Expressiveness of Guarded Existential Rule Languages

Georg Gottlob
Department of Computer Science and Oxford Man Institute
University of Oxford, UK
georg.gottlob@cs.ox.ac.uk

Sebastian Rudolph
Institute of Artificial Intelligence
Technische Universität Dresden, Germany
sebastian.rudolph@tu-dresden.de

Mantas Šimkus
Institute of Information Systems
Vienna University of Technology, Austria
simkus@dbai.tuwien.ac.at

ABSTRACT

The so-called *existential rules* have recently gained attention, mainly due to their adequate expressiveness for ontological query answering. Several decidable fragments of such rules have been introduced, employing restrictions such as various forms of *guardedness* to ensure decidability. Some of the more well-known languages in this arena are *(weakly) guarded* and *(weakly) frontier-guarded* fragments of existential rules. In this paper, we explore their relative and absolute expressiveness. In particular, we provide a new proof that queries expressed via frontier-guarded and guarded rules can be translated into plain Datalog queries. Since the converse translations are impossible, we develop generalizations of frontier-guarded and guarded rules to *nearly frontier-guarded* and *nearly guarded* rules, respectively, which have exactly the expressive power of Datalog. We further show that weakly frontier-guarded rules can be translated into weakly guarded rules, and thus, weakly frontier-guarded and weakly guarded rules have exactly the same expressive power. Such rules cannot be expressed in Datalog since their query answering problem is ExpTime-complete in data complexity. We strengthen this completeness result by proving that on ordered databases with input negation available, weakly guarded rules capture all queries computable in exponential time. We then show that weakly guarded rules extended with *stratified negation* are expressive enough to capture all database queries decidable in exponential time, without any assumptions on the input databases. Finally, we note that the translations of this paper are, in general, exponential in size, but lead to worst-case optimal algorithms for query answering with the considered languages.

Categories and Subject Descriptors

H.2 [**Database Management**]: General

Keywords

Existential rules; Expressiveness; Descriptive Complexity

PODS'14, June 22–27, 2014, Snowbird, UT, USA.

http://dx.doi.org/10.1145/2594538.2594556.

1. INTRODUCTION

Rule-based logical formalisms play a pivotal role in databases and knowledge representation. They are used in databases as expressive constraint and query languages, and in knowledge representation for declarative problem solving via various forms of logic programming, and, more recently, to represent and reason about ontological information.

Consequently, rule languages can be employed in at least two different ways: as ontology languages and as query languages. In the ontological setting, a set of facts and rules may be used to specify domain knowledge, in this way forming a *knowledge base*. In this setting, we are interested in answering user queries over a knowledge base, where queries may be expressed in a standard query language such as *conjunctive queries*. Alternatively, the rule part can be understood as part of the specification of a query that is executed over a plain database (such queries are sometimes called *ontology-mediated queries*[12]).

Given the plethora of different rule (and thus querying) languages, it is of utmost importance to understand and determine their expressivity. Facing with the problem of picking an adequate query language for some scenario, one has to have a clear picture of which information needs the languages at hand can express, which other query languages they subsume and also if they allow to express all possible queries whose answers can be computed at a certain cost.

To support this crucial considerations, database theory has come up with appropriate notions to compare and characterize query languages: *relative* and *absolute expressiveness*. Relative expressiveness considers if, given two query languages, every query formulated in the first language can be expressed by means of the second (and vice versa). Note that two languages might be equally expressive and still differ significantly in terms of the size of expressions needed to express the same query. Such *succinctness* differences impact the combined complexity of the corresponding entailment problem, whereas the data complexity (where only the database is assumed to vary while the query is fixed) of two equally expressive query languages must be the same. On the other hand, two query languages with coinciding data complexity do not necessarily have the same expressivity. Hence it makes sense to identify the query languages with a certain data complexity that have the maximal expressiveness. This leads us to *absolute expressiveness*, where expressive power is measured in terms of complexity classes using *descriptive complexity theory* [21]. A query language is said to *capture* a complexity class $\mathcal{C}$ if it can express any query that can be answered by a computation in $\mathcal{C}$. Such a lan-

guage is guaranteed to semantically subsume all other query languages with the same data complexity, and can therefore be considered as the best value-for-money (i.e. expressiveness for computation cost) choice in class $\mathcal{C}$.

This paper provides an expressivity analysis for important query languages based on the so-called *existential* rules, which support *value invention*. They allow to reason about objects whose identity is unknown yet whose existence is implied by the specified knowledge. This basic form of inference with incomplete information also lies at the core of reasoning in ontology languages such as *description logics* [5].

Existential rules are first-order logic sentences of the form $\forall\vec{x}\forall\vec{y}\alpha(\vec{x},\vec{y}) \rightarrow \exists\vec{z}\beta(\vec{y},\vec{z})$, where α, β are conjunctions of atoms over constants and variables. Such rules occur in many scenarios and are widely known under different names such as *tuple-generating dependencies* or *Datalog with value invention* in databases [2, 19], *Datalog*$\pm$ in ontological knowledge representation (see [14] and references there in), and *conceptual graph rules* in diagrammatic reasoning [26].

The typical reasoning problem considered in the context of existential rules is whether a ground atom is logically *entailed* by a collection of ground atoms and existential rules.

EXAMPLE 1. *As a running example we use the set Σ_p of rules, which describe (a part of) a publication database:*

$$\sigma_1 = \mathsf{Publication}(x) \rightarrow \exists k_1, k_2.\mathsf{Keywords}(x, k_1, k_2),$$

$$\sigma_2 = \mathsf{Keywords}(x, k_1, k_2) \rightarrow \mathsf{hasTopic}(x, k_1),$$

$$\sigma_3 = \mathsf{hasTopic}(x, z),\ \mathsf{hasAuthor}(x, u),\ \mathsf{hasAuthor}(y, u),\ \mathsf{hasTopic}(y, z'),\ \mathsf{Scientific}(z'),\ \mathsf{citedIn}(y, x) \rightarrow \mathsf{Scientific}(z).$$

In particular, the above rules state that every publication must have at least two keywords, where the first keyword describes the (main) topic of the publication at hand. The last rule provides a recipe to infer scientific topics: a topic is scientific if it is a topic of a paper that cites a scientific paper and shares with it a coauthor.

Suppose we are interested in persons who have authored scientific publications. This can be expressed using the following rule, which we add to Σ_p:

$$\sigma_4 = \mathsf{hasAuthor}(x, y), \mathsf{hasTopic}(x, z), \mathsf{Scientific}(z) \rightarrow \mathsf{Q}(y)$$

We consider the following atom set $D = \{\mathsf{Publication}(\mathrm{p}_1),\ \mathsf{Publication}(\mathrm{p}_2),\ \mathsf{citedIn}(\mathrm{p}_1, \mathrm{p}_2),\ \mathsf{hasAuthor}(\mathrm{p}_1, \mathrm{a}_1),\ \mathsf{hasAuthor}(\mathrm{p}_2, \mathrm{a}_1),\ \mathsf{hasAuthor}(\mathrm{p}_2, \mathrm{a}_2),\ \mathsf{hasTopic}(\mathrm{p}_1, \mathrm{t}_1),\ \mathsf{Scientific}(\mathrm{t}_1)\}$. Intuitively, Σ_p and D together entail $\mathsf{Q}(\mathrm{a}_1)$ and $\mathsf{Q}(\mathrm{a}_2)$, thus a_1 and a_2 are answers to our query.

Entailment checking over existential rules is nontrivial, because in the general case there is no bound on the number of unnamed objects that need to be considered for inferring the relevant information. In fact, theories of existential rules are undecidable already in very restricted cases [6]. To circumvent this, several syntactic and semantic conditions defining decidable fragments of existential rules have been introduced, inspired by positive decidability results in modal and description logics.

For many widely known existential rule languages, decidability is guaranteed by means of various versions of *guardedness*. Among the most expressive such fragments are *guarded and weakly guarded* rules [14] as well as *frontier-guarded* and *weakly frontier-guarded* rules [6]. In a nutshell, a rule is guarded (resp., frontier-guarded) if it has a body atom

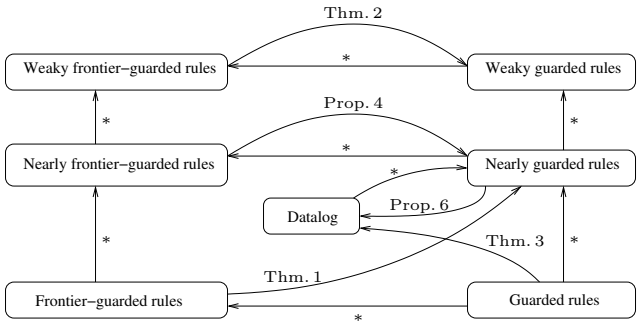

Figure 1: Semantic relations between the considered languages. An arrow from a fragment $\mathcal{L}$ to a fragment $\mathcal{L}'$ indicates that $\mathcal{L}'$ can be expressed in $\mathcal{L}$. Here '*' indicates syntactic membership.

that contains (i.e., "guards") all the universally quantified variables of the rule (resp., of the rule head). The *weakly* guarded and the *weakly* frontier-guarded fragments are obtained by restricting guarding to only "dangerous" variables, i.e. the ones that may be forced to be instantiated with unnamed objects. Our paper focuses on the above formalisms, plain Datalog, as well as the *nearly guarded* and *nearly frontier-guarded* rules that we introduce in this paper.

While there are several results on the combined and the data complexity of these fragments, little is known about their relative and absolute expressiveness. A notable exception is the sketch in [7] and the thorough proof in [8] that frontier-guarded rules are expressible in Datalog. Importantly, such rules can be translated into a small fragment of Datalog, where e.g. FO-rewritability is decidable [8, 10].

In this paper we close significant gaps concerning relative and absolute expressiveness and more practicable translations between the mentioned fragments. Our main contributions are as follows (Figure 1 summarizes the relationships between the considered formalisms):

- We show that all considered query languages with PTIME data complexity have the same expressive power as Datalog. This is a crucial insight as it shows that query answering in those languages can be realized via appropriate translations and subsequent execution on one of the existing optimized Datalog engines.

- In particular, noting that weakly (frontier-)guarded rules have EXPTIME data complexity and thus unfortunately cannot be expressed in Datalog, we introduce *nearly guarded* and *nearly frontier-guarded* rules, which retain as much expressivity of weakly (frontier-)guarded rules (such as syntactically containing plain Datalog) as possible while still being in PTIME and thus expressible in Datalog.

- To show our claims, we propose several translations between rule fragments. These do not depend on an input database and preserve entailment of ground atoms as well as answers to conjunctive queries. Moreover, compared to previously presented translations [7, 8], ours are modular, demonstrably worst-case optimal, much more goal-directed, and hence well-suited for an effective implementation.

  (i) We provide a translation from frontier-guarded into nearly guarded rules (Theorem 1). The translation capitalizes

on a suitable characterization of the relevant models of a frontier-guarded theory in terms of tree-shaped structures.

(ii) The above translation can be lifted to translate nearly frontier-guarded rules into nearly guarded rules (Proposition 4) and weakly frontier-guarded rules into weakly guarded rules (Theorem 2).

(iii) We show that nearly guarded rules can be translated into plain Datalog. This result is established by first providing a resolution based translation from fully guarded rules to plain Datalog (Theorem 3). We then show certain modularity properties that allow to extend the translation to cover nearly guarded rules as well (Proposition 6).

- Our results also show that answering conjunctive queries in knowledge bases where a database is enriched with nearly frontier-guarded rules, can be reduced to answering a Datalog query over the same database.

- Although the presented translations are not polynomial in general, they yield worst-case optimal algorithms for the considered problems. These findings allow to conclude that in most cases, the exponential blow-ups are unavoidable.

- We take a closer look at weakly guarded and weakly frontier-guarded rules, which are known to have ExpTime-complete data complexity. We strengthen this result by showing that on ordered databases with input negation available, weakly guarded rules in fact *capture* all queries computable in exponential time. This result is reminiscent of the fact that semipositive Datalog on ordered databases captures exactly queries computable in polynomial time [31, 28] (see [1] for an overview of this fundamental result).

- We show that weakly guarded rules extended with *stratified negation* are expressive enough to capture ExpTime, without any assumptions on the input databases. Consequently, despite being a rather plain query language, weakly guarded rules endowed with a mild form of negation subsume all languages with ExpTime data complexity. This supplements Cabibbo's proof that existential rules equipped with stratified negation capture all computable queries [13], and also exposes the expressive power of TriQ, a recent RDF query language based on stratified weakly guarded rules [3].

In summary, we show that (a) all considered notions of guardedness for which the data complexity is in PTime can be expressed in plain Datalog and that (b) the variants with ExpTime data complexity, endowed with a very moderate form of negation, actually capture this complexity class, that is, they are able to express all other ExpTime query languages including formalisms as powerful as second-order logic extended with a least fixed point operator.

The paper is organized as follows. After introducing the considered fragments of existential rules and recalling the standard notion of database *chase*, we show in Section 4 that the chase of a frontier-guarded rule set can be seen as a specially constructed tree. In Section 5, we exploit this property to provide translations from frontier-guarded and nearly frontier-guarded rules to nearly guarded rules, and a translation from weakly frontier-guarded rules to weakly guarded rules. In Section 6 we specially tailored inference rules to translate guarded rules into Datalog rules. We then lift this translation to nearly guarded rules. By composing the introduced transformations, we obtain the desired translations. In Section 7 we discuss how our results apply to the problem of answering conjunctive queries over databases enriched with existential rules. In Section 8 we present the capturing result, and then discuss some related work and conclude in Sections 9 and 10.

2. PRELIMINARIES

Existential Rules Let Δ_c, Δ_n and Δ_v be infinite mutually disjoint sets of *constants*, *labeled nulls*, and *variables*, respectively. Elements in $\Delta_c \cup \Delta_n \cup \Delta_v$ are *terms*. An *atom* α is an expression of the form $R(t_1, \ldots, t_n)$, where R is a *relation name* with *arity* n, and $t_1, \ldots, t_n$ are terms. We let $\mathsf{terms}(\alpha) = \{t_1, \ldots, t_n\}$ and $\mathsf{vars}(\alpha) = \mathsf{terms}(\alpha) \cap \Delta_v$. If $\mathsf{terms}(\alpha) \subseteq \Delta_c$, then α is *ground*. For a set Γ of atoms, we let $\mathsf{terms}(\Gamma) = \bigcup_{\alpha \in \Gamma} \mathsf{terms}(\alpha)$ and $\mathsf{vars}(\Gamma) = \mathsf{terms}(\Gamma) \cap \Delta_v$. An *(existential) rule* σ is an expression of the form

$$B_1 \wedge \ldots \wedge B_n \rightarrow \exists y_1, \ldots, y_k . H_1 \wedge \ldots \wedge H_m, \qquad (1)$$

where $B_1, \ldots, B_n$, with $n \geq 0$, and $H_1, \ldots, H_m$, with $m \geq 1$, are atoms with terms from $\Delta_c \cup \Delta_v$ only. We let $\mathsf{body}(\sigma) := \{B_1, \ldots, B_n\}$ and $\mathsf{head}(\sigma) := \{H_1, \ldots, H_m\}$. Let $\mathsf{terms}(\sigma) = \mathsf{terms}(\mathsf{body}(\sigma)) \cup \mathsf{terms}(\mathsf{head}(\sigma))$ and $\mathsf{vars}(\sigma) = \mathsf{terms}(\sigma) \cap \Delta_v$. Moreover, we let $\mathsf{uvars}(\sigma) = \mathsf{vars}(\mathsf{body}(\sigma))$, and let $\mathsf{evars}(\sigma) = \{y_1, \ldots, y_k\}$. The sets $\mathsf{uvars}(\sigma)$ and $\mathsf{evars}(\sigma)$ contain the *universal* and *existential* variables of σ, respectively. The set $\mathsf{fvars}(\sigma) = \mathsf{vars}(\mathsf{head}(\sigma)) \setminus \mathsf{evars}(\sigma)$ is called the *frontier* of σ. We assume that all rules are *safe*, i.e. $\mathsf{fvars}(\sigma) \subseteq \mathsf{vars}(\mathsf{body}(\sigma))$. If $\mathsf{evars}(\sigma) = \emptyset$, then σ is a *Datalog* rule.

A set Σ of rules is called a *theory*. A *Datalog program* is a theory consisting of Datalog rules only.

Databases A *database* D is any set of atoms with terms from $\Delta_c \cup \Delta_n$.[1] Given a set of atoms Γ and a database D, a *homomorphism* from Γ into D is a mapping $h : \Delta_c \cup \Delta_n \cup \Delta_v \rightarrow \Delta_c \cup \Delta_n$ such that

(i) $h(c) = c$ for each $c \in \Delta_c$;
(ii) if $R(t_1, \ldots, t_n) \in \Gamma$, then $R(h(t_1), \ldots, h(t_n)) \in D$.

A database D *satisfies* a rule σ if for any homomorphism h from $\mathsf{body}(\sigma)$ into D, there exists a homomorphism h' from $\mathsf{head}(\sigma)$ into D such that $h'(x) = h(x)$ for all $x \in \mathsf{uvars}(\sigma)$. A database D *satisfies* a theory Σ if D satisfies each $\sigma \in \Sigma$. Given a database D and a theory Σ, a *solution* to (Σ, D) is a database D' such that $D \subseteq D'$ and D' satisfies Σ. Given a ground atom α, we write $\Sigma, D \models \alpha$ if $\alpha \in D'$ for every solution D' to (Σ, D).

Queries A *query* is a pair (Σ, Q), where Σ is a theory and Q is a relation symbol. Given a query (Σ, Q) and a database D, we let $\mathsf{ans}((\Sigma, Q), D) = \{\vec{c} \in (\Delta_c)^n \mid \Sigma, D \models Q(\vec{c})\}$, where n is the arity of Q. We call Q the *output relation* of (Σ, Q) and $\mathsf{ans}((\Sigma, Q), D)$ the *answer* to (Σ, Q) over D.

Chase We recall the notion of *(oblivious) chase* [24, 11]. Assume a database D, a rule σ and a homomorphism h from $\mathsf{body}(\sigma)$ into D. A set of atoms Γ is called a *consequence* of σ w.r.t. D and h if Γ can be obtained from $\mathsf{head}(\sigma)$ by replacing each universal variable x by $h(x)$ and each existential variable by a fresh null $c \in \Delta_n$ not occurring in D. A *chase* of a database D w.r.t. a theory Σ is a potentially infinite sequence $D_0, D_1, \ldots$ of databases such that:
(a) $D_0 = D$;

[1] Whenever a database D is part of an input in a computational problem, D is assumed to be finite.

(b) for each $i > 0$, D_i is a consequence of some $\sigma \in \Sigma$ w.r.t. $D_0 \cup \ldots \cup D_{i-1}$ and some h;
(c) if there is some $\sigma \in \Sigma$, $i \geq 0$, and a homomorphism h from $\mathsf{body}(\sigma)$ into $D_0 \cup \ldots \cup D_i$, then there is $j \geq 0$ s.t. D_j has a consequence of σ w.r.t. $D_0 \cup \ldots \cup D_i$ and h.

Let $\mathsf{chase}(\Sigma, D) = \bigcup_{i \geq 0} D_i$. We remind the reader that $\mathsf{chase}(\Sigma, D)$ is unique up to homomorphic equivalence. It is also well known that $\mathsf{chase}(\Sigma, D)$ is a *universal solution* to (Σ, D), i.e. $\mathsf{chase}(\Sigma, D)$ is a solution to (Σ, D), and there is a homomorphism from $\mathsf{chase}(\Sigma, D)$ into any solution D' to (Σ, D) (see [14]). Observe that due to the universality of the chase, $\Sigma, D \models \alpha$ iff $\alpha \in \mathsf{chase}(\Sigma, D)$, for any ground atom α. This also implies that, given an n-ary relation symbol Q, $\mathsf{ans}((\Sigma, Q), D)$ equals the set constant tuples $\vec{c}$ with $Q(\vec{c}) \in \mathsf{chase}(\Sigma, D)$. With a slight abuse of notation, we will later write $\mathsf{chase}(\Sigma, D) \subseteq \mathsf{chase}(\Sigma', D')$ if there is a homomorphism from $\mathsf{chase}(\Sigma, D)$ to $\mathsf{chase}(\Sigma', D')$. If $\mathsf{chase}(\Sigma, D)$ and $\mathsf{chase}(\Sigma', D')$ are homomorphically equivalent, then we simply write $\mathsf{chase}(\Sigma, D) = \mathsf{chase}(\Sigma', D')$.

Relation name annotations It will sometimes be useful to encode some information as part of relation names. To this end, we will consider *annotated* relation names that have the form $R[\vec{t}]$, where $\vec{t}$ is a tuple of terms. A theory Σ is *safely annotated* if the following is true for every rule $\sigma \in \Sigma$:

(i) If $R[\vec{t}](\vec{v})$ is an atom in σ, then none of the variables of $\vec{t}$ occurs as an argument in an atom of σ.
(ii) If $x \in \Delta_v$ occurs in the annotation of an atom in $\mathsf{head}(\sigma)$, then x also occurs in the annotation of some atom in $\mathsf{body}(\sigma)$.

Further Notions We assume a unary *active constant domain* relation ACDom whose extension is fixed: for any database D, $\mathsf{ACDom}(c) \in D$ iff c occurs in some atom $R(\vec{v}) \in D$ with $R \neq \mathsf{ACDom}$. This assumption will help presentation, but won't affect generality. We also prohibit ACDom from rule heads.

For a possibly partial function f, let $\mathsf{dom}(f)$ and $\mathsf{ran}(f)$ denote the domain and the range of f, respectively. For a set of variables X, $\vec{X}$ is the tuple obtained by enumerating X. The enumeration of variable sets is globally fixed, i.e. for two sets X, Y of variables, $X = Y$ implies $\vec{X} = \vec{Y}$.

3. VARIATIONS OF GUARDED RULES

We remind that syntactic restriction on existential rules are required to ensure decidability of the basic reasoning tasks such as query answering. In this section we recall the so-called guarded, frontier-guarded, weakly guarded and weakly frontier-guarded rules. We also define nearly guarded and nearly frontier-guarded rules.

DEFINITION 1. *((Frontier-)Guarded rules) We say a rule σ is* guarded *(resp.,* frontier-guarded*) if there exists an atom $\alpha \in \mathsf{body}(\sigma)$ such that $\mathsf{uvars}(\sigma) \subseteq \mathsf{vars}(\alpha)$ (resp., such that $\mathsf{fvars}(\sigma) \subseteq \mathsf{vars}(\alpha)$). A theory is* guarded *(resp.,* frontier-guarded*) if all its rules are guarded (resp., frontier-guarded). For a frontier-guarded rule σ, let $\mathsf{fg}(\sigma)$ be an arbitrary but fixed frontier-guard α in σ, i.e. an atom containing all the variables of $\mathsf{fvars}(\sigma)$.*

EXAMPLE 2. *It is easy to see that Σ_p as defined in Example 1 is frontier-guarded. In Figure 2 we present the corresponding chase(Σ_p, D), witnessing $\Sigma_p, D \models \mathsf{Q}(\mathrm{a}_1)$ and $\Sigma_p, D \models \mathsf{Q}(\mathrm{a}_2)$.*

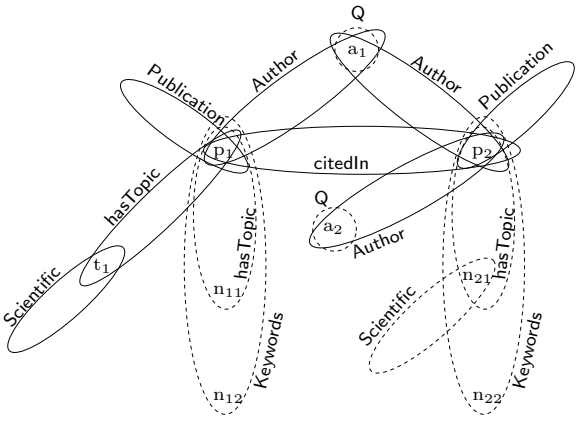

Figure 2: Illustration of $chase(\Sigma_p, D)$ for Σ_p and D from Ex. 1. The solid ellipses indicate the atoms of D and dashed ones indicate the inferred atoms.

Guarded and frontier-guarded theories cannot express all Datalog queries. To see this, assume a query (Σ, Q), where Σ is a frontier-guarded theory with no constants. It can be easily seen that for any tuple $\vec{c} \in \mathsf{ans}((\Sigma, Q), D)$, where D is a database, all constants from $\vec{c}$ must occur together in some database fact $R(\vec{t}) \in D$. In other words, Σ cannot relate constants that are not explicitly related in the input database (allowing for constants in Σ does not solve the issue in general). This property rules out using frontier-guarded rules to query e.g. the transitive closure of a binary relation, which is a classic task for Datalog.

The above expressiveness limitations can be resolved by employing weakly (frontier-)guarded theories, obtained by relaxing (frontier-)guardedness.

DEFINITION 2. *(Weakly (frontier-)guarded rules) Given a variable x and a set of atoms Γ, let $\mathsf{pos}(\Gamma, x)$ be the set of pairs (R, i) such that Γ has an atom $R(t_1, \ldots, t_n)$ with $t_i = x$, i.e. Γ has an atom where x occurs in position i. We collect the* affected positions *in relations of a theory Σ. Let $\mathsf{ap}(\Sigma)$ be the smallest set such that:*

i) $\mathsf{pos}(\mathsf{head}(\sigma), x) \subseteq \mathsf{ap}(\Sigma)$ for all $\sigma \in \Sigma$ and $x \in \mathsf{evars}(\sigma)$, i.e. all positions where existential variables occur are affected; and

ii) if $\sigma \in \Sigma$ and $x \in \mathsf{uvars}(\sigma)$ is such that $\mathsf{pos}(\mathsf{body}(\sigma), x) \subseteq \mathsf{ap}(\Sigma)$, then $\mathsf{pos}(\mathsf{head}(\sigma), x) \subseteq \mathsf{ap}(\Sigma)$, i.e. if all positions where x occurs in the body are affected, then all positions where x occurs in the head are affected as well.

Assume a theory Σ. We say a variable x of a rule σ is unsafe *w.r.t. Σ if $\mathsf{pos}(\mathsf{body}(\sigma), x) \subseteq \mathsf{ap}(\Sigma)$. We use $\mathsf{unsafe}(\sigma, \Sigma)$ to denote the variables of σ that are unsafe w.r.t. Σ. We say a rule σ is* weakly guarded *(resp.,* weakly frontier-guarded*) in Σ if it has a body atom B that contains all the variables of $\mathsf{uvars}(\sigma) \cap \mathsf{unsafe}(\sigma, \Sigma)$ (resp., of $\mathsf{fvars}(\sigma) \cap \mathsf{unsafe}(\sigma, \Sigma)$). Then Σ is* weakly guarded *(resp.,*weakly frontier-guarded*) if each rule of Σ is weakly-guarded (resp., weakly frontier-guarded) w.r.t. Σ.*

Weakly guarded and weakly frontier-guarded theories are significantly more expressive than plain Datalog. This is due to the fact that query answering in these languages is

ExpTime-hard in data complexity [14]. Since query answering in plain Datalog is PTime-complete in data complexity (see e.g. [16]), one generally cannot convert a query (Σ, Q), where Σ is weakly guarded or weakly frontier-guarded, into a Datalog query while preserving the query answers.

We introduce next *nearly guarded* and *nearly frontier-guarded* rules, which strictly extend guarded and frontier-guarded rules, yet less expressive than weakly guarded and weakly-frontier guarded theories, respectively. We will see later that they have exactly the expressive power of Datalog.

Definition 3. *(Nearly (frontier-)guarded rules) We say a rule σ is* nearly guarded *(resp.,* nearly frontier-guarded*) in a theory Σ if*

(i) σ is guarded (resp., frontier-guarded), or

(ii) $\mathsf{unsafe}(\sigma, \Sigma) = \mathsf{evars}(\sigma) = \emptyset$.

Then Σ is nearly guarded *(resp.,* nearly frontier-guarded*) if each rule of Σ is nearly guarded (resp., nearly frontier-guarded) w.r.t. Σ.*

Intuitively, every non-guarded rule of a nearly guarded theory only "operates" on constants from the input database. This is similar in spirit to the definition of description logics extended with *DL-safe* rules [25].

Normalization To make presentation easier, we will manipulate structurally simplified theories, defined as follows:

Definition 4. *(Normalized theories) A theory Σ is in* normal form *(or, is* normal*) if the following are satisfied:*

(i) For every $\sigma \in \Sigma$, we have $|\mathsf{head}(\sigma)| = 1$.

(ii) Every $\sigma \in \Sigma$ with $\mathsf{evars}(\sigma) \neq \emptyset$ is guarded. In other words, if σ is not guarded, then it is a Datalog rule.

(iii) If $\sigma \in \Sigma$ has an occurrence of a constant, then σ is of the form $\rightarrow R(c)$.

Proposition 1. *Every query (Σ, Q) can be transformed in logarithmic space into a query (Σ', Q') such that*

(a) Σ' is normal;

(b) $\mathsf{ans}((\Sigma, Q), D) = \mathsf{ans}((\Sigma', Q'), D)$ for any database D;

(c) if Σ is weakly frontier-guarded (resp., weakly guarded, nearly frontier-guarded, nearly guarded), then Σ' is weakly frontier-guarded (resp., weakly guarded, nearly frontier-guarded, nearly guarded);

4. CHASE TREE

The translations of this paper build on the fact the chase of a database w.r.t. a frontier-guarded theory has the shape of a tree. It is a known fact that $\mathsf{chase}(\Sigma, D)$, where Σ is weakly-guarded, has treewidth that is polynomial in the size of Σ and D [14]. We will (indirectly) make use of this property. We show that the chase of a frontier-guarded theory can be seen as the construction of a tree (the *chase tree*), whose root stores the atoms over the original constants of the input database, while the non-root nodes store atoms with labeled nulls. This representation turns out to be more informative than the existing representations in terms of tree decompositions. In particular, we exploit the fact that the structure of non-leaf nodes is only dependent on the theory and is independent from the database.

Definition 5. *(Minimal nodes) Assume a tree T whose nodes are sets of atoms. Given a set C of terms, a node d in T is called C-*minimal *if $C \subseteq \mathsf{terms}(d)$ and d has no parent d' with $C \subseteq \mathsf{terms}(d')$.*

We are ready to the define the notion of *chase tree*.

Definition 6. *(Chase tree) Assume a database D, a normal frontier-guarded theory Σ, and suppose*

$$D, \{R_1(\vec{t}_1)\}, \{R_2(\vec{t}_2)\} \ldots$$

is a chase of D w.r.t. Σ. Suppose each $\{R_i(\vec{t}_i)\}$ is a consequence of $\sigma_i \in \Sigma$ w.r.t. $D \cup \{R_1(\vec{t}_1)\} \cup \ldots \cup \{R_{i-1}(\vec{t}_{i-1})\}$ and a homomorphism h_i. Then a chase tree *T of D w.r.t. Σ is a tree built as follows. Initially, T consists of the single node*

$$d_0 = D \cup \{R(c) \mid \rightarrow R(c) \in \Sigma\}.$$

Subsequently each $R_i(\vec{t}_i)$ from $R_1(\vec{t}_1), R_2(\vec{t}_2), \ldots$ is added to T in the given order as follows:

(C1) if T has a node d with $\vec{t}_i \subseteq \mathsf{terms}(d)$, then add $R_i(\vec{t}_i)$ to a $\vec{t}_i$-minimal node in T;

(C2) otherwise, create a new node $\{R_i(\vec{t}_i)\}$ and set it as a child of some $\{h(x_0), \ldots, h(x_n)\}$-minimal node of T, where $\{x_0, \ldots, x_n\} = \mathsf{fvars}(\sigma_i)$.

Proposition 2. *Assume a database D, a normal frontier-guarded theory Σ, and let T be a chase tree of D w.r.t. Σ. Let m be the highest relation arity over all relations in Σ, and let k be the number of constants occurring in rules of Σ (recall that due to normalization such rules have the form $\rightarrow R(c)$). Then the following hold:*

(P1) $|\mathsf{terms}(d_0)| \leq |\mathsf{terms}(D)| + k$ for the root d_0 of T,

(P2) $|\mathsf{terms}(d)| \leq m$ for all non-root nodes d of T.

(P3) For any set C of terms, there is at most one C-minimal node in T.

The above proposition also shows the small tree-width of $\mathsf{chase}(\Sigma, D)$ for frontier-guarded rules. Take the chase tree T of D w.r.t. Σ and take the function L that maps each node d in T to $\mathsf{terms}(d)$. As easily seen, (T, L) is a tree decomposition of (the hypergraph of) $\mathsf{chase}(\Sigma, D)$ of width $max(|D|+k, m)$ with m the highest relation arity in Σ and k the number of constants in Σ. We only note that the satisfaction of the *connectedness condition* for (T, L) is guaranteed by the uniqueness of C-minimal nodes in T (the property (P3)). Indeed, if T had a term t that would induce several disconnected trees in T, then there would exist several $\{t\}$-minimal nodes in T.

5. FROM FRONTIER-GUARDEDNESS TO GUARDEDNESS

We start by providing a translation from frontier-guarded rules to nearly guarded rules. We then extend it to also cover nearly frontier-guarded and weakly frontier-guarded rules.

5.1 From Frontier-guarded to Nearly Guarded Rules

We show how a normal frontier-guarded theory Σ can be rewritten into a nearly guarded theory $\mathsf{rew}(\Sigma)$ such that $\Sigma, D \models \alpha$ iff $\mathsf{rew}(\Sigma), D \models \alpha$ for any database D and a ground

atom $\alpha = R(\vec{t})$, where R is a relation from Σ. Note that this implies $\mathsf{ans}((\Sigma, Q), D) = \mathsf{ans}((\mathsf{rew}(\Sigma), Q), D)$ for any query (Σ, Q) and database D, where Σ is frontier-guarded. The translation exploits the fact that frontier-guarded theories enjoy universal solutions that can be represented as trees (see Proposition 2).

Clearly, the difficult part is that $\mathsf{body}(\sigma)$ of a frontier-guarded rule σ can be seen as a complex (possibly cyclic) structure whose variables may have to be mapped rather arbitrarily into labeled nulls during the construction of the chase. This is in contrast to weakly and nearly guarded theories, where the possibly "cyclic part" of a body of a each rule only maps to original constants. Nevertheless, due to Proposition 2, the chase of a frontier-guarded theory is tree-like, and hence the relevant homomorphisms from $\mathsf{body}(\sigma)$ are only the ones that map into tree-like structures, given by partially constructed chase trees. In our translation we capture the set such homomorphisms for σ by a (potentially exponential) set of nearly guarded rules.

We first define the notion of *selection* in a rule. Intuitively, we select a set variables and obtain a group of at most k variables, where k is the maximal arity over all relations appearing in Σ.

Definition 7. *(Selection) Let Σ be a normal frontier-guarded theory and assume $\sigma \in \Sigma$. A* selection *for σ is a partial function μ from $\mathsf{uvars}(\sigma)$ to $\mathsf{uvars}(\sigma)$ such that $|\mathsf{ran}(\mu)| \leq k$ with k the maximal arity over all relations appearing in Σ. Given a set Γ of atoms, we use $\mu(\Gamma)$ to denote the result of replacing in Γ each variable $x \in \mathsf{dom}(\mu)$ by $\mu(x)$.*

We collect the rule atoms that are *covered* by a selection:

Definition 8. *(Covered atoms) Assume a normal frontier-guarded theory Σ, $\sigma \in \Sigma$, and a selection μ for σ. We define the following:*

$$\mathsf{cov}(\sigma, \mu) = \{B \in \mathsf{body}(\sigma) \mid \mathsf{vars}(B) \subseteq \mathsf{dom}(\mu)\}.$$

We will use two kinds of transformations. Intuitively, given a rule σ and a selection μ, the first transformation will pull out from σ all the atoms that are covered by μ. The second one will instead pull out of σ all the atoms that are *not* covered by μ. The removed atoms will be moved to a fresh rule. To preserve soundness, we must keep track of variables that occur both in the range of the selection and in a body atom not covered by the selection. The same holds for variables that occur in the head of σ.

Definition 9 (Variables to keep). *Assume a normal frontier-guarded theory Σ, $\sigma \in \Sigma$, and a selection μ for σ. We let $\mathsf{keep}(\sigma, \mu)$ be the set of all $\mu(x)$ such that $x \in \mathsf{dom}(\mu)$ and x occurs in $\mathsf{body}(\sigma) \setminus \mathsf{cov}(\sigma, \mu)$ or $\mathsf{head}(\sigma)$.*

We are now ready to define the two rewriting steps that we use. In both cases, a frontier-guarded rule σ is split into a pair σ', σ'' of rules, where one of them becomes guarded, while the other one is frontier-guarded but structurally less complex than σ. In the rewriting we employ annotated relation names, but they will only become relevant in next section for dealing with weakly frontier-guarded rules.

Definition 10. *(Remove-covered (rc)) Assume a non-guarded Datalog rule σ in a normal frontier-guarded theory Σ, and a selection μ for σ. An* rc-rewriting *of σ w.r.t. μ is any pair of σ', σ'' of rules obtained as follows:*

(i) $\sigma' = R(\vec{x}) \wedge \mu(\mathsf{cov}(\sigma, \mu)) \rightarrow H(\vec{y})$, where

- *(a) (guarding) R is an arbitrary relation from Σ, and $\vec{x}$ contains each variable of σ',*
- *(b) (variable projection) $\mu(\mathsf{cov}(\sigma, \mu))$ has a variable z such that $z \notin \vec{y}$, and*
- *(c) (fresh atom) H is a fresh relation name and $\vec{y} = \mathsf{keep}(\sigma, \mu)$. H has the annotation of $\mathsf{head}(\sigma)$.*

(ii) $\sigma'' = H(\vec{y}) \wedge \mu(\mathsf{body}(\sigma) \setminus \mathsf{cov}(\sigma, \mu)) \rightarrow \mu(\mathsf{head}(\sigma))$.

Example 3. *Consider the rule $\sigma = R(x_0, x_1), R(x_1, x_2), R(x_2, x_3), R(x_3, x_4), R(x_4, x_1) \rightarrow P(x_1)$. Clearly, σ is not guarded, but is frontier-guarded. Take the partial function $\mu = \{x_4 \rightarrow x_2, x_2 \rightarrow x_2, x_3 \rightarrow x_3\}$. Then $\mathsf{cov}(\sigma, \mu) = \{R(x_2, x_3), R(x_3, x_4)\}$ and $\mathsf{keep}(\sigma, \mu) = \{x_2\}$. An rc-rewriting σ', σ'' of σ w.r.t. μ can be obtained as follows. Let*

$$\sigma'' = R(x_0, x_1), R(x_1, x_2), R(x_2, x_1), A(x_2) \rightarrow P(x_1),$$

where A is a fresh relation name. In case, Q is a relation in Σ of arity 3, we can let

$$\sigma' = R(x_2, x_3), R(x_3, x_2), Q(x_2, x_3, y) \rightarrow A(x_2).$$

Observe that σ' is guarded, while σ'' is not guarded but frontier-guarded with the variables x_4, x_3 vanished.

Example 4. *Recall the rule σ_4 from Example 1. Take the function $\mu = \{x \rightarrow x, z \rightarrow z\}$. Then $\mathsf{cov}(\sigma_4, \mu) = \{\mathsf{hasTopic}(x, z), \mathsf{Scientific}(z)\}$ and $\mathsf{keep}(\sigma_4, \mu) = \{x\}$. Then the following pair is an rc-rewriting of σ_4 w.r.t. μ:*

$$\begin{aligned} \mathsf{hasAuthor}(x, y), \mathsf{Aux}(x) &\rightarrow \mathsf{Q}(y), \\ \mathsf{Keywords}(x, z, u), \mathsf{hasTopic}(x, z), \mathsf{Scientific}(z) &\rightarrow \mathsf{Aux}(x), \end{aligned}$$

where $\mathsf{Aux}(y)$ is a fresh relation name. In this example, we use $\mathsf{Keywords}(y, z, u)$ as a guard in the second rule. Observe that the two resulting rules are guarded.

Definition 11. *(Remove-non-covered (rnc)) Assume a non-guarded Datalog rule σ in a normal frontier-guarded theory Σ, and a selection μ for σ. An* rnc-rewriting *of σ w.r.t. μ is any pair of σ', σ'' of rules obtained as follows:*

(i) $\sigma' = R(\vec{x}) \wedge \mu(\mathsf{body}(\sigma) \setminus \mathsf{cov}(\sigma, \mu)) \rightarrow H(\vec{y})$, where

- *(a) (frontier-guarding) R is a relation from Σ and $\vec{x}$ contains each variable in $\vec{y}$;*
- *(b) (variable projection) $\vec{x}$ has a variable z such $z \notin \vec{y}$ and z occurs in $\mu(\mathsf{body}(\sigma) \setminus \mathsf{cov}(\sigma, \mu))$;*
- *(c) (fresh atom) H is a fresh relation name and $\vec{y} = \mathsf{keep}(\sigma, \mu)$; H has the annotation of $\mathsf{head}(\sigma)$;*

(ii) $\sigma'' = P(\vec{z}) \wedge H(\vec{y}) \wedge \mu(\mathsf{cov}(\sigma, \mu)) \rightarrow \mu(\mathsf{head}(\sigma))$, where

- *(guarding) P is a relation from Σ and $\vec{z}$ contains each variable of σ''.*

Example 5. *Assume the rule $\sigma = R(x_1, x_2), R(x_2, x_3), R(x_3, x_4), R(x_4, x_1), R(x_4, x_5) \rightarrow P(x_1, x_2)$. Again, σ is not guarded, but is frontier-guarded. Take the partial function $\mu = \{x_1 \rightarrow x_1, x_2 \rightarrow x_2, x_3 \rightarrow x_3\}$. Then $\mathsf{cov}(\sigma, \mu) = \{R(x_1, x_2), R(x_2, x_3)\}$ and $\mathsf{keep}(\sigma, \mu) = \{x_1, x_3\}$. An rnc-rewriting σ', σ'' of σ w.r.t. μ can be obtained as follows. Let σ' be the next rule:*

$$W(x_1, x_3, x_4), R(x_3, x_4), R(x_4, x_1), R(x_4, x_5) \rightarrow A(x_1, x_3),$$

where A is a fresh relation name and W is a ternary relation in Σ. In case Q is a relation in Σ of arity 3, we can let σ'' be as follows:

$$Q(x_1, x_3, x_2), A(x_1, x_3), R(x_1, x_2), R(x_2, x_3) \rightarrow P(x_1, x_2).$$

We have that σ'' is guarded, while σ' is frontier-guarded.

EXAMPLE 6. *Recall the rule σ_3 from Example 1. As in the previous example, take the function $\mu = \{x \rightarrow x, z \rightarrow z\}$. Then $\mathsf{cov}(\sigma_3, \mu) = \{\mathsf{hasTopic}(x, z), \mathsf{Scientific}(z)\}$ and $\mathsf{keep}(\sigma_3, \mu) = \{x\}$. Then the following pair of rules is an rnc-rewriting of σ_3 w.r.t. μ:*

$$\begin{array}{rcl} \mathsf{hasAuthor}(x, u), \mathsf{hasAuthor}(y, u), & & \\ \mathsf{hasTopic}(y, z'), \mathsf{Scientific}(z'), \mathsf{citedIn}(y, x) & \rightarrow & \mathsf{Aux}(x) \\ \mathsf{Keywords}(x, z, u), \mathsf{hasTopic}(x, z), \mathsf{Aux}'(x) & \rightarrow & \mathsf{Scientific}(z), \end{array}$$

where $\mathsf{Aux}'(y)$ is a fresh relation. The atom $\mathsf{Keywords}(y, z, u)$ is used as a guard in the second rule. Observe that the first rule is frontier-guarded and second rule is fully guarded.

An *expansion* of a frontier-guarded theory is obtained by closing it under rc-rewritings and rnc-rewritings.

DEFINITION 12. *(Expansion) An* expansion $\mathsf{ex}(\Sigma)$ *of a normal frontier-guarded theory Σ is obtained by exhaustively adding rules to Σ as follows: for each Datalog rule $\sigma \in \Sigma$ and each selection μ for σ, add to Σ all the rc-rewritings and rnc-rewritings of σ w.r.t. μ.*

Note that applying an rc-rewriting or an rnc-rewriting on a non-guarded rule leads to a guarded rule and a frontier-guarded rule that has, due to variable projection, strictly less variables that do not occur in a frontier guard. Thus the expansion can be computed in exponential time in the size of the input theory.

We can now finalize our transformation from frontier-guarded rules to nearly guarded rules. To this end, we simply add atoms of the form $\mathsf{ACDom}(x)$ to bodies of rules in $\mathsf{ex}(\Sigma)$ to ensure near guardedness of the resulting theory.

DEFINITION 13. *(Rewriting) The* rewriting $\mathsf{rew}(\Sigma)$ *of a normal frontier-guarded theory Σ is obtained from $\mathsf{ex}(\Sigma)$ by adding to each non-guarded rule $\sigma \in \mathsf{ex}(\Sigma)$ the atom $\mathsf{ACDom}(x)$ for each universal variable x of σ.*

The translation above leads to a nearly guarded theory.

PROPOSITION 3. *Let Σ be a normal frontier-guarded theory. Then $\mathsf{rew}(\Sigma)$ is nearly guarded.*

We are ready to state the main result of this section.

THEOREM 1. *Let Σ be a normal frontier-guarded theory. Then $\Sigma, D \models \alpha$ iff $\mathsf{rew}(\Sigma), D \models \alpha$ for any database D and ground atom α over Σ.*

PROOF SKETCH. To prove the claim, it suffices to show that $\alpha \in \mathsf{chase}(\Sigma, D)$ iff $\alpha \in \mathsf{chase}(\mathsf{rew}(\Sigma), D)$. For the "if" direction, one can show that, considering only atoms over the signature of Σ, each inference of an atom made in the construction of $\mathsf{chase}(\mathsf{rew}(\Sigma), D)$ can also be made by employing rules of Σ. For the "only if" direction, we can employ Proposition 2. Assume a non-guarded rule $\sigma \in \Sigma$ and suppose $\mathsf{body}(\sigma)$ has a homomorphism h into the (partially constructed) $\mathsf{chase}(\Sigma, D)$ that maps m variables to labeled nulls. By Proposition 2, $\mathsf{chase}(\Sigma, D)$ can be decomposed into a tree of atom sets such that each non-leaf node is over at most k constants, where k is the maximal arity of relations in Σ. Take a node d such that d has an atom from $h(\mathsf{body}(\sigma))$ and no descendant of d has an atom from $h(\mathsf{body}(\sigma))$. In case $h(\mathsf{fg}(\sigma)) \notin d$ (resp., $h(\mathsf{fg}(\sigma)) \in d$), one can pick a selection μ and define an rc-rewriting (resp., rnc-rewriting) of σ', σ'' of σ together with respective homomorphisms that give the same consequence as σ. Importantly, if the rewriting produces a non-guarded rule, then its body homomorphism employs strictly less than d labeled nulls. In this way one shows that every ground $\alpha \in \mathsf{chase}(\Sigma, D)$ can be derived by rules in $\mathsf{ex}(\Sigma)$ that are either guarded or don't employ labeled nulls in body homomorphisms. □

For the class of frontier-guarded theories with singleton heads, our translation into nearly guarded rules does not increase the maximal arity of relations. Entailment of ground atoms in frontier-guarded theories with singleton heads is 2-EXPTIME-hard even in case relation arities are bounded by a constant [7]. On the other hard, the same problem under bounded arities is EXPTIME-complete for weakly-guarded rules [14]. Thus the existence of polynomial arity-preserving translation would imply 2-EXPTIME = EXPTIME.

It is not difficult to extend the previous translation to cover nearly frontier-guarded rules. Assume a nearly frontier-guarded theory Σ. If a rule $\sigma \in \Sigma$ is not frontier guarded, then by definition of near frontier-guardedness, σ has no unsafe variable and thus during chasing its body is never mapped to labeled nulls. Such rules simply need no rewriting, which we see more formally as follows.

DEFINITION 14. *Assume a nearly frontier-guarded theory Σ. Let Σ_f denote the frontier-guarded rules in Σ and let $\Sigma_d = \Sigma \setminus \Sigma_f$ (note that $\mathsf{unsafe}(\sigma, \Sigma) = \mathsf{evars}(\sigma) = \emptyset$ for all $\sigma \in \Sigma_d$). We define $\mathsf{rew}(\Sigma) = \mathsf{rew}(\Sigma_f) \cup \Sigma_d$.*

PROPOSITION 4. *Assume a query (Σ, Q) with Σ a normal nearly frontier-guarded theory. We have that $\mathsf{rew}(\Sigma)$ is nearly guarded and $\mathsf{ans}((\Sigma, Q), D) = \mathsf{ans}((\mathsf{rew}(\Sigma), Q), D)$ for any database D.*

Axiomatizing the Relation ACDom Due to Definition 13, the theory $\mathsf{rew}(\Sigma)$ resulting from a theory Σ may contain atoms of the form $\mathsf{ACDom}(x)$. Such atoms can be eliminated from $\mathsf{rew}(\Sigma)$ while preserving ground atomic consequences, modulus renaming of relation names.

DEFINITION 15. *Assume a nearly guarded theory Σ. For every relation R of Σ, let R^* be a fresh relation of the same arity as R. Note that this includes the fresh relation ACDom^* in case ACDom occurs in Σ. Let Σ^* be the theory obtained from Σ by replacing every atom $R(\vec{t})$ by $R^*(\vec{t})$ and adding the following rules:*

(a) $R(x_1, \ldots, x_n) \rightarrow R^*(x_1, \ldots, x_n)$ for every n-ary R of Σ,

(b) $R(x_1, \ldots, x_n) \rightarrow \mathsf{ACDom}^*(x_i)$ for every n-ary relation R of Σ and $i \in \{1, \ldots, n\}$, and

(c) $\rightarrow \mathsf{ACDom}^*(c)$ for every constant c of Σ.

The following is not difficult to check:

PROPOSITION 5. *Assume a query (Σ, Q) with Σ nearly guarded. Then (Σ^*, Q^*) is a query such that*

i) Σ^* is a nearly guarded theory with no occurrences of the built-in relation ACDom, and

ii) $\mathsf{ans}((\Sigma, Q), D) = \mathsf{ans}((\Sigma^*, Q^*), D)$, for any database D.

5.2 Translating Weakly Frontier-guarded Rules

We have seen above a translation from frontier-guarded and nearly frontier-guarded rules to nearly guarded rules. We show next that it can be extended to also cover weakly frontier-guarded rules. Such rules are treated in three steps:

(a) converting them into frontier-guarded rules by removing the terms occurring in non-affected positions from atoms and storing them in relation annotations,

(b) converting the resulting frontier-guarded theory into a nearly guarded theory using the translation of the previous section, and

(c) reconstructing original atoms from relation annotations, thus obtaining a weakly guarded theory.

We make this more precise next.

To simplify presentation, we will use a convent ordering of positions in relations of a weakly frontier-guarded theory.

Definition 16. *(Proper theories) We say a weakly frontier-guarded theory Σ is* proper *if the following holds: $(R, i) \notin \mathsf{ap}(\Sigma)$ with $i \geq 1$ implies $(R, i+1) \notin \mathsf{ap}(\Sigma)$.*

In other words, if Σ is proper, then each atom $R(\vec{t})$ in Σ has an initial sequence of terms in affected positions, followed by terms that appear in non-affected positions only. For the rest of the section, we w.l.o.g. assume only proper theories. Any theory can be transformed in logarithmic space into a proper theory by reordering arguments in atoms.

We next define how atoms with terms in non-affected positions are converted into atoms where such terms are moved into a relation's annotation.

Definition 17. *Assume a weakly frontier-guarded theory Σ. We let $a_\Sigma(R(t_1, \ldots, t_n)) = R[t_{i+1}, \ldots, t_n](t_1, \ldots, t_i)$, where i is the last affected position in R. Given a database D, we let $a_\Sigma(D) = \{a_\Sigma(A) \mid A \in D\}$. Finally, we let $a(\Sigma)$ denote the theory obtained from Σ by replacing each atom A of Σ with $a_\Sigma(A)$.*

As easily seen, if Σ is weakly frontier-guarded, then $a(\Sigma)$ is frontier-guarded. We are ready to finalize the translation from weakly frontier-guarded theories:

Definition 18. *For a frontier-guarded theory Σ with a safe annotation, we use $a^-(\Sigma)$ to denote the theory obtained from Σ by replacing each atom $R[\vec{v}](\vec{t})$ by $R(\vec{t}, \vec{v})$.*

For a normal weakly frontier-guarded theory Σ, we let $\mathsf{rew}(\Sigma) = a^-(\mathsf{rew}(a(\Sigma)))$.

We are ready to state the main result of this section.

Theorem 2. *Take a query (Σ, Q) with Σ a normal weakly frontier-guarded theory. Then $\mathsf{rew}(\Sigma)$ is weakly guarded and $\mathsf{ans}((\Sigma, Q), D) = \mathsf{ans}((\mathsf{rew}(\Sigma), Q), D)$ for any database D.*

Proof Sketch. For the second part of the claim, assume (Σ, Q) as above and let D be a database. For a theory Σ', let $\mathsf{pg}(\Sigma, D)$ denote the theory that can be obtained from Σ' by all possible substitutions of safe variables by constants in D. Let $\Sigma_{pg} = \mathsf{pg}(\Sigma, D)$. It can be verified that $\mathsf{ans}((\Sigma, Q), D) = \mathsf{ans}((\Sigma_{pg}, Q), D)$ and

$$\mathsf{ans}((\mathsf{rew}(\Sigma), Q), D) = \mathsf{ans}((\mathsf{pg}(\mathsf{rew}(\Sigma), D), Q), D).$$

Thus due to Theorem 1, to prove the claim it suffices to show that $\mathsf{rew}(a(\Sigma_{pg}))$ and $a(\mathsf{pg}(\mathsf{rew}(\Sigma), D))$ have the same ground consequences from $a_\Sigma(D)$. This true because the two theories are equal modulo renaming of variables and auxiliary relation symbols introduced during the rewritings. □

$$\frac{\alpha \to \beta \wedge A}{\alpha \to A}\ A \text{ has no existential variables}$$

$$\frac{\alpha \to \beta \quad \gamma_1 \wedge \gamma_2 \to \delta}{\alpha \wedge h(\gamma_1) \to \beta \wedge h(\delta)}\ \begin{array}{l}\gamma_1 \wedge \gamma_2 \to \delta \text{ is a Datalog rule,}\\ h \text{ is a homomorphism from}\\ \gamma_2 \text{ to } \beta \text{ with } \mathsf{vars}(h(\gamma_1)) \subseteq \\ \mathsf{vars}(\alpha).\end{array}$$

$$\frac{\alpha \to \beta}{g(\alpha) \to g(\beta)}\ g : \mathsf{vars}(\alpha) \to \mathsf{vars}(\alpha)$$

Figure 3: Inference Rules

6. FROM NEARLY GUARDED RULES TO DATALOG

In this section, we give a translation from nearly guarded rules to Datalog. In line with the previous sections, the translation preserves answers to ground atomic queries. We first translate fully guarded theories in Datalog. The translation is based on specially tailored inference rules, which we use to saturate the input set of guarded rules. The desired Datalog program is obtained by dropping all existential rules. We then show that this methods easily extends to translate nearly guarded rules into Datalog.

Definition 19. *For a guarded theory Σ, let $\Xi(\Sigma)$ be the closure of Σ under the inference rules in Figure 3. Let $\mathsf{dat}(\Sigma)$ be the set of Datalog rules obtained from $\Xi(\Sigma)$ by dropping all rules containing existential variables in the head.*

We note that applying any of the inference rules to guarded rules, only guarded rules will be produced. Consequently $\Xi(\Sigma)$ (and thus $\mathsf{dat}(\Sigma)$) is guarded whenever Σ is.

Theorem 3. *Let Σ be a guarded theory. Then $\Sigma, D \models \alpha$ iff $\mathsf{dat}(\Sigma), D \models \alpha$, for any ground atom α and database D.*

Example 7. *We provide an illustrative example. Let our rule set Σ contain the rules*

- $\sigma_1 = \mathsf{A}(x) \to \exists y.\mathsf{R}(x, y)$,
- $\sigma_2 = \mathsf{R}(x, y) \to \mathsf{S}(y, y)$,
- $\sigma_3 = \mathsf{S}(x, y) \to \exists z.\mathsf{T}(x, y, z)$,
- $\sigma_4 = \mathsf{T}(x, x, y) \to \mathsf{B}(x)$,
- $\sigma_5 = \mathsf{C}(x), \mathsf{R}(x, y), \mathsf{B}(y) \to \mathsf{D}(x)$.

Let our database D consist of $\mathsf{A}(c)$ and $\mathsf{C}(c)$, and let our query α be $\mathsf{D}(c)$. The following chase confirms the query:

$\{\mathsf{A}(c), \mathsf{C}(c)\}$
$\Longrightarrow \{\mathsf{R}(c, n_1)\}$ *apply σ_1 with $x \mapsto c$*
$\Longrightarrow \{\mathsf{S}(n_1, n_1)\}$ *apply σ_2 with $y \mapsto n_1$*
$\Longrightarrow \{\mathsf{T}(n_1, n_1, n_2)\}$ *apply σ_3 with $x \mapsto n_1, y \mapsto n_1$*
$\Longrightarrow \{\mathsf{B}(n_1)\}$ *apply σ_4 with $x \mapsto n_1, y \mapsto n_2$*
$\Longrightarrow \{\mathsf{D}(c)\}$ *apply σ_5 with $x \mapsto c, y \mapsto n_1$.*

We now show how this query can be answered using only the datalog program $\mathsf{dat}(\Sigma)$. To this end, we first provide derivations of new rules from Σ

- $\sigma_6 = \mathsf{S}(y,y) \rightarrow \exists z.\mathsf{T}(y,y,z)$
  (from σ_3 via third rule with $g = \{x \mapsto y, y \mapsto y, z \mapsto z\}$),

- $\sigma_7 = \mathsf{S}(y,y) \rightarrow \exists z.\mathsf{T}(y,y,z) \wedge B(y)$
  (from σ_6 and σ_4 via second rule, $\gamma_1 = \emptyset$, $h = \{x \mapsto y, y \mapsto z\}$),

- $\sigma_8 = \mathsf{S}(y,y) \rightarrow \mathsf{B}(y)$ *(from σ_7 via first rule),*

- $\sigma_9 = \mathsf{A}(x) \rightarrow \exists y.\mathsf{R}(x,y) \wedge \mathsf{S}(y,y)$
  (from σ_1 and σ_2 via second rule, $\gamma_1 = \emptyset$, $h = id$),

- $\sigma_{10} = \mathsf{A}(x) \rightarrow \exists y.\mathsf{R}(x,y) \wedge \mathsf{S}(y,y) \wedge \mathsf{B}(y)$
  (from σ_9 and σ_8 via second rule, $\gamma_1 = \emptyset$, $h = id$),

- $\sigma_{11} = \mathsf{A}(x) \wedge \mathsf{C}(x) \rightarrow \exists y.\mathsf{R}(x,y) \wedge \mathsf{S}(y,y) \wedge \mathsf{B}(y) \wedge \mathsf{D}(x)$
  (from σ_{10} and σ_5 via second rule, $\gamma_1 = \mathsf{C}(x)$, $h = id$),

- $\sigma_{12} = \mathsf{A}(x) \wedge \mathsf{C}(x) \rightarrow \mathsf{D}(x)$, *(from σ_{11} via first rule).*

Clearly, σ_{12} is contained in $\mathsf{dat}(\Sigma)$ as it is a plain Datalog rule. Now it is straightforward to see that $\mathsf{D}(c)$ can be derived from $\{\mathsf{A}(c), \mathsf{C}(c)\}$ by one application of σ_{12}.

We saw how a (fully) guarded theory can be converted into a Datalog program while preserving atomic consequences. We show here that the method can also be applied to nearly guarded theories.

PROPOSITION 6. *Let Σ be a nearly guarded theory, α a ground atom, and D a database. Let Σ_g denote the guarded rules in Σ and let $\Sigma_d = \Sigma \setminus \Sigma_g$. Then $\Sigma, D \models \alpha$ iff $\mathsf{dat}(\Sigma_g) \cup \Sigma_d, D \models \alpha$.*

In other words, a nearly guarded theory Σ is translated into the Datalog program $\mathsf{dat}(\Sigma_g) \cup \Sigma_d$ while preserving ground atomic consequences of Σ.

The above translation leads to a Datalog program that is of size double exponential in the size of the input guarded or nearly-guarded theory. To see this, assume a nearly guarded theory Σ. Let Σ_g denote the guarded rules in Σ and let $\Sigma_d = \Sigma \setminus \Sigma_g$. Let v be the maximal number of variables per rule in Σ, and let p be the maximal arity of relations in Σ, and let c be the number of constants in Σ. Observe that the rewriting of Σ_g into the datalog program $\mathsf{dat}(\Sigma_g)$ does not introduce new variables, new constants or new relation symbols into the constructed rules, but may introduce new atoms in rule bodies and heads (see the second rule in Figure 3). Assuming the number of relation symbols in Σ is m, the number of possible rules is bounded by $2^{(v+c)^p m}$. In case the arity of the relations in Σ is bounded by a constant, the number of rules resulting from the translation in 19 is single exponential in the size of Σ.

The above translation does not increase the number of variables per rule. If we place this as a requirement, which is reasonable for practical purposes, then the double exponential blow-up in the size of the input theory Σ is unavoidable. This follows from the 2-EXPTIME-hardness of query answering in guarded theories [14]. Indeed, a (singly) exponentially sized Datalog program could be evaluated in single exponential time because the grounding of the program would remain of single exponential size. The double exponential blowup, under the common assumptions in complexity theory, is unavoidable even if we relax the requirement a bit: instead of preserving the number of variables per rule, we require the maximal predicate arity to be preserved. In this case, querying the resulting program would be feasible in NEXPTIME to due the fact that query answering in Datalog under bounded predicate arities is in NP [17].

7. CONJUNCTIVE QUERY ANSWERING

We look next at conjunctive queries over databases enriched with existential rules (see e.g. [9, 14] for similar results and motivation), which are fully supported in our setting. A *knowledge base query* is of the form $(\Sigma \cup \{\alpha \rightarrow Q(\vec{x})\}, Q)$, where Σ is a weakly frontier-guarded theory and Q does not occur in Σ. Note that $\alpha \rightarrow Q(\vec{x})$ need not be weakly frontier-guarded in $\Sigma \cup \{\alpha \rightarrow Q(\vec{x})\}$. However, by employing the built-in relation ACDom, the rule $\alpha \rightarrow Q(\vec{x})$ can be converted into a weakly frontier-guarded rule. Indeed, $(\Sigma \cup \{\alpha \rightarrow Q(\vec{x})\}, Q)$ and the weakly frontier-guarded query $(\Sigma \cup \{\alpha \wedge \mathsf{ACDom}(x_1) \wedge \ldots \wedge \mathsf{ACDom}(x_n) \rightarrow Q(x_1, \ldots, x_n)\}, Q)$ output the same for any database D, where $\langle x_1, \ldots, x_n \rangle = \vec{x}$.

We recall that checking $\Sigma, D \models \alpha$, where Σ is weakly frontier-guarded, is 2-EXPTIME-complete in combined complexity [30]. It is not difficult to see that the translations of this paper provide an alternative proof for the double exponential time upper bound. The high-level procedure to check $\Sigma, D \models \alpha$ is as follows:

1. Compute the weakly guarded theory $\mathsf{rew}(\Sigma)$.
2. Compute the *partial grounding* Σ_1 of $\mathsf{rew}(\Sigma)$ w.r.t. D. That is, Σ_1 is the set of rules that can be obtained from $\mathsf{rew}(\Sigma)$ by instantiating variables in non-affected positions with constants from D. The resulting Σ_1 is guarded, is of exponential size in the size of Σ and D, but has linearly many variables per rule.
3. Translate Σ_1 into Datalog with the translation of the previous section, obtaining $\Sigma_2 = \mathsf{dat}(\Sigma_1)$. The resulting Σ_2 is of double exponential size in the size of Σ and D, but has linearly many variables per rule.
4. Ground Σ_2 with the constants of D, obtaining a ground theory Σ_3 with doubly exponentially many rules.
5. If $\Sigma_3, D \models \alpha$, then $\Sigma, D \models \alpha$. Otherwise, $\Sigma, D \not\models \alpha$.

8. CAPTURING QUERIES COMPUTABLE IN EXPONENTIAL TIME

We have seen that many of the query languages investigated here can be expressed by plain Datalog queries. However, queries based on weakly frontier-guarded or weakly guarded rules cannot be expressed in Datalog due to the increase in data complexity from completeness for PTIME to EXPTIME [14]. We next explore settings in which these formalisms capture *all* queries computable in exponential time.

We first show that all exponential time computable queries posed over *string databases* can be expressed by weakly guarded theories. We later show that weakly guarded theories extended with *stratified negation* are in fact expressive enough to capture *all queries* computable in exponential time, without any assumptions about input databases.

DEFINITION 20. *(String databases and queries) Assume an integer k and a set Ω of k-ary relation symbols. Assume also k-ary relations $\mathsf{First}^k, \mathsf{Last}^k$, and a $2k$-ary relation Next^{2k}. A database D is called a* string database *(of degree k) if it has the following properties:*

- *Let Dom be the set of constants in D. Then for all k-tuples $\vec{v} \in Dom^k$, there is exactly one $\sigma \in \Omega$ such that $\sigma(\vec{v}) \in D$.*

- *The relation $\{(\vec{v}_1, \vec{v}_2) \mid \mathsf{Next}^{2k}(\vec{v}_1, \vec{v}_2) \in D, \vec{v}_1, \vec{v}_2 \in Dom^k\}$ is a successor relation from some total $<$ order on Dom^k. Moreover, $\mathsf{First}^k(\vec{v}_1)$ and $\mathsf{Last}^k(\vec{v}_2)$ are true exactly for the minimal and the maximal element in $<$, respectively.*

We let $w(D) \in \Omega^*$ denote the word such that, for each $1 \leq i \leq |Dom^k|$, the ith symbol of $w(D)$ is exactly the symbol $\sigma \in \Omega$ such that $\sigma(\vec{v}) \in D$, where $\vec{v}$ is the ith smallest element in $<$. In other words, $w(D)$ extracts the string encoded in D.

A string query *is a triple (k, Ω, T), where k and Ω are as above, and T is a set of string databases of degree k such that T is closed under isomorphic databases. We say (k, Ω, T) is* decidable in exponential time *if there is deterministic Turing machine M with alphabet Ω that accepts exactly the language $\{w(D) \in \Omega^* \mid D \in T\}$, and M operates in time that is bounded by an exponential in the size D.*

THEOREM 4. *Any string query (k, Ω, T) decidable in exponential time can be expressed as a query (Σ, Q) where Σ is weakly guarded.*

PROOF SKETCH. One shows that from a Turing machine M to decide (k, Ω, T) one can build a query (Σ_M, Q), where Σ_M is weakly guarded and Q is 0-ary (i.e. a propositional variable), such that $D \in T$ iff $\Sigma_M, D \models Q$. In particular, one can write a theory Σ_M to implement an alternating polynomial space algorithm for checking whether M accepts $w(D)$. □

Weakly guarded rules do not capture all database queries decidable in exponential (or, even polynomial) time. E.g. due to monotonicity of query answers with plain existential rules, it is impossible to express a query that checks whether the number of constants occurring in a database is even. This already hints that we need to add some non-monotonicity to the formalism in order to also cover non-monotonic queries. But first, we define our goal more formally.

DEFINITION 21. *A* (generic) Boolean database query *is a tuple (A, BQ), where is A is a set of relation names (a signature), and BQ is a set of databases over A such that BQ is closed under isomorphic databases. We assume a coding C that encodes databases over A into words, such that a database D with d constants is encoded as a string of length d^k over a fixed alphabet Ω, where k is a sufficiently large yet fixed constant k.*[2] *We say BQ is* decidable in exponential time *if there is an exponential time bounded deterministic Turing machine M that decides, for any database D over A, whether $C(D) \in \{C(D') \mid D' \in BQ\}$ holds.*

Assume (A, BQ) as above. Observe that there is only one problem that needs to be resolved before we can employ the previous result for string databases to show that (A, BQ) can also be captured. We need a set of rules to implement an encoding C. In particular, it suffices to define a theory Σ_{code} to transform an input database D into a string database D_s of level k, for some fixed k, with $C(D)$ being the string written in D_s, i.e. with $w(D_s) = C(D)$.

Implementing Σ_{code} is straightforward, assuming each input database provides via relations Succ, Min and Max a total order over its constants, and we have means to infer the absence of tuples in the input database. We sketch such an encoding for the case A has only one n-ary relation R. We employ a standard semipositive Datalog program, which we in fact also introduce formally later. Let Ω consist of n-ary relations Zero and One. The first step is to define relations First^n, Next^{2n} and Last^n to store a lexicographically ordered sequence of n-tuples of constants from D, which can be done using plain Datalog rules [16]. We then can simply use the rules $R(\vec{x}) \rightarrow \mathsf{One}(\vec{x})$ and $\neg R(\vec{x}) \wedge \mathsf{ACDom}(x_1) \wedge \ldots \wedge \mathsf{ACDom}(x_n) \rightarrow \mathsf{Zero}(\vec{x})$ to represent the characteristic function of R in D, where $\langle x_1, \ldots, x_n \rangle = \vec{x}$. Thus we can infer that weakly guarded rules extended with negation on input relations capture all exponential time computable queries over ordered databases.

We next show that the desired total order (represented by Succ, Min and Max) can be generated by weakly guarded rules with *stratified negation*, which generalizes negation on input relations. Thus weakly guarded rules with stratified negation capture all exponential time computable queries, without any assumptions on the input databases.

DEFINITION 22. *(Stratified theories – syntax) An existential rule with negation is an expression of the form*

$$B_1 \wedge \ldots \wedge B_n \rightarrow H_1 \wedge \ldots \wedge H_m, \tag{2}$$

where $B_1, \ldots, B_n$ can be atoms $R(\vec{t})$ or negated atoms $\neg R(\vec{t})$. Negated atoms are called negative*; the remaining atoms are called* positive*. We consider* safe *theories where every variable of a negative body atom also occurs in some positive body atom. A theory Σ is called* stratified *if it can be partitioned into mutually disjoint $\Sigma_1, \ldots, \Sigma_n$ such that:*

- *if Σ_i has a body atom $A(\vec{t})$, then there is no rule in $\Sigma_{i+1} \cup \ldots \cup \Sigma_n$ with A in the head;*

- *if Σ_i has a body atom $\neg A(\vec{t})$, then there is no rule in $\Sigma_i \cup \ldots \cup \Sigma_n$ with A in the head.*

If $n = 1$, then Σ is semipositive*.*

The semantics of such theories is given by an iterative chasing following the stratification, which we recall here [15].

DEFINITION 23. *(Stratified theories – semantics) Let Σ be a stratified theory and D a database. For every $\neg A(\vec{t})$ in Σ we introduce a predicate name $\bar{A}$ of the same arity as A. We also define $p(\Sigma)$ to be the theory obtained by uniformly replacing each atom $\neg A(\vec{t})$ in Σ by the atom $\bar{A}(\vec{t})$.*

The semantics is defined inductively. Assume a database D and Σ with stratification $\Sigma_1, \ldots, \Sigma_n$. We define the following databases:

- *$S_0 = D$, and*

- *For each $1 \leq i \leq n$, the database S_i is obtained by restricting $chase(p(\Sigma_i), S'_{i-1})$ to the original symbols of Σ, where $S'_{i-1} = S_{i-1} \cup \{\bar{A}(\vec{t}) \mid A(\vec{t}) \notin S_{i-1}\}$.*

We let $chase(\Sigma, D) = S_n$.

Introducing negation requires to revisit the notion of weak guardedness. Assume a stratified theory Σ and let Σ' be the theory obtained from Σ by dropping all negative atoms. We say Σ is *weakly guarded* if every rule $\sigma \in \Sigma$ has a body atom that contains all the variables of $\mathsf{unsafe}(\sigma, \Sigma') \cap \mathsf{uvars}(\sigma)$.

[2]Note that this is possible due to the fixed signature. We assume that D has at least two constants.

It has been shown in [3] that query answering in stratified weakly guarded theories is ExpTime-complete in data complexity, i.e. stratified negation does not increase the complexity of reasoning with weakly guarded rules.

Theorem 5. *Any Boolean database query (A, BQ) decidable in exponential time can be expressed as a query (Σ, Q) where Σ is a stratified, weakly guarded theory.*

Proof Sketch. We only define using a stratified theory (a variant of) the relations Succ, Min and Max to store a total order over the constants of the input database. To this end, we write a program Σ_{succ} that creates an infinite forest such that each possible total ordering of constants of an input database is represented by a prefix of a branch. At the end, each different total ordering will be represented by a distinct null value, and all relevant facts regarding a specific ordering will be indexed by the null value corresponding to that ordering.

In particular, we use the following relations:

- $\mathsf{Min}(a, u)$ means that a constant a is the smallest element of the ordering identified by u.
- $\mathsf{Max}(a, u)$ means that a constant a is the greatest element of the ordering identified by u.
- $\mathsf{Lt}(a, b, u)$ means that a precedes b in the ordering identified by u.
- $\mathsf{Succ}(a, b, u)$ means that b is the immediate successor of a in the ordering identified by u.
- $\mathsf{Good}(u)$ means that u effectively represents a linear ordering over the set of constants in the input database.
- $\mathsf{Repetition}(u)$ means that u *does not* represent a linear ordering due to a cycle in the Succ relation.
- $\mathsf{Omission}(u)$ means that u *does not* represent a linear ordering due to a missing element.
- $\mathsf{New}(a, u)$ means that a is the new element introduced for a candidate ordering u.
- $\mathsf{Old}(a, u)$ means that a is an element already treated in the construction of the current ordering.

We will employ the active constant domain relation ACDom, which can be axiomatized as discussed previously. The theory Σ_{succ} has the following rules:

(1) $\mathsf{ACDom}(x) \rightarrow \exists u.\mathsf{Min}(x, u) \wedge \mathsf{New}(x, u)$.

(2) $\mathsf{New}(x, u), \mathsf{ACDom}(y) \rightarrow \exists v.\mathsf{Succ}(x, y, u, v) \wedge \mathsf{New}(y, v)$.

(3) $\mathsf{New}(x, u) \rightarrow \mathsf{Old}(x, u)$.

(4) $\mathsf{Succ}(x, y, u, v), \mathsf{Old}(x', u) \rightarrow \mathsf{Old}(x', v)$.

(5) $\mathsf{Succ}(x, y, u, v), \mathsf{Min}(x', u) \rightarrow \mathsf{Min}(x', v)$.

(6) $\mathsf{Succ}(x, y, u, v), \mathsf{Succ}(x', y', u) \rightarrow \mathsf{Succ}(x', y', v)$.

(7) $\mathsf{Succ}(x, y, u) \rightarrow \mathsf{Lt}(x, y, u)$

(8) $\mathsf{Lt}(x, y, u), \mathsf{Lt}(y, z, u) \rightarrow \mathsf{Lt}(x, z, u)$.

(9) $\mathsf{Lt}(x, x, u) \rightarrow \mathsf{Repetition}(u)$.

(10) $\mathsf{Old}(y, u), \mathsf{ACDom}(x), \neg\mathsf{Old}(x, u) \rightarrow \mathsf{Omission}(u)$.

(11) $\mathsf{Old}(x, u), \neg\mathsf{Repetition}(u), \neg\mathsf{Omission}(u) \rightarrow \mathsf{Good}(u)$.

(12) $\mathsf{New}(x, u), \mathsf{Good}(u) \rightarrow \mathsf{Max}(x, u)$.

For each u for which $\mathsf{Good}(u)$ holds, the relations $\mathsf{Min}(\cdot, u)$, $\mathsf{Max}(\cdot, u)$, and $\mathsf{Succ}(\cdot, \cdot, u)$ jointly constitute a (u-"labeled") linear order on the constants of an input database. Vice-versa, for each such linear order, some u exists such that $\mathsf{Good}(u)$ holds and the order is represented by the relations $\mathsf{Min}(\cdot, u)$, $\mathsf{Max}(\cdot, u)$, and $\mathsf{Succ}(\cdot, \cdot, u)$. □

We remark that the above capturing result cannot be obtained for semipositive weakly guarded theories. This is because queries (Σ, Q), where Σ is semipositive, are monotonic on full databases (where all relations have all possible tuples). For this reason, it is impossible to express e.g. the query to check whether the input database is full and its domain has an odd number of constants.

9. RELATED WORK

There exists prior work on translations from frontier-guarded rules to Datalog [7, 8]. Exploiting a boundedness property, these earlier approaches essentially enumerate all (i.e., best-case exponentially many) possible queries with a certain number of variables. Contrarily, our approach was designed to be much more selective. In fact, methods which are close in spirit (i.e., inspired by knowledge compilation and consequence-driven reasoning) have been successfully implemented and evaluated for related but easier logics (cf. [4, 20, 18, 22]) As another difference, we give an intermediate translation into nearly guarded rules, which takes only single exponential time and generalizes nicely to a translation from weakly frontier-guarded to weakly-guarded rules. These results, which are based on new ideas and a new class of rules, and are proven from first principles, are of their own interest; they provide interesting insights and do not follow from previous work.

Recently, Bienvenu et al. have investigated the expressiveness of queries formulated by coupling a standard query language (e.g. conjunctive queries) and various description logics, and also richer logics such as extensions of the guarded fragment [12]. The authors show that such queries can be expressed in various fragments of disjunctive Datalog and also provide descriptive complexity results by establishing a tight connection with constraint satisfaction problems. We note that the query rewriting technique of [12] does not easily apply in our non-disjunctive setting, as it intrinsically relies on disjunctive rules to, intuitively speaking, compute the possible consistent completions of the database with ground atoms. Several further data-independent reformulations of query answering in description logics in terms of Datalog have been developed previously, e.g. for instance queries [20, 29, 27] and conjunctive queries [18]. All query languages above are not harder than NP in data complexity, and thus are subsumed by stratified weakly guarded existential rules.

Translations between existential rule fragments have been used to provide expressiveness and complexity results: in particular, there is a polynomial translation from weakly (frontier-)guarded into (frontier-)guarded rules [7] to show that they have the same combined complexity, however, this translation is not data-independent. Translations into Datalog exist for other existential rule fragments, e.g., for those based on notions of acyclicity [23].

The calculus from Definition 19 is inspired by results in description logics, where similar procedures have been developed for $\mathcal{EL}$ [4] and Horn-$\mathcal{SHIQ}$ [22].

10. CONCLUSION

In this paper, we have explored the relative and absolute expressiveness of diverse database query languages based on existential rules with various forms of guardedness.

We have seen that the considered languages with polynomial time data complexity (all up to nearly frontier-guarded

theories) can in fact be expressed in plain Datalog. To this end, we have provided translations which are worst-case optimal in terms of the incurred blow-up of the query size.

For the discussed languages with exponential time data complexity (i.e. weakly guarded and weakly frontier-guarded rules), we showed that, extended with negation on input facts, they capture all exponential time computable queries over ordered databases. For arbitrary databases, an extension with stratified negation is required in order to fully capture ExpTime. This result has far-reaching implications—it shows that queries with stratified weakly guarded rules can express an overwhelming range of query formalisms: first-order and second-order logic (even extended via least fixed point operators), any of the known concrete decidable fragments of existential rules or formalisms based on descriptions logics (to the best of our knowledge, no such formalisms with data complexity beyond ExpTime have been proposed).

Exploring existential rules under more fine grained notions of expressiveness such as *modular expressive power* [3] is a natural next step for future work.

11. ACKNOWLEDGMENTS

We thank the anonymous reviewers for their helpful comments. This work was partially supported by the Austrian Science Fund's (FWF) projects P25518-N23 and P25207-N23. Georg Gottlob's work was supported by the European Research Council under grant FP7/2007-2013/ERC 246858 "DIADEM"; Georg Gottlob is a James Martin Senior Fellow.

12. REFERENCES

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[2] S. Abiteboul and V. Vianu. Datalog extensions for database queries and updates. *Journal of Computer and System Sciences*, 43(1):62 – 124, 1991.

[3] M. Arenas, G. Gottlob, and A. Pieris. Expressive languages for querying the semantic web. In *Proc. of PODS'14*. ACM Press, 2014.

[4] F. Baader, S. Brand, and C. Lutz. Pushing the el envelope. In *In Proc. of IJCAI 2005*, pages 364–369. Morgan-Kaufmann Publishers, 2005.

[5] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, second edition, 2007.

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

[7] J.-F. Baget, M.-L. Mugnier, S. Rudolph, and M. Thomazo. Walking the complexity lines for generalized guarded existential rules. In *Proc. of IJCAI' 11*, pages 712–717. IJCAI/AAAI, 2011.

[8] V. Bárány, M. Benedikt, and B. ten Cate. Rewriting guarded negation queries. In *Proc. of MFCS' 13*, pages 98–110. ACM, 2013.

[9] V. Barany, G. Gottlob, and M. Otto. Querying the guarded fragment. *Logic in Computer Science, Symposium on*, 0:1–10, 2010.

[10] V. Bárány, B. ten Cate, and M. Otto. Queries with guarded negation. *PVLDB*, 5(11):1328–1339, 2012.

[11] C. Beeri and M. Y. Vardi. A proof procedure for data dependencies. *J. ACM*, 31(4):718–741, Sept. 1984.

[12] M. Bienvenu, B. ten Cate, C. Lutz, and F. Wolter. Ontology-based data access: A study through disjunctive datalog, CSP, and MMSNP. In *Proc. of PODS' 13*. ACM Press, 2013.

[13] L. Cabibbo. The expressive power of stratified logic programs with value invention. *Information and Computation*, 147(1):22 – 56, 1998.

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

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

[16] E. Dantsin, T. Eiter, G. Gottlob, and A. Voronkov. Complexity and expressive power of logic programming. *ACM Computing Surveys*, 33(3):374–425, 2001.

[17] T. Eiter, W. Faber, M. Fink, and S. Woltran. Complexity results for answer set programming with bounded predicate arities and implications. *Ann. Math. Artif. Intell.*, 51(2-4):123–165, 2007.

[18] T. Eiter, M. Ortiz, M. Simkus, T.-K. Tran, and G. Xiao. Query rewriting for horn-shiq plus rules. In *Proc. of AAAI' 2012*. AAAI Press, 2012.

[19] R. Hull and M. Yoshikawa. Ilog: Declarative creation and manipulation of object identifiers. In *Proc. VLDB 1990*, pages 455–468. Morgan Kaufmann, 1990.

[20] U. Hustadt, B. Motik, and U. Sattler. Reasoning in description logics by a reduction to disjunctive datalog. *J. Autom. Reasoning*, 39(3):351–384, 2007.

[21] N. Immerman. *Descriptive complexity*. Graduate texts in computer science. Springer, 1999.

[22] Y. Kazakov. Consequence-driven reasoning for horn shiq ontologies. In *Proc. of IJCAI'09*, pages 2040–2045, 2009.

[23] M. Krötzsch and S. Rudolph. Extending decidable existential rules by joining acyclicity and guardedness. In *Proc. of IJCAI' 11*, pages 963–968, 2011.

[24] D. Maier, A. O. Mendelzon, and Y. Sagiv. Testing implications of data dependencies. *ACM Trans. Database Syst.*, 4(4):455–469, Dec. 1979.

[25] B. Motik, U. Sattler, and R. Studer. Query answering for OWL-DL with rules. *Web Semantics*, 3(1), 2005.

[26] M.-L. Mugnier. Conceptual graph rules and equivalent rules: A synthesis. In *Proc. of ICCS 2009*, 2009.

[27] M. Ortiz, S. Rudolph, and M. Simkus. Worst-case optimal reasoning for the horn-dl fragments of owl 1 and 2. In *Proc. of KR'10*. AAAI Press, 2010.

[28] C. H. Papadimitriou. A note the expressive power of prolog. *Bulletin of the EATCS*, 26:21–22, 1985.

[29] S. Rudolph, M. Krötzsch, and P. Hitzler. Type-elimination-based reasoning for the description logic SHIQbs using decision diagrams and disjunctive datalog. *Logical Methods in Comp. Science*, 8(1), 2012.

[30] M. Thomazo, J.-F. Baget, M.-L. Mugnier, and S. Rudolph. A generic querying algorithm for greedy sets of existential rules. In *Proc. of KR 2012*, 2012.

[31] M. Y. Vardi. The complexity of relational query languages. In *Proc. of STOC' 82*. ACM, 1982.