Capturing Homomorphism-Closed Decidable Queries with Existential Rules (Extended Abstract)
Camille Bourgaux, David Carral, Markus Krötzsch, Sebastian Rudolph, Michaël Thomazo

To cite this version:

HAL Id: hal-03801049
https://hal.science/hal-03801049
Submitted on 6 Oct 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Capturing Homomorphism-Closed Decidable Queries with Existential Rules (Extended Abstract)∗

Camille Bourgaux1, David Carral2, Markus Krötzsch3, Sebastian Rudolph3 and Michaël Thomazo1

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 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 terminating1 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]

1We always mean universal termination, i.e., for every database.
2 Preliminaries

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

Rules We consider first-order formulas over countably infinite sets. There are no semi-decidable query languages that (i) express all decidable, homomorphism-closed queries and (ii) for which query answering is decidable.

Databases, Interpretations, and Entailment The semantics of formulas is based on interpretations, defined as relations over a countably infinite set. An interpretation $I$ is a mapping from variables to nulls of length $Ar$. A database $D$ is a finite set of variables. An interpretation $I$ over $S$ is a set of expressions $p_i$ with $p \in S$ and a list of nulls of length $Ar(p)$. A database is a finite interpretation.

An homomorphism $h : I_1 \rightarrow I_2$ between interpretations $I_1$ and $I_2$ is a mapping $h$ from the nulls in $I_1$ to the nulls in $I_2$, such that $p(i) \in I_1$ implies $p(h(i)) \in I_2$.

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

An interpretation $I$ satisfies a set $\Sigma$ of rules if it satisfies every rule in $\Sigma$. An interpretation $J$ is satisfied by an interpretation $I$ if there is a homomorphism $h : J \rightarrow I$. $I$ is a model of a rule set in an interpretation/database $X$ if $X$ is a model for $I$.

Abstract Queries, Expressivity, and Decidability An abstract query $Q$ over a schema $S$ is a set of databases over $S$ that is closed under isomorphism, i.e., such that whenever $D \in Q$ and $D' \in Q$, $D \times D'$ is obtained from $D$ by bijective renaming of

nulls, then $D' \in Q$. The query $Q$ is further closed under homomorphisms if, for all $D \in Q$ and all homomorphisms $h : D \rightarrow D'$, we have $D' \in Q$.

Definition 1. Let $\sigma \equiv \emptyset$ be a nullary predicate. A query $Q$ over $S$ is expressed by a set $\Sigma$ of rules if, for every database $D$ over $S$, we have $D \in Q$ if and only if $\Sigma, D \models \sigma \equiv \emptyset$.

To discuss decidability of queries, we need to see databases as Turing machine (TM) inputs over a fixed alphabet. A seriesalisation for a schema $S$ is a word $s \in \{0, 1\}^* \cup \{\\text{S}\\}$ of the form $e_1 \cdots e_n$ where $n \geq 0$ and $e_i = p_1 || w_{i1} || \cdots || w_{iAr(p_i)}$ for $w_{ij} \in \{0, 1\}^+$ and $p_i \in S$. Given $s$ of this form and an injection $\eta : \{0, 1\}^+ \rightarrow \text{Nulls}$, let $\eta(s)$ denote the database $\{p_i(\eta(w_{i1}), \ldots, \eta(w_{iAr(p_i)})) | 1 \leq i \leq n\}$. Then $s$ corresponds to a database $D$ if $\eta(s)$ is isomorphic to $D$; note that this does not depend on the choice of $\eta$.

A query $Q$ with schema $S$ is decidable if the set of all serialisations for $S$ that correspond to some $D \in Q$ 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 $S$ and $R$ of interpretations, $I$ is universal for $R$ if, for all $K \in R$, there is an interpretation $I \models C$ and a homomorphism $I \rightarrow K$. Consider a rule set $\Sigma$ and database $D$, and let $\mathcal{N}$ be the set of all models of $\Sigma, D$. Then $I$ is a universal model set for $\Sigma$ and $D$ if $I \subseteq \mathcal{N}$ and $I$ is universal for $\mathcal{N}$.

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

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 $\Sigma$ and $D$ is a (finite or infinite) tree where each node is labelled by a database, such that:

1. The root is labelled with $D$.
2. For every node with label $E$ that has $\ell$ children labelled $C_1, \ldots, C_\ell$, there is a rule $\rho \in \Sigma$ and a substitution $\sigma : \bar{x} \rightarrow \text{Nulls}$ such that (i) $(\rho, \sigma)$ is applicable to $E$, (ii) $\rho$ has $\ell$ head disjunctions, and (iii) $C_i = E \cup \sigma(\bar{\eta}_i)$ where $\sigma_i$ extends $\sigma$ by mapping each $\bar{\eta}_i$ to a fresh null.
3. For each $\rho \in \Sigma$ and $\sigma$, there is $i \geq 1$ such that $(\rho, \sigma)$ 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 $\Sigma$ and database $D$ is a universal model set for $\Sigma$ and $D$.

The pair $(\Sigma, D)$ is chase-terminating if all its chase trees are finite; this corresponds to all-strategy termination. $\Sigma$ is chase-terminating if $(\Sigma, D)$ is chase-terminating for every database $D$; 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 $\Omega$ over signature $S$, and a Turing machine $M_\Omega$ that decides $\Omega$, we describe in the next section how to construct a set of standard-chase-terminating existential rules $\Sigma$ that expresses $\Omega$.

**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 schema $S$ is a function from a set $L$ to $2^{D_S}$, where $D_S$ is the set of all databases over schema $S$. 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 $M_1$ that takes as input some $(l, D) \in (L \times D_S)$ and decides whether $D \in \mathfrak{F}(l)$.

The set of chase-terminating 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 $D_{\{\text{eq}\}}$) 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 $\Sigma$ that expresses the homomorphism-closed query $\Omega$ to prove Theorem 3.

#### 4.1 Overview

The construction works in three steps:

1. We express $\Omega$ with a set of disjunctive existential rules simulating a TM $M_\Omega$ deciding $\Omega$. 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 $D$.

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 $M_\Omega$ 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

\[ \exists y. \text{First}(y) \land \text{DbDom}(y) \]  
\[ \exists z. \text{Last}(z) \land \text{DbDom}(z) \]  
\[ p(x) \rightarrow \text{In}_p(x) \land \bigwedge_{x \in \bar{x}} \text{DbDom}(x) \]  
\[ \text{DbDom}(x) \rightarrow \text{Eq}(x, x) \]  
\[ \text{Eq}(x, y) \rightarrow \text{Eq}(y, x) \]  
\[ \text{NEq}(x, y) \rightarrow \text{NEq}(y, x) \]  
\[ R(x) \land \text{Eq}(x, y) \rightarrow R(x_{i \rightarrow y}) \]  
\[ \text{DbDom}(x) \land \text{DbDom}(y) \rightarrow \text{Eq}(x, y) \lor \text{NEq}(x, y) \]  
\[ \text{LT}(x, y) \land \text{LT}(y, z) \rightarrow \text{LT}(x, z) \]  
\[ \text{First}(x) \land \text{NEq}(x, y) \rightarrow \text{LT}(x, y) \]  
\[ \text{NEq}(x, y) \land \text{Last}(y) \rightarrow \text{LT}(x, y) \]  
\[ \text{NEq}(x, y) \rightarrow \text{LT}(x, y) \lor \text{LT}(y, x) \]  
\[ \bigwedge_{x \in \bar{x}} \text{DbDom}(x) \rightarrow \text{In}_p(x) \lor \text{NI}_p(x) \]

where $\text{In}_p(x)$ and $\text{NI}_p(x)$ are witnesses of the cases $p(x)$ and $\neg p(x)$, respectively.

#### 4.2 Expressing $\Omega$ 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 $D$ over $S$, there is a universal model set $\mathcal{M}$ of $\mathcal{R}_5$ and $D$ s.t. $D \in \mathcal{M}$ iff $\text{Goal}(I)$ for every $I \in \mathcal{M}$.

Intuitively, $\mathcal{R}_1$ constructs all possible linear orders over the nulls in $D$, as well as all possible completions of $D$ 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).
possible linear orders over nulls in \( \text{LT} \) encode representations of possible orderings over nulls in \( D \) (collected in predicate \( \text{DbDom} \)); and (2) predicates \( \text{In}_{p} \) and \( \text{NIn}_{p} \) for each \( p \in S \) explicitly encode positive and negative (absent) facts in \( D \). 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 \( \text{Root} \), \( \text{Chi} \) (“child”), and \( \text{Leaf} \) – 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 \( \text{In}_{p}^{+} \) and \( \text{NIn}_{p}^{-} \) that reflect the relations for \( \text{In}_{p} \) and \( \text{NIn}_{p} \) that hold between the represented elements (in the database completion of the considered model \( I \) 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 \( \text{Nxt} \)-connected chain of nulls with unary predicates \( S_{0} \) and \( S_{1} \) 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 \( \text{Enc}(u, t_{s}, t_{e}) \). Facts \( \text{Cpy}(a_{c}, b_{s}, b_{e}) \) are used to create a tape between \( b_{s} \) and \( b_{e} \) that contains a copy of the information on the tape between \( a_{c} \) and \( a_{e} \). Predicate \( \text{Cpy} \) 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 \([\text{Deutsch et al., 2008}]\) 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 \([\text{Baget et al., 2011a}]\) that is not chase-terminating and encompasses many well-known existential rule fragments with decidable query entailment, including guarded \([\text{Cali et al., 2008}]\), frontier-guarded \([\text{Baget et al., 2011a}]\), and glut-guarded existential rules \([\text{Krötzsch and Rudolph, 2011}]\), as well as greedy bts \([\text{Baget et al., 2011b}]\):

**Theorem 6.** Let \( \Sigma \) be a bounded-treewidth set of rules and \( Q \) a conjunctive query. There is a chase-terminating set \( \Sigma_{Q} \) of existential rules such that \( D, \Sigma \models Q \) iff \( D, \Sigma_{Q} \models \text{Goal} \).

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 \( \Sigma_{Q} \). Asking for a “uniform” chase-terminating existential rules set \( \Sigma' \) satisfying \( D, \Sigma' \models Q \) iff \( D, \Sigma \models Q \) would change the game \([\text{Zhang et al., 2015}]\). Such a set will not exist in all cases.

As we learned recently, Cabibbo \([\text{1998}]\) 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


