Proceedings of the 21st International Conference on Principles of Knowledge Representation and Reasoning
Main Track

Ontology-Based Query Answering over Datalog-Expressible Rule Sets is Undecidable

David Carral[1] , **Lucas Larroque**[2] , **Michaël Thomazo**[2]
[1]LIRMM, Inria, University of Montpellier, CNRS, Montpellier, France
[2]Inria, DI ENS, ENS, CNRS, PSL University, Paris, France
{david.carral, lucas.larroque, michael.thomazo}@inria.fr,

Abstract

Ontology-based query answering is a problem that takes as input an ontology $\mathcal{R}$ (typically expressed by existential rules), a set $\mathcal{F}$ of facts, and a Boolean conjunctive query (CQ) q, and asks whether $\mathcal{R}, \mathcal{F} \models q$. This problem is undecidable in general, and a widely investigated approach to tackle it in some cases is query rewriting: given some "rule query" $\langle \mathcal{R}, q \rangle$, we compute a Boolean query $q_{\mathcal{R}}$ such that, for any fact set $\mathcal{F}$, it holds that $\mathcal{R}, \mathcal{F} \models q$ if and only if $\mathcal{F} \models q_{\mathcal{R}}$. Insofar, previous work has mostly focused on output queries $q_{\mathcal{R}}$ expressed as union of Boolean conjunctive queries (UCQs), and an effective algorithm that computes such a query $q_{\mathcal{R}}$ whenever it exists has been proposed in the literature. However, UCQ-rewritability is not a very general notion and many real-world interesting rule queries do no admit UCQ-rewritings. This raises the question whether such a generic algorithm can be designed for a more expressive target language, such as datalog. We solve this question by the negative, by studying the difference between datalog-expressibility and datalog-rewritability. More precisely, we show that query answering under datalog-expressible rule queries is undecidable.

1 Introduction

Efficiently accessing data is an important step in many real-world applications. Ontologies have been identified as an important tool to help a user to express their information needs, allowing them to use a vocabulary they are familiar with, while enabling a system to perform automated reasoning, leading to more complete answers. Ontology-based query answering (OBQA) is a core problem therein, where a set of facts is queried while taking into account the domain knowledge expressed in an ontology. These ontologies may be expressed in a variety of formalisms, such as Description Logics or existential rules. The OBQA problem is typically framed as, given a fact set $\mathcal{F}$, an ontology $\mathcal{R}$, and a Boolean CQ q, check if $\mathcal{F}, \mathcal{R} \models q$, where $\models$ denotes the classical first-order logic entailment.

This problem is undecidable when the ontology can range over any set of existential rules. Thus, a lot of research has focused on finding decidable and even tractable classes of rule sets; see (Mugnier and Thomazo 2014) for an introduction to these. Particularly relevant to us are classes based on the so-called *query rewriting* approach. Given an ontology $\mathcal{R}$ and a Boolean CQ q, one computes a Boolean UCQ $q_{\mathcal{R}}$ such that for any fact set $\mathcal{F}$, it holds that $\mathcal{F}, \mathcal{R} \models q$ if and only if $\mathcal{F} \models q_{\mathcal{R}}$. As most data is stored in relational databases, which have been designed to efficiently process CQs, most research has focused on rewriting the output query $q_{\mathcal{R}}$ as a UCQ. A natural question is then, given an ontology $\mathcal{R}$ and a BCQ q, is there a UCQ-rewriting $q_{\mathcal{R}}$ for $\mathcal{R}$ and q? In other terms, is that true that the rule query $\langle \mathcal{R}, q \rangle$ is *UCQ-expressible*? This does not always hold, and it is actually undecidable to check whether it is the case (Baget et al. 2011). However, there exists an effective procedure algorithm that *computes* a UCQ-rewriting when given as input a UCQ-expressible rule query (König et al. 2015). In other terms, the UCQ-expressibility of every rule query (the existence of a UCQ-rewriting) in a class and UCQ-rewritability of that class (the computability of UCQ-rewritings for all rule queries in that class) are two notions that coincide, which possibly explains why they have not been introduced separately in the literature.

Syntactic conditions such as linearity (Calì, Gottlob, and Kifer 2013) or stickiness (Calì, Gottlob, and Pieris 2010) guarantee the existence of UCQ-rewritings for any BCQ. Moreover, there is also DL-Lite (Artale et al. 2009), which is a widely used Description Logic that can be translated into existential rules. However, the expressivity of these languages is too limited for many real-world ontologies. A natural task is to consider a more expressive target query language for the rewritings. It is known that considering first-order queries instead of union of conjunctive queries does not allow covering more classes of existential rules (Rossman 2008). We then focus on another classical language, namely datalog. Note that all UCQ-expressible rule queries are also datalog-expressible but the converse is not true.

As discussed in Section 5, there are many known and interesting classes for which specific datalog-rewriting algorithms have been designed. However, no generic algorithm, such as in the case of UCQ-expressibility, is known so far. The contribution of this paper is to show that, unfortunately, no such algorithm exists. This is done by proving that the problem of checking $\mathcal{R}, \mathcal{F} \models q$ under the assumption that $\langle \mathcal{R}, q \rangle$ is datalog-expressible is undecidable, contradicting the existence of a rewriting algorithm for datalog-expressible queries. We prove the result by reduction from the halting problem of Turing machines to OBQA, where the difficulty lies in ensuring that rule queries produced by the reduction are datalog-expressible.

2 Preliminaries

We assume that the reader is familiar with first-order logic and basic concepts from computability theory. We only provide a brief recap of these topics in this section.

2.1 First-Order Logic

We define Preds, Funs, Cons, and Vars to be mutually disjoint and countably infinite sets of predicates, function symbols, constants, and variables, respectively. We associate every $s \in \mathsf{Preds} \cup \mathsf{Funs}$ with some *arity* $\mathsf{ari}(s) \geq 0$. For every $n \geq 0$, the sets of all n-ary predicates and all n-ary function symbols are also countably infinite. The set Terms includes Cons, Vars, and the set of all nullary function symbols; and contains $f(t_1, \ldots, t_n)$ for every $n \geq 1$, every n-ary $f \in \mathsf{Funs}$, and every $t_1, \ldots, t_n \in \mathsf{Terms}$. We write lists $t_1, \ldots, t_n$ of terms as $\vec{t}$ and often treat these lists as sets.

An *atom* is a first-order formula of the form $\mathtt{P}(\vec{t})$ with $\vec{t}$ a list of terms and $\mathtt{P}$ a $|\vec{t}|$-ary predicate. An atom that features a predicate $\mathtt{P}$ is a $\mathtt{P}$-*atom*. A *fact* is a function- and variable-free atom. For a nullary predicate $\mathtt{P}$, we write $\mathtt{P}$ as a shortcut for the nullary atom $\mathtt{P}()$. For a first-order formula F and a list $\vec{x}$ of variables, we write $F[\vec{x}]$ to indicate that $\vec{x}$ is the set of all free variables occurring in F.

Definition 1. *An* (existential) rule R *is a function- and constant-free first-order formula of the form* $\forall \vec{x}.(B[\vec{x}] \rightarrow \exists \vec{z}.H[\vec{y}, \vec{z}])$ *where* B *and* H *are atom conjunctions,* H *is non-empty, and all variables in* $\vec{y}$ *occur in* $\vec{x}$*. We refer to* $\vec{y}$ *as the* frontier *of* R*, and to* B *and* $\exists \vec{z}.H$ *as the* body *and the* head *of* R*, respectively. Such a rule is* datalog *if it does not feature existential variables; that is, if* $\vec{z}$ *is the empty list.*

We omit universal quantifiers when writing rules. Also, we replace wedges with commas when writing atom conjunctions, which we often identify with sets.

A *substitution* π is a function from variables to terms. For an atom $\mathtt{P}(\vec{t})$, let $\pi(\mathtt{P}(\vec{t}))$ be the atom that results from replacing all occurrences of every variable x in $\mathtt{P}(\vec{t})$ with $\pi(x)$ if the latter is defined. For some atom sets $\mathcal{A}$ to $\mathcal{B}$, a *homomorphism* π from $\mathcal{A}$ to $\mathcal{B}$ is a substitution such that $\pi(\mathcal{A}) \subseteq \mathcal{B}$. Often, we abuse notation and consider substitutions (and thus homomorphisms) that map terms to terms. An *isomorphism* π from $\mathcal{A}$ to $\mathcal{B}$ is an injective homomorphism $\mathcal{A}$ to $\mathcal{B}$ such that $\pi(\mathcal{A}) = \mathcal{B}$ and $(\pi^{-}(\mathcal{B}) = \mathcal{A})$.

A *Boolean conjunctive query* (BCQ) is a function-free first-order formula of the form $\exists \vec{z}.B[\vec{z}]$ with B a non-empty conjunction of atoms. A *union of BCQs* (UBCQs) is defined in the obvious manner. Under standard first-order semantics, a fact set $\mathcal{F}$ entails a UBCQ $\bigvee_{i=1}^{n} \exists \vec{z}_i.B_i$ if there exists a homomorphism from B_k to $\mathcal{F}$ for some $1 \leq k \leq n$.

A *knowledge base* is a pair $\langle \mathcal{R}, \mathcal{F} \rangle$ with $\mathcal{R}$ a finite rule set and $\mathcal{F}$ a finite fact set. Without loss of generality, we assume that $(\dagger)$ existentially quantified variables do not reoccur across different rules in the same rule set. For a BCQ q, we write $\langle \mathcal{R}, \mathcal{F} \rangle \models q$ to indicate that $\mathcal{R} \cup \mathcal{F}$ entails q under standard first-order semantics.

2.2 The Chase Algorithm

We use the chase to define entailment procedurally. More precisely, we present a chase variant previously considered by (Urbani et al. 2018) in which datalog rules are applied with higher priority, and non-datalog rules are applied in parallel and only if they are not already satisfied. The use of this specific variant, which produces a single chase sequence from an input knowledge base, simplifies some of the proofs presented later.

Consider a rule $\forall \vec{x}.\forall \vec{y}.B[\vec{x}] \rightarrow \exists \vec{z}.H[\vec{y}, \vec{z}]$. For every $z \in \vec{z}$, let f_z be a fresh $|\vec{y}|$-ary function symbol unique for z. Moreover, let $\mathsf{Sk}(\exists \vec{z}.H)$ be the atom set that results from replacing all occurrences of every $z \in \vec{z}$ in H with the term $f_z(\vec{y})$; note that, because of $(\dagger)$, this symbol is also unique for a rule within a given rule set. In the following, we simply write z as a shortcut for a (nullary) function symbol of the form f_z. Moreover, we write $w(t)$ or $w^1(t)$ as a shortcut for a unary term such as $f_w(t)$, $w^2(t)$ as a shortcut for $f_w(f_w((t)))$, and so on.

A *trigger* t is a pair $\langle R, \pi \rangle$ where $R = \forall \vec{x}.\forall \vec{y}.B \rightarrow \exists \vec{z}.H$ is a rule and π is a homomorphism with domain $\vec{x} \cup \vec{y}$. Let $\mathsf{support}(t) = \pi(B)$ and $\mathsf{output}(t) = \pi(\mathsf{Sk}(\exists \vec{z}.H))$. The trigger t is *applicable* to an atom set $\mathcal{A}$ if $\mathsf{support}(t) \subseteq \mathcal{A}$ and $\mathcal{A}$ does not include $\hat{\pi}(H)$ for all extensions $\hat{\pi}$ of π. A trigger is *datalog* if it features a datalog rule.

For a rule R and an atom set $\mathcal{A}$, let $R(\mathcal{A})$ be the set that includes $\mathsf{output}(t)$ for every trigger t with R that is applicable to $\mathcal{A}$. For a rule set $\mathcal{R}$, let $\mathcal{R}(\mathcal{A}) = \bigcup_{R \in \mathcal{R}} R(\mathcal{A}) \cup \mathcal{A}$. Moreover, let $\mathcal{R}^{\forall}$ and $\mathcal{R}^{\exists}$ be the sets of all datalog and non-datalog rules in $\mathcal{R}$, respectively. The *datalog closure* of $\mathcal{A}$ and $\mathcal{R}$ is the minimal superset $\mathcal{A}'$ of $\mathcal{A}$ such that $\mathcal{R}^{\forall}(\mathcal{A}') = \mathcal{A}'$; that is, $\mathcal{A}'$ is the (unique) minimal superset of $\mathcal{A}$ that satisfies all of the rules in $\mathcal{R}^{\forall}$ under standard first-order semantics. Note that an atom set satisfies a rule if no trigger with the latter is applicable to the former.

Definition 2. *For a knowledge base* $\mathcal{K} = \langle \mathcal{R}, \mathcal{F} \rangle$*; let* $\mathsf{Ch}_1(\mathcal{K}) = \mathcal{F}$*, let* $\mathsf{Ch}_i(\mathcal{K})$ *be the datalog closure of* $\mathcal{R}$ *and* $\mathsf{Ch}_{i-1}(\mathcal{K})$ *for every even* $i \geq 1$*, and let* $\mathsf{Ch}_i(\mathcal{K}) = \mathcal{R}^{\exists}(\mathsf{Ch}_{i-1}(\mathcal{K}))$ *for every odd* $i \geq 2$*. Also, let* $\mathsf{Ch}(\mathcal{K}) = \bigcup_{i \geq 1} \mathsf{Ch}_i(\mathcal{K})$ *be the* chase *of* $\mathcal{K}$*.*

The chase is a handy tool to answer conjunctive queries over a knowledge base.

Proposition 3. *A knowledge base* $\mathcal{K}$ *entails a UBCQ* q *if and only if* $\mathsf{Ch}(\mathcal{K})$ *entails* q*.*

The above holds because $\mathsf{Ch}(\mathcal{K})$ is a universal model for $\mathcal{K} = \langle \mathcal{R}, \mathcal{F} \rangle$. That is, there is a homomorphism from the chase of $\mathcal{K}$ to every model of this first-order theory.

2.3 Computability Theory

Definition 4. *A* transition function *for a set* Q *of states is a (total) function from* $(Q \setminus \{q_f\}) \times \{0, 1, B\}$ *to* $(Q \setminus \{q_s\}) \times \{0, 1\} \times \{L, R\}$ *where* q_s *and* q_f *are two different states. A* (Turing) machine *is a pair* $\langle Q, \delta \rangle$ *where* Q *is a set of states that contains* q_s *and* q_f*, and* δ *is a transition function for* Q*.*

As per our definition, all machines reuse the same *initial* q_s and *final* q_f states, as well as the same binary alphabet $\{0, 1, B\}$. Moreover, machines do not write blanks, may not reenter the starting state after the initial configuration, and halt if they reach the final state.

Definition 5. *A* configuration *is a tuple $\langle n, t, p, q\rangle$ where $n \geq 1$ is a natural number, t is a function from $\{1, \ldots, n\}$ to $\{0, 1, B\}$ that maps n to B, p is some number in $\{1, \ldots, n\}$, and q is a state. The* starting configuration *on some* word *$w_1, \ldots, w_n \in \{0, 1\}^*$ is the tuple $\langle n + 1, t, 1, q_s\rangle$ where t is the function that maps every $i \in \{1, \ldots, n\}$ to w_i, (and $n + 1$ to B).*

Given some configuration $\langle n, t, p, q\rangle$, we use t to encode the contents of the tape at each position; moreover, we use p and q to encode the position of the head and the current state of the machine, respectively.

Definition 6. *Consider a machine $M = \langle Q, \delta\rangle$, a configuration $C = \langle n, t, p, q\rangle$ with $q \in Q \setminus \{q_f\}$, and the tuple $\delta(t(p), q) = \langle r, a, D\rangle$. Then, let $\mathsf{Next}_M(C)$ be the configuration $\langle n + 1, t', p', r\rangle$ such that:*

- *For every $1 \leq i \leq n$ with $i \neq p$, let $t'(i) = t(i)$. Moreover, let $t'(p) = a$ and $t'(n + 1) = B$.*
- *If $D = L$; then $p' = p - 1$ if $p \geq 2$, and $p' = p$ otherwise. If $D = R$, then $p' = p + 1$.*

As described above, any given machine defines a function that maps non-final configurations to configurations. An exhaustive iteration through these consecutive configurations that begins with a starting configuration yields a run.

Definition 7. *Consider a machine M and a word $\vec{w}$.*

- *Define $\mathsf{Run}_M(\vec{w})$ as the (possibly infinite) sequence $C_0, \ldots, C_n, \ldots$ of configurations such that C_0 is the start configuration on $\vec{w}$, and $C_{i+1} = \mathsf{Next}_M(C_i)$ for all i such that C_i does not feature the final state.*
- *The machine M* halts *on $\vec{w}$ if $\mathsf{Run}_M(\vec{w})$ is finite.*

Note that, as per our definition, a machine halts on some word if and only if it reaches the final configuration. We do not discuss acceptance or rejection of a word by a machine; this is unnecessary for our purposes since the halting problem is already undecidable.

Proposition 8. *The problem of checking if a machine halts on the empty word is undecidable.*

3 Query Languages and Expressivity

The focus of this paper is on the query rewriting approach. To address it, we must first introduce the notions of rule query and of a $\mathcal{L}$-rewriting for some query language $\mathcal{L}$.

Definition 9. *A* rule query *is a pair $\langle \mathcal{R}, q\rangle$ where $\mathcal{R}$ is a rule set and q is a BCQ. A* datalog query *is a rule query with a datalog rule set.*

The expressivity of rule queries has been studied from both a complexity viewpoint and a model-theoretic one (Rudolph and Thomazo 2015; Bourgaux et al. 2021). A classical and practically important question is whether a given rule query can be expressed within a less expressive query language. This question is formalized with the notion of *$\mathcal{L}$-rewriting* of a rule query $\langle \mathcal{R}, q\rangle$, for a query language $\mathcal{L}$ (such as UCQ, datalog, first-order logic,...). We will consider only Boolean queries, hence we regard each query q as a set of fact sets, and we denote $\mathcal{F} \models q$ if $\mathcal{F} \in q$.

Definition 10 ($\mathcal{L}$-rewriting)**.** *Let $\mathcal{L}$ be a query language. An* $\mathcal{L}$-rewriting *of a rule query $\langle \mathcal{R}, q\rangle$ over a set of predicates $\mathbf{P}_e$ is a query $q' \in \mathcal{L}$ such that for any finite fact set $\mathcal{F}$ over $\mathbf{P}_e$, $\langle \mathcal{R}, \mathcal{F}\rangle \models q$ if and only if $\mathcal{F} \models q'$.*

Unless specified otherwise, we set $\mathbf{P}_e = \mathsf{Preds}$, and discuss the other cases in the related work section. This means we consider rewritings that preserve BCQ entailment with respect to the the initial rule query over any fact set.

Definition 11 ($\mathcal{L}$-Expressibility)**.** *Given some query language $\mathcal{L}$, a rule query $\langle \mathcal{R}, q\rangle$ is* $\mathcal{L}$-expressible *if it admits an* $\mathcal{L}$-rewriting.

Related to $\mathcal{L}$-expressibility, but possibly a stronger requirement, is the notion of of $\mathcal{L}$-rewritability: an $\mathcal{L}$-rewriting must not only exist, but we should be able to compute it.

Definition 12 ($\mathcal{L}$-Rewritability)**.** *Let $\mathcal{L}$ be a query language. A set $\mathcal{Q}$ of rule queries is* $\mathcal{L}$-rewritable *if there exists a procedure, which given q, computes an $\mathcal{L}$-rewriting q' of q if $q \in \mathcal{Q}$, and whose behavior is not specified (and may even not terminate) if $q \notin \mathcal{Q}$.*

Note that the absence of condition on the behavior of the algorithm in the case where $q \notin \mathcal{L}$ allows us to consider query languages whose membership is not decidable – this is in particular the case of UBCQ-expressible rule sets (Baget et al. 2011). In that case, $\mathcal{L}$-expressibility and $\mathcal{L}$-rewritability are identical notions: this is a direct consequence of the algorithm presented in (König et al. 2015) (see Algorithm 1 and Theorem 7).

Theorem 13. *The class of all UBCQ-expressible rule queries is UBCQ-rewritable.*

From this theorem, we can derive a useful corollary:

Corollary 14. *There is a procedure to check if a knowledge base $\langle \mathcal{R}, \mathcal{F}\rangle$ entails a BCQ q that is sound, complete, and terminates if $\langle \mathcal{R}, q\rangle$ is UBCQ-expressible.*

Proof. First, we compute a UBCQ-rewriting q' of $\langle \mathcal{R}, q\rangle$ using the procedure described by (König et al. 2015), which is guaranteed to terminate if $\langle \mathcal{R}, q\rangle$ is UBCQ-expressible. Then, we can simply check if $\mathcal{F} \models q'$; the result of this check indicates if $\langle \mathcal{R}, \mathcal{F}\rangle \models q$. □

The very same reasoning procedure described in the proof of Corollary 14 can be applied off-the-shelf for any class of rule queries that is UBCQ-expressible such as linear (Calì, Gottlob, and Kifer 2013), sticky (Calì, Gottlob, and Pieris 2010), non-local rule sets (Ostropolski-Nalewaja et al. 2022), DL-Lite (Artale et al. 2009), or any other class of UCQ-expressible rule queries yet to be defined.

In this paper, we investigate whether analogous generic procedures can be developped for more expressive query languages. As it is known that BFO-rewritability and UBCQ-rewritability coincide (Rossman 2008) for rule queries, we set $\mathcal{L}$ to datalog and turn our attention to the relationships between datalog-expressibility and datalog-rewritability. As datalog is a major query language, its expressivity has been studied (Feder and Vardi 2003; Dawar and Kreutzer 2008; Rudolph and Thomazo 2016), and it has served as a target query language for numerous

classes of rule queries (see related work). So far, however, the relationship between datalog-expressibility and datalog-rewritability remains unclear. We prove that:

Theorem 15. *The class of all datalog-expressible rule queries is not datalog-rewritable.*

Proof. If the above result does not hold, then we can define a procedure that solves BCQ entailment for datalog-rewritable inputs analogous to the one discussed in Corollary 14. This clashes with Theorem 16 to be proven in the next section, and hence, this theorem follows by contradiction. □

4 Entailment for Datalog-Expressible Queries is Undecidable

Our only goal in this section is to show that:

Theorem 16. *There is no procedure to check if a knowledge base $\langle \mathcal{R}, \mathcal{F} \rangle$ entails a BCQ q that is sound, complete, and terminates if $\langle \mathcal{R}, q \rangle$ is datalog-expressible.*

Here is our high-level strategy to prove this result:

⊡ In Definition 18, we introduce a reduction that takes a machine M as input and produces the rule set $\mathcal{R}_M$.

⚀ We show that a machine M halts on the empty word ε if and only if $\langle \mathcal{R}_M, \emptyset \rangle \models \texttt{Halt}$ where $\texttt{Halt}$ is a nullary predicate occurring in $\mathcal{R}_M$; see Lemma 22. The knowledge base $\langle \mathcal{R}_M, \emptyset \rangle$ will be denoted by $\mathcal{K}_M$.

⚁ We show that, if a machine M does not halt on ε, then the rule query $\langle \mathcal{R}_M^{\forall}, \texttt{Halt} \rangle$ is a datalog-rewriting of $\langle \mathcal{R}_M, \texttt{Halt} \rangle$;[1] see Lemma 24.

Once we establish all the above claims, we can readily prove the main result in this section:

Proof of Theorem 16. First, we prove that the rule query $\langle \mathcal{R}_M, \texttt{Halt} \rangle$ is datalog-expressible for any given machine M. If M halts on ε, then $\langle \mathcal{R}_M, \emptyset \rangle \models \texttt{Halt}$ by ⚀. Therefore, $\langle \mathcal{R}_M, \mathcal{F} \rangle \models \texttt{Halt}$ for every fact set $\mathcal{F}$ since first-order logic entailment is monotonic, and the rule query $\langle \{\rightarrow \texttt{Halt}\}, \texttt{Halt} \rangle$ is a valid datalog-rewriting of $\langle \mathcal{R}_M, \texttt{Halt} \rangle$ in this case. Otherwise, we conclude that $\langle \mathcal{R}_M, \texttt{Halt} \rangle$ is datalog-expressible by ⚁.

Now suppose for a contradiction that there is a procedure such as the one discussed in Theorem 16. Given a machine M, we can first compute $\mathcal{R}_M$ and then use such a procedure to effectively decide if $\langle \mathcal{R}_M, \emptyset \rangle \models \texttt{Halt}$ since $\langle \mathcal{R}_M, \texttt{Halt} \rangle$ is datalog-expressible. Moreover, we can use the result of this check to verify if M halts on ε by ⚀. Note the clash with Proposition 8. □

We address Points ⊡, ⚀, and ⚁ separately in each of followin subsections.

$$\rightarrow \exists s.\texttt{q}_\texttt{s}(s,s) \qquad (R^{\texttt{q}_\texttt{s}})$$
$$\texttt{q}_\texttt{s}(x,x) \rightarrow \texttt{RS}(x) \qquad (R^{\texttt{RS}}_{\texttt{q}_\texttt{s}})$$
$$\texttt{N}(\ell_i,\ell_{i+1}) \rightarrow \texttt{N}^+(\ell_i,\ell_{i+1}) \qquad (R^{\texttt{N}^+}_{init})$$
$$\texttt{N}(\ell_i,\ell_{i+1}),\texttt{N}^+(\ell_{i+1},\ell_j) \rightarrow \texttt{N}^+(\ell_i,\ell_j) \qquad (R^{\texttt{N}^+}_{tr})$$
$$\texttt{q}_\texttt{s}(\ell_1,\ell_1),\texttt{N}^+(\ell_1,\ell_i) \rightarrow \texttt{RS}(\ell_i) \qquad (R^{\texttt{RS}})$$
$$\texttt{RS}(x) \rightarrow \texttt{B}(x,x) \qquad (R^{\texttt{B}}_{\texttt{RS}})$$
$$\texttt{q}_\texttt{f}(p,c) \rightarrow \texttt{Halt} \qquad (R^{\texttt{Halt}}_{final})$$

$$\texttt{Eq}(x,y) \rightarrow \texttt{Eq}(y,x) \qquad (R^{\texttt{Eq}}_{sym})$$
$$\texttt{Eq}(x,y),\texttt{Eq}(y,z) \rightarrow \texttt{Eq}(x,z) \qquad (R^{\texttt{Eq}}_{tr})$$

$$\texttt{Chaos} \rightarrow \texttt{Halt} \qquad (R^{\texttt{Halt}}_{\texttt{Chaos}})$$
$$\texttt{N}^+(\ell_i,\ell_i) \rightarrow \texttt{Chaos} \qquad (R^{\texttt{Ch}}_{loop})$$
$$\texttt{N}^+(\ell_{-i},\ell_1),\texttt{q}_\texttt{s}(\ell_1,\ell_1) \rightarrow \texttt{Chaos} \qquad (R^{\texttt{Ch}}_{\texttt{N}^+\cdot q_s})$$
$$\texttt{q}_\texttt{f}(p,c_i),\texttt{N}^+(c_i,c_j) \rightarrow \texttt{Chaos} \qquad (R^{\texttt{Ch}}_{q_f\cdot\texttt{N}^+})$$
$$\texttt{N}(\ell_i,\ell_{i+1}),\texttt{N}(\ell_i,\ell'_{i+1}) \rightarrow \texttt{Eq}(\ell_{i+1},\ell'_{i+1}) \qquad (R^{\texttt{Eq}}_{<2\texttt{N}})$$
$$\texttt{N}(\ell_i,\ell_{i+1}),\texttt{N}(\ell'_i,\ell_{i+1}) \rightarrow \texttt{Eq}(\ell_i,\ell'_i) \qquad (R^{\texttt{Eq}}_{<2\texttt{N}^-})$$
$$\texttt{q}_\texttt{s}(\ell_1,\ell'_1),\texttt{q}_\texttt{s}(\ell''_1,\ell'''_1) \rightarrow \texttt{Eq}(\ell_1,\ell'''_1) \qquad (R^{\texttt{Eq}}_{<2q_s})$$

Figure 1: Some of the Rules in $\mathcal{R}_M$

4.1 Point ⊡: The Reduction

Before presenting our reduction in Definition 18, we define the signature of the output rule set:

Definition 17. *For a machine $M = \langle Q, \delta \rangle$; let* $\mathsf{Preds}(M)$ *be the set of predicates that contains the nullary predicates* $\texttt{Halt}$ *and* $\texttt{Chaos}$*; the unary predicate* $\texttt{RS}$*; the binary predicates* $\texttt{N}$*,* $\texttt{N}^+$*,* $\texttt{0}$*,* $\texttt{1}$*,* $\texttt{B}$*, and* $\texttt{Eq}$*; and a fresh binary predicate* $\texttt{q}$ *unique for every $q \in Q$.*[2]

Before reading ahead, remember that rules are function- and constant-free by Definition 1. Hence, all terms occurring in these formulas are variables.

Definition 18. *For a machine $M = \langle Q, \delta \rangle$, let $\mathcal{R}_M$ be the rule set over the predicates in* $\mathsf{Preds}(M)$ *that includes $\mathcal{R}_M^{Em}$, $\mathcal{R}_M^{Eq}$, and $\mathcal{R}_M^{Ch}$, which are defined below. In these definitions, we write $\langle x, y \rangle^+_{\texttt{RS}}$ for some variables x and y as a shortcut for the conjunction* $\texttt{RS}(x), \texttt{N}^+(x,y), \texttt{RS}(y)$.

The rule set $\mathcal{R}_M^{Em}$ contains the first seven rules in Figure 1 plus all of the following:

- *For every $q \in Q \setminus \{q_f\}$, we add:*

$$\texttt{q}(p,c), \texttt{RS}(c) \rightarrow \exists n.\texttt{N}(c,n) \qquad (R^{\texttt{N}}_q)$$

[1] Remember that $\mathcal{R}_M^{\forall}$ is the set of all datalog rules in $\mathcal{R}_M$. We introduced this notation in Section 2.2.

[2] Predicate names $\texttt{RS}$, $\texttt{N}$, and $\texttt{Eq}$ are shortcuts for "reachable from start state", "next", and "equality", respectively.

- *For every* $\langle q, a\rangle \mapsto \langle r, b, L\rangle \in \delta$*, we add:*

$$\mathtt{q}(p_i, c_j), \mathtt{a}(p_i, c_j), \mathtt{N}(p_{i-1}, p_i), \mathtt{N}(c_j, c_{j+1}), \langle p_i, c_{j+1}\rangle^{+}_{\mathtt{RS}} \rightarrow \mathtt{r}(p_{i-1}, c_{j+1}), \mathtt{b}(p_i, c_{j+1}) \quad (R^{r,b,L}_{q,a})$$

$$\mathtt{q}(p_1, c_j), \mathtt{a}(p_1, c_j), \mathtt{q_s}(p_1, p_1), \mathtt{N}(c_j, c_{j+1}), \langle p_1, c_{j+1}\rangle^{+}_{\mathtt{RS}} \rightarrow \mathtt{r}(p_1, c_{j+1}), \mathtt{b}(p_1, c_{j+1}) \quad (R^{r,b,S}_{q,a})$$

- *For every* $\langle q, a\rangle \mapsto \langle r, b, R\rangle \in \delta$*, we add:*

$$\mathtt{q}(p_i, c_j), \mathtt{a}(p_i, c_j), \mathtt{N}(p_i, p_{i+1}), \mathtt{N}(c_j, c_{j+1}), \langle p_{i+1}, c_{j+1}\rangle^{+}_{\mathtt{RS}} \rightarrow \mathtt{r}(p_{i+1}, c_{j+1}), \mathtt{b}(p_i, c_{j+1}) \quad (R^{r,b,R}_{q,a})$$

$$\mathtt{q}(\ell_i, \ell_i), \mathtt{a}(\ell_i, \ell_i), \mathtt{N}(\ell_i, \ell_{i+1}), \mathtt{RS}(\ell_{i+1}) \rightarrow \mathtt{r}(\ell_{i+1}, \ell_{i+1}), \mathtt{b}(\ell_i, \ell_{i+1}) \quad (S^{r,b,R}_{q,a})$$

- *For every* $q \in Q \setminus \{q_f\}$ *and* $a \in \{0, 1, B\}$*, we add:*

$$\mathtt{a}(p_i, c_j), \mathtt{q}(p_k, c_j), \mathtt{N}(c_j, c_{j+1}), \mathtt{N}^{+}(p_i, p_k), \langle p_k, c_{j+1}\rangle^{+}_{\mathtt{RS}}, \langle p_i, c_{j+1}\rangle^{+}_{\mathtt{RS}} \rightarrow \mathtt{a}(p_i, c_{j+1}) \quad (R^{a\cdot L}_{copy})$$

$$\mathtt{q}(p_i, c_j), \mathtt{a}(p_k, c_j), \mathtt{N}(c_j, c_{j+1}), \mathtt{N}^{+}(p_i, p_k), \langle p_k, c_{j+1}\rangle^{+}_{\mathtt{RS}}, \langle p_i, c_{j+1}\rangle^{+}_{\mathtt{RS}} \rightarrow \mathtt{a}(p_k, c_{j+1}) \quad (R^{a\cdot R}_{copy})$$

The rule set $\mathcal{R}^{Eq}_M$ *contains the two rules in the middle of Figure 1 plus the following for every* $\mathtt{P} \in \mathsf{Preds}(M) \setminus \{\mathtt{Eq}, \mathtt{Chaos}, \mathtt{Halt}\}$ *and* $1 \leq i \leq \mathsf{ar}(P)$*:*

$$\mathtt{P}(x_1, \ldots, x_n) \rightarrow \mathtt{Eq}(x_i, x_i) \quad (R^{\mathtt{Eq}}_{\mathtt{P},i})$$

$$\mathtt{P}(x_1, \ldots, x_n), \mathtt{Eq}(x_i, y_i) \rightarrow \mathtt{P}(x_1, \ldots, x_{i-1}, y_i, x_{i+1}, \ldots, x_n) \quad (R^{\mathtt{P},i}_{cg})$$

The rule set $\mathcal{R}^{Ch}_M$ *contains the last seven rules in Figure 1, and includes all of the following rule sets:*

$$\{\mathtt{q}(p, c), \mathtt{q}(p', c) \rightarrow \mathtt{Eq}(p, p') \mid q \in Q\} \quad (R^{\mathtt{Eq}}_{q})$$

$$\{\mathtt{q}(p, c), \mathtt{r}(p', c) \rightarrow \mathtt{Chaos} \mid q \neq r \text{ in } Q\} \quad (R^{\mathtt{Ch}}_{q\neq r})$$

$$\{\mathtt{s}(p, c), \mathtt{N}^{+}(c, p) \rightarrow \mathtt{Chaos} \mid s \in \{0, 1, B\} \cup Q\} \quad (R^{\mathtt{Ch}}_{s\cdot \mathtt{N}^{+}})$$

$$\{\mathtt{a}(p, c), \mathtt{b}(p, c) \rightarrow \mathtt{Chaos} \mid a \neq b \text{ in } \{0, 1, B\}\} \quad (R^{\mathtt{Ch}}_{a\neq b})$$

To ease the understanding, we first provide a high-level explanation for each of the rule sets defined in Definition 3; more detailed explanations specific to each rule will follow in Sections 4.2 and 4.3.

For a machine M, the rule set $\mathcal{R}^{Em}_M$ ensures that the result of the chase on $\langle \mathcal{R}_M, \emptyset\rangle$ encodes the information in the run of M on the empty word ε. Put differently, this rule set ensures that the chase on $\langle \mathcal{R}_M, \emptyset\rangle$ becomes a procedure that closely emulates the computation of M on this specific input. This rule set is prominently used in the following section to show Lemma 22.

The rule set $\mathcal{R}^{Eq}_M$ ensures that Eq, which is just a regular binary predicate, behaves as an axiomatization of first-order equality; that is, of the special predicate $\approx$. In particular, note the rules of type $R^{\mathtt{P},i}_{cg}$, which ensure that Eq defines congruent relation in the result of the chase. The other rules in this rule set ensure that it defines an equivalence relation.

The rule set $\mathcal{R}^{Ch}_M$ allows us to derive Halt in many cases only using datalog rules. For example, using these rules we can readily show that Halt is in the result of the chase on a knowledge base $\langle \mathcal{R}^{\forall}_M, \mathcal{F}\rangle$ if $\mathcal{F}$ is a fact set that includes an N-cycle; that is, a non-empty set of facts of the form $\{\mathtt{N}(d_1, d_2), \ldots, \mathtt{N}(d_{n-1}, d_n), \mathtt{N}(d_n, d_1)\}$. Thus, we can show that $\langle \mathcal{R}_M, \mathcal{F}\rangle \models \mathtt{Halt}$ if and only if $\langle \mathcal{R}^{\forall}_M, \mathcal{F}\rangle \models \mathtt{Halt}$ for such a fact set $\mathcal{F}$. In turn, this greatly simplifies the proof of Lemma 24.

4.2 Point ⚀: Machine Emulation

Our only goal in this subsection is to show that $\mathcal{R}_M$ correctly simulates M on the empty word (Lemma 22). Let us clarify this intuition with the following example.

Example 19. *Consider a state q and a machine $M = \langle Q, \delta\rangle$ such that $Q = \{q_s, q_f, q\}$, $\delta(q_s, B) = \langle q, 1, R\rangle$, and $\delta(q, B) = \langle q_f, 0, L\rangle$. We depict $\mathsf{Run}_M(\varepsilon)$ and the chase of $\langle \mathcal{R}_M, \emptyset\rangle$ in Figure 2; note how the latter structure encodes the information in the former.*

For instance, the alphabet symbol 1 *occurs in the second position of the tape in the third configuration of $\mathsf{Run}_M(\varepsilon)$; this information is encoded in $\mathsf{Ch}(\mathcal{K}_M)$ with the atom $\mathtt{1}(n(s), n^2(s))$.*[3] *The alphabet symbol* 1 *is encoded using the corresponding binary predicate* 1*; the fact that this symbol occurs in the second position of the tape in the third configuration is encoded using the second $n(s)$ and third $n^2(s)$ elements of the* N*-chain, respectively. In other words, the first and second arguments of an* 1*-atom encode the information about the position and configuration of some occurrence of the alphabet symbol* 1*, respectively. Note that this intuition is reflected in the naming of the variables appearing in rules, where p stands for position and c stands for configuration.*

Information pertaining to the position of the head as well as the state of the machine is analogously encoded using binary atoms over q*,* $\mathtt{q_s}$*, and* $\mathtt{q_f}$*. For instance, the atom $\mathtt{q_f}(s, n^2(s)) \in \mathsf{Ch}(\mathcal{K}_M)$ is used to encode that, in the third configuration, the machine enters the final state q_f and its head is in the first position of the tape. Finally, note that the nullary fact* Halt *is in $\mathsf{Ch}(\mathcal{K}_M)$ since this set contains an atom over the predicate* $\mathtt{q_f}$*.*

After the previous example, we can now elucidate the purpose of the rules in the rule set $\mathcal{R}^{Em}_M$ for a given machine M. Roughly speaking, we can divide this rule set into five mutually disjoint partitions:

- First, we have the first six rules in Figure 1 as well as the rules of type $R^{\mathtt{N}}_q$, which instantiate the N-chain in $\mathsf{Ch}(\mathcal{K}_M)$. Moreover, these rules also materialize the atoms encoding the starting configuration on the empty word as well as every new blank symbol at the end of every subsequent configuration.
- Second, we have all rules of type $R^{r,b,L}_{q,a}$, $R^{r,b,S}_{q,a}$, $R^{r,b,R}_{q,a}$, or $S^{r,b,R}_{q,a}$. These rules instantiate the only rewritten alphabet symbol in every non-starting configuration, and update the state of the machine and move the head to the left ($R^{r,b,L}_{q,a}$ and $R^{r,b,S}_{q,a}$) or right ($R^{r,b,R}_{q,a}$ and $S^{r,b,R}_{q,a}$).

[3] Remember that $n^2(s)$ is a shortcut for the unary term $f_n(f_n(f_s))$. We introduced this notation in Section 2.2.

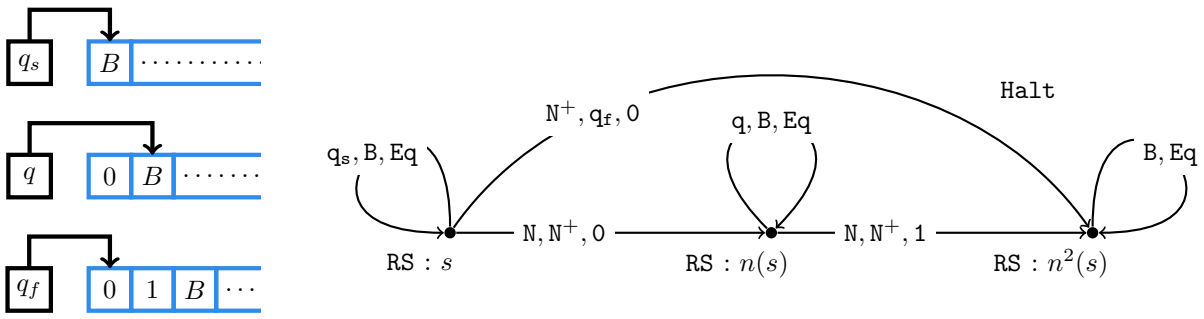

Figure 2: The Sequence $\mathsf{Run}_M(\varepsilon)$ and the Chase of $\langle \mathcal{R}_M, \emptyset \rangle$ where M is the machine from Example 19

- Fourth, the rules of type $R^{a \cdot L}_{copy}$ and $R^{a \cdot R}_{copy}$ take care of copying the alphabet symbols that are to the left and to the right of the head to every subsequent configuration, respectively. Note the use of the predicate $\mathtt{N}^+$ to detect if a position for a given configuration is to the left or the right.
- Finally, Rule $R^{\mathtt{Halt}}_{final}$, which produces $\mathtt{Halt}$ if the machine reaches the final configuration in the run.

All other rules in $\mathcal{R}_M$ are mostly inactive during the computation of the chase on $\langle \mathcal{R}_M, \emptyset \rangle$. Namely, these rules solely derive facts of the form $\mathtt{Eq}(t, t)$ and thus have no impact on deriving $\mathtt{Halt}$ when we only consider the knowledge base $\langle \mathcal{R}_M, \emptyset \rangle$ with the empty set.

We are ready to formalize the intuition presented in Example 19. Namely, for a machine M, we first define some atom sets that encode the information in every configuration in $\mathsf{Run}_M(\varepsilon)$, and then show that the union of all of these sets is the result of the chase on $\langle \mathcal{R}_M, \emptyset \rangle$.

Definition 20. *Consider a machine M, some $1 \leq k \leq |\mathsf{Run}_M(\varepsilon)|$, and the k-th configuration $\langle k, t, p, q \rangle$ of $\mathsf{Run}_M(\varepsilon)$. Let $\mathcal{C}_\mathcal{M}(k)$ be the atom set that includes $\mathcal{C}_\mathcal{M}(k-1)$ if $k > 1$, contains $\mathtt{Halt}$ if $q = q_f$, and includes*

$$\{\mathtt{N}(n^i(s), n^{i+1}(s)) \mid 0 \leq i < k-1\} \cup$$
$$\{\mathtt{N}^+(n^i(s), n^j(s)) \mid 0 \leq i < j < k\} \cup$$
$$\{\mathtt{RS}(n^i(s)), \mathtt{Eq}(n^i(s), n^i(s)) \mid 0 \leq i < k\} \cup$$
$$\{\mathsf{Pred}(t(i+1))(n^i(s), n^{k-1}(s)) \mid 0 \leq i < k\}$$

where $\mathsf{Pred}(0) = \mathtt{0}$, $\mathsf{Pred}(1) = \mathtt{1}$, and $\mathsf{Pred}(B) = \mathtt{B}$. Moreover, let $\mathcal{C}_\mathcal{M}$ be the atom set that includes $\mathcal{C}_\mathcal{M}(i)$ for every $1 \leq i \leq |\mathsf{Run}_M(\varepsilon)|$.

Lemma 21. *For a machine M, we have $\mathsf{Ch}(\mathcal{K}_M) = \mathcal{C}_\mathcal{M}$.*

Sketch. First, we verify that $\mathsf{Ch}_{2j+2}(\mathcal{R}_M) = \mathcal{C}_\mathcal{M}(j)$ for every $1 \leq j \leq |\mathsf{Run}_M(\varepsilon)|$ by induction on j. Furthermore, we show that, if M halts on the empty word ε, then $\mathsf{Ch}(\mathcal{K}_M) = \mathsf{Ch}_{2|\mathsf{Run}_M(\varepsilon)|+2}(\langle \mathcal{R}_M, \emptyset \rangle)$. □

For a machine M, the previous lemma establishes the formal correspondence between the information in $\mathsf{Run}_M(\varepsilon)$ and the result of the chase of $\langle \mathcal{R}_M, \emptyset \rangle$. Applying this result, we can readily show the main result of this subsection.

Lemma 22. *A machine M halts on the empty word if and only if the knowledge base $\langle \mathcal{R}_M, \emptyset \rangle$ entails $\mathtt{Halt}$.*

Proof. If $\langle \mathcal{R}_M, \emptyset \rangle$ entails $\mathtt{Halt}$ for some machine M, then the nullary atom $\mathtt{Halt}$ is in the result of the chase on $\langle \mathcal{R}_M, \emptyset \rangle$ by Proposition 3. Therefore, $\mathtt{Halt}$ is in $\mathcal{C}_\mathcal{M}(k)$ for some $1 \leq k \leq |\mathsf{Run}_M(\varepsilon)|$ by Lemma 21. By Definition 20, this is the case only if the k-th configuration in $\mathsf{Run}_M(\varepsilon)$ features the final state q_f.

If a machine M halts on ε, then the nullary atom $\mathtt{Halt}$ is in $\mathcal{C}_\mathcal{M}(|\mathsf{Run}_M(\varepsilon)|)$ by Definition 20. Hence, $\mathtt{Halt}$ is in the result of the chase on $\langle \mathcal{R}_M, \emptyset \rangle$ by Lemma 21. □

If our only goal was to show Lemma 22, we could reuse classical machine simulations relying on a grid structure, as done for instance in (Baget et al. 2011). However, to prove datalog-expressibility of $\langle \mathcal{R}_M, \mathtt{Halt} \rangle$, it is convenient to have a simpler structure of Skolem terms: $\mathcal{R}_M$ has only two existential rules, $R^{\mathtt{q_s}}$ of empty frontier and $R^{\mathtt{N}}_q$ of frontier one. This facilitates the detection (through datalog rules) of fact sets that either do not encode a valid run of a machine.

4.3 Point ⚂: Datalog-Expressibility

For the remainder of the subsection, we fix a machine $M = \langle Q, \delta \rangle$ and a fact set $\mathcal{F}$. Moreover, we define another fact set that results from collapsing every set of constants in $\mathsf{Ch}(\mathcal{R}^\forall_M, \mathcal{F})$ mutually interconnected by $\mathtt{Eq}$.

Definition 23. *Since rules $R^{\mathtt{Eq}}_{\mathtt{P},i}$, $R^{\mathtt{Eq}}_{sym}$, and $R^{\mathtt{Eq}}_{tr}$ are in $\mathcal{R}^\forall_M$, the $\mathtt{Eq}$ predicate defines an equivalence relation in $\mathsf{Ch}(\mathcal{R}^\forall_M, \mathcal{F})$. For a constant c occurring in this atom set, let $[c]$ be a fresh constant unique for the equivalence class of c induced by $\mathtt{Eq}$. Let $\mathcal{F}^{\mathtt{Eq}}_M = \{\mathtt{P}([c_1], \ldots, [c_n]) \mid \mathtt{P}(c_1, \ldots, c_n) \in \mathsf{Ch}(\mathcal{R}^\forall_M, \mathcal{F})\}$.*

The fact set $\mathcal{F}^{\mathtt{Eq}}_M$ is isomorphic to the chase of $\langle \mathcal{R}^\forall_M, \mathcal{F} \rangle$ if we had used first-order equality in $\mathcal{R}_M$ instead of an axiomatization of this special predicate. If $\mathtt{Chaos} \notin \mathcal{F}^{\mathtt{Eq}}_M$, then this fact set has a clean structure that is quite useful in our proofs. After this short preamble, let us consider the main result of this subsection:

Lemma 24. *If M does not halt on ε, then $\langle \mathcal{R}^\forall_M, \mathtt{Halt} \rangle$ is a datalog-rewriting of $\langle \mathcal{R}_M, \mathtt{Halt} \rangle$*

Proof. Since $\mathcal{R}_M$ includes $\mathcal{R}^\forall_M$ and first-order entailment is monotonic, we conclude that $\langle \mathcal{R}^\forall_M, \mathcal{F} \rangle \models \mathtt{Halt}$ implies $\langle \mathcal{R}_M, \mathcal{F} \rangle \models \mathtt{Halt}$. If the premise of the lemma holds, then

we conclude that $\langle \mathcal{R}_M^\forall, \mathcal{F}\rangle \not\models \texttt{Halt}$ implies $\langle \mathcal{R}_M, \mathcal{F}\rangle \not\models \texttt{Halt}$ from Lemmas 25 and 33. □

Because of the rules in $\mathcal{R}_M^{Eq}$, the predicate Eq behaves as an axiomatization of equality in the chase of $\langle \mathcal{R}_M^\forall, \mathcal{F}\rangle$. Hence, when we collapse constants interconnected via Eq to create $\mathcal{F}_M^{\texttt{Eq}}$, we preserve entailment of nullary facts and do not violate existing datalog rules.

Lemma 25. *We have that $\langle \mathcal{R}_M, \mathcal{F}_M^{\texttt{Eq}}\rangle \not\models \texttt{Halt}$ if and only $\langle \mathcal{R}_M, \mathcal{F}\rangle \not\models \texttt{Halt}$. Moreover, $\mathcal{F}_M^{\texttt{Eq}}$ satisfies all rules in $\mathcal{R}_M^\forall$ (and therefore, $\mathsf{Ch}(\mathcal{R}_M^\forall, \mathcal{F}_M^{\texttt{Eq}}) = \mathcal{F}_M^{\texttt{Eq}}$).*

Intuitively, the above lemma implies that we can disregard $\mathcal{F}$ and focus on $\mathcal{F}_M^{\texttt{Eq}}$ for the remainder of the section. Now, let us explore the structure of the latter, which is not evident in the former. Namely, we show that, if $\texttt{Chaos} \notin \mathcal{F}_M^{\texttt{Eq}}$, then the set of all constants in $\mathcal{F}_M^{\texttt{Eq}}$ can be partitioned into a disjoint union of N-chains.

Definition 26. *An* N-chain *C is a non-empty (finite) list $c_1, \ldots, c_n$ of constants such that $c_i \neq c_j$ for every $1 \leq i < j \leq n$; $\texttt{N}(c_i, c_{i+1}) \in \mathcal{F}_M^{\texttt{Eq}}$ for every $1 \leq i \leq n-1$; and, if there is a fact of the form $\texttt{N}(d, e) \in \mathcal{F}_M^{\texttt{Eq}}$ such that either d or e occur in C, then $d = c_i$ and $e = c_{i+1}$ for some $1 \leq i \leq n-1$. A* starting chain *is an* N-chain *$c_1, \ldots, c_n$ such that $\texttt{q}_\texttt{s}(c_1, c_1) \in \mathcal{F}_M^{\texttt{Eq}}$.*

Lemma 27. *If $\texttt{Chaos} \notin \mathcal{F}_M^{\texttt{Eq}}$; then the set of all constants occurring in $\mathcal{F}_M^{\texttt{Eq}}$ is a disjoint union of chains, there is at most one $\texttt{q}_\texttt{s}$-atom in $\mathcal{F}_M^{\texttt{Eq}}$ (and hence is at most one starting chain), and there is one starting chain if and only if $\mathcal{F}_M^{\texttt{Eq}}$ contains some $\texttt{q}_\texttt{s}$-atom.*

Sketch. The first, second, and third implications hold because $\mathcal{R}_M^\forall$ includes $\{R_{init}^{\texttt{N}^+}, R_{tr}^{\texttt{N}^+}, R_{loop}^{\texttt{Ch}}, R_{<2\texttt{N}}^{\texttt{Eq}}, R_{<2\texttt{N}^-}^{\texttt{Eq}}\}$, $\{R_{<2q_s}^{\texttt{Eq}}\}$, and $\{R_{init}^{\texttt{N}^+}, R_{\texttt{N}^+\cdot q_s}^{\texttt{Ch}}, R_{<2q_s}^{\texttt{Eq}}\}$, respectively. Note that $\mathcal{F}_M^{\texttt{Eq}}$ satisfies all rules in $\mathcal{R}_M^\forall$ by Lemma 25, and $c = d$ if $\texttt{Eq}(c, d) \in \mathcal{F}_M^{\texttt{Eq}}$ for some constants c and d. □

Furthermore, we can show that, if there is a starting chain, then the fact set associated to this structure is isomorphic to a finite prefix of the chain in $\mathsf{Ch}(\mathcal{K}_M)$.

Definition 28. *Let π_S be the (injective) substitution that maps s to the first element of (the starting chain) S, $n(s)$ to the second, $n^2(s)$ to the third, and so on. For a list $\vec{t}$ of terms, let $\mathsf{Facts}(\vec{t})$ be the set of all facts in $\mathcal{F}_M^{\texttt{Eq}}$ that can be defined using some non-nullary predicate and some terms occurring in $\vec{t}$.*

Remember that, since we slightly abuse notation, substitutions may be defined for non-variable terms.

Lemma 29. *If $\texttt{Chaos} \notin \mathcal{F}_M^{\texttt{Eq}}$ and there is a starting chain S, then the substitution π_S is an isomorphism from $\mathcal{C}_\mathcal{M}(|S|)$ to $\mathsf{Facts}(S)$.*[4]

[4]Remember that $\mathcal{C}_\mathcal{M}(|S|)$ was introduced in Definition 20.

Sketch. We can use an argument analogous to the one in the proof of Lemma 21 to show that π_S is a homomorphism from $\mathcal{C}_\mathcal{M}(|S|)$ to $\mathsf{Facts}(S)$. Then, for the other direction, we use Rules $R_{loop}^{\texttt{Ch}}$, $R_{\texttt{N}^+\cdot q_s}^{\texttt{Ch}}$, $R_{q_f\cdot\texttt{N}^+}^{\texttt{Ch}}$, $R_{<2\texttt{N}}^{\texttt{Eq}}$, $R_{<2\texttt{N}^-}^{\texttt{Eq}}$, $R_{<2q_s}^{\texttt{Eq}}$, $R_{q\neq r}^{\texttt{Ch}}$, $R_q^{\texttt{Eq}}$, $R_{a\neq b}^{\texttt{Ch}}$ and $R_{s\cdot\texttt{N}^+}^{\texttt{Ch}}$ to show that $\mathsf{Facts}(S)$ only contains atoms in $\pi_S(\mathcal{C}_\mathcal{M}(|S|))$. For instance, if an atom of the form $\texttt{N}^+(c, d)$ is in $\mathsf{Facts}(S)$ but not in $\pi_S(\mathcal{C}_\mathcal{M}(|S|))$, then either $c = d$, or c comes after d in the starting chain.

- If $c = d$, then $R_{loop}^{\texttt{Ch}}$ implies that $\texttt{Chaos} \in \mathsf{Facts}(S)$, which contradicts the premise of the lemma.
- If c comes after d in the starting chain, then there is a list $c_1, \ldots, c_n$ of constants such that $c_1 = d$, $c_n = c$, and $\texttt{N}(c_i, c_{i+1}) \in \mathsf{Facts}(S)$ for every $1 \leq i < n$. Then, $\texttt{N}^+(d, d) \in \mathsf{Facts}(S)$ because of Rule $R_{tr}^{\texttt{N}^+}$, and $\texttt{Chaos} \in \mathsf{Facts}(S)$ because of Rule $R_{loop}^{\texttt{Ch}}$, which contradicts the premise of the lemma.

Thus, there are no $\texttt{N}^+$-atom in $\mathsf{Facts}(S) \setminus \pi_S(\mathcal{C}_\mathcal{M}(|S|))$. A careful analysis of all the other predicates then concludes the proof. □

Now that we understand the structure and content of $\mathcal{F}_M^{\texttt{Eq}}$, we can declaratively describe how to extend this fact set to obtain the chase of $\langle \mathcal{R}_M, \mathcal{F}_M^{\texttt{Eq}}\rangle$.

Definition 30. *An* N-chain *$C = c_1, \ldots, c_m$ is* incomplete *if the set $\mathcal{F}_M^{\texttt{Eq}}$ contains $\texttt{RS}(c_m)$ and some fact of the form $\texttt{q}(d, c_m)$ such that $q \neq q_f$. Given such an* N-chain*, let $\mathsf{Compl}(C)$ be the atom set that contains $\texttt{N}(c_m, n(c_m))$, $\texttt{Eq}(n(c_m), n(c_m))$, and $\texttt{N}^+(c_i, n(c_m))$ for every $1 \leq i \leq m$. Let $\mathsf{Compl}(\mathcal{F}_M^{\texttt{Eq}})$ be the atom set that includes $\mathsf{Compl}(C)$ for every incomplete chain C.*

Lemma 31. *If $\texttt{Chaos} \notin \mathcal{F}_M^{\texttt{Eq}}$, then*

$$\mathsf{Ch}(\mathcal{R}_M, \mathcal{F}_M^{\texttt{Eq}}) = \mathcal{F}_M^{\texttt{Eq}} \cup \mathsf{Compl}(\mathcal{F}_M^{\texttt{Eq}}) \cup \pi_S(\mathsf{Ch}(\mathcal{K}_M)).$$

Roughly speaking, if $\texttt{Chaos} \notin \mathcal{F}_M^{\texttt{Eq}}$, we can define the chase of $\langle \mathcal{R}_M, \mathcal{F}_M^{\texttt{Eq}}\rangle$ by taking $\mathcal{F}_M^{\texttt{Eq}}$, completing every incomplete chain, and appending the (possibly infinite) chain in $\mathsf{Ch}(\mathcal{K}_M)$ to the starting chain in $\mathcal{F}_M^{\texttt{Eq}}$. To understand why this suffices, see the following example.

Example 32. *Consider a machine that has three states, q_s, q and q_f, that always goes right, and that goes to state q whenever it reads a blank, and goes to q_f whenever it reads a* 0 *or a* 1*. Obviously, this machine does not halt on the empty word, but does on every other word. The fact set on the left of Figure 3 contains two* N*-chains: the starting chain at the top, and another at the bottom. After one existential step and one datalog step, note that the atom $\texttt{q}_\texttt{f}(n(d), n(d))$ has not been derived: this is due to the fact that $\texttt{RS}(n(d))$ is not derivable, as $n(d)$ is not reachable through an* N*-chain from c_0, the unique element for which $q_s(c_0, c_0)$ holds. This blocks the application of Rule $S_{q,a}^{r,b,R}$.*

If ones try to circumvent that problem as on the right of Figure 3, with two N*-chains connected by a* q *atom, the atom*

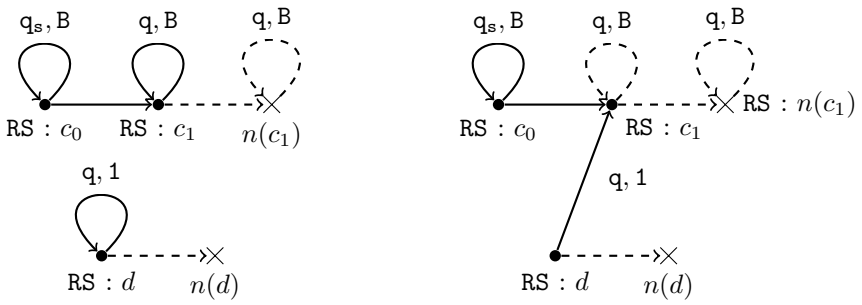

Figure 3: Fact sets from Example 32 after one existential step and one closure under datalog rules. Dots represent constants occurring in the fact set, and crosses represent Skolem terms. Solid atoms are in the original fact set, dashed ones created by rule applications. Unlabelled arrows represent the $\mathtt{N}$ predicate. Predicates $\mathtt{N}^+$ and $\mathtt{Eq}$ are not represented.

$\mathtt{N}^+(n(d), n(c_2))$ *cannot be mapped, hence Rule* $R^{r,b,R}_{q,a}$ *cannot be applied. In turn, this prevents the instantiation of the atom* $\mathtt{q_f}(n(c_2), n(c_2))$.

These two examples illustrate the use of predicates $\mathtt{RS}$ *and* $\mathtt{N}^+$*, allowing to block the development of simulations that do not correspond to the empty word.*

Leveraging the contingent definition of the chase of $\langle \mathcal{R}_M, \mathcal{F}^{\mathtt{Eq}}_M \rangle$ given in Lemma 31, we can better understand when $\mathtt{Halt}$ is not entailed by this knowledge base:

Lemma 33. *If* $\mathtt{Halt} \notin \mathcal{F}^{\mathtt{Eq}}_M$ *and* M *does not halt on* ε*, then* $\langle \mathcal{R}_M, \mathcal{F}^{\mathtt{Eq}}_M \rangle$ *does not entail* $\mathtt{Halt}$.

Proof. By Proposition 3, we have that $\langle \mathcal{R}_M, \mathcal{F}^{\mathtt{Eq}}_M \rangle$ does not entail $\mathtt{Halt}$ if $\mathtt{Halt}$ is not in the chase of $\langle \mathcal{R}_M, \mathcal{F}^{\mathtt{Eq}}_M \rangle$. By Lemma 31, this is the case if $\mathtt{Halt}$ is not in $\mathcal{F}^{\mathtt{Eq}}_M$, in $\mathsf{Compl}(\mathcal{F}^{\mathtt{Eq}}_M)$, or in $\pi_S(\mathsf{Ch}(\mathcal{K}_M))$. Indeed, $\mathtt{Halt}$ is not in the first set by the premise of the lemma, it is not in the second by Definition 30, and it is not in the third by the premise of the lemma and Lemma 22. Note that the premise of the lemma implies that $\langle \mathcal{R}_M, \mathcal{F}^{\mathtt{Eq}}_M \rangle$ does not entail $\mathtt{Chaos}$ since the Rule $R^{\mathtt{Halt}}_{\mathtt{Chaos}}$ is in $\mathcal{R}_M$. □

5 Related Work

Let us first point out that there are several variations around the notion of datalog-rewriting. A first dimension is that, rather than rewriting a specific rule query $\langle \mathcal{R}, q \rangle$, one can wish to rewrite $\mathcal{R}$ into a datalog rule set $\mathcal{R}'$, and use $\mathcal{R}'$ to compute answers for a class of queries. Another variation is focused on the fact sets on which the rewriting should output the same answer as the original rule query. In this paper, we consider the strong version where answers should be the same on every fact set over Pred. Quite often, defined datalog rewritings only preserve answers over fact sets on the original signature – this allows to introduce fresh predicates which are known not to belong to fact sets on which the datalog program is to be evaluated. There are cases for which there exists datalog-rewritings of rule queries for this relaxed definition but not for our restricted one; this is a consequence of Theorem 3 in (Krötzsch 2011). Our undecidability result implies undecidability of this more relaxed notion.

The use of datalog as a target language for rewritings has been studied over the last 15 years. The goal was to reduce reasoning task over expressive ontologies towards query answering over datalog, for which optimization techniques have been developed in the database community. This is even more pregnant today, as a variety of efficient datalog reasoners have been implemented (Nenov et al. 2015; Urbani et al. 2018). Such an approach has been proposed for providing disjunctive[5] datalog rewritings for $\mathcal{SHIQ}$ for fact entailment over the original signature (Hustadt, Motik, and Sattler 2007), later generalized to $\mathcal{SHIQ}b_S$ (Rudolph, Krötzsch, and Hitzler 2012). More recently, such reductions for Horn description logics have been implemented and evaluated (Carral, Dragoste, and Krötzsch 2018; Carral, González, and Koopmann 2019). Such datalog rewritings have also been studied for existential rules, for guarded (Benedikt et al. 2022), nearly guarded (Gottlob, Rudolph, and Simkus 2014), warded (Berger et al. 2022) and shy (Leone et al. 2019) rule sets.

Beyond these fragment specific reductions, the limits of datalog-rewritability have been explored. In (Marnette 2012), it is shown that whenever rule queries have bounded depth (meaning that if they are entailed, they are entailed by a portion of the chase that uses only Skolem terms of bounded depth), they are datalog-rewritable. This result applies for all syntactic fragment for which the chase is known to terminate (Grau et al. 2013), but datalog rewritability is not guaranteed (and not always possible) for rule sets having terminating restricted chase (Krötzsch, Marx, and Rudolph 2019) – this is proven by a data complexity argument.

Another question of interest is the size of the obtained rewritings. In (Ahmetaj, Ortiz, and Simkus 2018), the authors provide polynomial (disjunctive) datalog rewritings for (disjunctive) guarded rules queries. Non-recursive datalog has also been studied: while it does not increase the expressivity with respect to UCQs, the re-use of predicates allows to significanlty reduce the size of rewritings, reaching polynomiality in some cases (Gottlob and Schwentick 2012; Gottlob et al. 2014).

All of these contributions are summarised in Table 1.

[5]For disjunctive existential rules and datalog, the reader is invited to consult (Deutsch and Tannen 2003)

Source Language	Target Language	Arbitrary Query Sig.	Implemented	Reference
$\mathcal{SHIQ}$	disj. dat.	×	✓	(Hustadt, Motik, and Sattler 2007)
$\mathcal{SHIQ}b_S$	disj. dat.	×	×	(Rudolph, Krötzsch, and Hitzler 2012)
Horn-$\mathcal{ALCHOIQ}$	datalog	×	✓	(Carral, Dragoste, and Krötzsch 2018)
Horn-$\mathcal{SRIQ}$	datalog	×	✓	(Carral, González, and Koopmann 2019)
Bounded Detph rules	datalog	×	×	(Marnette 2012)
Frontier guarded rules	datalog	✓	×	(Bárány, Benedikt, and ten Cate 2013)
Nearly Guarded Rules	datalog	✓	×	(Gottlob, Rudolph, and Simkus 2014)
Guarded disj. rules	disj. datalog	×	×	(Ahmetaj, Ortiz, and Simkus 2018)
Guarded rules	datalog	✓	✓	(Benedikt et al. 2022)
Warded rules	datalog	✓	✓	(Berger et al. 2022)
Linear	non rec. dat.	×	×	(Gottlob and Schwentick 2012)
Sticky(-join)	non rec. dat.	×	×	(Gottlob and Schwentick 2012)

Table 1: Summary of datalog rewriting approaches applicable for fact entailment

6 Future Work

To conclude, we discuss two distinct lines for future research, which naturally follow from our work.

Answer Expressible Rule Sets We intend to study an even more restrictive class of rewritable rule sets: a rule set $\mathcal{R}$ is *answer datalog-expressible* if, for every CQ $q[\vec{x}]$, the rule query $\langle \mathcal{R}, q[\vec{x}] \rangle$ admits some *datalog-rewriting that preserves all answers*. That is, a datalog query $\langle \mathcal{R}', q'[\vec{x}] \rangle$ such that, for every fact set $\mathcal{F}$ and every list $\vec{a}$ of constants occurring in $\mathcal{F}$, we have that $\langle \mathcal{R}, \mathcal{F} \rangle \models q[\vec{x}/\vec{a}]$ if and only if $\langle \mathcal{R}', \mathcal{F} \rangle \models q[\vec{x}/\vec{a}]$. With this definition in place, we wonder about the following problem:

Open Problem 34. *Consider a knowledge base $\mathcal{K} = \langle \mathcal{R}, \mathcal{F} \rangle$, a CQ q, and a list $\vec{a}$ of constants occurring in $\mathcal{F}$. Is there a procedure to check if $\vec{a}$ is an answer of q with respect to $\langle \mathcal{R}, \mathcal{F} \rangle$ that is sound, complete, and terminating if $\mathcal{R}$ is answer datalog-expressible?*

The above question is quite relevant for our field of research, where we often study the theoretical properties of classes of rule sets and not of rule queries.

At the moment, we believe that the answer to this open problem is negative, which would yield a result strictly stronger than Theorem 16. However, a different proof strategy is required to show this since there are rule sets in the range of the reduction described in Definition 18 that are not answer datalog-rewritable.

Alternative Rewriting Languages In this paper, our primary focus is on datalog; in the future, we plan to study alternative query languages for rewritings. For instance, one could consider unions of Boolean conjunctive regular path queries (UBCRPQs) (Florescu, Levy, and Suciu 1998) and then consider the following problem:

Open Problem 35. *Is the class of all UBCRPQ-expressible queries is UBCRPQ-rewritable? Is there a procedure to check if a knowledge base $\langle \mathcal{R}, \mathcal{F} \rangle$ entails a BCQ q that is sound, complete, and terminates if the rule query $\langle \mathcal{R}, q \rangle$ is UCRPQ-expressible?*

We can instantiate different versions of this open problem by considering different output rewriting languages. For instance, we could consider as unions of (non-conjunctive) regular path queries, monadic datalog, query languages based on context-free grammars (Medeiros, Musicante, and da Costa 2022), or any of the query languages considered by (Bourhis, Krötzsch, and Rudolph 2015).

As a closing remark, note that the answers to the first and second questions in Open Problem 35 might be negative and positive, respectively. That is, it is possible that we can solve entailment for UCRPQ-expressible rule queries even if we cannot effectively compute rewritings for these. This is an exciting possibility that may lead us to the discovery of a novel kind of reasoning procedure for this expressive class of rule queries. Or perhaps future research will just result in another undecidability result, which would be strictly stronger than Theorem 16. Either way, we look forward to researching (and hopefully settling!) these questions.

References

Ahmetaj, S.; Ortiz, M.; and Simkus, M. 2018. Rewriting guarded existential rules into small datalog programs. In Kimelfeld, B., and Amsterdamer, Y., eds., *21st International Conference on Database Theory, ICDT 2018, March 26-29, 2018, Vienna, Austria*, volume 98 of *LIPIcs*, 4:1–4:24. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Artale, A.; Calvanese, D.; Kontchakov, R.; and Zakharyaschev, M. 2009. The dl-lite family and relations. *J. Artif. Intell. Res.* 36:1–69.

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

Bárány, V.; Benedikt, M.; and ten Cate, B. 2013. Rewriting guarded negation queries. In Chatterjee, K., and Sgall, J., eds., *Mathematical Foundations of Computer Science 2013 - 38th International Symposium, MFCS 2013, Klosterneuburg, Austria, August 26-30, 2013. Proceedings*, volume 8087 of *Lecture Notes in Computer Science*, 98–110. Springer.

Benedikt, M.; Buron, M.; Germano, S.; Kappelmann, K.;

and Motik, B. 2022. Rewriting the infinite chase. *Proc. VLDB Endow.* 15(11):3045–3057.

Berger, G.; Gottlob, G.; Pieris, A.; and Sallinger, E. 2022. The space-efficient core of vadalog. *ACM Trans. Database Syst.* 47(1):1:1–1:46.

Bourgaux, C.; Carral, D.; Krötzsch, M.; Rudolph, S.; and Thomazo, M. 2021. Capturing homomorphism-closed decidable queries with existential rules. In Bienvenu, M.; Lakemeyer, G.; and Erdem, E., eds., *Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning, KR 2021, Online event, November 3-12, 2021*, 141–150.

Bourhis, P.; Krötzsch, M.; and Rudolph, S. 2015. Reasonable highly expressive query languages - IJCAI-15 distinguished paper (honorary mention). In Yang, Q., and Wooldridge, M. J., eds., *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, 2826–2832. AAAI Press.

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

Calì, A.; Gottlob, G.; and Pieris, A. 2010. Query answering under non-guarded rules in datalog+/-. In Hitzler, P., and Lukasiewicz, T., eds., *Web Reasoning and Rule Systems - Fourth International Conference, RR 2010, Bressanone/Brixen, Italy, September 22-24, 2010. Proceedings*, volume 6333 of *Lecture Notes in Computer Science*, 1–17. Springer.

Carral, D.; Dragoste, I.; and Krötzsch, M. 2018. The combined approach to query answering in horn-alchoiq. In Thielscher, M.; Toni, F.; and Wolter, F., eds., *Principles of Knowledge Representation and Reasoning: Proceedings of the Sixteenth International Conference, KR 2018, Tempe, Arizona, 30 October - 2 November 2018*, 339–348. AAAI Press.

Carral, D.; González, L.; and Koopmann, P. 2019. From horn-sriq to datalog: A data-independent transformation that preserves assertion entailment. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, 2736–2743. AAAI Press.

Dawar, A., and Kreutzer, S. 2008. On datalog vs. LFP. In Aceto, L.; Damgård, I.; Goldberg, L. A.; Halldórsson, M. M.; Ingólfsdóttir, A.; and Walukiewicz, I., eds., *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part II - Track B: Logic, Semantics, and Theory of Programming & Track C: Security and Cryptography Foundations*, volume 5126 of *Lecture Notes in Computer Science*, 160–171. Springer.

Deutsch, A., and Tannen, V. 2003. Reformulation of XML queries and constraints. In Calvanese, D.; Lenzerini, M.; and Motwani, R., eds., *Database Theory - ICDT 2003, 9th International Conference, Siena, Italy, January 8-10, 2003, Proceedings*, volume 2572 of *Lecture Notes in Computer Science*, 225–241. Springer.

Feder, T., and Vardi, M. Y. 2003. Homomorphism closed vs. existential positive. In *Proceedings of the 18th IEEE Symposium on Logic in Computer Science (LICS 2003)*, 311–320.

Florescu, D.; Levy, A. Y.; and Suciu, D. 1998. Query containment for conjunctive queries with regular expressions. In Mendelzon, A. O., and Paredaens, J., eds., *Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 1-3, 1998, Seattle, Washington, USA*, 139–148. ACM Press.

Gottlob, G., and Schwentick, T. 2012. Rewriting ontological queries into small nonrecursive datalog programs. In Brewka, G.; Eiter, T.; and McIlraith, S. A., eds., *Principles of Knowledge Representation and Reasoning: Proceedings of the Thirteenth International Conference, KR 2012, Rome, Italy, June 10-14, 2012*. AAAI Press.

Gottlob, G.; Kikot, S.; Kontchakov, R.; Podolskii, V. V.; Schwentick, T.; and Zakharyaschev, M. 2014. The price of query rewriting in ontology-based data access. *Artif. Intell.* 213:42–59.

Gottlob, G.; Rudolph, S.; and Simkus, M. 2014. Expressiveness of guarded existential rule languages. In Hull, R., and Grohe, M., eds., *Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS'14, Snowbird, UT, USA, June 22-27, 2014*, 27–38. ACM.

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

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

König, M.; Leclère, M.; Mugnier, M.; and Thomazo, M. 2015. Sound, complete and minimal ucq-rewriting for existential rules. *Semantic Web* 6(5):451–475.

Krötzsch, M.; Marx, M.; and Rudolph, S. 2019. The power of the terminating chase (invited talk). In Barceló, P., and Calautti, M., eds., *22nd International Conference on Database Theory, ICDT 2019, March 26-28, 2019, Lisbon, Portugal*, volume 127 of *LIPIcs*, 3:1–3:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Krötzsch, M. 2011. Efficient rule-based inferencing for OWL EL. In Walsh, T., ed., *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, 2668–2673. IJCAI/AAAI.

Leone, N.; Manna, M.; Terracina, G.; and Veltri, P. 2019. Fast query answering over existential rules. *ACM Trans. Comput. Log.* 20(2):12:1–12:48.

Marnette, B. 2012. Resolution and Datalog rewriting under value invention and equality constraints. *CoRR* abs/1212.0254.

Medeiros, C. M.; Musicante, M. A.; and da Costa, U. S. 2022. Querying graph databases using context-free grammars. *J. Comput. Lang.* 68:101089.

Mugnier, M., and Thomazo, M. 2014. An introduction to ontology-based query answering with existential rules. In Koubarakis, M.; Stamou, G. B.; Stoilos, G.; Horrocks, I.; Kolaitis, P. G.; Lausen, G.; and Weikum, G., eds., *Reasoning Web. Reasoning on the Web in the Big Data Era - 10th International Summer School 2014, Athens, Greece, September 8-13, 2014. Proceedings*, volume 8714 of *Lecture Notes in Computer Science*, 245–278. Springer.

Nenov, Y.; Piro, R.; Motik, B.; Horrocks, I.; Wu, Z.; and Banerjee, J. 2015. Rdfox: A highly-scalable RDF store. In Arenas, M.; Corcho, Ó.; Simperl, E.; Strohmaier, M.; d'Aquin, M.; Srinivas, K.; Groth, P.; Dumontier, M.; Heflin, J.; Thirunarayan, K.; and Staab, S., eds., *The Semantic Web - ISWC 2015 - 14th International Semantic Web Conference, Bethlehem, PA, USA, October 11-15, 2015, Proceedings, Part II*, volume 9367 of *Lecture Notes in Computer Science*, 3–20. Springer.

Ostropolski-Nalewaja, P.; Marcinkowski, J.; Carral, D.; and Rudolph, S. 2022. A journey to the frontiers of query rewritability. In Libkin, L., and Barceló, P., eds., *PODS '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022*, 359–367. ACM.

Rossman, B. 2008. Homomorphism preservation theorems. *J. ACM* 55(3):15:1–15:53.

Rudolph, S., and Thomazo, M. 2015. Characterization of the expressivity of existential rule queries. In Yang, Q., and Wooldridge, M. J., eds., *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, 3193–3199. AAAI Press.

Rudolph, S., and Thomazo, M. 2016. Expressivity of datalog variants - completing the picture. In Kambhampati, S., ed., *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, 1230–1236. IJCAI/AAAI Press.

Rudolph, S.; Krötzsch, M.; and Hitzler, P. 2012. Type-elimination-based reasoning for the description logic shiqbs using decision diagrams and disjunctive datalog. *Log. Methods Comput. Sci.* 8(1).

Urbani, J.; Krötzsch, M.; Jacobs, C. J. H.; Dragoste, I.; and Carral, D. 2018. Efficient model construction for horn logic with vlog - system description. In Galmiche, D.; Schulz, S.; and Sebastiani, R., eds., *Automated Reasoning - 9th International Joint Conference, IJCAR 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14-17, 2018, Proceedings*, volume 10900 of *Lecture Notes in Computer Science*, 680–688. Springer.