relation symbols from $\mathcal{R}$. The *domain* of a database D, denoted $\mathsf{dom}(D)$, is the set of all constants appearing in D. *General instances (databases)* are the extensions of instances (databases) by allowing nulls to be used. Given a general instance (database) D and a relational schema $\mathcal{R}$, the *restriction* of D to $\mathcal{R}$, denoted $D|_{\mathcal{R}}$, is the set of facts in D involving only relation symbols from $\mathcal{R}$.

A *substitution* is a function $h : \Delta \cup \Delta_n \cup \Delta_v \to \Delta \cup \Delta_n \cup \Delta_v$ with (i) $h(c) = c$ for all $c \in \Delta$ and (ii) $h(\mathsf{n}) \in \Delta \cup \Delta_n$ for all $\mathsf{n} \in \Delta_n$. Let D and D_0 be general instances of the same schema. Then D is called *homomorphic* to D_0, written $D \to D_0$, if there is a substitution h with $h(D) \subseteq D_0$ where h is assumed to be extended to atoms and general instances naturally. In this case, the function is called a *homomorphism* from D to D_0. Moreover, D is *homomorphically equivalent* to D_0 if D is homomorphic to D_0 and vice versa.

Every *conjunctive query (CQ)* q over a relational schema $\mathcal{R}$ has the form $q(\boldsymbol{x}) := \exists \boldsymbol{y} \varphi(\boldsymbol{x}, \boldsymbol{y})$, where $\boldsymbol{x}, \boldsymbol{y}$ are tuples of variables, and $\varphi(\boldsymbol{x}, \boldsymbol{y})$ is a conjunction (sometimes we regard it as a set) of atoms with variables from $\boldsymbol{x}$ and $\boldsymbol{y}$, and relation symbols from $\mathcal{R}$. A *Boolean CQ (BCQ)* is a CQ of the form $q()$. Actually, BCQs can be regarded as general databases if we omit the quantifiers and regard the variables as nulls. Given any BCQ q and any general instance D over the same schema, the *answer* to q over D is "*Yes*", written $D \models q$, if there exists a homomorphism from q to D.

Existential Rules and Skolem Chase. Given a relational schema $\mathcal{R}$, every *(existential) rule* over $\mathcal{R}$ is a first-order sentence γ of the form $\forall \boldsymbol{x} \forall \boldsymbol{y}(\varphi(\boldsymbol{x}, \boldsymbol{y}) \to \exists \boldsymbol{z} \psi(\boldsymbol{x}, \boldsymbol{z}))$, where φ and ψ are conjunctions of atoms with relation symbols from $\mathcal{R}$ and variables from $\boldsymbol{x} \cup \boldsymbol{y}$ and $\boldsymbol{x} \cup \boldsymbol{z}$, respectively. We call φ the *body* of γ and ψ the *head* of γ, and write them as $\mathsf{body}(\gamma)$ and $\mathsf{head}(\gamma)$, respectively. When writing a rule, for simplicity, we will omit the universal quantifiers.

A *rule ontology* is a triple $(\Sigma, \mathcal{D}, \mathcal{Q})$, where Σ is *finite* and *nonempty* set Σ of rules, $\mathcal{D}$, called *database schema*, is a relational schema consisting of the relation symbols to be used in databases, and $\mathcal{Q}$, called *query schema*, is a relational schema consisting of the relation symbols to be used in queries. Relation symbols appearing in Σ but neither $\mathcal{D}$ nor $\mathcal{Q}$ are called *auxiliary symbols*. Note that $\mathcal{D}$ and $\mathcal{Q}$ could be the same. Without loss of generality, in any rule ontology, each variable is assumed to be quantified at most once.

Let γ be a rule $\varphi(\boldsymbol{x}, \boldsymbol{y}) \to \exists \boldsymbol{z} \psi(\boldsymbol{x}, \boldsymbol{z})$. We introduce a function symbol f_z of arity $|\boldsymbol{x}|$ for each variable $z \in \boldsymbol{z}$. From now on, we will regard terms built from constants and the introduced function symbols as a special class of nulls. The *functional transformation* of γ, denoted $\mathsf{sk}(\gamma)$, is the formula obtained from γ by substituting $\mathsf{f}_z(\boldsymbol{x})$ for each variable $z \in \boldsymbol{z}$. Given a set Σ of rules, the *functional transformation* of Σ, denoted $\mathsf{sk}(\Sigma)$, is the set of rules $\mathsf{sk}(\gamma)$ for all $\gamma \in \Sigma$.

Now we are in the position to define the *(Skolem) chase*. Let D be a database and Σ a rule set. We let $\mathsf{chase}^0(D, \Sigma) = D$ and, for all $n > 0$, let $\mathsf{chase}^n(D, \Sigma)$ denote the union of $\mathsf{chase}^{n-1}(D, \Sigma)$ and $h(\mathsf{head}(\gamma))$ for all rules $\gamma \in \mathsf{sk}(\Sigma)$ and all substitutions h such that $h(\mathsf{body}(\gamma)) \subseteq \mathsf{chase}^{n-1}(D, \Sigma)$. Let $\mathsf{chase}(D, \Sigma)$ be the union of $\mathsf{chase}^n(D, \Sigma)$ for all $n \geq 0$. It is well-known that, for all BCQs q, $D \cup \Sigma \models q$ (under the semantics of first-order logic) if and only if $\mathsf{chase}(D, \Sigma) \models q$. Given a rule ontology $O = (\Sigma, \mathcal{D}, \mathcal{Q})$, we say that O *has finite chase* if for all $\mathcal{D}$-databases D, $\mathsf{chase}(D, \Sigma)$ is finite. For more details, please refer to (Marnette 2009).

More Notations. Given a set Σ of rules and a BCQ q, let $\|\Sigma\|$ and $\|q\|$ denote the numbers of symbols occurring in Σ and q, respectively. We assume that the reader is familiar with complexity theory. Given a unary function T on natural numbers, by $\mathrm{DTIME}(T(n))$ we mean the class of complexity languages decidable in time $T(n)$ by a deterministic Turing machine. For $k \geq 0$ we let $\mathsf{exp}_k(n)$ denote the function that maps n to n if $k = 0$, and $2^{\mathsf{exp}_{k-1}(n)}$ otherwise. By k-EXPTIME we mean the class $\bigcup_{c\geq 0} \mathrm{DTIME}(\mathsf{exp}_k(n^c))$.

For simplicity, we denote relation symbols (nulls/function symbols, respectively) by capitalized (lower-case, respectively) sans-serif letters, constants by lower-case italic letters a, b, c, variables by lower-case italic letters u, v, w, x, y, z, and terms by lower-case italic letters s, t. All of these symbols may be written with subscripts or superscripts. In addition, bold italic letters $\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w}, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}$ and $\boldsymbol{s}, \boldsymbol{t}$ are used to range over tuples of variables and terms, respectively.

Bounded Classes

In this section, we define a family of rule languages with finite chase, and study its general properties.

We first define some notations. Given a ground term t, the *height* of t, denoted $\mathsf{ht}(t)$, is defined as follows:

$$\mathsf{ht}(t) := \begin{cases} 0 & \text{if } t \in \Delta; \\ \max\{\mathsf{ht}(s) : s \in \boldsymbol{s}\} + 1 & \text{if } t = \mathsf{f}(\boldsymbol{s}) \text{ for some } \mathsf{f}. \end{cases}$$

Given any general instance A, the *height* of A, denoted $\mathsf{ht}(A)$, is defined as the maximum height of terms that have at least one occurrence in A if it exists, and ∞ otherwise.

Definition 1. Every *bound function* is a function from positive integers to positive integers. Let δ be a bound function. A rule ontology $(\Sigma, \mathcal{D}, \mathcal{Q})$ is called *δ-bounded* if, for all $\mathcal{D}$-databases D, $\mathsf{ht}(\mathsf{chase}(D, \Sigma)) \leq \delta(\|\Sigma\|)$. We let δ-BOUNDED denote the class of δ-bounded rule ontologies.

As there exist an infinite number of bound functions, it is interesting to know if there is a "maximum" bound function that captures all δ-bounded rule ontologies for any bound function δ (or all rule ontologies with finite chase). The following result shows that the answer is definitely "yes".

Proposition 1. *There is a bound function δ such that, for every rule ontology O, O has finite chase iff it is δ-bounded.*

Proof. (Sketch) We first construct a bound function δ, and then it suffices to show that every rule ontology with finite chase is δ-bounded. To define δ, we want to prove that, for every rule ontology $O = (\Sigma, \mathcal{D}, \mathcal{Q})$, there exists a database D^*_O such that $\mathsf{ht}(\mathsf{chase}(D, \Sigma)) \leq \mathsf{ht}(\mathsf{chase}(D^*_O, \Sigma))$ for all $\mathcal{D}$-databases D. This can be done by employing the so-called *critical database technique*, which was developed in (Marnette 2009). Define $\delta(n)$ as the maximum $\mathsf{ht}(\mathsf{chase}(D^*_O, \Sigma))$ among all rule ontologies $O = (\Sigma, \mathcal{D}, \mathcal{Q})$ with finite chase such that $\|\Sigma\| \leq n$; we then have the desired function δ. ☐

Remark 1. Let BOUNDED be the union of δ-BOUNDED for all bound functions δ. A rule ontology is called *bounded* if it belongs to BOUNDED. As all bounded rule ontologies have finite chase, by Proposition 1 we have that BOUNDED contains exactly the rule ontologies with finite chase.

Next, let us define a class of interesting bound functions.

Definition 2. Let k be a natural number and let exp_k be the function defined in the previous section. A rule ontology is called *k-exponentially bounded* if it is exp_k-bounded.

Remark 2. The MFA class (Grau et al. 2013), which was shown to extend many existing languages with an acyclicity restriction, is defined by restricting the recursive uses of existential variables in Skolem chase. It is not difficult to see that MFA $\subseteq$ exp_0-BOUNDED. The following example shows that the inclusion is in fact strict.

Example 1. Let $O = (\Sigma, \mathcal{D}, \mathcal{Q})$ be a rule ontology, where $\mathcal{D} = \{\mathsf{R}\}$ and Σ consists of the following rules:

$$\begin{aligned} \mathsf{R}(x, x) &\rightarrow \exists yz\, \mathsf{S}(x, y) \wedge \mathsf{S}(y, z) \\ \mathsf{R}(x, y) \wedge \mathsf{S}(x, z) &\rightarrow \exists v\, \mathsf{R}(z, v) \end{aligned}$$

This rule ontology does not belong to the MFA class because the existential variable v might be recursively applied in the Skolem chase (one can check it by letting the database D be $\{\mathsf{R}(a, a)\}$). As each existential variable can be recursively used at most twice, O is 0-exponentially bounded. ☐

One might ask if all bounded ontologies can be captured by exponential bound functions (or computable bound functions). The proposition below shows that this is impossible.

Proposition 2. *There is no computable bound function δ such that every bounded rule ontology is δ-bounded.*

Complexity

Now we study the complexity of bounded classes. We are interested in the complexity of two kinds of important computations: query answering and language membership.

Boolean Query Answering. The problem to be investigated here, also known as *query entailment*, is defined as follows: Given a set Σ of rules, a database D and a Boolean query q, decide if $D \cup \Sigma \models q$. We first consider the upper bound.

Proposition 3. *Let δ be a bound function. Then for any δ-bounded rule ontology $(\Sigma, \mathcal{D}, \mathcal{Q})$, any $\mathcal{D}$-database D and any BCQ q over $\mathcal{Q}$, it is in*

$$\mathrm{DTIME}((|\mathsf{dom}(D)| + \|\Sigma\|)^{\|\Sigma\|^{\mathcal{O}(\delta(\|\Sigma\|))} \cdot \|q\|^{\mathcal{O}(1)}})$$

to check whether $D \cup \Sigma \models q$.

Proof. (Sketch) First evaluate the size of $\mathsf{chase}(D, \Sigma)$. By this we know how many stages are needed for the chase to terminate. Counting the cost of each chase stage and querying on $\mathsf{chase}(D, \Sigma)$, we then have the desired result. ☐

A lower bound for the combined complexity is as follows.

Proposition 4. *It is $(k + 2)$-EXPTIME-hard (for the combined complexity) to check, given a k-exponentially bounded rule ontology $(\Sigma, \mathcal{D}, \mathcal{Q})$, a $\mathcal{D}$-database D and a BCQ q over $\mathcal{Q}$, whether $D \cup \Sigma \models q$.*

Proof. (Sketch) Let M be any deterministic Turing machine that terminates in $\mathsf{exp}_{k+2}(n)$ steps on any input of length n. Let $\mathcal{D} = \emptyset$ and $\mathcal{Q} = \{\mathsf{Accept}\}$ where Accept is a nullary relation symbol. To show the desired result, it suffices to show that, for each input (a binary string) x, there is an exp_k-bounded rule ontology $(\Sigma, \mathcal{D}, \mathcal{Q})$ such that M terminates on input x if and only if $\emptyset \cup \Sigma \models \mathsf{Accept}$. Let x be an input of length n. To construct the rule set Σ, the main difficulty is to define a linear order of length $\mathsf{exp}_{k+2}(n)$. If the order is defined, by an encoding similar to that in (Dantsin et al. 2001), one can construct a set of datalog rules to encode both M and x. Here we only explain how to define the linear order.

Let us first consider the case where k is even. The general idea is to construct a sequence of rule sets $(\Sigma_i)_{i\geq 0}$. For each i, let $\mathsf{Succ}_i, \mathsf{Min}_i$ and Max_i be relation symbols intended to define the (immediate) successor relation, the minimum element and the maximum element, respectively, of a linear order. For $i > 0$, the function of Σ_i is as follows: If $\mathsf{Succ}_{i-1}, \mathsf{Min}_{i-1}$ and Max_{i-1} define a linear order of length n, then $\mathsf{Succ}_i, \mathsf{Min}_i$ and Max_i define a linear order on integers (represented in binary strings) from 0 to 2^{2^n}. To implement each Σ_i, we generalize a technique used in the proof of Theorem 1 in (Calì, Gottlob, and Pieris 2010).

The first task is to define the binary strings of length one, i.e. "0" and "1". This can be done by the following rule:

$$\mathsf{Min}_{i-1}(v) \rightarrow \mathsf{BS}_i(v, 0) \wedge \mathsf{BS}_i(v, 1)$$

where $\mathsf{BS}_i(v, x)$ states that x is a binary string of length 2^v.

The following rules are used to generate binary strings of length 2^{v+1} by combining two binary strings of length 2^v:

$$\begin{aligned} \mathsf{BS}_i(v, x) \wedge \mathsf{BS}_i(v, y) &\rightarrow \exists z\, \mathsf{C}_i(v, x, y, z) \\ \mathsf{C}_i(v, x, y, z) \wedge \mathsf{Succ}_{i-1}(v, w) &\rightarrow \mathsf{BS}_i(w, z) \end{aligned}$$

Then, some rules to define a successor relation (w.r.t. the lexicographic order) on strings of length 2^{v+1} are as follows:

$$\begin{bmatrix} \mathsf{C}_i(v, x, y, z) \wedge \mathsf{C}_i(v, x, y_0, z_0) \\ \wedge\, \mathsf{Succ}^*_i(v, y, y_0) \wedge \mathsf{Succ}_{i-1}(v, w) \end{bmatrix} \rightarrow \mathsf{Succ}^*_i(w, z, z_0)$$

$$\begin{bmatrix} \mathsf{C}_i(v, x, y, z) \wedge \mathsf{C}_i(v, x_0, y_0, z_0) \\ \wedge\, \mathsf{Max}^*_i(v, y) \wedge \mathsf{Min}^*_i(v, y_0) \\ \wedge\, \mathsf{Succ}^*_i(v, x, x_0) \wedge \mathsf{Succ}_{i-1}(v, w) \end{bmatrix} \rightarrow \mathsf{Succ}^*_i(w, z, z_0)$$

where $\mathsf{Succ}^*_i(v, x, y)$ is intended to assert that y is the immediate successor of x, and both x and y are of length 2^v.

The minimum and the maximum binary strings of length 2^{v+1} are defined by the following rules:

$$\begin{bmatrix} \mathsf{Min}^*_i(v, x) \wedge \mathsf{Min}^*_i(v, y) \\ \wedge\, \mathsf{C}_i(v, x, y, z) \wedge \mathsf{Succ}_{i-1}(v, w) \end{bmatrix} \rightarrow \mathsf{Min}^*_i(w, z)$$

$$\begin{bmatrix} \mathsf{Max}^*_i(v, x) \wedge \mathsf{Max}^*_i(v, y) \\ \wedge\, \mathsf{C}_i(v, x, y, z) \wedge \mathsf{Succ}_{i-1}(v, w) \end{bmatrix} \rightarrow \mathsf{Max}^*_i(w, z)$$

Now the desired relations $\mathsf{Num}_i, \mathsf{Succ}_i, \mathsf{Min}_i$ and Max_i can be obtained by applying the following rules:

$$\begin{aligned} \mathsf{Succ}^*_i(v, x, y) \wedge \mathsf{Max}_{i-1}(v) &\rightarrow \mathsf{Succ}_i(x, y) \\ \mathsf{Min}^*_i(v, x) \wedge \mathsf{Max}_{i-1}(v) &\rightarrow \mathsf{Min}_i(x) \\ \mathsf{Max}^*_i(v, x) \wedge \mathsf{Max}_{i-1}(v) &\rightarrow \mathsf{Max}_i(x) \end{aligned}$$

For all $i > 0$, let Σ_i consist of all of the above rules. It is easy to see that Σ_i is as desired. Let $0, \ldots, n-1$ be distinct constants. Let Σ_0 denote the following rule set:

$$\begin{gathered} \mathsf{Min}_0(0) \wedge \mathsf{Max}_0(n-1) \\ \mathsf{Succ}_0(0, 1) \wedge \cdots \wedge \mathsf{Succ}_0(n-2, n-1) \end{gathered}$$

Next, let $\ell = k/2 + 1$ and let Σ_{num} be the union of Σ_i for all i with $0 \leq i \leq \ell$. By the previous analysis, it is not difficult to see that $\mathsf{Succ}_\ell, \mathsf{Min}_\ell$ and Max_ℓ define a linear order on (the binary represention of) integers from 0 to $\mathsf{exp}_{k+2}(n)$. It is also not difficult to check that the rule ontology $(\Sigma_{\mathsf{num}}, \emptyset, \{\mathsf{Accept}\})$ is exp_k-bounded.

For the case where k is odd, we can achieve the goal by some slight modifications to Σ_{num}: (i) substituting the least integer greater than or equal to $\log n$ for n in Σ_0, and then (ii) setting $\ell = k/2 + 2$. Similarly, we can show that the resulting rule set Σ_{num} satisfies the desired property. $\square$

Now, by combining Propositions 3, 4, and the data complexity of Datalog (see, e.g., (Dantsin et al. 2001)), for any k-exponentially bounded class we then have the exact bound of the complexity w.r.t. Boolean query answering.

Theorem 5. *For all integers $k \geq 0$, the Boolean query answering problem of the k-exponentially bounded language is $(k+2)$-EXPTIME-complete for the combined complexity, and PTIME-complete for the data complexity.*

Membership of Language. Now we consider the membership problem of bounded languages. The problem is as follows: Given a rule ontology, check whether it belongs to the bounded language under consideration. Since the boundedness is defined in a semantical way, it is interesting to know how to check whether a rule ontology is δ-bounded.

Proposition 6. *Let δ be a bound function that is computable in DTIME$(T(n))$ for some function $T(n)$. Then for every δ-bounded rule ontology $O = (\Sigma, \mathcal{D}, \mathcal{Q})$, it is in*

$$\mathrm{DTIME}(\|\Sigma\|^{\|\Sigma\|^{\mathcal{O}(\delta(\|\Sigma\|))}} + T(\log \|\Sigma\|)^{\mathcal{O}(1)})$$

to check whether O is δ-bounded.

The above proposition can be proved by using Marnette's critical database technique (2009) and then by an analysis similar to that in the proof of Proposition 3.

Remark 3. Two immediate corollaries of Proposition 6 are: It is in $(k+2)$-EXPTIME to check whether a rule ontology is k-exponentially bounded; moreover, the membership for δ-bounded language is decidable whenever δ is computable.

Expressiveness and Succinctness

Though all rule languages with finite chase are tractable for data complexity (Marnette 2009), in the last section we have shown that their combined complexity could be very high. Hence, a natural question is as follows: Are the languages with high combined complexity really necessary for representing ontological knowledge? In this section, we address this question on two aspects: What is the expressiveness of these languages? How about the succinctness among them?

Universal Worldview Mapping. We first propose a semantic (and more general) definition for rule ontologies.

Definition 3. Let $\mathcal{D}$ and $\mathcal{Q}$ be two relational schemas. A *universal worldview mapping*, or UWM for short, over $(\mathcal{D}, \mathcal{Q})$ is a function that maps every $\mathcal{D}$-database D to a general instance Q over $\mathcal{Q}$. Let Φ and Ψ be two UWMs over $(\mathcal{D}, \mathcal{Q})$. We say that Φ is *equivalent* to Ψ, written $\Phi \approx \Psi$, if for all $\mathcal{D}$-databases, $\Phi(D)$ is homomorphically equivalent to $\Psi(D)$.

It is clear that $\approx$ is an equivalence relation on the UWMs. Next, we show how to define UWMs from rule ontologies.

Definition 4. Let $O = (\Sigma, \mathcal{D}, \mathcal{Q})$ be any rule ontology. We define $[\![O]\!]$ as the function that maps every $\mathcal{D}$-database D to the general instance $\mathsf{chase}(D, \Sigma)|_{\mathcal{Q}}$.

Given any rule ontology O, it is clear that $[\![O]\!]$ is a UWM. We say that two rule ontologies O_1 and O_2 are *equivalent* if the corresponding UWMs are equivalent, i.e., $[\![O_1]\!] \approx [\![O_2]\!]$. The following property explains why this is desired.

Proposition 7. *Let $O_1 = (\Sigma_1, \mathcal{D}, \mathcal{Q})$ and $O_2 = (\Sigma_2, \mathcal{D}, \mathcal{Q})$ be two rule ontologies with finite chase. Then $[\![O_1]\!] \approx [\![O_2]\!]$ iff, for all $\mathcal{D}$-databases D and all BCQs q over $\mathcal{Q}$, we have*

$$D \cup \Sigma_1 \models q \quad \textit{iff} \quad D \cup \Sigma_2 \models q.$$

In addition, for a technical reason, given a rule ontology $O = (\Sigma, \mathcal{D}, \mathcal{Q})$, we require that $\mathcal{D}$ and $\mathcal{Q}$ are *disjoint* and *no* relation symbol in $\mathcal{D}$ has an occurrence in the head of any rule in Σ.[1] We call such rule ontologies *normal*. These assumptions do not change the expressiveness since, for every relation symbol $\mathsf{R} \in \mathcal{D} \cap \mathcal{Q}$, we can always replace R in $\mathcal{D}$ with a fresh relation symbol R' of the same arity, and then add a *copy rule* $\mathsf{R}'(\mathbf{x}) \to \mathsf{R}(\mathbf{x})$ into the rule set Σ.

From Bounded Classes to the WA Class. In this subsection, we show that any bounded ontology can be rewritten to a rule ontology that is weakly acyclic (Fagin et al. 2005).

Let us first review the notion of weak acyclicity. Fix Σ as a set of rules and $\mathcal{R}$ its schema. A *position* of Σ is a pair (R, i) where $\mathsf{R} \in \mathcal{R}$ is of an arity n and $1 \le i \le n$. The *dependency graph* of Σ is a directed graph with each position of Σ as a node, and with each pair $((\mathsf{R}, i), (\mathsf{S}, j))$ as an edge if there is a rule $\varphi(\boldsymbol{x}) \to \exists \boldsymbol{y} \psi(\boldsymbol{x}, \boldsymbol{y})$ from Σ such that either

- there is a variable $x \in \boldsymbol{x}$ such that x occurs both in the position (R, i) in φ and in the position (S, j) in ψ, or
- there are variables $x \in \boldsymbol{x}$ and $y \in \boldsymbol{y}$ such that x occurs in the position (R, i) in φ and y occurs in the position (S, j) in ψ (in this case, the edge is called a *special edge*).

A rule ontology $(\Sigma, \mathcal{D}, \mathcal{Q})$ is *weakly acyclic (WA)* if no cycle in the dependency graph of Σ goes through a special edge.

It is well-known that the class of WA rule ontologies enjoys the finite chase property. In the last few years, a number of classes have been proposed to extend it. However, the next theorem shows that, in view of the expressiveness, the WA class is no weaker than any class with finite chase.

Theorem 8. *For every normal rule ontology $O = (\Sigma, \mathcal{D}, \mathcal{Q})$ with finite chase, there exists a weakly acyclic normal rule ontology $O^* = (\Sigma^*, \mathcal{D}, \mathcal{Q})$ such that $[\![O]\!] \approx [\![O^*]\!]$.*

We prove this theorem by developing a translation. The general idea is as follows. Given any normal rule ontology $O = (\Sigma, \mathcal{D}, \mathcal{Q})$ with finite chase, we need to construct a weakly acyclic rule ontology $O^* = (\Sigma^*, \mathcal{D}, \mathcal{Q})$ such that any computation on O can be simulated by a computation on O^*. The main difficulty is how to simulate the cyclic existential quantifications by weakly acyclic existential quantifications. Fortunately, by Proposition 1, O is always bounded, which means that the size of any (possibly functional) term generated by the chase procedure of O on any database is bounded by some integer ℓ. Therefore, every (possibly functional) term generated by the chase procedure of O can be encoded by an ℓ-tuple of function-free terms. For every relation symbol R occurring in O, we introduce an auxiliary relation symbol R^* with a larger arity so that every fact of the form $\mathsf{R}(\boldsymbol{t})$ can be encoded by a fact of the form $\mathsf{R}^*(\boldsymbol{t}^*)$.

With these settings, we can then construct some rules to simulate the chase procedure of O by that of O^* so that, for every $\mathcal{D}$-database D and every fact $\mathsf{R}(\boldsymbol{t}) \in \mathsf{chase}(D, \Sigma)$, there exists a fact $\mathsf{R}^*(\boldsymbol{t}^*) \in \mathsf{chase}(D, \Sigma^*)$ that encodes $\mathsf{R}(\boldsymbol{t})$. These rules can be constructed by an approach similar to that in (Krötzsch and Rudolph 2011). So, the remaining task is to decode $\mathsf{R}(\boldsymbol{t})$ from $\mathsf{R}^*(\boldsymbol{t}^*)$. The difficulty of decoding is how to assure that, for each term (that might occur as arguments in different facts) in $\mathsf{chase}(D, \Sigma)$, there is exactly one null to be allocated to it. Fortunately again, it can be overcome by some encoding techniques. We will explain this later.

Now, let us define the translation formally. Let $\mathcal{D}$, $\mathcal{Q}$ and Σ be defined as in the theorem. Let δ be a bound function such that Σ is δ-bounded. Let m be the maximum arity of function symbols appearing in $\mathsf{sk}(\Sigma)$. Let $\ell = \sum_{i=0}^{\delta(\|\Sigma\|)} m^i$ and $s = (\ell - 1)/m$. Introduce a fresh $\ell \cdot n$-ary relation symbol[2] R^* for each n-ary relation symbol R, introduce a fresh variable v^i for each variable v and each i with $1 \le i \le \ell$, and introduce $\square$ (as a blank symbol to fill the gaps) and all non-nullary function symbols in $\mathsf{sk}(\Sigma)$ as fresh constants.

To represent a (functional) term t, we first regard t as an m-complete tree with each symbol (function or constant) as a node and arguments of a function symbol f as the children of f. If some node is empty, we then fill it with $\square$. The tuple encodes t is then the symbol sequence obtained by a depth-first traversal. If the tuple is of length $< \ell$, we then fill $\square$s in the tail. Lastly, we let $[t]$ denote this tuple. For example, if $m = 2$, $\ell = 7$ and $t = \mathsf{f}(\mathsf{g}(a), b)$, then $[t] = \mathsf{fg}a\square b\square\square$.

To activate the simulation, some rules are needed to copy the data of R to R^*. Formally, for each n-ary relation symbols $\mathsf{R} \in \mathcal{D}$, we need a rule ϱ_{R}, defined as follows:

$$\mathsf{R}(x_1, \ldots, x_n) \to \mathsf{R}^*(x_1, \star, \cdots x_n, \star)$$

where $\star$ denotes the tuple consisting of $\ell - 1$ consecutive $\square$s.

Let γ be a rule from Σ of the form $\varphi(\boldsymbol{x}, \boldsymbol{y}) \to \exists \boldsymbol{z}\, \psi(\boldsymbol{x}, \boldsymbol{z})$ where $\boldsymbol{x} = x_1 \cdots x_k$ is a permutation of all the variables occurring in both φ and ψ. We need a rule γ^*, which will be defined shortly, to simulate γ. For any term t in γ, we let

$$\tau(t) := \begin{cases} a\,\square \cdots \square & \text{if } t = a \in \Delta; \\ v^1 \cdots v^s\,\square \cdots \square & \text{if } t = v \in \boldsymbol{x}\,; \\ \mathsf{f}_v x_1^1 \cdots x_1^s x_2^1 \cdots x_k^s \square \cdots \square & \text{if } t = v \in \boldsymbol{z}\,; \\ v^1 \cdots v^\ell & \text{if } t = v \in \boldsymbol{y}\,. \end{cases}$$

where, in each of the first three cases, the tail of $\tau(t)$ is filled with the symbol $\square$ such that the length of $\tau(t)$ is exactly ℓ. Now we define γ^* as the rule $\varphi^* \to \psi^*$, where φ^* and ψ^* denote the formulas obtained from φ and ψ, respectively, by

[1] This is similar to that in data exchange (Fagin et al. 2005).

[2] In fact, we can use some relation symbol with a smaller arity, but this will make the presentation more complicated.

- substituting $\tau(t)$ for each term $t \in \Delta \cup \Delta_v$, followed by
- substituting R^* for each relation symbol R.

In the chase procedure for the new ontology, by applying above rules on a $\mathcal{D}$-database D, we obtain a fact set S^* that encodes $\mathsf{chase}(D, \Sigma)$. Thus, as mentioned previously, the remaining task is to construct rules for the decoding. The idea is as follows: (i) let Dom^* be the set of all ℓ-tuples that encode terms with occurrences in $\mathsf{chase}(D, \Sigma)$; (ii) for each ℓ-tuple $\boldsymbol{s}^* \in \mathsf{Dom}^*$, generate a null n for it (by appying an existential quantifier once), and use $\mathsf{Map}(\boldsymbol{s}^*, \mathsf{n})$ to record the correspondence between $\boldsymbol{s}^*$ and n; (iii) translate each fact $\mathsf{R}^*(\boldsymbol{t}^*)$ to a fact $\mathsf{R}(\boldsymbol{t})$ by looking up the relation Map.

To collect the ℓ-tuples in stage (i), we need the following rules. Given an n-ary relation symbol $\mathsf{R} \in \mathcal{Q}$, let λ_{R} denote

$$\mathsf{R}^*(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n) \rightarrow \mathsf{Dom}^*(\boldsymbol{v}_1) \wedge \cdots \wedge \mathsf{Dom}^*(\boldsymbol{v}_n)$$

where each $\boldsymbol{v}_i$ is a tuple of distinct variables $v_i^1 \cdots v_i^s$, and Dom^* a fresh relation symbol of arity ℓ.

Next, we define some rules to generate nulls, which implement stage (ii). For each function symbol f_x in $\mathsf{sk}(\Sigma)$ where x is an existential variable in Σ, let ζ_x denote

$$\mathsf{Dom}^*(\mathsf{f}_x, \boldsymbol{v}) \rightarrow \exists x\, \mathsf{Map}(\mathsf{f}_x, \boldsymbol{v}, x)$$

where $\boldsymbol{v}$ is a tuple of distinct variables $v_1 \cdots v_{\ell-1}$, and Map a fresh $(\ell+1)$-ary relation symbol. In addition, let ζ_c denote

$$\mathsf{Dom}^*(x, \square, \ldots, \square) \rightarrow \mathsf{Map}(x, \square, \ldots, \square, x).$$

Informally, this rule asserts that, for any ℓ-tuple that encodes a single-symbol term, we do not need to generate any null.

Now, we can define rules to carry out the decoding. For each n-ary relation symbol $\mathsf{R} \in \mathcal{Q}$, let ϑ_{R} denote

$$\begin{bmatrix} \mathsf{R}^*(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n) \wedge \mathsf{Map}(\boldsymbol{v}_1, x_1) \\ \wedge \cdots \wedge \mathsf{Map}(\boldsymbol{v}_n, x_n) \end{bmatrix} \rightarrow \mathsf{R}(x_1, \ldots, x_n).$$

Finally, we let Σ^* denote the rule set consisting of (1) ϱ_{R} for every relation symbol $\mathsf{R} \in \mathcal{D}$, (2) γ^* for every rule $\gamma \in \Sigma$, (3) λ_{R} for every relation symbol $\mathsf{R} \in \mathcal{Q}$, (4) ζ_x for every existential variable x in Σ such that f_x is of a positive arity, (5) ζ_c, and (6) ϑ_{R} for every relation symbol $\mathsf{R} \in \mathcal{Q}$.

Example 2. By adding a copy rule into the rule ontology O defined in Example 1, we then obtain a normal rule ontology $O_0 = (\Sigma_0, \mathcal{D}_0, \mathcal{Q}_0)$, where Σ_0 is the following rule set:

$$\begin{aligned} \mathsf{D}(x, y) &\rightarrow \mathsf{R}(x, y) \\ \mathsf{R}(x, x) &\rightarrow \exists yz\, \mathsf{S}(x, y) \wedge \mathsf{S}(y, z) \\ \mathsf{R}(x, y) \wedge \mathsf{S}(x, z) &\rightarrow \exists v\, \mathsf{R}(z, v) \end{aligned}$$

and $\mathcal{D}_0 = \{\mathsf{D}\}$, $\mathcal{Q}_0 = \{\mathsf{R}\}$. Next, we will illustrate the translation by the rule ontology O_0.

All the function symbols in $\mathsf{sk}(\Sigma_0)$ are clearly unary, and as analyzed in Example 1, O_0 is δ-bounded for some bound function δ with $\delta(\|\Sigma_0\|) = 2$. So, we have $\ell = 1^0 + 1^1 + 1^2 = 3$, i.e., terms generated by the chase procedure of O_0 will be encoded by triples of function-free terms. Now, we use the following rule to initialize the auxiliary relation symbol D^*:

$$\mathsf{D}(x, y) \rightarrow \mathsf{D}^*(x\square\square\, y\square\square)$$

To simulate the chase procedure of O_0, we need the following rules, which correspond to the rules in Σ_0:

$$\begin{aligned} \mathsf{D}^*(x_1x_2x_3\, y_1y_2y_3) &\rightarrow \mathsf{R}^*(x_1x_2x_3\, y_1y_2y_3) \\ \mathsf{R}^*(x_1x_2\square\, x_1x_2\square) &\rightarrow \mathsf{S}^*(x_1x_2\square\, \mathsf{f}_yx_1x_2) \wedge \mathsf{S}^*(\mathsf{f}_yx_1x_2\, \mathsf{f}_zx_1x_2) \\ \mathsf{R}^*(x_1x_2x_3\, y_1y_2y_3) \wedge \mathsf{S}^*(y_1y_2y_3\, z_1z_2\square) &\rightarrow \mathsf{R}^*(z_1z_2\square\, \mathsf{f}_vz_1z_2) \end{aligned}$$

The following rules are used to implement the decoding:

$$\begin{aligned} \mathsf{R}^*(x_1x_2x_3\, y_1y_2y_3) &\rightarrow \mathsf{Dom}^*(x_1x_2x_3) \wedge \mathsf{Dom}^*(y_1y_2y_3) \\ \mathsf{Dom}^*(\mathsf{f}_yx_1x_2) &\rightarrow \exists y\, \mathsf{Map}(\mathsf{f}_yx_1x_2\, y) \\ \mathsf{Dom}^*(\mathsf{f}_zx_1x_2) &\rightarrow \exists z\, \mathsf{Map}(\mathsf{f}_zx_1x_2\, z) \\ \mathsf{Dom}^*(\mathsf{f}_vx_1x_2) &\rightarrow \exists v\, \mathsf{Map}(\mathsf{f}_vx_1x_2\, v) \\ \mathsf{Dom}^*(x\square\square) &\rightarrow \mathsf{Map}(x\square\square\, x) \\ \begin{bmatrix} \mathsf{R}^*(x_1x_2x_3\, y_1y_2y_3) \wedge \\ \mathsf{Map}(x_1x_2x_3\, x) \wedge \mathsf{Map}(y_1y_2y_3\, y) \end{bmatrix} &\rightarrow \mathsf{R}(x\, y) \end{aligned}$$

Finally, let Σ_0^* consist of the set of all rules defined above. It is not difficult to check that $[\![O_0]\!] \approx [\![(\Sigma_0^*, \mathcal{D}_0, \mathcal{Q}_0)]\!]$. $\square$

Capturing PTIME by the WA Class. We have proved that all the rule languages with finite chase are of the same expressiveness as the WA class in the last subsection. However, this characterization is syntactic. In this subsection, we will give a complexity-theoretic characterization. Before presenting the result, we need some definitions.

Like in traditional Datalog (Dantsin et al. 2001), we will study the expressiveness on ordered databases. Every *ordered database* is a database whose domain is an integer set $\{0, \ldots, n\}$ for some integer $n \geq 0$; whose schema contains three special relation symbols Succ, Min and Max (we call such a schema *ordered*); in which Succ is interpreted as the immediate successor relation on natural numbers, and Min and Max are interpreted as $\{0\}$ and $\{n\}$, respectively. By *ordered UWMs* we mean the restrictions of UWMs to ordered databases. We generalize definitions of $[\![\cdot]\!]$ and $\approx$ to ordered UWMs by replacing "database" with "ordered database". Note that the ordered version of Proposition 7 still holds.

We fix a natural way for representing (general) databases in binary strings. Given a general database D, let $\langle D \rangle$ denote its binary represention. Let $\mathcal{D}$ and $\mathcal{Q}$ be any two disjoint relational schemas where $\mathcal{D}$ is ordered. Let Φ be an ordered UWM over $(\mathcal{D}, \mathcal{Q})$. We say that Φ is *computed* by a Turing machine M if M halts on any input $\langle D \rangle$ where D is an ordered $\mathcal{D}$-database, and there is a general $\mathcal{Q}$-database Q such that Q is homomorphically equivalent to $\Phi(D)$ and the output w.r.t. $\langle D \rangle$ is $\langle Q \rangle$, the binary representation of Q.

On syntax, we also need a slightly richer language defined as follows. Let $\mathcal{D}$ be a relational schema (as a database schema). A *semipositive rule* w.r.t. $\mathcal{D}$ is a generalized rule defined by allowing negated atoms with relation symbols from $\mathcal{D}$ to appear in the body. Semipositive rule ontologies are then generalized from rule ontologies by allowing semipositive rules w.r.t. its database schema. A semipositive rule ontology is called *weakly acyclic* if the rule ontology obtained by omitting negative atoms is weakly acyclic.

Theorem 9. *For every ordered UWM Φ that is computable in deterministic polynomial time, there is a weakly acyclic and semipositive rule ontology O such that $[\![O]\!] \approx \Phi$.*

Remark 4. By a slight generalization of the critical database technique proposed in (Marnette 2009), one can show that every semipositive rule ontology with finite Skolem chase is computable in deterministic polynomial time. Therefore, the above theorem implies that every semipositive rule language with finite Skolem chase that extends the semipositive WA class captures the class of PTIME-computable UWMs.

Succinctness. Our previous results show that all the rule languages with finite chase that extend the weakly acyclic class are of the same expressiveness. Now we further consider the following question: Is it possible to compare the efficiency of rule languages with finite chase for representing ontological knowledge? In general, it is not an easy task to compare the succinctness for fragments of first-order logic. However, the following theorem provides us with such a result for rule languages, which states that the bounded rule languages with higher combined complexity are normally more succinct than those with lower combined complexity.

Theorem 10. *For all $k > 0$, there exists a k-exponentially bounded rule ontology $O = (\Sigma, \mathcal{D}, \mathcal{Q})$ such that, for any $(k-1)$-exponentially bounded rule ontology $O_0 = (\Sigma_0, \mathcal{D}, \mathcal{Q})$ where Σ_0 is of polynomial size w.r.t. Σ, we have $[\![O_0]\!] \not\approx [\![O]\!]$.*

Proof. (Sketch) Let n, ℓ and Σ_{num} be defined as in the proof of Proposition 4. Let $\mathcal{D} = \emptyset$ and $\mathcal{Q} = \{\mathsf{Min}_\ell, \mathsf{Max}_\ell, \mathsf{Succ}_\ell\}$. By using the notion of *core* (see, e.g., (Deutsch, Nash, and Remmel 2008)), we show a lower bound for the number of nulls in universal models. By estimating the number of nulls to be used in the chase, we then prove that $(\Sigma_{\mathsf{num}}, \mathcal{D}, \mathcal{Q})$ is not equivalent to any $(k-1)$-exponentially bounded ontology $(\Sigma, \mathcal{D}, \mathcal{Q})$ if Σ is of a polynomial size w.r.t. Σ_{num}. $\square$

Remark 5. Theorem 10 tells us that, although extending the WA class to larger classes with finite chase does not increase the expressiveness, the succinctness could be a bonus.

Remark 6. It would be interesting to compare the succinctness of finite-chase rule languages with the same combined complexity under query answering. For instance, is the MFA class more succinct than the WA class? But this is beyond the scope of this work. We will pursue it in the future.

Concluding Remarks

We have studied the existential rule languages with finite chase in this paper. Instead of considering specific rule languages like most current works on this topic, here we have defined a family of rule languages based on a new concept called *δ-boundedness*, from which the overall complexity and expressiveness characterizations on these languages have been provided. Our study on this topic may be further undertaken in various directions. One interesting yet challenging future work is to investigate disjunctive existential rule languages. It is important to discover whether our approach can be extended to identify decidable disjunctive existential rule languages and to characterize relevant complexity and expressiveness properties. Results on this aspect may significantly enhance our current understanding on ontological reasoning with disjunctive existential rules (Alviano et al. 2012; Bourhis, Morak, and Pieris 2013).

References

Alviano, M.; Faber, W.; Leone, N.; and Manna, M. 2012. Disjunctive datalog with existential quantifiers: Semantics, decidability, and complexity issues. *Theor. Pract. Log. Prog.* 12:701–718.

Arenas, M.; Gottlob, G.; and Pieris, A. 2014. Expressive languages for querying the semantic web. In *Proc. PODS-2014*, 14–26.

Baget, J.; Leclère, M.; Mugnier, M.; and Salvat, E. 2011a. On rules with existential variables: Walking the decidability line. *Artif. Intell.* 175(9-10):1620–1654.

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

Baget, J.; Garreau, F.; Mugnier, M.; and Rocher, S. 2014. Extending acyclicity notions for existential rules. In *Proc. ECAI-2014*, 39–44.

Beeri, C., and Vardi, M. Y. 1981. The implication problem for data dependencies. In *Proc. ICALP-1981*, 73–85.

Bourhis, P.; Morak, M.; and Pieris, A. 2013. The impact of disjunction on query answering under guarded-based existential rules. In *Proc. IJCAI-2013*, 796–802.

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

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

Calì, A.; Gottlob, G.; and Pieris, A. 2010. Query answering under non-guarded rules in Datalog+/-. In *Proc. RR-2010*, 1–17.

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

Dantsin, E.; Eiter, T.; Gottlob, G.; and Voronkov, A. 2001. Complexity and expressive power of logic programming. *ACM Comput. Surv.* 33(3):374–425.

Deutsch, A.; Nash, A.; and Remmel, J. 2008. The chase revisited. In *Proc. PODS-2008*, 149–158.

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

Gottlob, G.; Rudolph, S.; and Simkus, M. 2014. Expressiveness of guarded existential rule languages. In *Proc. PODS-2014*, 27–38.

Grau, B. C.; Horrocks, I.; Krötzsch, M.; Kupke, C.; Magka, D.; Motik, B.; and Wang, Z. 2013. Acyclicity notions for existential rules and their application to query answering in ontologies. *J. Artif. Intell. Res.* 47:741–808.

Greco, S.; Spezzano, F.; and Trubitsyna, I. 2011. Stratification criteria and rewriting techniques for checking chase termination. *PVLDB* 4(11):1158–1168.

Krötzsch, M., and Rudolph, S. 2011. Extending decidable existential rules by joining acyclicity and guardedness. In *Proc. IJCAI-2011*, 963–968.

Leone, N.; Manna, M.; Terracina, G.; and Veltri, P. 2012. Efficiently computable datalog$^\exists$ programs. In *Proc. KR-2012*, 13–23.

Marnette, B. 2009. Generalized schema-mappings: from termination to tractability. In *Proc. PODS-2009*, 13–22.

Zhang, H.; Zhang, Y.; and You, J.-H. 2014. Existential rule languages with finite chase: Complexity and expressiveness. *CoRR* abs/1411.5220.