Capturing Homomorphism-Closed Decidable Queries with Existential Rules (Extended Abstract)[*]

Camille Bourgaux[1] , **David Carral**[2] , **Markus Krötzsch**[3] ,
Sebastian Rudolph[3] and **Michaël Thomazo**[1]
[1] DI ENS, ENS, CNRS, PSL University & Inria, Paris, France
[2] LIRMM, Inria, University of Montpellier, CNRS, Montpellier, France
[3] Technische Universität Dresden, Dresden, Germany
{camille.bourgaux, david.carral, michael.thomazo}@inria.fr,
{markus.kroetzsch, sebastian.rudolph}@tu-dresden.de

Abstract

Existential rules are a very popular ontology-mediated query language for which the chase represents a generic computational approach for query answering. It is straightforward that existential rule queries exhibiting chase termination are decidable and can only recognize properties that are preserved under homomorphisms. This paper is an extended abstract of our eponymous publication at KR 2021 where we show the converse: every decidable query that is closed under homomorphism can be expressed by an existential rule set for which the standard chase universally terminates. Membership in this fragment is not decidable, but we show via a diagonalisation argument that this is unavoidable.

1 Introduction

At the core of contemporary logic-based knowledge representation is the concept of *querying* data sources, often using elaborate query formalisms that allow for taking background knowledge into account. The classical decision problem related to such knowledge-aware querying is *Boolean query entailment*. From an abstract point of view, a Boolean query identifies a class of *databases* $\mathcal{D}$ – those that satisfy the query, i.e., to which the query "matches". This view allows us to define and investigate properties of (abstract) queries independently from the syntax used to specify them. Such properties can be structural (morphisms, closure properties) or computational (decidability, complexity).

A very popular querying formalism are *existential rules*, also referred to as *tuple-generating dependencies*. It is straightforward that the class of databases satisfying some existential rule query is closed under homomorphisms and recursively enumerable. Conversely, it was established that *every* homomorphism-closed query that is recursively enumerable can be expressed using existential rules [Rudolph and Thomazo, 2015]. That is, plain existential rules already realize their full potential; further syntactic extensions within these boundaries do not enhance expressivity.

For questions related to automated deduction, however, decidability rather than recursive enumerability is of central interest. The crucial question we tackle in this paper is thus:

Can we characterize an existential rules fragment capable of expressing every *decidable homomorphism-closed query?*

The generic computational paradigm for existential rules, the *chase* [Beeri and Vardi, 1984], is based on repetitive, forward-chaining rule application, starting from the database. As this may cause the iterated introduction of new domain elements, this procedure is not guaranteed to terminate – yet, termination is a crucial criterion for decidability. The chase comes in several variants, mainly differing in their (increasingly thorough) mechanisms to prevent unnecessary rule applications: While the *Skolem* chase [Marnette, 2009] essentially just avoids duplicate rule applications, the *standard* [Fagin *et al.*, 2005] and the *core* chase [Deutsch *et al.*, 2008] check for redundancy on a local and global level, respectively.

The class of existential rule sets with terminating[1] Skolem chase has already been weighed and found wanting: it only comprises those queries that are already expressible in plain Datalog – and hence can be evaluated in polynomial time [Marnette, 2009; Krötzsch and Rudolph, 2011; Zhang *et al.*, 2015]. For the standard-chase-terminating and the core-chase-terminating existential rules classes, on the other hand, we only know that the former is contained in the latter [Grahne and Onet, 2018], but little more than that [Krötzsch *et al.*, 2019]. In this paper, we clarify the situation significantly by showing the following:

Standard-chase-terminating existential rules capture the class of all decidable homomorphism-closed queries.

Notably, this implies that standard-chase-terminating and core-chase-terminating existential rule queries are equally expressive and no decidable enhancement of this formalism that preserves homomorphism-closedness (e.g. by allowing disjunction in rule heads) can be strictly more expressive.

As a downside, the existential rules fragment thus identified is not even semi-decidable, that is, one cannot semi-

[*]This is an extended abstract of the paper that won the 2021 Ray Reiter Best Paper Prize at the Int. Conf. on Principles of Knowledge Representation and Reasoning [Bourgaux et al., 2021]

[1]We always mean universal termination, i.e., for every database.

decide whether a set of rules is universally terminating for the standard chase. We show that this downside is unavoidable:

There are no semi-decidable query languages that (i) *express all decidable, homomorphism-closed queries and* (ii) *for which query answering is decidable.*

2 Preliminaries

We first briefly introduce the notions we need in this paper.

Rules We consider first-order formulas over countably infinite sets Vars of variables and Preds of *predicates*, where each $\mathtt{p} \in \mathsf{Preds}$ has an arity $Ar(\mathtt{p}) \geq 0$. An *atom* is an expression $\mathtt{p}(\vec{x})$ with $\mathtt{p} \in \mathsf{Preds}$ and $\vec{x}$ a list of variables of length $Ar(\mathtt{p})$. The fragment of *disjunctive existential rules* consists of formulas of the form:

$$\forall \vec{x}. \Big(\beta[\vec{x}] \to \bigvee\nolimits_{i=1}^{k} \exists \vec{y}_i . \eta_i[\vec{x}_i, \vec{y}_i] \Big), \tag{1}$$

where $\beta[\vec{x}]$ and $\eta_i[\vec{x}_i, \vec{y}_i]$ $(i = 1, \ldots, k)$ are conjunctions of atoms with variables $\vec{x}$ and $\vec{x}_i \cup \vec{y}_i$, respectively. We call β *body* and $\bigvee_{i=1}^{k} \exists \vec{y}_i . \eta_i$ *head.* Bodies can be empty, but heads must be non-empty. We require that $\vec{x}$ and $\vec{y}_i$ are mutually disjoint and that $\vec{x}_i \subseteq \vec{x}$ for all $i = 1, \ldots, k$. We single out the fragment of *existential rules* by disallowing disjunction, i.e. requiring $k = 1$, and *Datalog rules* by disallowing existential quantifiers. We omit the universal quantifiers from rules.

Databases, Interpretations, and Entailment The semantics of formulas is based on interpretations, defined as relational structures over a countably infinite set Nulls of *nulls*. A *schema* $\mathcal{S}$ is a finite set of predicates. An *interpretation* $\mathcal{I}$ over $\mathcal{S}$ is a set of expressions $\mathtt{p}(\vec{n})$ with $\mathtt{p} \in \mathcal{S}$ and $\vec{n}$ a list of nulls of length $Ar(\mathtt{p})$. A *database* is a finite interpretation.

A *homomorphism* $h : \mathcal{I}_1 \to \mathcal{I}_2$ between interpretations $\mathcal{I}_1$ and $\mathcal{I}_2$ is a mapping h from the nulls in $\mathcal{I}_1$ to the nulls in $\mathcal{I}_2$, such that $\mathtt{p}(\vec{n}) \in \mathcal{I}_1$ implies $\mathtt{p}(h(\vec{n})) \in \mathcal{I}_2$.

A *substitution* σ is a mapping from variables to nulls. A rule ρ as in (1) is *satisfied* by interpretation $\mathcal{I}$ if every substitution $\sigma : \vec{x} \to \mathsf{Nulls}$ with $\sigma(\beta) \subseteq \mathcal{I}$ can be extended to $\sigma' : \vec{x} \cup \vec{y}_i \to \mathsf{Nulls}$ for some $i \in \{1, \ldots, k\}$ such that $\sigma'(\eta_i) \subseteq \mathcal{I}$. Otherwise, if $\sigma(\beta) \subseteq \mathcal{I}$ but no extension σ' of σ verifies $\sigma'(\eta_i) \subseteq \mathcal{I}$ for some i, then $\langle \rho, \sigma \rangle$ is *applicable* to $\mathcal{I}$.

An interpretation $\mathcal{I}$ *satisfies* a set Σ of rules if it satisfies every rule in Σ. An interpretation $\mathcal{J}$ is *satisfied* by an interpretation $\mathcal{I}$ if there is a homomorphism $h : \mathcal{J} \to \mathcal{I}$. $\mathcal{I}$ is a *model* of a rule/rule set/interpretation/database $\mathcal{X}$ if $\mathcal{X}$ is satisfied by $\mathcal{I}$, written $\mathcal{I} \models \mathcal{X}$. As usual, we also write $\mathcal{X} \models \mathcal{Y}$ if every model of $\mathcal{X}$ is a model of $\mathcal{Y}$, where $\mathcal{X}$ and $\mathcal{Y}$ might be rules, rule sets, databases, or lists of several such elements. Note that the semantics of a database $\mathcal{D}$ in this context corresponds to the semantics of a *Boolean conjunctive query* $\exists \vec{x}. \bigwedge \{ \mathtt{p}(x_{n_1}, \ldots, x_{n_\ell}) \mid \mathtt{p}(n_1, \ldots, n_\ell) \in \mathcal{D} \}$ – we will therefore not introduce such queries as a separate notion. Also note that entailment and satisfaction between interpretations/databases coincide.

Abstract Queries, Expressivity, and Decidability An *(abstract) query* $\mathfrak{Q}$ over a schema $\mathcal{S}$ is a set of databases over $\mathcal{S}$ that is *closed under isomorphism*, i.e., such that whenever $\mathcal{D} \in \mathfrak{Q}$ and $\mathcal{D}'$ is obtained from $\mathcal{D}$ by bijective renaming of nulls, then $\mathcal{D}' \in \mathfrak{Q}$. The query $\mathfrak{Q}$ is further *closed under homomorphisms* if, for all $\mathcal{D} \in \mathfrak{Q}$ and all homomorphisms $h : \mathcal{D} \to \mathcal{D}'$, we have $\mathcal{D}' \in \mathfrak{Q}$.

Definition 1. Let $\mathtt{Goal}$ be a nullary predicate. A query $\mathfrak{Q}$ over $\mathcal{S}$ is *expressed by* a set Σ of rules if, for every database $\mathcal{D}$ over $\mathcal{S}$, we have $\mathcal{D} \in \mathfrak{Q}$ if and only if $\Sigma, \mathcal{D} \models \mathtt{Goal}$.

To discuss decidability of queries, we need to see databases as Turing machine (TM) inputs over a fixed alphabet. A *serialisation* for a schema $\mathcal{S}$ is a word $s \in (\{0, 1, \|\} \cup \mathcal{S})^*$ of the form $e_1 \cdots e_n$ where $n \geq 0$ and $e_i = \mathtt{p}_i \| w_{i1} \| \cdots \| w_{iAr(\mathtt{p}_i)} \|$ for $w_{ij} \in \{0, 1\}^+$ and $\mathtt{p}_i \in \mathcal{S}$. Given s of this form and an injection $\eta : \{0, 1\}^+ \to \mathsf{Nulls}$, let $\eta(s)$ denote the database $\{ \mathtt{p}_i(\eta(w_{i1}), \ldots, \eta(w_{iAr(\mathtt{p}_i)})) \mid 1 \leq i \leq n \}$. Then *$s$ corresponds to* a database $\mathcal{D}$ if $\eta(s)$ is isomorphic to $\mathcal{D}$; note that this does not depend on the choice of η.

A query $\mathfrak{Q}$ with schema $\mathcal{S}$ is *decidable* if the set of all serialisations for $\mathcal{S}$ that correspond to some $\mathcal{D} \in \mathfrak{Q}$ is decidable.

Universal Models and the Chase Entailment of databases (corresponding to Boolean conjunctive queries) can be decided by considering only a subset of all models. Given sets $\mathfrak{I}$ and $\mathfrak{K}$ of interpretations, $\mathfrak{I}$ is *universal* for $\mathfrak{K}$ if, for all $\mathcal{K} \in \mathfrak{K}$, there is $\mathcal{I} \in \mathfrak{I}$ and a homomorphism $\mathcal{I} \to \mathcal{K}$. Consider a rule set Σ and database $\mathcal{D}$, and let $\mathfrak{M}$ be the set of all models of $\Sigma, \mathcal{D}$. Then $\mathfrak{I}$ is a *universal model set* for Σ and $\mathcal{D}$ if $\mathfrak{I} \subseteq \mathfrak{M}$ and $\mathfrak{I}$ is universal for $\mathfrak{M}$.

Fact 1. *If $\mathfrak{I}$ is a universal model set for Σ and $\mathcal{D}$ then, for every database $\mathcal{C}$, we have $\Sigma, \mathcal{D} \models \mathcal{C}$ iff $\mathcal{I} \models \mathcal{C}$ for all $\mathcal{I} \in \mathfrak{I}$.*

Universal model sets can be computed with the *chase* algorithm. We consider a variation of the *standard* (or *restricted*) chase for rules with disjunctions [Carral *et al.*, 2017].

Definition 2. A *chase tree* for Σ and $\mathcal{D}$ is a (finite or infinite) tree where each node is labelled by a database, such that:

1. The root is labelled with $\mathcal{D}$.
2. For every node with label $\mathcal{E}$ that has ℓ children labelled $\mathcal{C}_1, \ldots, \mathcal{C}_\ell$, there is a rule $\rho \in \Sigma$ and a substitution $\sigma : \vec{x} \to \mathsf{Nulls}$ such that (i) $\langle \rho, \sigma \rangle$ is applicable to $\mathcal{E}$, (ii) ρ has ℓ head disjuncts, and (iii) $\mathcal{C}_i = \mathcal{E} \cup \sigma_i(\eta_i)$ where σ_i extends σ by mapping each $y \in \vec{y}_i$ to a fresh null.
3. For each $\rho \in \Sigma$ and σ, there is $i \geq 1$ such that $\langle \rho, \sigma \rangle$ is not applicable to the label of any node of depth $\geq i$.

The *result* that corresponds to a chase tree is the set of all interpretations that can be obtained as the union of all interpretations along a path in the tree.

Point (3) ensures fair rule application, but different orders of application can lead to different chase trees, and different results. Nevertheless, every result is semantically correct:

Fact 2. *Every result of a chase on a rule set Σ and database $\mathcal{D}$ is a universal model set for Σ and $\mathcal{D}$.*

The pair $\langle \Sigma, \mathcal{D} \rangle$ is *chase-terminating* if all its chase trees are finite; this corresponds to *all-strategy termination*. Σ is *chase-terminating* if $\langle \Sigma, \mathcal{D} \rangle$ is chase-terminating for every database $\mathcal{D}$; this corresponds to *universal termination*.

3 Main Results

We are now ready to state the main results of the paper.

Capturing Decidable Homomorphism-Closed Queries Given a homomorphism-closed query $\mathfrak{Q}$ over signature $\mathcal{S}$, and a Turing machine $M_{\mathfrak{Q}}$ that decides $\mathfrak{Q}$, we describe in the next section how to construct a set of standard-chase-terminating existential rules Σ that expresses $\mathfrak{Q}$.

Theorem 3. *Chase-terminating existential rules capture the class of all decidable homomorphism-closed queries.*

Limitations of Semi-Decidable Languages A *query language* $\mathfrak{F}$ over a schema $\mathcal{S}$ is a function from a set L to $2^{\mathfrak{D}_{\mathcal{S}}}$, where $\mathfrak{D}_{\mathcal{S}}$ is the set of all databases over schema $\mathcal{S}$. We say that $\mathfrak{F}$ is *semi-decidable* if membership to L is semi-decidable, and that its *query answering problem is decidable* if there exists a TM $M_{\mathfrak{F}}$ that takes as input some $(l, \mathcal{D}) \in (L \times \mathfrak{D}_{\mathcal{S}})$ and decides whether $\mathcal{D} \in \mathfrak{F}(l)$.

The set of chase-terminating existential rule sets is a query language that is not semi-decidable [Grahne and Onet, 2018] and for which the query answering problem is decidable (by running the chase). In fact, we show that one cannot find a semi-decidable query language with similar properties.

Theorem 4. *There are no semi-decidable query languages that (i) express all decidable, homomorphism-closed queries and (ii) for which query answering is decidable.*

To show this result, we define a set $\mathcal{M}$ of TMs (intuitively, $\mathcal{M}$ is the set of all deciders that solve homomorphism-closed queries over databases in $\mathfrak{D}_{\{\mathtt{ed}\}}$), show that $\mathcal{M}$ can be enumerated up to equivalence if there is a semi-decidable language that satisfies (i) and (ii) above, and finally prove that $\mathcal{M}$ is not enumerable up to equivalence.

4 Construction for Theorem 3

In this section we explain how we construct the standard-chase-terminating existential rules Σ that expresses the homomorphism-closed query $\mathfrak{Q}$ to prove Theorem 3.

4.1 Overview

The construction works in three steps:

1. We express $\mathfrak{Q}$ with a set of disjunctive existential rules simulating a TM $M_{\mathfrak{Q}}$ deciding $\mathfrak{Q}$. Disjunction is actually used in a very light way: It is only used to guess an ordering on the database elements, and to guess which atoms are not present in $\mathcal{D}$.
2. We ensure termination of the standard chase by modifying our rule set. Known syntactic criteria to ensure termination come with an upper bound on the length of the chase (possibly exponential) and are thus not applicable in our setting, as $M_{\mathfrak{Q}}$ may take arbitrarily long to halt. We thus adapt the "emergency brake" technique [Krötzsch *et al.*, 2019] by casting it as a general rule set transformation, and apply it by detecting the situations in which the previous rule set leads to non-termination. This essentially happens when the "linear order" relation that is guessed contains cycles.
3. Finally, we remove disjunction by noticing that the rule set has a specific shape, in which all disjunctive rules are Datalog rules that can be applied before any existential rules. For such a rule set, we adapt another technique by Krötzsch *et al.* [2019] to create an equivalent set of non-disjunctive existential rules: It builds all possible worlds corresponding to choices made by disjunctive Datalog rules, simulates the application of non-disjunctive existential rules in each world independently, and aggregates the results from all worlds.

In the remaining of this section, we focus on the first step.

$$\rightarrow \exists y.\mathtt{First}(y) \wedge \mathtt{DbDom}(y) \qquad (2)$$
$$\rightarrow \exists z.\mathtt{Last}(z) \wedge \mathtt{DbDom}(z) \qquad (3)$$
$$\mathtt{p}(\vec{x}) \rightarrow \mathtt{In_p}(\vec{x}) \wedge \textstyle\bigwedge_{x \in \vec{x}} \mathtt{DbDom}(x) \qquad (4)$$
$$\mathtt{DbDom}(x) \rightarrow \mathtt{Eq}(x, x) \qquad (5)$$
$$\mathtt{Eq}(x, y) \rightarrow \mathtt{Eq}(y, x) \qquad (6)$$
$$\mathtt{NEq}(x, y) \rightarrow \mathtt{NEq}(y, x) \qquad (7)$$
$$\mathtt{R}(\vec{x}) \wedge \mathtt{Eq}(x_i, y) \rightarrow \mathtt{R}(\vec{x}_{x_i \mapsto y}) \qquad (8)$$
$$\mathtt{DbDom}(x) \wedge \mathtt{DbDom}(y) \rightarrow \mathtt{Eq}(x, y) \vee \mathtt{NEq}(x, y) \qquad (9)$$
$$\mathtt{LT}(x, y) \wedge \mathtt{LT}(y, z) \rightarrow \mathtt{LT}(x, z) \qquad (10)$$
$$\mathtt{First}(x) \wedge \mathtt{NEq}(x, y) \rightarrow \mathtt{LT}(x, y) \qquad (11)$$
$$\mathtt{NEq}(x, y) \wedge \mathtt{Last}(y) \rightarrow \mathtt{LT}(x, y) \qquad (12)$$
$$\mathtt{NEq}(x, y) \rightarrow \mathtt{LT}(x, y) \vee \mathtt{LT}(y, x) \qquad (13)$$
$$\textstyle\bigwedge_{x \in \vec{x}} \mathtt{DbDom}(x) \rightarrow \mathtt{In_p}(\vec{x}) \vee \mathtt{NIn_p}(\vec{x}) \qquad (14)$$

Figure 1: The rule set $\mathcal{R}_1$, where rules (4) and (14) are instantiated for each $\mathtt{p} \in \mathcal{S}$, and rules (8) are instantiated for each $\mathtt{R} \in \{\mathtt{First}, \mathtt{Last}, \mathtt{Eq}, \mathtt{NEq}, \mathtt{LT}\} \cup \{\mathtt{In_p}, \mathtt{NIn_p} \mid \mathtt{p} \in \mathcal{S}\}$ and $1 \leq i \leq Ar(\mathtt{R})$, and $\vec{x}_{x_i \mapsto y}$ denotes $\vec{x}$ with x_i replaced by y.

$$\mathtt{First}(x) \rightarrow \exists u.\mathtt{Root}(u) \wedge \mathtt{Rep}(x, u) \qquad (15)$$
$$\mathtt{Rep}(x, v) \wedge \mathtt{LT}(x, z) \rightarrow \exists w.\mathtt{Chi}(v, w) \wedge \mathtt{Rep}(z, w) \qquad (16)$$
$$\mathtt{Last}(x) \wedge \mathtt{Rep}(x, u) \rightarrow \mathtt{Leaf}(u) \qquad (17)$$
$$\mathtt{Rep}(x, u) \wedge \mathtt{Eq}(x, y) \rightarrow \mathtt{Rep}(y, u) \qquad (18)$$
$$\mathtt{In_p}(\vec{x}) \wedge \textstyle\bigwedge_{i=1}^{|\vec{x}|} \mathtt{Rep}(x_i, u_i) \rightarrow \mathtt{In'_p}(\vec{u}) \qquad (19)$$
$$\mathtt{NIn_p}(\vec{x}) \wedge \textstyle\bigwedge_{i=1}^{|\vec{x}|} \mathtt{Rep}(x_i, u_i) \rightarrow \mathtt{NIn'_p}(\vec{u}) \qquad (20)$$

Figure 2: The rule set $\mathcal{R}_2$ contains $\mathcal{R}_1$ (see Figure 1) and all above rules, where (19) and (20) are instantiated for each $\mathtt{p} \in \mathcal{S}$.

4.2 Expressing $\mathfrak{Q}$ with Disjunctive Rules

We build five rule sets $\mathcal{R}_1 \subseteq \mathcal{R}_2 \subseteq \mathcal{R}_3 \subseteq \mathcal{R}_4 \subseteq \mathcal{R}_5$ such that for any database $\mathcal{D}$ over $\mathcal{S}$, there is a universal model set $\mathfrak{M}$ of $\mathcal{R}_5$ and $\mathcal{D}$ s.t. $\mathcal{D} \in \mathfrak{Q}$ iff $\mathtt{Goal} \in \mathcal{I}$ for every $\mathcal{I} \in \mathfrak{M}$.

Intuitively, $\mathcal{R}_1$ constructs all possible linear orders over the nulls in $\mathcal{D}$, as well as all possible completions of $\mathcal{D}$ with facts built using these nulls; $\mathcal{R}_2 \setminus \mathcal{R}_1$ extracts successor relations from the linear orders; $\mathcal{R}_3 \setminus \mathcal{R}_2$ associates to nulls representations of their positions in successor relations; $\mathcal{R}_4 \setminus \mathcal{R}_3$ encodes all initial TM configurations corresponding to some linear order and completion; and $\mathcal{R}_5 \setminus \mathcal{R}_4$ simulates the run of the TM on these configurations. The two last steps being more standard (e.g. [Abiteboul *et al.*, 1995] and [Baget *et al.*, 2011a]), we focus on $\mathcal{R}_1$ to $\mathcal{R}_3$ (see Figures 1-3).

$$\mathtt{Root}(u) \to \exists y_1, y_2.\mathtt{Enc}(u, y_1, y_2) \wedge \mathtt{S}_0(y_1) \wedge \mathtt{Nxt}(y_1, y_2) \wedge \mathtt{S}_1(y_2) \tag{21}$$
$$\mathtt{Enc}(u, y_1, y_\_) \wedge \mathtt{Chi}(u, v) \to \exists z_1, z_\_.\mathtt{Enc}(v, z_1, z_\_) \wedge \mathtt{Cpy}_{+1}(y_1, y_\_, z_1, z_\_) \tag{22}$$
$$\mathtt{Cpy}_{+1}(y_1, y_2, z_1, z_\_) \wedge \mathtt{S}_0(y_1) \wedge \mathtt{Nxt}(y_1, y_2) \to \mathtt{S}_1(z_1) \wedge \mathtt{Nxt}(z_1, z_\_) \wedge \mathtt{S}_1(z_\_) \tag{23}$$
$$\mathtt{Cpy}_{+1}(y_1, y_2, z_1, z_\_) \wedge \mathtt{S}_1(y_1) \wedge \mathtt{Nxt}(y_1, y_2) \to \exists z_2.\mathtt{S}_0(z_1) \wedge \mathtt{Nxt}(z_1, z_2) \wedge \mathtt{S}_0(z_2) \wedge \mathtt{Nxt}(z_2, z_\_) \wedge \mathtt{S}_1(z_\_) \tag{24}$$
$$\mathtt{Cpy}_{+1}(y_1, y_\_, z_1, z_\_) \wedge \mathtt{S}_0(y_1) \wedge \mathtt{Nxt}(y_1, y_2) \wedge \mathtt{Nxt}(y_2, y_3) \to \exists z_2.\mathtt{Cpy}(y_2, y_\_, z_2, z_\_) \wedge \mathtt{S}_1(z_1) \wedge \mathtt{Nxt}(z_1, z_2) \tag{25}$$
$$\mathtt{Cpy}_{+1}(y_1, y_\_, z_1, z_\_) \wedge \mathtt{S}_1(y_1) \wedge \mathtt{Nxt}(y_1, y_2) \wedge \mathtt{Nxt}(y_2, y_3) \to \exists z_2.\mathtt{Cpy}_{+1}(y_2, y_\_, z_2, z_\_) \wedge \mathtt{S}_0(z_1) \wedge \mathtt{Nxt}(z_1, z_2) \tag{26}$$
$$\mathtt{Cpy}(y_1, y_2, z_1, z_2) \wedge \mathtt{S}_*(y_1) \wedge \mathtt{Nxt}(y_1, y_2) \to \mathtt{S}_*(z_1) \wedge \mathtt{Nxt}(z_1, z_2) \wedge \mathtt{S}_1(z_2) \tag{27}$$
$$\mathtt{Cpy}(y_1, y_\_, z_1, z_\_) \wedge \mathtt{S}_*(y_1) \wedge \mathtt{Nxt}(y_1, y_2) \wedge \mathtt{Nxt}(y_2, y_3) \to \exists z_2.\mathtt{Cpy}(y_2, y_\_, z_2, z_\_) \wedge \mathtt{S}_*(z_1) \wedge \mathtt{Nxt}(z_1, z_2) \tag{28}$$

Figure 3: The rule set $\mathcal{R}_3$ contains $\mathcal{R}_2$ (Figure 2) and all of the above rules, where (27) and (28) are instantiated for each $* \in \{0, 1\}$.

Guessing an Order and Completing the Database $\mathcal{R}_1$ serves two distinct purposes: (1) predicates First, Last, Eq ("="), NEq ("≠"), and LT ("<") encode representations of possible linear orders over nulls in $\mathcal{D}$ (collected in predicate DbDom); and (2) predicates $\mathtt{In}_\mathtt{p}$ and $\mathtt{NIn}_\mathtt{p}$ for each $\mathtt{p} \in \mathcal{S}$ explicitly encode positive and negative (absent) facts in $\mathcal{D}$. Both purposes require disjunctive reasoning. Possible models include representations of strict, total linear orders (1) and the exact database completion (2), but also models for collapsed orders and inconsistent completions. The latter is not problematic since we consider homomorphism-closed queries.

Building a Tree Structure According to the Order The purpose of $\mathcal{R}_2$ is to extract successor relations from the transitive linear order LT. It builds a tree structure – defined using predicates Root, Chi ("child"), and Leaf – where each path is a sequence of nulls representing a sequence of elements that respects the linear order LT. Unavoidably, some elements may be skipped in some sequences, but one path is complete, and it is the *successor relation* with respect to LT. Note that it may lead to non-termination of the restricted chase when the guessed LT contains a cycle. This problem is dealt with in the second part of the construction. Moreover, nulls are related via predicates $\mathtt{In}'_\mathtt{p}$ and $\mathtt{NIn}'_\mathtt{p}$ that reflect the relations for $\mathtt{In}_\mathtt{p}$ and $\mathtt{NIn}_\mathtt{p}$ that hold between the represented elements (in the database completion of the considered model $\mathcal{I}$ of $\mathcal{R}_1$).

Creating a Binary Encoding of the Distance from the Root The purpose of $\mathcal{R}_3$ is to associate each node in the tree of $\mathcal{R}_2$ with a binary encoding of its distance from the root (the root starts with "distance" 2 for technical reasons). Encodings start at the least significant bit and always end in 1 (i.e., have no leading 0s). To simplify upcoming steps, encodings take the form of little TM tapes, represented by a Nxt-connected chain of nulls with unary predicates $\mathtt{S}_0$ and $\mathtt{S}_1$ encoding the symbol at each position. Nodes u relate to the first and last null t_s and t_e of their "tape" through facts $\mathtt{Enc}(u, t_s, t_e)$. Facts $\mathtt{Cpy}(a_s, a_e, b_s, b_e)$ are used to create a tape between b_s and b_e that contains a copy of the information on the tape between a_s and a_e. Predicate $\mathtt{Cpy}_{+1}$ is analogous, but creates a representation of the successor of the number that is copied.

5 Discussion and Conclusion

We characterized all decidable homomorphism-closed Boolean queries: These are exactly the chase-terminating existential rule queries, that is, queries that can be expressed by a set of (non-disjunctive) existential rules for which the standard chase universally terminates irrespective of the order of rule applications (as long as it is fair).

By its nature, our result immediately shows that various extensions of our framework do not increase its expressivity:

Theorem 5. *Chase-terminating existential rule queries have the same expressivity as*

1. *existential rule queries with guaranteed existence of some finite chase tree (for every database)*
2. *existential rule queries for which the chase terminates according to some fair strategy (such as datalog-first),*
3. *core-chase-terminating [Deutsch* et al.*, 2008] existential rule queries,*
4. *disjunctive chase-terminating existential rule queries.*

Moreover, our result also applies to query languages and querying formalisms of different types. An interesting example is the existential rules fragment of *bounded treewidth sets (bts) of rules* [Baget *et al.*, 2011a] that is *not* chase-terminating and encompasses many well-known existential rule fragments with decidable query entailment, including guarded [Calì *et al.*, 2008], frontier-guarded [Baget *et al.*, 2011a], and glut-guarded existential rules [Krötzsch and Rudolph, 2011], as well as greedy bts [Baget *et al.*, 2011b]:

Theorem 6. *Let Σ be a bounded-treewidth set of rules and Q a conjunctive query. There is a chase-terminating set Σ_Q of existential rules such that $\mathcal{D}, \Sigma \models Q$ iff $\mathcal{D}, \Sigma_Q \models$* Goal.

While possibly surprising, this is a consequence of decidability of conjunctive query entailment from bts and of homomorphism-closedness of existential rule queries. Note, however, that every Q would give rise to a different Σ_Q. Asking for a "uniform" chase-terminating existential rules set Σ' satisfying $\mathcal{D}, \Sigma' \models Q$ iff $\mathcal{D}, \Sigma \models Q$ would change the game [Zhang *et al.*, 2015]. Such a set will not exist in all cases.

As we learned recently, Cabibbo [1998] shows a result akin to Theorem 3 for monotonic queries and Horn logic with inequality. This is important prior work that nonetheless also has important differences, explaining why our proof required several new techniques. A detailed discussion is planned for future work. Another direction of future research is to look at *complexity* instead of *expressivity*. Indeed, even though several chase variants can express the same query, we believe that not all of them lead to worst-case optimal computations.

Acknowledgements

The authors thank Jan van den Bussche for pointing them to the related work of Cabibbo [1998] during a discussion at ICDT conference 2022.

This work is partly supported by DFG in project number 389792660 (TRR 248, Center for Perspicuous Systems), by BMBF in the Center for Scalable Data Analytics and Artificial Intelligence (ScaDS.AI), by the Center for Advancing Electronics Dresden (cfaed), by the ERC Consolidator Grant DeciGUT (project number 771779), and by the ANR project CQFD (ANR-18-CE23-0003).

References

[Abiteboul *et al.*, 1995] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

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

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

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

[Bourgaux *et al.*, 2021] Camille Bourgaux, David Carral, Markus Krötzsch, Sebastian Rudolph, and Michaël Thomazo. Capturing homomorphism-closed decidable queries with existential rules. In Meghyn Bienvenu, Gerhard Lakemeyer, and Esra Erdem, editors, *Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning (KR 2021)*, pages 141–150, 2021.

[Cabibbo, 1998] Luca Cabibbo. The expressive power of stratified logic programs with value invention. *Inf. Comput.*, 147(1):22–56, 1998.

[Calì *et al.*, 2008] Andrea Calì, Georg Gottlob, and Michael Kifer. Taming the infinite chase: Query answering under expressive relational constraints. In Gerhard Brewka and Jérôme Lang, editors, *Proc. 11th Int. Conf. on Knowledge Representation and Reasoning (KR'08)*, pages 70–80. AAAI Press, 2008.

[Carral *et al.*, 2017] David Carral, Irina Dragoste, and Markus Krötzsch. Restricted chase (non)termination for existential rules with disjunctions. In Carles Sierra, editor, *Proc. 26th Int. Joint Conf. on Artificial Intelligence, IJCAI'17*, pages 922–928. ijcai.org, 2017.

[Deutsch *et al.*, 2008] Alin Deutsch, Alan Nash, and Jeffrey B. Remmel. The chase revisited. In Maurizio Lenzerini and Domenico Lembo, editors, *Proc. 27th Symposium on Principles of Database Systems (PODS'08)*, pages 149–158. ACM, 2008.

[Fagin *et al.*, 2005] Ronald Fagin, Phokion G. Kolaitis, Renée J. Miller, and Lucian Popa. Data exchange: semantics and query answering. *Theoretical Computer Science*, 336(1):89–124, 2005.

[Grahne and Onet, 2018] Gösta Grahne and Adrian Onet. Anatomy of the chase. *Fundam. Informaticae*, 157(3):221–270, 2018.

[Krötzsch and Rudolph, 2011] Markus Krötzsch and Sebastian Rudolph. Extending decidable existential rules by joining acyclicity and guardedness. In Walsh [2011], pages 963–968.

[Krötzsch *et al.*, 2019] Markus Krötzsch, Maximilian Marx, and Sebastian Rudolph. The power of the terminating chase. In Pablo Barceló and Marco Calautti, editors, *Proc. 22nd Int. Conf. on Database Theory, ICDT'19*, volume 127 of *LIPIcs*, pages 3:1–3:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[Marnette, 2009] Bruno Marnette. Generalized schema-mappings: from termination to tractability. In Jan Paredaens and Jianwen Su, editors, *Proc. 28th Symposium on Principles of Database Systems (PODS'09)*, pages 13–22. ACM, 2009.

[Rudolph and Thomazo, 2015] Sebastian Rudolph and Michaël Thomazo. Characterization of the expressivity of existential rule queries. In Qiang Yang and Michael Wooldridge, editors, *Proc. 24th Int. Joint Conf. on Artificial Intelligence (IJCAI'15)*, pages 3193–3199. AAAI Press, 2015.

[Walsh, 2011] Toby Walsh, editor. *Proc. 22nd Int. Joint Conf. on Artificial Intelligence (IJCAI'11)*. AAAI Press/IJCAI, 2011.

[Zhang *et al.*, 2015] Heng Zhang, Yan Zhang, and Jia-Huai You. Existential rule languages with finite chase: Complexity and expressiveness. In Blai Bonet and Sven Koenig, editors, *Proc. 29th AAAI Conf. on Artificial Intelligence (AAAI'15)*. AAAI Press, 2015.