Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning
Main Track

Capturing Homomorphism-Closed Decidable Queries with Existential Rules

Camille Bourgaux[1] , **David Carral**[2] , **Markus Krötzsch**[3] , **Sebastian Rudolph**[3] , **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. In this paper, 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, a more restricted requirement than recursive enumerability is of central interest: decidability. Therefore, the crucial question we tackle in this paper is:

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, Nash, and Remmel 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, Zhang, and You 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, Marx, and Rudolph 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, but we show via a diagonalisation argument that this downside is, in fact, unavoidable.

Additional proofs and details are given in the appendix of (Bourgaux et al. 2021).

2 Preliminaries

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 $Ar(\mathtt{p}) \geq 0$. Lists of variables are denoted $\vec{x} = x_1, \ldots, x_k$ and will be treated like sets when order is not relevant. An *atom* is an expression $\mathtt{p}(\vec{x})$ with $\mathtt{p} \in \mathsf{Preds}$ and $|\vec{x}| = Ar(\mathtt{p})$.

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

The fragment of *disjunctive existential rules* consists of formulae of the form:

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

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 (we then omit $\to$), but heads must be non-empty. We require that $\vec{x}$ and $\vec{y}_i$ $(i = 1, \ldots, k)$ 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 often omit the universal quantifiers from rules and treat conjunctions of atoms as sets of atoms.

Databases, Interpretations, and Entailment The semantics of formulas is based on logical interpretations, which we define 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 schema $\mathcal{S}$ is a set of expressions $\mathtt{p}(\vec{n})$ with $\mathtt{p} \in \mathcal{S}$ and $\vec{n}$ a list of nulls of length $Ar(\mathtt{p})$. We write $\mathsf{Nulls}(\mathcal{I})$ for the set of nulls in $\mathcal{I}$. A *database* is a finite interpretation. See also the remarks on this notation below.

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 $\mathtt{p}(h(\vec{n})) \in \mathcal{I}_2$, where $h(\vec{n})$ denotes the list of h-values over $\vec{n}$. We also write $h(\mathcal{I})$ for $\{\mathtt{p}(h(\vec{n})) \mid \mathtt{p}(\vec{n}) \in \mathcal{I}\}$.

A *substitution* σ is a mapping from variables to nulls, which we extend to lists of variables and formulas as usual. 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 a substitution $\sigma' : \vec{x} \cup \vec{y}_i \to \mathsf{Nulls}$ 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 $i \in \{1, \ldots, k\}$, then $\langle \rho, \sigma \rangle$ is *applicable* to $\mathcal{I}$.

$\mathcal{I}$ *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 $\mathcal{X} \models \mathcal{Y}$ ("$\mathcal{X}$ entails $\mathcal{Y}$") 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 $\mathcal{D}' \in \mathfrak{Q}$. $\mathfrak{Q}$ is further *closed under homomorphisms* if, for all $\mathcal{D} \in \mathfrak{Q}$ and 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 conceive databases as Turing machine 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 a decidable language.

Remarks on our Notation Many works consider constants to appear in databases (not just nulls), but complexity and expressivity is usually studied for queries that are closed under isomorphisms, a.k.a. *generic* (Abiteboul, Hull, and Vianu 1995, Ch. 16), and nulls are more natural there. One can admit finitely many exceptions (elements that must not be renamed), but such "constants" can be simulated by marking them with dedicated unary predicates.

Specifying logical interpretations as sets of "atoms" that may use nulls is a notational convenience with some side effects: our interpretations cannot contain elements that do not stand in any relation, but they can have an empty domain. Both aspects do not change the notion of logical entailment *on the formulas we consider*.

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. Here, we consider a variation of the *standard* (or *restricted*) chase for rules with disjunction, introduced by (Carral, Dragoste, and Krötzsch 2017).

Definition 2. A *chase tree* for a rule set Σ and database $\mathcal{D}$ 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 $\rho \in \Sigma$ of the form (1) and a substitution $\sigma : \vec{x} \to \mathsf{Nulls}$ such that (i) $\langle \rho, \sigma \rangle$ is applicable to $\mathcal{E}$, (ii) ρ has $k = \ell$ head disjuncts, and (iii) $\mathcal{C}_i = \mathcal{E} \cup \sigma_i(\eta_i)$ where σ_i extends σ by mapping each variable $y \in \vec{y}_i$ to a fresh null.

3. For each rule $\rho \in \Sigma$ and each substitution 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.

Condition (3) ensures fair, exhaustive rule application, but different orders of application can lead to different chase trees, which can also have different results. Nevertheless, every result is semantically correct in the following sense:

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 – by König's Lemma, this is equivalent to all chase results for $\langle \Sigma, \mathcal{D} \rangle$ containing only finite interpretations; 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*.

Turing Machines We will use (deterministic) Turing machines (TM), denoted as a tuple $M = \langle Q, \Gamma, \delta \rangle$, with states Q, tape alphabet Γ with blank $\textvisiblespace \in \Gamma$, and transition function δ. M has a distinguished initial state $q_S \in Q$, and accepting and rejecting halting states $q_A, q_R \in Q$. For all states $q \in Q \setminus \{q_A, q_R\}$ and tape symbols $a \in \Gamma$, there is exactly one transition $(q, a) \mapsto (r, b, D) \in \delta$. We assume that TM tapes are unbounded to the right but bounded to the left, and that TMs will never attempt to move left on the first position of the tape (this is w.l.o.g., since one can modify any TM to insert a marker at the tape start to recognise this case).

3 On the Expressivity of Disjunctive Rules

In this section, we show how to express a homomorphism-closed, decidable query with a disjunctive rule set. This construction will be the basis for finding a chase-terminating set of (deterministic) existential rules. Throughout this section, $\mathfrak{Q}$ is a fixed but arbitrary homomorphism-closed query over signature $\mathcal{S}$, and $M = \langle Q, \Gamma, \delta \rangle$ is a TM that decides $\mathfrak{Q}$.

To express $\mathfrak{Q}$, we specify five rule sets $\mathcal{R}_1 \subseteq \mathcal{R}_2 \subseteq \mathcal{R}_3 \subseteq \mathcal{R}_4 \subseteq \mathcal{R}_5$ (see Figures 1–6), which will be explained later. We want to show the following result:

Theorem 3. *The set $\mathcal{R}_5$ of disjunctive existential rules expresses the query $\mathfrak{Q}$.*

To show this, we fix an arbitrary database $\mathcal{D}$ over $\mathcal{S}$. Theorem 3 then follows from Fact 1 and the next lemma:

Lemma 4. *There is a universal model set $\mathfrak{M}$ of $\mathcal{R}_5$ and $\mathcal{D}$ such that $\mathcal{D} \in \mathfrak{Q}$ iff $\mathtt{Goal} \in \mathcal{I}$ for every $\mathcal{I} \in \mathfrak{M}$.*

The universal model set $\mathfrak{M}$ is a complicated structure that we describe step by step, by specifying five sets of interpretations – $\mathfrak{I}_1$, $\mathfrak{I}_2$, $\mathfrak{I}_3$, $\mathfrak{I}_4$, and $\mathfrak{I}_5$ –, such that (1) $\mathfrak{I}_i$ is a universal model set of $\mathcal{R}_i, \mathcal{D}$ for each $1 \leq i \leq 5$, (2) $|\mathfrak{I}_1| = |\mathfrak{I}_2| = |\mathfrak{I}_3| = |\mathfrak{I}_4| = |\mathfrak{I}_5|$, and (3) for each $1 \leq i < j \leq 5$ and each $\mathcal{I} \in \mathfrak{I}_i$, there is exactly one $\mathcal{J} \in \mathfrak{I}_j$ with $\mathcal{I} \subseteq \mathcal{J}$. Lemma 4 can then be shown using $\mathfrak{M} = \mathfrak{I}_5$.

Before dwelling into the details of each rule set and universal model, we give an overview of the construction: $\mathcal{R}_1$ 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.

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

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.

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

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 $\mathtt{p} \in \mathcal{S}$.

$\mathcal{R}_1$: Linear Order and Database Completion $\mathcal{R}_1$ serves two distinct purposes: (1) predicates $\mathtt{First}$, $\mathtt{Last}$, $\mathtt{Eq}$ ("="), $\mathtt{NEq}$ ("$\neq$"), and $\mathtt{LT}$ ("$<$") encode representations of possible linear orders over nulls in $\mathcal{D}$ (collected in predicate $\mathtt{DbDom}$); and (2) predicates $\mathtt{In_p}$ and $\mathtt{NIn_p}$ 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.

We define the interpretation set $\mathfrak{I}_1$ on the set $\Delta = \mathsf{Nulls}(\mathcal{D}) \cup \{u_\alpha, u_\omega\}$, for fresh nulls $u_\alpha, u_\omega \notin \mathsf{Nulls}(\mathcal{D})$. A (partially collapsed) linear order can be represented by

$$\texttt{Root}(u) \rightarrow \exists y_1, y_2.\texttt{Enc}(u, y_1, y_2) \wedge \texttt{S}_0(y_1) \wedge \texttt{Nxt}(y_1, y_2) \wedge \texttt{S}_1(y_2) \quad (21)$$

$$\texttt{Enc}(u, y_1, y_\_) \wedge \texttt{Chi}(u, v) \rightarrow \exists z_1, z_\_.\texttt{Enc}(v, z_1, z_\_) \wedge \texttt{Cpy}_{+1}(y_1, y_\_, z_1, z_\_) \quad (22)$$

$$\texttt{Cpy}_{+1}(y_1, y_2, z_1, z_\_) \wedge \texttt{S}_0(y_1) \wedge \texttt{Nxt}(y_1, y_2) \rightarrow \texttt{S}_1(z_1) \wedge \texttt{Nxt}(z_1, z_\_) \wedge \texttt{S}_1(z_\_) \quad (23)$$

$$\texttt{Cpy}_{+1}(y_1, y_2, z_1, z_\_) \wedge \texttt{S}_1(y_1) \wedge \texttt{Nxt}(y_1, y_2) \rightarrow \exists z_2.\texttt{S}_0(z_1) \wedge \texttt{Nxt}(z_1, z_2) \wedge \texttt{S}_0(z_2) \wedge \texttt{Nxt}(z_2, z_\_) \wedge \texttt{S}_1(z_\_) \quad (24)$$

$$\texttt{Cpy}_{+1}(y_1, y_\_, z_1, z_\_) \wedge \texttt{S}_0(y_1) \wedge \texttt{Nxt}(y_1, y_2) \wedge \texttt{Nxt}(y_2, y_3) \rightarrow \exists z_2.\texttt{Cpy}(y_2, y_\_, z_2, z_\_) \wedge \texttt{S}_1(z_1) \wedge \texttt{Nxt}(z_1, z_2) \quad (25)$$

$$\texttt{Cpy}_{+1}(y_1, y_\_, z_1, z_\_) \wedge \texttt{S}_1(y_1) \wedge \texttt{Nxt}(y_1, y_2) \wedge \texttt{Nxt}(y_2, y_3) \rightarrow \exists z_2.\texttt{Cpy}_{+1}(y_2, y_\_, z_2, z_\_) \wedge \texttt{S}_0(z_1) \wedge \texttt{Nxt}(z_1, z_2) \quad (26)$$

$$\texttt{Cpy}(y_1, y_2, z_1, z_2) \wedge \texttt{S}_*(y_1) \wedge \texttt{Nxt}(y_1, y_2) \rightarrow \texttt{S}_*(z_1) \wedge \texttt{Nxt}(z_1, z_2) \wedge \texttt{S}_1(z_2) \quad (27)$$

$$\texttt{Cpy}(y_1, y_\_, z_1, z_\_) \wedge \texttt{S}_*(y_1) \wedge \texttt{Nxt}(y_1, y_2) \wedge \texttt{Nxt}(y_2, y_3) \rightarrow \exists z_2.\texttt{Cpy}(y_2, y_\_, z_2, z_\_) \wedge \texttt{S}_*(z_1) \wedge \texttt{Nxt}(z_1, z_2) \quad (28)$$

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 $* \in \{0, 1\}$.

an ordered partition $\vec{T} = T_1, \ldots, T_k$ ($k \geq 1$) of Δ where $u_\alpha \in T_1$ and $u_\omega \in T_k$. Let Ords be the set of all such $\vec{T}$, and let $\mathit{Compls}(\vec{T})$ be the set of all interpretations with nulls Δ that are set-minimal among the models of $\mathcal{R}_1$ and $\mathcal{D}$, and that contain the database

$$\{\texttt{First}(u) \mid u \in T_1\} \cup \{\texttt{Last}(u) \mid u \in T_k\} \cup$$
$$\{\texttt{Eq}(t, u) \mid 1 \leq i \leq k; t, u \in T_i\} \cup$$
$$\{\texttt{LT}(t,u), \texttt{NEq}(t,u), \texttt{NEq}(u,t) \mid 1 \leq i < j \leq k; t \in T_i; u \in T_j\}.$$

By minimality, each $\mathcal{I} \in \mathit{Compls}(\vec{T})$ contains exactly one of $\{\texttt{In}_\texttt{p}(\vec{t}), \texttt{NIn}_\texttt{p}(\vec{t})\}$ for every $\vec{t} \subseteq \Delta$ with $|\vec{t}| = \mathit{Ar}(\texttt{p})$.

Lemma 5. *$\mathfrak{I}_1 = \bigcup_{\vec{T} \in \mathit{Ords}} \mathit{Compls}(\vec{T})$ is a universal model set of $\mathcal{R}_1$ and $\mathcal{D}$.*

Proof Sketch. We can construct a chase tree for $\mathcal{R}_1$ and $\mathcal{D}$ that prioritises the application of rules in the order of their appearance in Figure 1. The result of that chase $\mathfrak{K}$ is a finite universal model set (Fact 2) and we can show that every $\mathcal{K} \in \mathfrak{K}$ is isomorphic to a unique $\mathcal{I} \in \mathfrak{I}_1$. □

Each set $\mathfrak{I}_i$ for some $2 \leq i \leq 5$ is obtained as $\mathfrak{I}_i = \{\mathit{Int}_i(\mathcal{I}) \mid \mathcal{I} \in \mathfrak{I}_{i-1}\}$ for a function Int_i as defined below. Every $\mathcal{I} \in \mathfrak{I}_i$ with $1 \leq i \leq 5$ contains a unique interpretation $\mathit{Seed}(\mathcal{I}) \in \mathfrak{I}_1$, and there is a unique ordered partition $\mathit{Order}(\mathcal{I}) \in \mathit{Ords}$ such that $\mathit{Seed}(\mathcal{I}) \in \mathit{Compls}(\mathit{Order}(\mathcal{I}))$.

$\mathcal{R}_2$: Representative Tree The purpose of $\mathcal{R}_2$ is to extract successor relations from the transitive linear order LT. Given an ordered partition $\mathit{Order}(\mathcal{I}) = T_1, \ldots, T_k$ of some model $\mathcal{I}$ of $\mathcal{R}_1$, $\mathcal{R}_2$ constructs a finite tree structure – defined using predicates Root, Chi ("child"), and Leaf – where each path represents a sub-sequence $T_{z_1}, \ldots, T_{z_p}$ of $T_1, \ldots, T_k$ with $z_1 = 1$ and $z_p = k$. Unavoidably, some paths will skip some T_i, but it suffices for our purposes if one path is complete. The elements of any set T_j are related to the tree nodes that represent T_j by a predicate Rep. Moreover, nodes are related via predicates $\texttt{In}'_\texttt{p}$ and $\texttt{NIn}'_\texttt{p}$ that reflect the relations for $\texttt{In}_\texttt{p}$ and $\texttt{NIn}_\texttt{p}$ that hold between the represented elements (in the database completion of the considered model $\mathcal{I}$ of $\mathcal{R}_1$).

Given $\mathcal{I}$ with $|\mathit{Order}(\mathcal{I})| = k$ as above, let $\mathbf{Z}$ be the set of all words $z_1 \cdots z_p \in \{1, \ldots, k\}^*$ such that $z_1 = 1$ and $z_i < z_{i+1}$ for all $i \in \{1, \ldots, p-1\}$. Moreover, let $\mathit{end}(z_1 \cdots z_p) := z_p$. Using fresh nulls $\{u_w \mid w \in \mathbf{Z}\}$, we define $\mathit{Tree}(\mathcal{I})$ to be the interpretation

$$\{\texttt{Root}(u_1)\} \cup \{\texttt{Chi}(u_w, u_{wz}) \mid wz \in \mathbf{Z}, 2 \leq z \leq k\} \cup$$
$$\{\texttt{Leaf}(u_w) \mid w \in \mathbf{Z}; \mathit{end}(w) = k\} \cup$$
$$\{\texttt{Rep}(x, u_w) \mid w \in \mathbf{Z}; x \in T_{\mathit{end}(w)}\}.$$

Now $\mathit{Int}_2(\mathcal{I})$ is the least interpretation that contains $\mathcal{I}$ and $\mathit{Tree}(\mathcal{I})$, and that further satisfies rules (19) and (20). We can extend Lemma 5 as follows. The required universal model set is obtained by any chase that prioritises rule (18).

Lemma 6. *$\mathfrak{I}_2 = \{\mathit{Int}_2(\mathcal{I}) \mid \mathcal{I} \in \mathfrak{I}_1\}$ is a universal model set of $\mathcal{R}_2$ and $\mathcal{D}$.*

$\mathcal{R}_3$: Position Binary Encodings 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 $\texttt{S}_0$ and $\texttt{S}_1$ 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 $\texttt{Enc}(u, t_s, t_e)$. Facts $\texttt{Cpy}(a_s, a_e, b_s, b_e)$ 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 $\texttt{Cpy}_{+1}$ is analogous, but creates a representation of the successor of the number that is copied.

Consider a model $\mathcal{I}$ of $\mathcal{R}_2$ and define the set of sequences $\mathbf{Z}$ as before. For $w \in \mathbf{Z}$ of length $|w|$, and $b_1 \cdots b_\ell$ the binary representation of $|w| + 1$, let $\mathit{EncPos}(w)$ be the database

$$\{\texttt{Enc}(u_w, e^1_w, e^\ell_w)\} \cup \{\texttt{S}_{b_i}(e^i_w) \mid 1 \leq i \leq \ell\} \cup$$
$$\{\texttt{Nxt}(e^{i-1}_w, e^i_w) \mid 2 \leq i \leq \ell\}.$$

Let $\mathcal{J} = \mathcal{I} \cup \bigcup_{w \in \mathbf{Z}} \mathit{EncPos}(w)$. We define $\mathit{Int}_3(\mathcal{I})$ as the smallest superset of $\mathcal{J}$ that satisfies all rules in $\mathcal{R}_3$ while including only the nulls in $\mathcal{J}$. $\mathit{Int}_3(\mathcal{I})$ extends $\mathcal{J}$ only by missing Cpy and $\texttt{Cpy}_{+1}$ relations, which can be inferred by slightly rewritten rules. For example, rule (22) is satisfied when applying the following rule to $\mathcal{J}$:

$$\texttt{Enc}(u, y_1, y_\_) \wedge \texttt{Chi}(u, v) \wedge \texttt{Enc}(v, z_1, z_\_) \rightarrow \texttt{Cpy}_{+1}(y_1, y_\_, z_1, z_\_).$$

$$\texttt{Ld}_{\texttt{p}}(u,t,\vec{v}) \wedge \texttt{In}'_{\texttt{p}}(\vec{v}) \rightarrow \exists \vec{x}, y.\texttt{S}_{\texttt{p}}(t) \wedge \texttt{Nxt}(t,x_1) \wedge \textstyle\bigwedge_{i=1}^{|\vec{v}|} \texttt{LdE}(v_i,x_i,x_{i+1}) \wedge \texttt{Nxt}(x_{|\vec{v}|+1},y) \wedge \texttt{Rdy}_{\texttt{p}}(u,y,\vec{v}) \quad (29)$$

$$\texttt{LdE}(v,x_s,x_e) \wedge \texttt{Enc}(v,y_1,y_\_) \rightarrow \exists z_1,z_\_.\texttt{S}_{\|}(x_s) \wedge \texttt{Nxt}(x_s,z_1) \wedge \texttt{Cpy}(y_1,y_\_,z_1,z_\_) \wedge \texttt{Nxt}(z_\_,x_e) \wedge \texttt{S}_{\|}(x_e) \quad (30)$$

Figure 4: The rule set $\mathcal{R}_4$ contains $\mathcal{R}_3$ (see Figure 3), the rules from Figure 5, and the above rules instantiated for all $\texttt{p} \in \mathcal{S}$.

$$\texttt{Leaf}(u) \rightarrow \exists t.\texttt{Ld}_1(u,t,u) \wedge \texttt{Hd}_{q_S}(t) \quad (31)$$

$$\texttt{Ld}_\ell(u,t,\vec{v}) \rightarrow \texttt{Ld}_{\texttt{p}_1^\ell}(u,t,\vec{v}) \quad (32)$$

$$\texttt{Rdy}_{\texttt{p}_j^\ell}(u,t,\vec{v}) \rightarrow \texttt{Ld}_{\texttt{p}_{j+1}^\ell}(u,t,\vec{v}) \quad (33)$$

$$\texttt{Rdy}_{\texttt{p}_{\bar{n}}^\ell}(u,t,\vec{v}) \rightarrow \texttt{Rdy}_\ell(u,t,\vec{v}) \quad (34)$$

$$\texttt{Rdy}_\ell(u,t,\vec{v}) \wedge \textstyle\bigwedge_{i=k+1}^{\ell} \texttt{Root}(v_i) \wedge \texttt{Chi}(w,v_k) \rightarrow \texttt{Ld}_\ell(u,t,v_1,\cdots,v_{k-1},w,u,\cdots,u) \quad (35)$$

$$\texttt{Rdy}_\ell(u,t,\vec{v}) \wedge \textstyle\bigwedge_{i=1}^{\ell} \texttt{Root}(v_i) \rightarrow \texttt{Ld}_{\ell+1}(u,t,u,\cdots,u) \quad (36)$$

$$\texttt{Rdy}_{\bar{m}}(u,t,\vec{v}) \wedge \textstyle\bigwedge_{i=1}^{\bar{m}} \texttt{Root}(v_i) \rightarrow \texttt{S}_\_(t) \wedge \texttt{End}(t) \quad (37)$$

$$\texttt{Ld}_{\texttt{p}}(u,t,\vec{v}) \wedge \texttt{NIn}'_{\texttt{p}}(\vec{v}) \rightarrow \texttt{Rdy}_{\texttt{p}}(u,t,\vec{v}) \quad (38)$$

Figure 5: Some rules of $\mathcal{R}_4$, to be instantiated for all $1 \leq j \leq \bar{n}-1$, $1 \leq k \leq \ell \leq \bar{m}$, and $\texttt{p} \in \mathcal{S}$.

All other rules can be rewritten analogously, since every existentially quantified variable is used in unique ways with predicates other than Cpy and $\texttt{Cpy}_{+1}$.

For every $\mathcal{I} \in \mathfrak{I}_2$, we show that $Int_3(\mathcal{I})$ is isomorphic to a result of the chase on $\mathcal{R}_3$ and $\mathcal{I}$. The next result then follows from Lemma 6.

Lemma 7. *$\mathfrak{I}_3 = \{Int_3(\mathcal{I}) \mid \mathcal{I} \in \mathfrak{I}_2\}$ is a universal model set of $\mathcal{R}_3$ and $\mathcal{D}$.*

$\mathcal{R}_4$: Initial TM Configuration For each leaf in the tree of completions, $\mathcal{R}_4$ creates the representation of an initial TM configuration. The tape is again represented by a Nxt-chain, using further unary predicates $\texttt{S}_{\|}$, $\texttt{S}_\_$, and $\texttt{S}_{\texttt{p}}$ (for all $\texttt{p} \in \mathcal{S}$) for additional tape symbols. $\texttt{Hd}_{q_S}$ marks the TM's starting position and initial state q_S, and End the end of the tape.

Let $\bar{m}$ be the maximal arity of predicates in $\mathcal{S}$. We require that there is some $\bar{n} > 0$ such that $\mathcal{S}$ contains exactly $\bar{n}$ predicates $\texttt{p}_1^i, \ldots, \texttt{p}_{\bar{n}}^i$ of arity i, for every $1 \leq i \leq \bar{m}$. This is without loss of generality, except for the exclusion of nullary predicates. Our results do not depend on this restriction, but it helps to simplify the presentation of our main ideas.

To serialise the data as a tape, we iterate over all predicate arities $\ell = 1, \ldots, \bar{m}$ and over all lists $\vec{v}$ of tree nodes with length ℓ. In this process, $\texttt{Ld}_\ell(u,t,\vec{v})$ expresses that, while encoding the leaf u, after constructing the tape until position t, we continue serialising ℓ-ary predicate data for arguments $\vec{v}$. Analogously, $\texttt{Rdy}_\ell(u,t,\vec{v})$ means that this was completed at tape position t. Similar predicates $\texttt{Ld}_{\texttt{p}}$ and $\texttt{Rdy}_{\texttt{p}}$ are used to consider a specific predicate $\texttt{p} \in \mathcal{S}$ during this process. The rules in Figure 5 start the serialisation (31), proceed over all predicates (32)–(34), iterate over parameter vectors (35) and arities (36), and finally end the tape (37).

Absent facts do not need to be serialised (38), while present facts can be treated by copying the encodings for each of their parameters (29) and (30). In the latter, LdE states that a specific argument is serialised between two given tape positions.

The resulting TM tapes serialise facts $\texttt{In}'_{\texttt{p}}(\vec{u})$ as introduced by $\mathcal{R}_2$, i.e., where $\vec{u}$ are nodes in the representative tree. Given a model $\mathcal{I} \in \mathfrak{I}_3$ with some $\texttt{Leaf}(u_w) \in \mathcal{I}$, let $branch(u_w)$ be the set of all nodes $u_{w'}$ on the branch of u_w, i.e. all nulls $u_{w'}$ where w' is a prefix of w. Elements of $branch(u_w)$ are totally ordered by setting $u_{w_1} \prec u_{w_2}$ if $|w_1| > |w_2|$. Predicates are totally ordered by setting $\texttt{p}_i^a \prec \texttt{p}_j^b$ if either $a < b$, or both $a = b$ and $i < j$. We can then order facts as $\texttt{p}(\vec{u}) \prec \texttt{q}(\vec{v})$ if $\vec{u}, \vec{v} \subseteq branch(u_w)$ and $\langle \texttt{p}, \vec{u} \rangle$ is lexicographically before $\langle \texttt{q}, \vec{v} \rangle$.

Now let $branchDb(\mathcal{I}, u_w) = \{\texttt{p}(\vec{u}) \mid \texttt{In}'_{\texttt{p}}(\vec{u}) \in \mathcal{I}, \vec{u} \subseteq branch(u_w)\}$ denote the set of all facts on the branch with leaf u_w, and let $branchTape(\mathcal{I}, u_w)$ denote the TM tape serialisation (as defined in Section 2) of $branchDb(\mathcal{I}, u_w)$ according to the total order $\prec$ and representing each node u_w by the binary representation of $|w|+1$ as before. Given $S = branchTape(\mathcal{I}, u_w)$, let $startConf(\mathcal{I}, u_w)$ be the following interpretation:

$$\{\texttt{Ld}_1(u_w, t_w^1, u_w), \texttt{Hd}_{q_S}(t_w^1), \texttt{End}(t_w^{|S|+1})\} \cup$$
$$\{\texttt{Nxt}(t_w^{j-1}, t_w^j) \mid 2 \leq j \leq |S|+1\} \cup$$
$$\{\texttt{S}_a(t_w^j) \mid 1 \leq j \leq |S|, a = S[j]\} \cup \{\texttt{S}_\_(t_w^{|S|+1})\}.$$

Let $\mathcal{J}$ be the extension of $\mathcal{I}$ with $startConf(\mathcal{I}, u_w)$ for every $\texttt{Leaf}(u_w) \in \mathcal{I}$. We define $Int_4(\mathcal{I})$ to be the smallest superset of $\mathcal{J}$ that satisfies all rules in $\mathcal{R}_4$ while including only the nulls in $\mathcal{J}$. As in the case of Int_3, the missing relations can easily be inferred using the original rules or, for (29) and (30), with simple rewritings thereof.

Lemma 8. *$\mathfrak{I}_4 = \{Int_4(\mathcal{I}) \mid \mathcal{I} \in \mathfrak{I}_3\}$ is a universal model set of $\mathcal{R}_4$ and $\mathcal{D}$.*

$\mathcal{R}_5$: TM Run The purpose of $\mathcal{R}_5$ is to simulate the run of the deterministic TM $\langle Q, \Gamma, \delta \rangle$ on each of the initial tapes created by $\mathcal{R}_4$. We continue to use predicate Nxt for neighbouring tape cells (augmented with its transitive closure $\texttt{Nxt}^+$), $\texttt{S}_b$ to encode tape symbols $b \in \Gamma$, and $\texttt{Hd}_q$ to encode head position and current state $q \in Q$. Predicate Stp connects tape cells in each configuration to the corresponding tape cells in the next configuration (provided the TM performs another step). The rules in Figure 6 are a standard TM encoding, with the slight exception of rule (43), which adds a new blank tape cell in each step (even if not used by the TM). Our rules use the assumptions on TMs in Section 2.

Consider some $\mathcal{I} \in \mathfrak{I}_4$. It is easy and only mildly laborious to define interpretations $Run(u_w)$ that represent all successor configurations of the starting configuration $startConf(\mathcal{I}, u_w)$, appropriately connected with Stp and

$$\mathtt{Hd}_{q_A}(x) \to \mathtt{Goal} \quad (39)$$
$$\mathtt{Nxt}(x,y) \to \mathtt{Nxt}^+(x,y) \quad (40)$$
$$\mathtt{Nxt}^+(x,y) \wedge \mathtt{Nxt}^+(y,z) \to \mathtt{Nxt}^+(x,z) \quad (41)$$
$$\mathtt{Nxt}(x,y) \wedge \mathtt{Stp}(x,z) \wedge \mathtt{Stp}(y,w) \to \mathtt{Nxt}(z,w) \quad (42)$$
$$\mathtt{End}(x) \wedge \mathtt{Stp}(x,z) \to \exists v.\mathtt{Nxt}(z,v) \wedge \mathtt{S\_}(v) \wedge \mathtt{End}(v) \quad (43)$$
$$\mathtt{Hd}_q(x) \wedge \mathtt{S}_a(x) \to \exists z.\mathtt{Stp}(x,z) \wedge \mathtt{S}_b(z) \quad (44)$$
$$\mathtt{Hd}_q(x) \wedge \mathtt{Nxt}^+(x,y) \wedge \mathtt{S}_c(y) \to \exists z.\mathtt{Stp}(y,z) \wedge \mathtt{S}_c(z) \quad (45)$$
$$\mathtt{Hd}_q(x) \wedge \mathtt{Nxt}^+(y,x) \wedge \mathtt{S}_c(y) \to \exists z.\mathtt{Stp}(y,z) \wedge \mathtt{S}_c(z) \quad (46)$$
$$\mathtt{Hd}_q(x) \wedge \mathtt{S}_a(x) \wedge \mathtt{Stp}(x,z) \wedge \mathtt{Nxt}(z,w) \to \mathtt{Hd}_r(w) \quad (47)$$
$$\mathtt{Hd}_q(x) \wedge \mathtt{S}_a(x) \wedge \mathtt{Stp}(x,z) \wedge \mathtt{Nxt}(w,z) \to \mathtt{Hd}_r(w) \quad (48)$$

Figure 6: The rule set $\mathcal{R}_5$ contains $\mathcal{R}_4$ (see Figure 4) and the above rules, where we instantiate rules (44)–(46) for all transitions $(q,a) \mapsto (r,b,X) \in \delta$ and $c \in \Gamma$; rule (47) for all $(q,a) \mapsto (r,b,+1) \in \delta$; and rule (48) for all $(q,a) \mapsto (r,b,-1) \in \delta$.

the transitive closure $\mathtt{Nxt}^+$. Moreover, let $\mathcal{J}$ be the extension of $\mathcal{I}$ with all $\mathtt{Nxt}^+$ required to satisfy (40) and (41) (note that $\mathtt{Nxt}$ also occurs in encodings from $\mathcal{R}_3$). We define $Int_5(\mathcal{I})$ as the union of $\mathcal{J}$ with the interpretations $Run(u_w)$ for all u_w with $\mathtt{Leaf}(u_w) \in \mathcal{I}$.

Lemma 9. *$\mathfrak{I}_5 = \{Int_5(\mathcal{I}) \mid \mathcal{I} \in \mathfrak{I}_4\}$ is a universal model set of $\mathcal{R}_5$ and $\mathcal{D}$.*

Proving Lemma 4 To complete the proof of Lemma 4, we set $\mathfrak{M} = \mathfrak{I}_5$. For $\mathcal{I} \in \mathfrak{M}$, let $Db(\mathcal{I}) = \{\mathtt{p}(\vec{t}) \mid \mathtt{In_p}(\vec{t}) \in \mathcal{I}\}$ denote the completed database created by $\mathcal{R}_1$. Due to rule (4) in Figure 1, there is a homomorphism $\mathcal{D} \to Db(\mathcal{I})$. Moreover, the representation tree constructed for $\mathcal{I}$ by $\mathcal{R}_2$ has a branch that is maximal, i.e., has $|Order(\mathcal{I})|$ nodes; this branch has a leaf u_w with $|w| = |Order(\mathcal{I})|$. We obtain a homomorphism $Db(\mathcal{I}) \to branchDb(\mathcal{I}, u_w)$. Lemma 4 now follows from Lemma 9 and Lemmas 11 and 10 below.

Lemma 10. *If $\mathcal{D} \in \mathfrak{Q}$, then $\mathtt{Goal} \in \mathcal{I}$ for each $\mathcal{I} \in \mathfrak{M}$.*

Proof. As shown above, there is a homomorphism $\mathcal{D} \to branchDb(\mathcal{I}, u_w)$ for the node u_w where $|w| = |Order(\mathcal{I})|$. Since $\mathfrak{Q}$ is closed under homomorphisms, $\mathcal{D} \in \mathfrak{Q}$ implies $branchDb(\mathcal{I}, u_w) \in \mathfrak{Q}$. By the correctness of our TM simulation, we obtain $\mathtt{Goal} \in \mathcal{I}$. □

Lemma 11. *If $\mathcal{D} \notin \mathfrak{Q}$, then $\mathtt{Goal} \notin \mathcal{I}$ for some $\mathcal{I} \in \mathfrak{M}$.*

Proof. Consider some $\mathcal{I} \in \mathfrak{M}$ such that $Db(\mathcal{I}) = \mathcal{D}$ and $\mathtt{NEq}(t,u) \in \mathcal{I}$ for each $t, u \in \mathsf{Nulls}(\mathcal{D})$ with $t \neq u$. Let u_w denote the leaf node with $|w| = |Order(\mathcal{I})|$ as before. Then $branchDb(\mathcal{I}, u_w)$ is isomorphic to $Db(\mathcal{I}) = \mathcal{D}$. By the correctness of our TM simulation, $\mathtt{Goal}$ is not derived from this maximal branch. Moreover, for all other leaf nodes u_v with $\mathtt{Leaf}(u_v) \in \mathcal{I}$, there is a homomorphism $branchDb(\mathcal{I}, u_v) \to branchDb(\mathcal{I}, u_w)$. Since $\mathfrak{Q}$ is closed under homomorphisms, the TM does not accept any such $branchDb(\mathcal{I}, u_v)$, so $\mathtt{Goal} \notin \mathcal{I}$. □

4 Ensuring Chase Termination

While the rules in Section 3 are semantically correct, the disjunctive chase may not terminate on them. Many known fragments of existential rules can guarantee chase termination, including for expressive cases where termination might be exponential (Carral et al. 2019), but they are not applicable to our case, since the runtime of TMs that decide a query can in general not be bounded by any elementary function. Indeed, we rely on the TM to stop "naturally", by virtue of being a decider. Nevertheless, our rules lead to infinite chase trees, e.g., if the disjunctive guessing of $\mathtt{LT}$ leads to a cycle, which enables rule (16) to create an infinite path in the representation tree. We will now show that this can be avoided:

Theorem 12. *Every homomorphism-closed decidable query is expressed by a set of disjunctive rules that is chase-terminating for all databases over the schema of the query.*

To show this, we refine and generalise the "emergency brake" technique of Krötzsch, Marx, and Rudolph (2019), and re-formulate it as a general rule set transformation. This not only yields a generic method that is of independent interest, but it also allows us to address potential termination problems in our prior modelling.

Definition 3. Consider a rule set Σ and a nullary predicate $\mathtt{Halt}$ that does not occur in Σ. For every predicate $\mathtt{p}$ in Σ, let $\hat{\mathtt{p}}$ be a fresh predicate of the same arity, and, for any formula ψ, let $\hat{\psi}$ be ψ with all predicates $\mathtt{p}$ replaced by $\hat{\mathtt{p}}$. Now the set $brake(\Sigma, \mathtt{Halt})$ consists of the following rules:

$$\to \exists v.\mathtt{Brake}(v) \quad (49)$$
$$\mathtt{Halt} \wedge \mathtt{Brake}(x) \to \mathtt{Real}(x) \quad (50)$$
$$\hat{\mathtt{p}}(\vec{x}) \wedge \textstyle\bigwedge_{x \in \vec{x}} \mathtt{Real}(x) \to \mathtt{p}(\vec{x}) \qquad \text{for all } \mathtt{p} \text{ in } \Sigma \quad (51)$$

For every rule $\rho : \beta[\vec{x}] \to \bigvee_{i=1}^{k} \exists \vec{y}_i.\eta_i[\vec{x}_i, \vec{y}_i]$:

$$\beta[\vec{x}] \wedge \mathtt{Brake}(v) \to \textstyle\bigvee_{i=1}^{k} \big(\mathtt{B}^i_\rho(\vec{x}_i) \wedge \hat{\eta}_i[\vec{x_i}, \vec{y}_i \mapsto v] \wedge \bigwedge_{x \in \vec{x}_i} \mathtt{Real}(x)\big) \quad (52)$$
$$\mathtt{B}^i_\rho(\vec{x}_i) \to \exists \vec{y_i}.\hat{\eta}_i[\vec{x_i}, \vec{y_i}] \wedge \textstyle\bigwedge_{y \in \vec{y}_i} \mathtt{Real}(y) \quad (53)$$

where $\hat{\eta}_i[\vec{x_i}, \vec{y}_i \mapsto v]$ is $\hat{\eta}_i$ with each variable $y \in \vec{y}_i$ replaced by v, and $\mathtt{Brake}$, $\mathtt{Real}$, and all $\mathtt{B}^i_\rho$ are fresh predicates with arities as indicated.

Note that $brake(\Sigma, \mathtt{Halt})$ does not define rules to derive $\mathtt{Halt}$, and indeed the transformation largely preserves the models of Σ in the following sense:

Lemma 13. *Consider a rule set Σ and database $\mathcal{D}$ over predicates that occur in Σ. For every model $\mathcal{I}$ of $brake(\Sigma, \mathtt{Halt})$ and $\mathcal{D}$, the set $\mathcal{I}^- = \{\mathtt{p}(\vec{n}) \mid \mathtt{p}(\vec{n}) \in \mathcal{I}, \mathtt{p}$ occurs in $\Sigma\}$ is a model of Σ and $\mathcal{D}$, and every model $\mathcal{J}$ of Σ and $\mathcal{D}$ is of this form.*

Proof. Consider a rule $\rho \in \Sigma$ as in Definition 3, and let σ be a substitution such that $\sigma(\beta) \subseteq \mathcal{I}^-$. Then we can apply rules (52), (53), and finally (51) to derive $\sigma'(\eta_i) \subseteq \mathcal{I}^-$ for a suitable extension σ' of σ. Hence $\mathcal{I}^- \models \Sigma$.

Conversely, let $\mathcal{J} \models \Sigma$. A model $\mathcal{I}$ of $brake(\Sigma, \mathtt{Halt})$ can be found by adding, for each matching body $\sigma(\beta) \subseteq \mathcal{J}$

of rule ρ, an atom $\sigma(\mathtt{B}^i_\rho(\vec{x}_i))$ for some i such that $\sigma'(\eta_i) \subseteq \mathcal{J}$ for an extension σ' of σ. To obtain the required model $\mathcal{I}$ of $\mathit{brake}(\Sigma, \mathtt{Halt})$, it remains to add facts $\mathtt{Brake}(b)$ for a fresh null b, $\sigma(\hat{\eta}_i[\vec{x}_i, \vec{y}_i \mapsto b])$ as in (52) for every $\sigma(\mathtt{B}^i_\rho(\vec{x}_i)) \in \mathcal{I}$, and $\mathtt{Real}(n)$ for every $\mathtt{p}(\vec{n}) \in \mathcal{J}$ and $n \in \vec{n}$. □

For $\mathit{brake}(\Sigma, \mathtt{Halt})$ to be useful, we need to add rules that can "pull the brake" by deriving $\mathtt{Halt}$. Doing so stops the chase in the following sense:

Lemma 14. *Consider a rule set Σ, a database $\mathcal{D}$ over predicates that occur in Σ, and a set Π of rules of the form $\beta \to \mathtt{Halt}$ where β only uses predicates in Σ. If $\mathcal{I}$ is the label of a node in a chase tree for $\Sigma \cup \Pi$ and $\mathcal{D}$ such that $\mathtt{Halt} \in \mathcal{I}$, then the tree starting at the node of $\mathcal{I}$ is finite.*

Proof. Since $\mathtt{Halt} \in \mathcal{I}$, there is a substitution σ such that $\langle \rho_{(50)}, \sigma \rangle$ is applicable (for $\rho_{(50)}$ in (50)). By fairness, $\mathtt{Real}(\sigma(x))$ will be derived at some depth of the tree. From this depth on, no rule of form (53) is applicable: given $\mathtt{Real}(\sigma(x))$, the head of rules of form (52) already satisfies the head of the rule (53) that could be applied to a newly derived atom for $\mathtt{B}^i_\rho$. Rules other than (53) do not contain existential quantifiers thus can only be applied a finite number of times before the chase on this part of the tree terminates. □

If $\mathtt{Halt}$ is derived, the semantic correspondence of Lemma 13 is weakened, but suffices to preserve entailments:

Lemma 15. *Consider Σ, $\mathcal{D}$, and Π as in Lemma 14. For every model $\mathcal{I}$ of $\mathit{brake}(\Sigma, \mathtt{Halt}) \cup \Pi$ and $\mathcal{D}$, $\mathcal{I}^-$ (as in Lemma 13) is a model of Σ and $\mathcal{D}$.*

Proof. This is immediate from Lemma 13 and the fact that every model of $\mathit{brake}(\Sigma, \mathtt{Halt}) \cup \Pi$ and $\mathcal{D}$ is also a model of $\mathit{brake}(\Sigma, \mathtt{Halt})$ and $\mathcal{D}$. □

Having established the key properties of the emergency brake construction, we can now apply it to show Theorem 12. Given the rule set $\mathcal{R}_5$ as defined for a query $\mathfrak{Q}$ in Section 3, let $\mathcal{R}_6$ denote the extension of $\mathit{brake}(\mathcal{R}_5, \mathtt{Halt})$ with the following rules:

$$\mathtt{In}_\mathtt{p}(\vec{x}) \wedge \mathtt{NIn}_\mathtt{p}(\vec{x}) \to \mathtt{Halt} \tag{54}$$

$$\mathtt{LT}(x, x) \to \mathtt{Halt} \tag{55}$$

$$\mathtt{Last}(x) \wedge \mathtt{LT}(x, y) \to \mathtt{Halt} \tag{56}$$

$$\mathtt{LT}(x, y) \wedge \mathtt{First}(y) \to \mathtt{Halt} \tag{57}$$

Lemma 16. *$\mathcal{R}_6$ expresses the query $\mathfrak{Q}$.*

Proof. For a database $\mathcal{D}$ over $\mathcal{S}$, let $\mathfrak{M}$ be the universal model set constructed in Section 3. If $\mathcal{D} \in \mathfrak{Q}$, then $\mathcal{R}_5, \mathcal{D} \models \mathtt{Goal}$ by Theorem 3. Then $\mathcal{R}_6, \mathcal{D} \models \mathtt{Goal}$ since any model of $\mathcal{R}_6$ and $\mathcal{D}$ must contain $\mathtt{Goal}$ by Lemma 15.

Conversely, if $\mathcal{D} \notin \mathfrak{Q}$, then there is $\mathcal{U} \in \mathfrak{M}$ with $\mathtt{Goal} \notin \mathcal{U}$. By Lemma 13, there is a model $\mathcal{I}$ of $\mathit{brake}(\mathcal{R}_5, \mathtt{Halt})$ with $\mathcal{I}^- = \mathcal{U}$, and hence $\mathtt{Goal} \notin \mathcal{I}$. By construction of $\mathfrak{M}$, none of the rules (54)–(57) applies to $\mathcal{U}$, and hence $\mathcal{I}$ is also a model of $\mathcal{R}_6$, i.e., $\mathcal{R}_6, \mathcal{D} \not\models \mathtt{Goal}$. □

Lemma 17. *$\mathcal{R}_6$ is chase-terminating for all databases over the schema $\mathcal{S}$ of the query $\mathfrak{Q}$.*

Proof. Consider a chase over $\mathcal{R}_6$ and input database $\mathcal{D}$. Chase branches where $\mathtt{Halt}$ is eventually part of a node label terminate by Lemma 14. Let b denote any branch of the chase where $\mathtt{Halt}$ is not derived, and let $\mathcal{I}$ be the union of all node labels on that branch. We want to show that $\mathcal{I}$ (and hence b) is finite.

By Lemma 13, $\mathcal{I} \models \mathcal{R}_1$. Moreover, since $\mathtt{Halt} \notin \mathcal{I}$, rules (54)–(57) are not applicable to $\mathcal{I}$. Both properties together suffice to show that the set $\mathcal{I}_{\mathcal{R}_1} = \{\mathtt{p}(\vec{n}) \in \mathcal{I} \mid \mathtt{p} \text{ is a predicate in } \mathcal{R}_1\}$ is an element of $\mathfrak{I}_1$ defined in Section 3.

Since the predicates in rule bodies of $\mathcal{R}_1$ do not occur in any rule head in $\mathcal{R}_5 \setminus \mathcal{R}_1$, we can assume without loss of generality (and without affecting chase termination), that the corresponding rules of $\mathit{brake}(\mathcal{R}_1, \mathtt{Halt}) \subseteq \mathcal{R}_6$ have been applied first. This shows that $\mathcal{I}$ is equal to the result of a chase with non-disjunctive rules $\mathcal{R}_6 \setminus \mathit{brake}(\mathcal{R}_1, \mathtt{Halt})$ on a database $\mathcal{I}_{\mathcal{R}_1} \in \mathfrak{I}_1$. The claim follows by noting that any such chase must terminate: this was shown in Section 3, where we described a deterministic process of defining the elements in the universal model set $\mathfrak{I}_5$ from those in $\mathfrak{I}_1$. Each steps in this construction is fully determined and introduces isomorphic sets of nulls irrespectively of the order of rule applications. The only exception are application of rules (15), (16), and (18). For example, given facts $\mathtt{First}(n_1)$, $\mathtt{First}(n_2)$, and $\mathtt{Eq}(n_1, n_2)$, the standard model of Section 3 contains one fact $\mathtt{Root}(u_1)$ with $\mathtt{Rep}(n_1, u_1)$ and $\mathtt{Rep}(n_2, u_1)$, which can be obtained using (15) (on n_1) and (18). If we apply (15) to both n_1 and n_2 before applying (18), we obtain two distinct $\mathtt{Root}(u_1)$ and $\mathtt{Root}(u'_1)$. Similar variations can occur with other tree nodes if (16) is applied before (18). If is easy to see that this does not endanger termination, but merely leads to several isomorphic paths in the representation tree. □

Together, Lemmas 16 and 17 show Theorem 12.

5 Removing Disjunctions

Our main result is that any decidable homomorphism-closed query is expressible by a chase-terminating existential rule set. To conclude the proof of this statement, we remove the disjunction from the rule set $\mathcal{R}_6$ of Section 4. We present this as a general technique of expressing disjunctive Datalog using existential rules, which is also of independent interest.

For a rule set Σ, the *input schema* $\mathcal{S}_{\mathsf{in}}(\Sigma)$ is the set of all predicates in Σ that do not occur in any rule head. We focus on rule sets that can be split into a disjunctive part and an existential part, such that it is admissible to completely apply the disjunctive rules first, and the existential ones afterwards:

Definition 4. A *split* of a set Σ of disjunctive existential rules consists of a set Σ_1 of disjunctive Datalog rules and a set Σ_2 of existential rules, such that $\Sigma = \Sigma_1 \cup \Sigma_2$ and:

> For every database $\mathcal{D}$ over $\mathcal{S}_{\mathsf{in}}(\Sigma)$, and for every chase result $\mathfrak{M}$ over $\langle \Sigma, \mathcal{D} \rangle$, there is a chase result $\mathfrak{M}_1$ over $\langle \Sigma_1, \mathcal{D} \rangle$, such that $\mathfrak{M} = \{\mathcal{C}_\mathcal{I} \mid \mathcal{I} \in \mathfrak{M}_1\}$ where each $\mathcal{C}_\mathcal{I}$ is the (unique) interpretation resulting from some chase over $\langle \Sigma_2, \mathcal{I} \rangle$.

$$\rightarrow \exists w.\texttt{Init}(w) \wedge \texttt{Done}(w) \wedge \texttt{Empty}(w) \tag{58}$$

$$\texttt{Done}(w) \wedge \texttt{Init}(w) \wedge \mathtt{p}(\vec{x}) \rightarrow \exists w'.\texttt{Ins}_{\mathtt{p}}(\vec{x}, w, w') \wedge \texttt{Subs}(w', w') \wedge \texttt{Init}(w') \tag{59}$$

$$\texttt{Done}(w) \wedge \textstyle\bigwedge_{\mathtt{p}(\vec{x})\in\beta} \texttt{Ins}_{\mathtt{p}}(\vec{x}, w, w) \rightarrow \exists w_1.\texttt{Ins}_{\mathtt{p}_1}(\vec{x_1}, w, w_1) \wedge \texttt{Subs}(w_1, w_1) \tag{60}$$

$$\texttt{Done}(w) \wedge \textstyle\bigwedge_{\mathtt{p}(\vec{x})\in\beta} \texttt{Ins}_{\mathtt{p}}(\vec{x}, w, w) \rightarrow \exists w_2.\texttt{Ins}_{\mathtt{p}_2}(\vec{x_2}, w, w_2) \wedge \texttt{Subs}(w_2, w_2) \tag{61}$$

$$\texttt{Ins}_{\mathtt{p}}(\vec{x}, w_0, w_1) \wedge \texttt{Subs}(w_1, w_2) \rightarrow \texttt{Ins}_{\mathtt{p}}(\vec{x}, w_2, w_2) \wedge \mathtt{p}'(\vec{x}, w_2) \wedge \texttt{Subs}(w_0, w_2) \tag{62}$$

$$\texttt{Empty}(w) \wedge \texttt{Subs}(w, w') \rightarrow \texttt{Done}(w') \tag{63}$$

Figure 7: The rule set Σ_1', where we instantiate (59) and (62) for all $\mathtt{p} \in \mathsf{Preds}$ in $\Sigma_1 \cup \Sigma_2$, and (60) and (61) for all $\beta \rightarrow \mathtt{p}_1(\vec{x_1}) \vee \mathtt{p}_2(\vec{x_2}) \in \Sigma_1$.

$$\texttt{Done}(w) \wedge \textstyle\bigwedge_{\mathtt{p}(\vec{x})\in\beta} \mathtt{p}'(\vec{x}, w) \rightarrow \exists \vec{z}.\, \bigwedge_{\mathtt{q}(\vec{y})\in\eta} \mathtt{q}'(\vec{y}, w) \tag{64}$$

Figure 8: The rule set Σ_2', where we instantiate (64) for all $\beta \rightarrow \exists \vec{z}.\eta \in \Sigma_2$.

$$\texttt{Goal}'(w) \rightarrow \texttt{Acc}(w) \tag{65}$$

$$\texttt{Ins}_{\mathtt{p}_1}(\vec{x}_1, w, w_1) \wedge \texttt{Acc}(w_1) \wedge \texttt{Ins}_{\mathtt{p}_2}(\vec{x}_2, w, w_2) \wedge \texttt{Acc}(w_2) \wedge \textstyle\bigwedge_{\mathtt{p}(\vec{x})\in\beta} \texttt{Ins}_{\mathtt{p}}(\vec{x}, w, w) \rightarrow \texttt{Acc}(w) \tag{66}$$

$$\texttt{Init}(w) \wedge \texttt{Acc}(w) \rightarrow \texttt{Goal} \tag{67}$$

Figure 9: The rule set Σ_3', where (66) is instantiated for all rules $\beta \rightarrow \mathtt{p}_1(\vec{x_1}) \vee \mathtt{p}_2(\vec{x_2}) \in \Sigma_1$

Lemma 18. *Consider a rule set Σ with split $\langle \Sigma_1, \Sigma_2 \rangle$. There is a set Σ' of existential rules, such that, for every database $\mathcal{D}$ over $\mathcal{S}_{\mathsf{in}}(\Sigma)$, we have:*

1. *$\mathcal{D}, \Sigma \models \texttt{Goal}$ iff $\mathcal{D}, \Sigma' \models \texttt{Goal}$, and*
2. *if $\langle \Sigma_2, \mathcal{D}_2 \rangle$ is chase-terminating for every database $\mathcal{D}_2$ over $\mathcal{S}_{\mathsf{in}}(\Sigma_2)$, then $\langle \Sigma', \mathcal{D} \rangle$ is also chase-terminating.*

To construct this set Σ', we assume w.l.o.g. that all disjunctive rules have exactly two disjuncts in the head. We define Σ' as the union of sets Σ_1', Σ_2' and Σ_3' as shown in Figures 7, 8 and 9, which we explain below.

Σ_1' uses a technique for modelling sets with chase-terminating existential rules (Krötzsch, Marx, and Rudolph 2019, Fig. 2). We adapt this to sets of ground atoms, called *worlds* and denoted by variables w in the figures. Facts $\texttt{Ins}_{\mathtt{p}}(\vec{t}, w, w')$ express that world w' is obtained by adding $\mathtt{p}(\vec{t})$ to world w. In particular, $\texttt{Ins}_{\mathtt{p}}(\vec{t}, w, w)$ states that $\mathtt{p}(\vec{t})$ is in w, and we define $\mathsf{world}(w) = \{\mathtt{p}(\vec{a}) \mid \texttt{Ins}_{\mathtt{p}}(\vec{a}, w, w) \in \mathcal{I}\}$ for any interpretation $\mathcal{I}$. Worlds are created by adding database facts (59) or by applying rules to existing worlds (60)–(61). Worlds containing only database facts are marked with `Init`. Predicate `Subs` defines the subset relation on worlds. Rules (62)–(63) copy all prior facts to a new world before marking it `Done`.

Proposition 19. *Σ_1' is chase-terminating and for every $\mathcal{D}$ over $\mathcal{S}_{\mathsf{in}}(\Sigma)$, the (unique) interpretation $\mathcal{I}$ resulting from some chase over Σ_1' and $\mathcal{D}$ is such that:*

- *if $\mathtt{p}(\vec{a}) \in \mathcal{D}$ and $\texttt{Done}(w) \in \mathcal{I}$, there exists w' such that $\texttt{Ins}_{\mathtt{p}}(\vec{a}, w, w') \in \mathcal{I}$ and $\mathsf{world}(w') = \mathsf{world}(w) \cup \{\mathtt{p}(\vec{a})\}$;*
- *if $\rho \in \Sigma_1$ is applicable to $\mathsf{world}(w)$, creating $\mathtt{p}_1(\vec{a})$ or $\mathtt{p}_2(\vec{b})$, there exists w_1 and w_2 such that $\{\texttt{Ins}_{\mathtt{p}_1}(\vec{a}, w, w_1), \texttt{Ins}_{\mathtt{p}_2}(\vec{b}, w, w_2)\} \subseteq \mathcal{I}$, $\mathsf{world}(w_1) = \mathsf{world}(w) \cup \{\mathtt{p}_1(\vec{a})\}$ and $\mathsf{world}(w_2) = \mathsf{world}(w) \cup \{\mathtt{p}_2(\vec{b})\}$.*

Note that we cannot distinguish worlds that are not containing all database facts, and that some worlds may contain more facts than needed to satisfy all disjunctive heads.

Σ_2' now simulates the application of rules from Σ_2 in any of the worlds. Computations relative to different worlds are independent from each other. Finally, Σ_3' aggregates results from all worlds: a world is accepting (`Acc`) if either `Goal` was derived locally (65) or it has two successor worlds for a disjunctive rule that are both accepting (66). `Goal` is a consequence if any initial world is accepting (67). This finishes the construction of Σ' as the main ingredient for proving Lemma 18.

Finally, we can apply Lemma 18 to $\mathcal{R}_6 = \mathit{brake}(\mathcal{R}_5, \texttt{Halt}) \cup \Pi$ from Section 4, where Π denotes rules (54)–(57). Intuitively, a possible split is $\mathit{brake}(\mathcal{R}_1, \texttt{Halt})$ and $\mathit{brake}(\mathcal{R}_5 \setminus \mathcal{R}_1, \texttt{Halt}) \cup \Pi$. Formally, however, $\mathit{brake}(\mathcal{R}_1, \texttt{Halt})$ is not disjunctive Datalog due to existential rules (2), (3), and (49). However, our result easily extends to such rules with empty body: we can just add them to Σ_1' and treat their inferences like facts from the initial database. The other properties of Definition 4 are easy to verify. The fact that both rule sets have some common rules, such as (50), is no concern. Finally, it remains to argue termination for Σ_2 as required for item (2) in Lemma 18. This is slightly stronger than Lemma 17 since we must also consider databases that use some inferred predicates of $\mathcal{R}_1$. However, the proof of Lemma 17 and the emergency brake technique in general served the main purpose of safeguarding against problematic structures among the inferred predicates of $\mathcal{R}_1$, and it is not hard to see that this already showed what we require here. Combining all of our insights, we finally obtain:

Theorem 20. *Chase-terminating existential rules capture the class of all decidable homomorphism-closed queries.*

6 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 21. *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 (cf. Definition 5), show that $\mathcal{M}$ can be enumerated up to equivalence if there is a semi-decidable language that satisfies (i) and (ii) above (cf. Lemma 22), and finally prove that the consequence of this implication does not hold (cf. Lemma 23).

Definition 5. Consider the set $\mathcal{M}$ of all TMs M such that: (i) The TM M halts on all inputs. (ii) If M accepts some word w, then w corresponds to a database over schema $\{\mathtt{ed}\}$. (iii) Consider some words w and v that correspond to some $\mathcal{D}$ and $\mathcal{E}$ in $\mathfrak{D}_{\{\mathtt{ed}\}}$, respectively. If M accepts w and there is a homomorphism $h : \mathcal{D} \to \mathcal{E}$, then M accepts v.

Intuitively, $\mathcal{M}$ is the set of all deciders that solve homomorphism-closed queries over databases in $\mathfrak{D}_{\{\mathtt{ed}\}}$.

Lemma 22. *If there is a semi-decidable query language $\mathfrak{F}$ that satisfies (i) and (ii) in Theorem 21, then $\mathcal{M}$ is enumerable up to equivalence.*

Proof. If there is a language such as $\mathfrak{F}$, then there is an enumerator P for L that prints out a sequence $l_1, l_2, \ldots$ and a decider $M_{\mathfrak{F}}$ that can be used to check if $\mathcal{D} \in \mathfrak{F}(l)$ for each $(l, \mathcal{D}) \in (L \times \mathfrak{D}_{\{\mathtt{ed}\}})$. For each $i \geq 1$, let M_i be the TM that, on input w, performs the following computation: if w corresponds to a database $\mathcal{D} \in \mathfrak{D}_{\{\mathtt{ed}\}}$ and $M_{\mathfrak{F}}$ accepts $(l_i, \mathcal{D})$, then *accept*; otherwise, *reject*. By modifying P we can define an enumerator that prints out the sequence $M_1, M_2, \ldots$, which contains $\mathcal{M}$ up to equivalence. □

Lemma 23. *The set $\mathcal{M}$ is not enumerable up to equivalence.*

Proof Sketch. Assume that there is an enumerator that outputs a sequence $M_1, M_2, \ldots$ that includes $\mathcal{M}$ up to equivalence. We obtain a contradiction by defining a sequence $\mathcal{D}_1, \mathcal{D}_2, \ldots$ of databases and a TM $M_d \in \mathcal{M}$ that diagonalises over $M_1, M_2, \ldots$ and $\mathcal{D}_1, \mathcal{D}_2, \ldots$ Namely, for each $i \geq 1$, let $\mathcal{D}_i = \{\mathtt{ed}(u_1, u_2), \ldots, \mathtt{ed}(u_{p_{i+1}}, u_1)\}$ where p_{i+1} is the $(i+1)$-th prime. Moreover, M_d is the TM that, on input w, performs the computation: (1) *Reject* if w does not correspond to some $\mathcal{D} \in \mathfrak{D}_{\{\mathtt{ed}\}}$. (2) *Reject* if $\mathcal{D}$ can be homembedded into a path over $\mathtt{ed}$. (3) *Accept* if $\mathtt{ed}(u, u) \in \mathcal{D}$ for some null u. (4) If there is some $i \geq 1$ such that there are less nulls in $\mathcal{D}_i$ than in $\mathcal{D}$, the TM M_i accepts some serialisation that corresponds to $\mathcal{D}_i$, and there is a homomorphism $h : \mathcal{D} \to \mathcal{D}_i$; then *reject*. Otherwise, *accept*. □

7 Discussion and Conclusion

In this paper, we have established a characterization of all decidable homomorphism-closed Boolean queries. We showed that 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 24. *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 existential rule queries,*
4. *disjunctive chase-terminating existential rule queries.*

Proof. (3) Standard-chase termination implies core-chase termination. On the other hand, core chase termination implies decidability and thus our result applies. (1) and (2) Standard-chase termination implies these weaker form of guarantees, which themselve imply core chase termination. (4) Obviously, every (non-disjunctive) existential rule set is a special case of a disjunctive one and for this special case, disjunctive chase termination coincides with termination of the (non-disjunctive) standard chase. On the other hand, disjunctive existential rule queries are also closed under homomorphisms, and disjunctive universal chase termination obviously implies decidability. So our result applies. □

However, the applicability of our result does not stop at (syntactic) extensions of our framework, as it applies to arbitrary query languages and querying formalisms of different types. In particular we would like to stress the relationship to the very comprehensive existential rules fragment of *bounded treewidth sets (bts) of rules* (Baget et al. 2011a) that is *not* chase-terminating and encompasses a plethora of well-known existential rule fragments with decidable query entailment, including guarded (Calì, Gottlob, and Kifer 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 25. *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 $\mathcal{D}, \Sigma_Q \models \mathtt{Goal}$.*

While possibly surprising, this is a straightforward consequence of decidability of conjunctive query entailment from bts and of homomorphism-closedness of existential rule queries in general. Note, however, that every Q would give rise to a different Σ_Q. In fact, 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, Zhang, and You 2015). Such a set will not exist in all cases.

While our result addresses many of the open questions regarding *expressivity* of the terminating chase (Krötzsch, Marx, and Rudolph 2019) an important avenue for future work is to investigate potential differences when it comes to the corresponding computational *complexities*. We deem it likely that not all of the discussed chase variants give rise to worst-case optimal computations.

Acknowledgements

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, S.; Hull, R.; and Vianu, V. 1995. *Foundations of Databases*. Addison-Wesley.

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

Baget, J.; Mugnier, M.; Rudolph, S.; and Thomazo, M. 2011b. Walking the complexity lines for generalized guarded existential rules. In Walsh (2011), 712–717.

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

Bourgaux, C.; Carral, D.; Krötzsch, M.; Rudolph, S.; and Thomazo, M. 2021. Capturing homomorphism-closed decidable queries with existential rules. arXiv:2107.07811 [cs.LO].

Calì, A.; Gottlob, G.; and Kifer, M. 2008. Taming the infinite chase: Query answering under expressive relational constraints. In Brewka, G., and Lang, J., eds., *Proc. 11th Int. Conf. on Knowledge Representation and Reasoning (KR'08)*, 70–80. AAAI Press.

Carral, D.; Dragoste, I.; Krötzsch, M.; and Lewe, C. 2019. Chasing sets: How to use existential rules for expressive reasoning. In Kraus, S., ed., *Proc. 28th Int. Joint Conf. on Artificial Intelligence, IJCAI'19*, 1624–1631. ijcai.org.

Carral, D.; Dragoste, I.; and Krötzsch, M. 2017. Restricted chase (non)termination for existential rules with disjunctions. In Sierra, C., ed., *Proc. 26th Int. Joint Conf. on Artificial Intelligence, IJCAI'17*, 922–928. ijcai.org.

Deutsch, A.; Nash, A.; and Remmel, J. B. 2008. The chase revisited. In Lenzerini, M., and Lembo, D., eds., *Proc. 27th Symposium on Principles of Database Systems (PODS'08)*, 149–158. ACM.

Fagin, R.; Kolaitis, P. G.; Miller, R. J.; and Popa, L. 2005. Data exchange: semantics and query answering. *Theoretical Computer Science* 336(1):89–124.

Grahne, G., and Onet, A. 2018. Anatomy of the chase. *Fundam. Informaticae* 157(3):221–270.

Krötzsch, M., and Rudolph, S. 2011. Extending decidable existential rules by joining acyclicity and guardedness. In Walsh (2011), 963–968.

Krötzsch, M.; Marx, M.; and Rudolph, S. 2019. The power of the terminating chase. In Barceló, P., and Calautti, M., eds., *Proc. 22nd Int. Conf. on Database Theory, ICDT'19*, volume 127 of *LIPIcs*, 3:1–3:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Marnette, B. 2009. Generalized schema-mappings: from termination to tractability. In Paredaens, J., and Su, J., eds., *Proc. 28th Symposium on Principles of Database Systems (PODS'09)*, 13–22. ACM.

Rudolph, S., and Thomazo, M. 2015. Characterization of the expressivity of existential rule queries. In Yang, Q., and Wooldridge, M., eds., *Proc. 24th Int. Joint Conf. on Artificial Intelligence (IJCAI'15)*, 3193–3199. AAAI Press.

Walsh, T., ed. 2011. *Proc. 22nd Int. Joint Conf. on Artificial Intelligence (IJCAI'11)*. AAAI Press/IJCAI.

Zhang, H.; Zhang, Y.; and You, J. 2015. Existential rule languages with finite chase: Complexity and expressiveness. In Bonet, B., and Koenig, S., eds., *Proc. 29th AAAI Conf. on Artificial Intelligence (AAAI'15)*. AAAI Press.