Ontology-Based Query Answering Over Datalog-Expressible Rule Sets Is Undecidable

Lucas Larroque

ONTOLOGY-BASED QUERY ANSWERING OVER DATALOG-EXPRESSIBLE RULE SETS IS UNDECIDABLE

Lucas Larroque

Supervised by Michaël Thomazo and David Carral

Master Logique et Fondements de l'Informatique
Septembre 2023

Contents

Chapter 1

Introduction

To handle the ever-increasing amount of data available, and to process it efficiently, querying data has become a ubiquitous task. In some cases, one also needs to be able to reason over data. A standard use case is the harmonization of databases coming from different organizations. Indeed, to query data efficiently, it is necessary for those different organizations to agree on the vocabulary used, which may not necessarily be possible. Introducing a logical layer to simplify this harmonization is often more practical than convincing everyone to use a unique standard. This logical layer is usually called an *ontology*, giving rise to the problem of *ontology-based query answering (OBQA)*.

There are many ways to express ontologies, the most usual being description logics. However, description logics are limited to unary and binary predicates, and can only encode knowledge in tree-like structures, which restricts their expressivity. Another language that is able to express more complex structures is Datalog, but it is limited in the sense that it cannot create any new individual: all the knowledge inferred is about already known individuals. Thus, to meet both those requirements, another framework has been studied, which is known as *Existential rules*. Existential rules extend Datalog rules with the ability to create new individuals. More precisely, an existential rule is a first order formula of the form $body \rightarrow head$, where *body* and *head* are both conjunctions of atoms, and variables appearing only in the head are existentially quantified.

While existential rules have both the ability to express complex structures and to infer the existence of new individuals, this expressivity comes with a cost: reasoning becomes semi-decidable. The usual reasoning task with existential rules, *boolean conjunctive query (BCQ) entailment*, can be summarized as the following: given a database and an ontology expressed as a set of existential rules, is a given query a logical consequence of the ontology if we take into account the facts that compose the database? In order to solve it, two main approaches have been developed.

The first approach to solve BCQ entailment is the *chase*. The chase is a family of algorithms that works by materializing new facts in the database using the rules in the ontology, until a fixpoint is reached. Then to solve BCQ entailment, it is enough to evaluate the query on the saturated database using traditional query evaluation algorithms. While this approach enables efficient query answering (through the use of standard database query evaluation over a query that stayed the same), it is not always feasible to modify the database,

either for security reasons, or because of size constraints.

The other standard way to solve BCQ entailment is through *query rewriting*. The idea is that instead of adding inferred knowledge to the database, one can modify the query to take into account the dependencies in the rule set, and then evaluate this query over the initial database. When starting from BCQs, using standard techniques, one ends up with a union of conjunctive queries (UCQ) as a rewriting, though other target languages have been studied. In this report, we focus particularly on Datalog rewritings, as it is a very natural restriction of existential rules, that ensures termination of the chase (as Datalog rules cannot create new individuals). Let us define theses two kinds of rewritings. Take note however that to end up with Datalog rewritings, one not only modifies the query, but also the rule set.

Definition 1. *The couple* $(\mathcal{R}', q')$ *is a* rewriting *of a rule set* $\mathcal{R}$ *and a query* q *if* $(\mathcal{R}, D) \models q$ *if and only if* $(\mathcal{R}', D) \models q'$ *for all databases* D*. An* UCQ rewriting *of a query* q *w.r.t. a rule set* $\mathcal{R}$ *is a rewriting* $(\emptyset, q')$ *where* q' *is a union of conjunctive queries. A* Datalog *rewriting of a rule set* $\mathcal{R}$ *and a query* q *is a rewriting* $(\mathcal{R}', q')$ *where all the rules in* $\mathcal{R}'$ *are Datalog. A couple rule set-query is* UCQ-expressible *(resp.* Datalog-expressible*) if it admits a UCQ rewriting (resp. a Datalog rewriting).*

An interesting characteristic of UCQ and Datalog rewritings is that query answering using these rewritings is decidable. Indeed, a UCQ rewriting does not feature a rule set so usual database theory algorithms work, and for Datalog rewritings, the chase terminates. Thus, from any rule set and query, if one can compute a UCQ or a Datalog rewriting, then one can solve query entailment in finite time. Thus, the question of finding these rewritings is of paramount interest, and gives rise to the following problems.

Query answering over UCQ-expressible Rule Sets

Input: A database D, a rule set $\mathcal{R}$ and a query q that is UCQ-expressible w.r.t. $\mathcal{R}$

Question: Do we have $(\mathcal{R}, D) \models q$?

Theorem 2. *[3] Query answering over UCQ-expressible rule sets is decidable.*

This means that we have an effective algorithm to compute UCQ rewritings. Though, the situation is very different for Datalog rewritings.

Query answering over Datalog-expressible Rule Sets

Input: A database D, a couple rule set-query $(\mathcal{R}, q)$ that is $\mathcal{R}$-Datalog-expressible

Question: Do we have $(\mathcal{R}, D) \models q$?

Theorem 3. *Query answering over Datalog-expressible rule sets is undecidable.*

Our contribution is proving Theorem 3. To do so, we present a reduction from the halting problem to the problem of query answering over Datalog-expressible rule sets.

Chapter 2

Preliminaries

2.1 Syntax

We work in a first-order setting limited to *predicates*, *constants*, and *variables*. A *term* is a variable or a constant. To each predicate we associate a natural integer, called the *arity* of the predicate. A *P-atom* (or just an atom) is a first order formula of the form $P(t_1, \ldots, t_n)$ where P is a predicate of arity n and $t_1, \ldots, t_n$ are some terms.

In the following, we may also group variables into tuples (which are seen as sets when convenient). Let $\vec{x_1}, \ldots, \vec{x_n}$ be pairwise disjoint tuples of variables. Then, $X[\vec{x_1}, \ldots, \vec{x_n}]$ denotes a conjunction of atoms whose variables are among $\vec{x_1} \cup \cdots \cup \vec{x_n}$.

Definition 4. *An* (existential) rule *R is a first-order formula of the following form:*

$$\forall \vec{x} \forall \vec{y}.\ B[\vec{x}, \vec{y}] \rightarrow \exists \vec{z}.\ H[\vec{x}, \vec{z}]$$

where $B[\vec{x}, \vec{y}]$ and $H[\vec{x}, \vec{z}]$ are conjunctions of atoms called the body *and the* head *of the rule, respectively. The set $\vec{x}$, which is shared between the body and the head, is called the* frontier *of the rule and is denoted by frR. An existential rule where $\vec{z}$ is empty is a* Datalog rule.

For the sake of readability, we omit universal quantifiers when writing existential rules, and we denote conjunctions of atoms with commas instead of wedges. For instance, the rule $\forall x, y.\ A(x) \wedge P(x, y) \rightarrow \exists z.\ P(y, z)$ will be written $A(x), P(x, y) \rightarrow \exists z.\ P(y, z)$.

An *instance* is an existentially closed conjunction of atoms, and a *database* is an instance featuring only constants. It is usual to identify an instance with the corresponding set of atoms. For instance, the instance $\exists x, y, z.\ A(x, y) \wedge B(x, z, y)$ is identified to the set $\{A(x, y), B(x, z, y)\}$. A *knowledge base* is a tuple $\mathcal{K} = (\mathcal{R}, D)$ where $\mathcal{R}$ is a rule set and D is an instance.

Definition 5. *A* conjunctive query *is a conjunction of atoms. We call the free variables of a conjunctive query its* answer variables. *A* Boolean conjunctive query *is a conjunctive query that has no answer variable.*

Note that by this definition, a Boolean conjunctive query (BCQ) is an instance.

In this setting, homomorphisms are often denoted as sets of individual variable mappings. For example, the homomorphism from $\{x, y, z\}$ to $\{a, b\}$ such that $\sigma(x) = a$, $\sigma(y) = a$ and $\sigma(z) = b$ can be denoted by $\{x \mapsto a, y \mapsto a, z \mapsto b\}$.

Definition 6. *A* retraction *r from a set of atoms D to $D' \subseteq D$ is a homomorphism from D to D' such that $r_{|D'} = id_{D'}$ and $r(D) = D'$.*

2.2 Semantics

We rely on classical first-order logic semantics. Indeed, the following definitions can be derived from the classical ones by considering the knowledge base $(\mathcal{R}, D)$ as the theory $\mathcal{R} \cup D$ (recall that an instance is a formula).

An instance D *satisfies* (or *models*) a rule $R = B \to H$, denoted with $D \models R$, if for every homomorphism π from B to D, there is an extension $\hat{\pi}$ of π to $B \cup H$ such that $\hat{\pi}(H) \subseteq D$. A *model* $\mathcal{M}$ of a knowledge base $(\mathcal{R}, D)$ is a set of atoms such that $D \subseteq \mathcal{M}$ that satisfies all the rules in $\mathcal{R}$. A model $\mathcal{M}$ of a knowledge base $\mathcal{K}$ is *universal* if for every model $\mathcal{M}'$ of $\mathcal{K}$, there is a homomorphism from $\mathcal{M}$ to $\mathcal{M}'$. An instance D_1 *entails* an instance (or a BCQ) D_2 if there is a homomorphism from D_2 to D_1. A knowledge base $(\mathcal{R}, D_1)$ *entails* an instance (or a BCQ) D_2 if every model of $(\mathcal{R}, D_1)$ entails D_2. In both cases, A entails B is denoted by $A \models B$.

Note that by this definition, if $\mathcal{U}$ is a universal model of a knowledge base $\mathcal{K}$, then for every model $\mathcal{M}$ of $\mathcal{K}$, $\mathcal{M} \models \mathcal{U}$. Thus, given a BCQ q, if $\mathcal{U} \models q$, then for any model $\mathcal{M}$ of $\mathcal{K}$, we have $\mathcal{M} \models q$, and if $\mathcal{U} \not\models q$, then there is a model of $\mathcal{K}$ that does not model q. Thus, $\mathcal{K} \models q$ if and only if $\mathcal{U} \models q$. This motivates the study of an algorithm to find universal models in order to solve BCQ entailment.

2.3 The chase

The chase is a family of algorithms mainly used to solve BCQ entailment, through repeated rule applications until a fixpoint is reached. The chase is indeed sound and complete [2], but since the problem of BCQ entailment is semi-decidable [1], it does not necessarily terminate. To define it properly, we first introduce the notion of a trigger.

Definition 7. *A* trigger *t for an instance D is a pair (R, π) consisting of a rule $R = B \to H$ and a homomorphism from B to D. We then define π^R as the extension of π that maps every existential variable z in R to the fresh variable z_t that is unique for z and t. A trigger $t = (R, \pi)$ is* Datalog *if R is Datalog, and* existential *otherwise. The trigger t is* applicable *on D if $\pi^R(H) \not\subseteq D$. We denote $\pi(B)$ and $\pi^R(H)$ by support(t) and output(t), and refer to them as the* support *and the* output *of t, respectively.*

Now, that we have a basic notion of applicability, we can define derivations, before refining the notion of applicability to exhibit different chase variants.

Definition 8. *Let $\mathcal{K} = (\mathcal{R}, D)$ be a knowledge base. A* derivation *from $\mathcal{K}$ is a (possibly infinite) sequence $\mathcal{D} = D_0, t_1, D_1, \ldots$ where $D_0 = D$, and for each $i > 0$, t_i is a trigger for the instance D_{i-1} and $D_i = D_{i-1} \cup$ output(t_i). The result of a derivation is the set res$(\mathcal{D}) = \bigcup_i D_i$.*

Note that $res(\mathcal{D})$ is well-defined for every derivation $\mathcal{D} = D_0, t_1, D_1, \ldots$ (finite or not) because $\{D_i\}_{i>0}$ is an increasing sequence for $\subset$ for every derivation.

In the following it will be useful to distinguish between two variants of the chase, namely the oblivious chase ($\mathbb{O}$) and the restricted chase ($\mathbb{R}$). They differ from each other in how they constrain trigger applicability, as specified in the next definition.

Definition 9. *Let $\mathcal{K} = (\mathcal{R}, D)$ be a knowledge base, $\mathcal{D} = D_0, t_1, D_1, \ldots$ be a derivation from $\mathcal{K}$, and $t = (R, \pi)$ a trigger applicable on $res(\mathcal{D})$. Then t is:*

- $\mathbb{O}$-applicable *on $\mathcal{D}$ if there is no i such that $t = t_i$.*
- $\mathbb{R}$-applicable *on $\mathcal{D}$ if there is no retraction from $res(\mathcal{D}) \cup output(t)$ to $res(\mathcal{D})$.*

An $\mathbb{X}$-derivation *(with $\mathbb{X} \in \{\mathbb{O}, \mathbb{R}\}$) is a derivation $\mathcal{D} = D_0, t_1, D_1, \ldots$ for which every trigger t_i is $\mathbb{X}$-applicable on D_{i-1}. A* Datalog-first $\mathbb{X}$-derivation *is an $\mathbb{X}$-derivation $\mathcal{D} = D_0, t_1, D_1, \ldots$ such that for all i, if t_i is existential, then there is no Datalog trigger $\mathbb{X}$-applicable on D_{i-1}.*

Datalog-first derivations consist is a succession of closure under Datalog rules and applications of a single existential trigger. Thus, in the following, we will write Datalog-first derivations from $(\mathcal{R}, D)$ as $\mathcal{D} = \mathcal{D}_0, t_1, \mathcal{D}_1, \ldots$, where for all i, t_i is an existential trigger and $\mathcal{D}_i$ is a derivation from $(\mathcal{R}, res(t_i))$ that features only Datalog triggers.

Theorem 10. *If $\mathcal{D}$ is a chase sequence from a knowledge base $(\mathcal{R}, D)$, then $(\mathcal{R}, D) \models q$ if and only if $res(\mathcal{D}) \models q$, for all queries q.*

2.4 Some computability theory

Definition 11. *A* Turing machine *M is a tuple (Q, q_0, q_f, δ) where Q is a set of* states*, q_0 and q_f are two special states called the* initial *and the* final state*, respectively, and $\delta : (Q \setminus \{q_f\}) \times \{0, 1, \mathtt{B}\} \to (Q \setminus \{q_0\}) \times \{0, 1, \mathtt{B}\} \times \{\leftarrow, \rightarrow\}$ is a function called the* transition function*.*

A configuration *of a Turing machine is a triple (q, h, T) where q is a state, h is an integer called the* position of the head*, and $T : \{0, 1, \ldots, n\} \to \{0, 1, \mathtt{B}\}$ is the* tape*, that is such that $T(n) = \mathtt{B}$, and where n is an integer. We then say that the tape is of size n. The* start configuration *C_0 of M on a word w (on the alphabet $\{0, 1\}$) is $(q_0, 0, T_0)$ where T_0 is a tape of size $|w|$ such that for all i between 0 and $|w|$, we have $T_0(i) = w[i]$.*

The run *of M on a word w is a sequence $C_0, C_1, \ldots$ of configurations such that C_0 is the start configuration of M on w, and for all t, if $C_t = (q_t, h_t, T_t)$, then T_t is of size t and:*

- *If $q_t = q_f$, then C_t is the last element of the run. In this case, we say M* halts on *w* at step *t.*
- *If $\delta(q_t, T_t(h_t)) = (q, c, \leftarrow)$ and $h_t = 0$, then we say M halts on w at step t.*
- *If $\delta(q_t, T_t(h_t)) = (q, c, \leftarrow)$ and $h_t > 0$, then $C_{t+1} = (q, h_t - 1, T_{t+1})$.*

- *If* $\delta(q_t, T_t(h_t)) = (q, c, \rightarrow)$*, then* $C_{t+1} = (q, h_t + 1, T_{t+1})$*.*

where $T_{t+1}(h_t) = c$*,* $T_{t+1}(t+1) = \mathtt{B}$*, and* $T_{t+1}(i) = T_t(i)$ *for all* $i \neq h_t$ *between* 0 *and* t*.*

In the following, we denote the unique run from M *on the empty word as* $r^M = C_0^M, C_1^M, \ldots,$ *with* $C_t^M = (q_t, h_t, T_t)$ *for all* $t \geq 0$*, and we write* $M(i,t) = T_t(i)$*, so* $M(i,t)$ *is the content of position* i *of* M*'s tape at step* t*.*

If M *halts on the empty word, we denote with* $\mathrm{time}_\varepsilon(M)$ *the index at which it does. Otherwise, we define* $\mathrm{time}_\varepsilon(M) = \infty$*.*

Note that this definition of a Turing machine is a bit unusual, to make the representation using existential rules and the proofs a bit simpler. First, we do not consider acceptance or rejection, but only halting and not halting. In addition, instead of the usual "infinite tape" that Turing machines feature, we use a tape of unbounded size, but which is finite at each step. In addition, the size of the tape is not its real size: as a tape T of size n is a function from $\{0, 1, \ldots, n\}$ to our alphabet, it actually features $n+1$ cases. This can be explained by the fact that we ask for $T(n)$ to be $\mathtt{B}$, which ensures that the head of the Turing machine always has something to read (otherwise, when starting from the empty word ε, the head would not be able to the tape in the starting configuration). Though, the size of the word represented on the tape is indeed the size of the tape as we define it (the empty word has size 0).

Chapter 3

OBQA over Datalog-expressible rule sets is undecidable

We present a reduction that takes a Turing machine M and produces a rule set $\mathcal{R}_M$ such that

1. The Turing machine M halts on the empty word if and only if $(\mathcal{R}_M, \emptyset) \models \texttt{Goal}$.
2. The couple rule set-query $(\mathcal{R}_M, \texttt{Goal})$ is Datalog-expressible.

Provided we can construct such a reduction, if we assume, by the absurd, that there is an algorithm that decides entailment for Datalog-expressible rule sets, then we can decide if M halts, which contradicts the undecidability of the halting problem. Thus, this is enough to prove Theorem 3. We then exhibit such a rule set, before proving it enjoys both these properties.

3.1 The rule set

Definition 12. *For a TM M, let $\mathcal{R}_M$ be the rule set containing the following rules:*

$$\rightarrow \exists w_0.\ \boldsymbol{q_0}(w_0, w_0) \qquad (R_0)$$

$$\boldsymbol{q_0}(w_0, w_0) \rightarrow \exists w_1.\ \boldsymbol{N}(w_0, w_1) \qquad (R_1)$$

$$\boldsymbol{q_0}(w_0, w_0), \boldsymbol{N}^+(w_0, w_i) \rightarrow \exists w_{i+1}.\ \boldsymbol{N}(w_i, w_{i+1}) \qquad (R_{+1})$$

$$\boldsymbol{N}(w_i, w_{i+1}) \rightarrow \boldsymbol{N}^+(w_i, w_{i+1}) \qquad (R_{\boldsymbol{N}^+})$$

$$\boldsymbol{N}(w_i, w_{i+1}), \boldsymbol{N}^+(w_{i+1}, w_k) \rightarrow \boldsymbol{N}^+(w_i, w_k) \qquad (R^{trans}_{\boldsymbol{N}^+})$$

$$\boldsymbol{q_0}(w_0, w_0) \rightarrow \texttt{B}(w_0, w_0) \qquad (R^0_{\texttt{B}})$$

$$\boldsymbol{q_0}(w_0, w_0), \boldsymbol{N}^+(w_0, w_i) \rightarrow \texttt{B}(w_i, w_i) \qquad (R_{\texttt{B}})$$

$$\boldsymbol{q_f}(x, y) \rightarrow \texttt{Goal} \qquad (R_{\texttt{Goal}})$$

For every $q, q' \in Q$, $c, c' \in \{0, 1, \mathtt{B}\}$ such that $\delta(q, c) = (q', c', \leftarrow)$,

$$\boldsymbol{N}(p_1, p_2), \boldsymbol{N}(t_1, t_2), \boldsymbol{q}(p_2, t_1), \boldsymbol{c}(p_2, t_1), \boldsymbol{N}^+(p_2, t_2) \to \boldsymbol{q'}(p_1, t_2), \boldsymbol{c'}(p_2, t_2) \qquad (R^{\leftarrow}_{q,c})$$

$$\boldsymbol{q_0}(w_0, w_0), \boldsymbol{q}(w_0, t), \boldsymbol{c}(w_0, t) \to \mathtt{Goal} \qquad (R^{0}_{q,c})$$

For every $q, q' \in Q$, $c, c' \in \{0, 1, \mathtt{B}\}$ such that $\delta(q, c) = (q', c', \rightarrow)$,

$$\boldsymbol{N}(p_1, p_2), \boldsymbol{N}(t_1, t_2), \boldsymbol{q}(p_1, t_1), \boldsymbol{c}(p_1, t_1), \boldsymbol{N}^+(p_2, t_2) \to \boldsymbol{q'}(p_2, t_2), \boldsymbol{c'}(p_1, t_2) \qquad (R^{\rightarrow}_{q,c})$$

$$\boldsymbol{N}(w_1, w_2), \boldsymbol{q}(w_1, w_1), \boldsymbol{c}(w_1, w_1) \to \boldsymbol{q'}(w_2, w_2), \boldsymbol{c'}(w_1, w_2) \qquad (R^{max}_{q,c})$$

For every $q \in Q$ and $c \in \{0, 1, \mathtt{B}\}$,

$$\boldsymbol{N}^+(p_i, p_k), \boldsymbol{N}(t_1, t_2), \boldsymbol{q}(p_k, t_1), \boldsymbol{c}(p_i, t_2) \to \boldsymbol{c}(p_i, t_2) \qquad (R^{L}_{q,c})$$

$$\boldsymbol{N}^+(p_i, p_k), \boldsymbol{N}(t_1, t_2), \boldsymbol{q}(p_i, t_1), \boldsymbol{c}(p_k, t_1) \to \boldsymbol{c}(p_k, t_2) \qquad (R^{R}_{q,c})$$

And the rules:

$$\boldsymbol{N}(x, y), \boldsymbol{N}(x, y') \to \mathtt{Eq}(y, y') \qquad (R^{+}_{func})$$

$$\boldsymbol{N}(x, y), \boldsymbol{N}(x', y) \to \mathtt{Eq}(x, x') \qquad (R^{-}_{func})$$

$$\boldsymbol{q_0}(x, y), \boldsymbol{q_0}(z, t) \to \mathtt{Eq}(x, y), \mathtt{Eq}(y, z), \mathtt{Eq}(z, t) \qquad (R_{q_0})$$

$$\boldsymbol{N}^+(x, x) \to \mathtt{Fail} \qquad (R^{cycle}_{\boldsymbol{N}^+})$$

$$\boldsymbol{N}(w_{-1}, w_0), \boldsymbol{q_0}(w_0, w_0) \to \mathtt{Fail} \qquad (R_{-1})$$

$$\boldsymbol{N}^+(x, y), \boldsymbol{q_1}(x, t), \boldsymbol{q_2}(y, t) \to \mathtt{Fail} \quad \textit{for every } q_1, q_2 \in Q \qquad (R^{step}_{q_1,q_2})$$

$$\boldsymbol{q_1}(x, t), \boldsymbol{q_2}(x, t) \to \mathtt{Fail} \quad \textit{for every } q_1 \neq q_2 \textit{ in } Q \qquad (R^{cell}_{q_1,q_2})$$

$$\boldsymbol{c_1}(x, t), \boldsymbol{c_2}(x, t) \to \mathtt{Fail} \quad \textit{for every } c_1 \neq c_2 \textit{ in } \{0, 1, \mathtt{B}\} \qquad (R^{cell}_{c_1,c_2})$$

$$\boldsymbol{N}^+(t, x), \boldsymbol{c}(x, t) \to \mathtt{Fail} \quad \textit{for every } c \in \{0, 1, \mathtt{B}\} \qquad (R^{OoB}_{c})$$

$$\mathtt{Fail} \to \mathtt{Goal} \qquad (R_{\mathtt{Fail}})$$

$$\mathtt{Eq}(x, y), \mathtt{Eq}(y, z) \to \mathtt{Eq}(x, z) \qquad (R^{trans}_{\mathtt{Eq}})$$

$$\mathtt{Eq}(x, y) \to \mathtt{Eq}(y, x) \qquad (R^{sym}_{\mathtt{Eq}})$$

For all predicates P and integers i between 1 and $ar(P)$,

$$P(x_1, \ldots, x_{ar(P)}), \mathtt{Eq}(x_i, y_i) \to P(x_1, \ldots, y_i, \ldots, x_{ar(P)}) \qquad (R^{repl}_{P,i})$$

3.2 $\mathcal{R}_M$ simulates M

This section is dedicated to proving the following theorem.

Theorem 13. *The Turing machine M halts on the empty word if and only if $(\mathcal{R}_M, \emptyset) \models \mathtt{Goal}$.*

Before proving this theorem, we explain the relevant parts of the rule set. For this rule set, we use an encoding derived from the usual grid representation of Turing machines. In the grid representation of Turing machines, one would

usually consider a two-way infinite grid (isomorphic to $\mathbb{N} \times \mathbb{N}$), on which the position (i, j) would represent the i-th cell of the machine's tape at step j. Then, to encode this representation in existential rules, one would have an individual for each cell and two binary predicates representing the relations "x is right below y" and "x is right at the left of y". Finally, one would have a unary predicate for each state and for each tape symbol, and $\boldsymbol{q}(x)$ would mean that the head is on the cell represented by x and the Turing machine is at step q at this time, and $\boldsymbol{c}(x)$ means that x contains c.

Here, we use the same representation, but we encode it differently: instead of using unary predicates for the content of the tapes, we use binary predicates. This means that we only need individuals representing $\mathbb{N}$ and not the full $\mathbb{N} \times \mathbb{N}$. In addition, we only use one successor predicate instead of two. In our case, this successor predicate is $\boldsymbol{N}$, and the rules that generate this encoding of $\mathbb{N}$ are rules R_0, R_1 and R_{+1}. Then, the predicate $\boldsymbol{N}^+$ is $\boldsymbol{N}$'s transitive closure, and is computed by rules $R_{\boldsymbol{N}^+}$ and $R^{trans}_{\boldsymbol{N}^+}$. In the following, we will refer to the elements of this representation of $\mathbb{N}$ by $(z_n)_{n\in\mathbb{N}}$: z_0 is the element that does not have a predecessor through $\boldsymbol{N}$, and z_{i+1} is z_i's successor, for all i.

Then, as we said, we model Turing machine simulation using a binary predicate for each state of the Turing machine and for each character that can appear on the tape. In more detail, we will show in Proposition 16 that if $\boldsymbol{q}(z_i, z_t)$ is in $\mathit{chase}(\mathcal{R}_M, \emptyset)$ for some state q, then the Turing machine is in state q at step t, and its head is on position i. Similarly, if $\boldsymbol{c}(z_i, z_t)$ is in $\mathit{chase}(\mathcal{R}_M, \emptyset)$, then the character at position i of the tape at step t is c. Rules R_0, $R^0_{\mathtt{B}}$ and $R_{\mathtt{B}}$ initialize this calculation, rules $R^{\leftarrow}_{q,c}$ and $R^{\rightarrow}_{q,c}$ move the head and change the character, and rules $R^L_{q,c}$ and $R^R_{q,c}$ propagate the symbols that do not change. Finally, rules $R_{\mathtt{Goal}}$ and $R^0_{q,c}$ generate $\mathtt{Goal}$ whenever the Turing machine encounters q_f or tries to move its head left when it is on position 0, that is to say whenever it halts. The remaining rules will not do much on the empty database, and will be useful later to ensure that the rule set is Datalog expressible. We will thus explain them later. We first show a quick lemma that will be useful in the proof of Theorem 13.

Lemma 14. *Let D be an instance and t a term in D. There is no trigger (R, π) that is applicable on D and such that $\mathtt{Eq}(t, t) \in \pi(\mathit{body}(R))$.*

Proof. Suppose for a contradiction that there is such a trigger. Then, since the only rules that feature an $\mathtt{Eq}$-atom in their body are rules $R^{trans}_{\mathtt{Eq}}$, $R^{sym}_{\mathtt{Eq}}$ and $R^{repl}_{P,i}$, the rule R must be one of these three.

- If $R = R^{trans}_{\mathtt{Eq}}$, then two cases are possible. Either $\pi(x) = \pi(y) = t$ and $\pi(z) = u$ such that $\mathtt{Eq}(t, u) \in D$, and (R, π) yields $\mathtt{Eq}(t, u)$ or $\pi(y) = \pi(z) = t$ and $\pi(x) = u$ with $\mathtt{Eq}(u, t) \in D$ and (R, π) yields $\mathtt{Eq}(u, t)$. In either case, $\mathit{output}(R, \pi) \subseteq D$ so the trigger is not applicable.
- If $R = R^{sym}_{\mathtt{Eq}}$, then $\pi(x) = \pi(y) = t$, and (R, π) yields $\mathtt{Eq}(t, t)$, which is by hypothesis in $D \cup \mathtt{Eq}(t, t)$, so this trigger is not applicable either.
- If $R = R^{repl}_{P,i}$, then there is an atom $P(t_1, \ldots, t_{i-1}, t, t_{i+1}, \ldots, t_n)$ (with $n = \mathit{ar}(P)$). Then, again, (R, π) yields $P(t_1, \ldots, t_{i-1}, t, t_{i+1}, \ldots, t_n)$, which is in $D \cup \{\mathtt{Eq}(t, t)\}$ by hypothesis, so (R, π) is not applicable in this case either. □

Then, to show Theorem 13, we will specify the result of Datalog-first derivations. We first define a few sets that will be useful to express this result.

Definition 15. *For all n, we define the following sets*

$$\begin{aligned}
D_n^{\boldsymbol{N}} &= \{\boldsymbol{N}(z_i, z_{i+1}) \mid 0 \leq i < n\} \\
D_n^{\boldsymbol{N}^+} &= \left\{\boldsymbol{N}^+(z_i, z_j) \;\middle|\; 0 \leq i < j \leq n\right\} \\
D_n^{Q} &= \{\boldsymbol{q_t}(z_{h_t}, z_t) \mid 0 \leq t \leq \min(n, \mathit{time}_\varepsilon(M))\} \\
D_n^{\Sigma} &= \{\boldsymbol{c}(z_i, z_t) \mid 0 \leq t \leq \min(n, \mathit{time}_\varepsilon(M)),\ 0 \leq i \leq t,\ M(i,t) = c\} \\
D_n^{\texttt{Eq}} &= \{\texttt{Eq}(z_i, z_i) \mid 0 \leq i \leq n\} \\
D_n^{\texttt{Goal}} &= \{\texttt{Goal}\} \textit{ if } n \geq \mathit{time}_\varepsilon(M) \textit{ and } \emptyset \textit{ otherwise.}
\end{aligned}$$

We can now formulate the following proposition.

Proposition 16. *For a Datalog-first derivation $\mathcal{D} = \mathcal{D}_{init}, t_0, \mathcal{D}_0, t_1, \mathcal{D}_1, \ldots$ from $(\mathcal{R}_M, \emptyset)$, we have that:*

- $t_0 = ((R_0), \emptyset)$ *and* $t_1 = (R_1, \{w_0 \mapsto z_0\})$
- *For all* $n > 1$, $t_n = (R_{+1}), \{w_0 \mapsto z_0,\ w_i \mapsto z_{n-1}\}$
- *For all* $n \in \mathbb{N}$, $\mathit{res}(\mathcal{D}'_n) = D_n^{\boldsymbol{N}} \cup D_n^{\boldsymbol{N}^+} \cup D_n^{Q} \cup D_n^{\Sigma} \cup D_n^{\texttt{Eq}} \cup D_n^{\texttt{Goal}}$.

Proof. We proceed by induction on n.

At step $n = 0$, on the database $\emptyset$, only rules with an empty body are applicable. As R_0 is the only rule with an empty body, it is applicable, so $\mathcal{D}_{init}$ is empty and $t_0 = (R_0, \emptyset)$, with $\mathit{res}(t_0) = \{\boldsymbol{q_0}(z_0, z_0)\}$. Then, rules $R^0_{\texttt{B}}$ and R_{q_0} are applicable and yield $\texttt{B}(z_0, z_0)$ and $\texttt{Eq}(z_0, z_0)$, respectively. Finally, if $\delta(q_0, \texttt{B}) = (q', c, \leftarrow)$ for some q', c', then rule $R^0_{q_0,\texttt{B}}$ is applicable, and if $q_0 = q_f$, $R_{\texttt{Goal}}$ is applicable. In both these cases, the application yields $\texttt{Goal}$, and by definition of a run, $\mathit{time}_\varepsilon(M) = 0$. We then show that $\mathit{res}(\mathcal{D}_0) = D_0^{\boldsymbol{N}} \cup D_0^{\boldsymbol{N}^+} \cup D_0^{Q} \cup D_0^{\Sigma} \cup D_0^{\texttt{Eq}} \cup D_0^{\texttt{Goal}}$ by double inclusion. For the converse inclusion, note that $D_0^{\boldsymbol{N}}$ and $D_0^{\boldsymbol{N}^+}$ are empty, and by definition of a starting configuration, we have $D_0^{Q} = \{\boldsymbol{q_0}(z_0, z_0)\}$ and $D_0^{\Sigma} = \{\texttt{B}(z_0, z_0)\}$. In addition, we also have $D_0^{\texttt{Eq}} = \{\texttt{Eq}(z_0, z_0)\}$. Then, $D_0^{\texttt{Goal}} = \{\texttt{Goal}\}$ if $\mathit{time}_\varepsilon(M) = 0$ (case in which $\texttt{Goal} \in \mathit{res}(\mathcal{D}_0)$), and $D_0^{\texttt{Goal}} = \emptyset$ otherwise. Thus, the converse inclusion holds.

Then, for the direct inclusion, we show that no Datalog rule is applicable on $D_0^{\boldsymbol{N}} \cup D_0^{\boldsymbol{N}^+} \cup D_0^{Q} \cup D_0^{\Sigma} \cup D_0^{\texttt{Goal}} = \{\boldsymbol{q_0}(z_0, z_0), \texttt{B}(z_0, z_0)\} \cup D_0^{\texttt{Goal}}$. First notice that all Datalog rules except $R_{\texttt{Goal}}$ $R^0_{q,c}$, R_{q_0}, $R^{cell}_{q_1,q_2}$, $R^{cell}_{c_1,c_2}$, $R_{\texttt{Fail}}$ and the axiomatization of equality have an $\boldsymbol{N}$-atom or an $\boldsymbol{N}^+$-atom in their body, so their bodies cannot be mapped to $\mathit{res}(t_0)$. In addition, the body of rules $R_{\texttt{Goal}}$ and $R^0_{q,c}$ can only be mapped into $\{\boldsymbol{q_0}(z_0, z_0), \texttt{B}(z_0, z_0)\} \cup D_0^{\texttt{Goal}}$ if $\mathit{time}_\varepsilon(M) = 0$. Indeed, if rule $R_{\texttt{Goal}}$'s body can be mapped, then there is $q_f(x, y)$ in the database, so $q_0 = q_f$, and if $R^0_{q,c}$'s body can be mapped, then $q = q_0$, $c = \texttt{B}$ and $\delta(q_0, \texttt{B}) = (q', c', \leftarrow)$. Thus, in both cases, $\mathit{time}_\varepsilon(M) = 0$, so $\texttt{Goal} \in D_0^{\texttt{Goal}} \subseteq \mathit{res}(\mathcal{D}_0)$, and these rules are not applicable.

Regarding rule R_{q_0}, as it has already been applied with its body mapped to the only $\boldsymbol{q_0}$-atom, it is not applicable either. Rule $R^{cell}_{q_1,q_2}$ (resp. $R^{cell}_{c_1,c_2}$) has two atoms that use distinct state (resp. character) predicates in its body. As there is

only one state (resp. character) atom in $res(\mathcal{D}'_0) \cup D_0^{\texttt{Eq}}$, rule $R^{cell}_{q_1,q_2}$ (resp. $R^{cell}_{c_1,c_2}$) is not applicable. As $\texttt{Fail} \notin res(\mathcal{D}'_0)$, rule $R_{\texttt{Fail}}$ is not applicable either. Finally, if $t = (R, \pi)$ is a trigger such that $R \in \{R^{trans}_{\texttt{Eq}}, R^{sym}_{\texttt{Eq}}, R^{repl}_{P,i}\}$, then there must be an $\texttt{Eq}$-atom in $\pi(body(R))$. As $\texttt{Eq}(z_0, z_0)$ is the only $\texttt{Eq}$-atom in $res(\mathcal{D}'_0) \cup D_0^{\texttt{Eq}}$, it must be that $\texttt{Eq}(z_0, z_0) \in \pi(body(R))$, which by Lemma 14 means that t is not applicable. Thus, the equality holds in the base case.

Before proving the inductive step, we prove that $t_1 = (R_1), \{w_0 \mapsto z_0\}$. Since $res(\mathcal{D}_0) = \{\boldsymbol{q_0}(z_0, z_0), \texttt{B}(z_0, z_0)\}$, rule R_{+1} is not applicable (it has an $\boldsymbol{N}^+$-atom in its body). Since rule R_0 has already been applied with the only homomorphism possible, the only applicable rule is R_1, so $t_1 = (R_1), \{w_0 \mapsto z_0\}$.

Now, for the inductive step, assume that $n > 0$ and that the lemma is true for all $i < n$. We first prove that if $n > 1$, $t_n = (R_{+1}), \{w_i \mapsto z_{n-1},\ w_0 \mapsto z_0\}$. By definition of a Turing machine, q_0 can only appear at step and position 0, so by induction hypothesis, the only term x such that $\boldsymbol{q_0}(x, x) \in res(\mathcal{D}'_{n-1})$ is z_0. Thus, all the homomorphisms from the body of rule R_{+1} to $res(\mathcal{D}'_{n-1})$ are of the form $\{w_i \mapsto z_j, w_0 \mapsto z_0\}$ with $0 \leq j \leq n-1$. Again by induction hypothesis, all triggers $(R_{+1}), \{x_1 \mapsto z_j, w_0 \mapsto z_0\}$ for $2 \leq j \leq n-2$ have been applied at earlier steps, so the only applicable existential trigger is $(R_{+1}), \{w_i \mapsto z_{n-1},\ w_0 \mapsto z_0\}$.

We now come back to the more general case $n > 0$, and we prove that $D_n^{\boldsymbol{N}} \cup D_n^{\boldsymbol{N}^+} \cup D_n^{Q} \cup D_n^{\Sigma} \cup D_n^{\texttt{Eq}} \cup D_n^{\texttt{Goal}} \subseteq res(\mathcal{D}'_n)$.

$D_n^{\boldsymbol{N}} \subseteq res(\mathcal{D}_n)$: By induction hypothesis, $D_{n-1}^{\boldsymbol{N}} \subseteq res(\mathcal{D}'_{n-1}) \subseteq res(\mathcal{D}'_n)$. Then, if $n = 1$, $output(t_1) = \{\boldsymbol{N}(z_0, z_1)\} = \{\boldsymbol{N}(z_{n-1}, z_n)\}$, and if $n > 1$, $output(t_n) = \{N(z_{n-1}, z_n)\}$, so in either case, $D_n^{\boldsymbol{N}} \subseteq res(\mathcal{D}'_n)$.

$D_n^{\boldsymbol{N}^+} \subseteq res(\mathcal{D}_n)$ By induction hypothesis, $D_{n-1}^{\boldsymbol{N}^+} \subseteq res(\mathcal{D}'_{n-1}) \subseteq res(\mathcal{D}'_n)$, so it is enough to show that for all i such that $0 \leq i < n$, we have $\boldsymbol{N}^+(z_i, z_n) \in res(\mathcal{D}'_n)$, or equivalently that for all i such that $1 \leq i \leq n$, we have $\boldsymbol{N}^+(z_{n-i}, z_n) \in res(\mathcal{D}'_n)$. We prove this last result by induction on i.

We first show that if $i = 1$, then $\boldsymbol{N}^+(z_{n-1}, z_n) \in res(\mathcal{D}'_n)$. As $D_n^{\boldsymbol{N}} \subseteq res(\mathcal{D}'_n)$, we have $\boldsymbol{N}(z_{n-1}, z_n) \in res(\mathcal{D}'_n)$. Thus, since $res(\mathcal{D}'_n)$ satisfies rule $R_{\boldsymbol{N}^+}$, the atom $\boldsymbol{N}^+(z_{n-1}, z_n)$ is in $res(\mathcal{D}'_n)$.

For the induction step, we show that if $1 < i \leq n$ and $\boldsymbol{N}^+(z_{n-(i-1)}, z_n) \in res(\mathcal{D}'_n)$, then $\boldsymbol{N}^+(z_{n-i}, z_n) \in res(\mathcal{D}'_n)$. Since $D_n^{\boldsymbol{N}} \subseteq res(\mathcal{D}'_n)$, we have $\boldsymbol{N}(z_{n-1}, z_{n-i+1}) \in res(\mathcal{D}'_n)$. As $n - i + 1 = n - (i - 1)$, the atoms $\boldsymbol{N}(z_{n-1}, z_{n-i+1})$ and $\boldsymbol{N}^+(z_{n-i+1}, z_n)$ are both in $res(\mathcal{D}'_n)$. Thus, since $res(\mathcal{D}'_n)$ models rule $R^{trans}_{\boldsymbol{N}^+}$, $\boldsymbol{N}^+(z_{n-(i-1)}, z_n) \in res(\mathcal{D}'_n)$.

$D_n^{\boldsymbol{Q}} \subseteq res(\mathcal{D}_n)$: First, notice that if $n > time_\varepsilon(M)$, then $D_n^{Q} = D_{n-1}^{Q}$, so the induction hypothesis suffices. We thus only consider the case where $n \leq time_\varepsilon(M)$ in the following. We then want to show that $\boldsymbol{q_n}(z_{h_n}, z_n) \in res(\mathcal{D}'_n)$. As $q_{n-1} \neq q_f$, there are q_n, c, c', d such that $c = M(h_{n-1}, n-1)$ and $\delta(q_{n-1}, c) = (q_n, c', d)$. By induction hypothesis, both the atoms $\boldsymbol{q_{n-1}}(z_{h_{n-1}}, z_{n-1})$ and $\boldsymbol{c}(z_{h_{n-1}}, z_{n-1})$ are in $res(\mathcal{D}'_n)$. In addition, as $h_{n-1} \leq n-1 < n$, we also have $\boldsymbol{N}^+(z_{h_{n-1}}, z_n) \in res(\mathcal{D}'_n)$, and we showed earlier that $\boldsymbol{N}(z_{n-1}, z_n) \in res(\mathcal{D}'_n)$. Thus:

- If $d = \rightarrow$, as $h_{n-1} < n$, we have $\boldsymbol{N}(z_{h_{n-1}}, z_{h_{n-1}+1}) \in res(\mathcal{D}'_n)$. Thus,

$$t_\rightarrow = (R^{\rightarrow}_{q_{n-1},c}, \{p_i \mapsto z_{h_{n-1}}, p_{i+1} \mapsto z_{h_{n-1}+1}, t_i \mapsto z_{n-1}, t_{i+1} \mapsto z_n\})$$

is applicable. In this case, by definition, $h_n = h_{n-1}+1$, so this trigger yields the atoms $\boldsymbol{q_n}(z_{h_n}, z_n)$ and $\boldsymbol{c'}(z_{h_{n-1}}, z_n)$.

- If $d = \leftarrow$, since $n-1 < \mathit{time}_\varepsilon(M)$, then $h_{n-1} > 0$, so we have $\boldsymbol{N}(z_{h_{n-1}-1}, z_{h_{n-1}}) \in \mathit{res}(\mathcal{D}'_n)$. Thus, the trigger

$$t_\leftarrow = (R^{\leftarrow}_{q_{n-1},c}, \{p_{i-1} \mapsto z_{h_{n-1}-1}, p_i \mapsto z_{h_{n-1}}, t_i \mapsto z_{n-1}, t_{i+1} \mapsto z_n\})$$

is applicable. In this case, by definition, $h_n = h_{n-1}-1$, so this trigger yields the atoms $\boldsymbol{q_n}(z_{h_n}, z_n)$ and $\boldsymbol{c'}(z_{h_{n-1}}, z_n)$.

Thus, in either cases, we have $\boldsymbol{q_n}(z_{h_n}, z_n)$ and $\boldsymbol{c'}(z_{h_{n-1}}, z_n)$ in $\mathit{res}(\mathcal{D}'_n)$, so $D^Q_n \subseteq \mathit{res}(\mathcal{D}'_n)$.

$\boldsymbol{D^\Sigma_n \subseteq res(\mathcal{D}_n)}$: First, as before, if $n > \mathit{time}_\varepsilon(M)$, then $D^\Sigma_n = D^\Sigma_{n-1}$, so the induction hypothesis suffices. In the following, we restrict ourselves to the case $n \leq \mathit{time}_\varepsilon(M)$. We thus need to prove that $\boldsymbol{c_i}(z_i, z_n) \in \mathit{res}(\mathcal{D}'_n)$ for all i such that $0 \leq i \leq n$ and $M(i,n) = c_i$. There are four cases:

- If $i < h_{n-1}$, then $c_i = M(i,n) = M(i,n-1)$. By induction hypothesis, the atoms $\boldsymbol{c_i}(z_i, z_{n-1})$, $\boldsymbol{q_{n-1}}(z_{h_{n-1}}, z_{n-1})$ and $\boldsymbol{N}^+(z_i, z_{h_{n-1}})$ are in $\mathit{res}(\mathcal{D}'_n)$. In addition, since $D^N_n \subseteq \mathit{res}(\mathcal{D}'_n)$, we have $\boldsymbol{N}(z_{n-1}, z_n) \in \mathit{res}(\mathcal{D}'_n)$. Thus, as $\mathit{res}(\mathcal{D}'_n)$ satisfies rule $R^L_{q_{n-1},c}$, we have $\boldsymbol{c_i}(z_i, z_n) \in \mathit{res}(\mathcal{D}'_n)$.
- If $i = h_{n-1}$, as seen in the previous case, we have $\boldsymbol{c'}(z_{h_{n-1}}, z_n)$, with c' such that $\delta(q_{n-1}, M(h_{n-1}, n-1)) = (q', c', d)$. By definition of a run, $c' = M(h_{n-1}, n) = c_i$, which concludes this case.
- If $h_{n-1} < i < n$, then $c_i = M(i,n) = M(i,n-1)$. As $i \leq n-1$, by induction hypothesis, the atoms $\boldsymbol{c_i}(z_i, z_{n-1})$, $\boldsymbol{q_{n-1}}(z_{h_{n-1}}, z_{n-1})$ and $\boldsymbol{N}^+(z_{h_{n-1}}, z_i)$ are in $\mathit{res}(\mathcal{D}'_n)$. In addition, since $D^N_n \subseteq \mathit{res}(\mathcal{D}'_n)$, we have $\boldsymbol{N}(z_{n-1}, z_n) \in \mathit{res}(\mathcal{D}'_n)$. Thus, as $\mathit{res}(\mathcal{D}'_n)$ satisfies rule $R^R_{q_{n-1},c}$, we have $\boldsymbol{c_i}(z_i, z_n) \in \mathit{res}(\mathcal{D}'_n)$.
- If $i = n$, a usual property of Turing machines states that $M(n,n) = \mathtt{B}$, as a Turing machine can only move one step at a step and edit the bit its head was on at a previous step. As $D^{N^+}_n \subseteq \mathit{res}(\mathcal{D}'_n)$, we have $\boldsymbol{N}^+(z_0, z_n) \in \mathit{res}(\mathcal{D}'_n)$. Thus, as $\mathit{res}(\mathcal{D}'_n)$ satisfies rule $R_\mathtt{B}$, the atom $\mathtt{B}(z_n, z_n)$ is in $\mathit{res}(\mathcal{D}'_n)$.

$\boldsymbol{D^{\mathtt{Eq}}_n \subseteq res(\mathcal{D}_n)}$: As we saw earlier, the atom $\boldsymbol{N}(z_{n-1}, z_n)$ is in $\mathit{res}(\mathcal{D}_n)$, so rule R^+_{func} is applicable by mapping x to z_{n-1}, and y and y' to z_n. This application yields the atom $\mathtt{Eq}(z_n, z_n)$, so $D^{\mathtt{Eq}}_n \subseteq \mathit{res}(\mathcal{D}_n)$ (as we already had $D^{\mathtt{Eq}}_{n-1} \subseteq \mathit{res}(\mathcal{D}_n)$ by induction hypothesis).

$\boldsymbol{D^{\mathtt{Goal}}_n \subseteq res(\mathcal{D}_n)}$: If $n < \mathit{time}_\varepsilon(M)$, the result follows from the fact that $D^{\mathtt{Goal}}_n$ is empty in this case. Otherwise, if $n \geq \mathit{time}_\varepsilon(M)$, then there are two cases:

- If $q_{\mathit{time}_\varepsilon(M)} = q_f$, since $D^Q_n \subseteq \mathit{res}(\mathcal{D}'_n)$, $\boldsymbol{q_f}(z_{h_{\mathit{time}_\varepsilon(M)}}, z_{\mathit{time}_\varepsilon(M)}) \in \mathit{res}(\mathcal{D}'_n)$. Thus, as $\mathit{res}(\mathcal{D}'_n)$ models rule $R_{\mathtt{Goal}}$, $\mathtt{Goal} \in \mathit{res}(\mathcal{D}'_n)$.

- Otherwise, we have $h_{time_\varepsilon(M)} = 0$ and

$$\delta(q_{time_\varepsilon(M)}, M(0, time_\varepsilon(M))) = (q, c, \leftarrow)$$

for some q and c. In this case, as $D_n^Q \cup D_n^\Sigma \subseteq res(\mathcal{D}'_n)$, the atoms $\boldsymbol{q}_{time_\varepsilon(M)}(z_0, z_{time_\varepsilon(M)})$ and $\boldsymbol{c_0}(z_0, z_{time_\varepsilon(M)})$ are in $res(\mathcal{D}'_n)$, with $c_0 = M(0, time_\varepsilon(M))$. Thus, as $res(\mathcal{D}'_n)$ satisfies rule $R^0_{q,c}$, $\texttt{Goal} \in res(\mathcal{D}'_n)$.

Thus, in all cases, $D_n^{\texttt{Goal}} \subseteq res(\mathcal{D}'_n)$.

This concludes the proof of the first inclusion. To show that the other inclusion holds, we will show that $D = D_n^{\boldsymbol{N}} \cup D_n^{\boldsymbol{N}^+} \cup D_n^Q \cup D_n^\Sigma \cup D_n^{\texttt{Eq}} \cup D_n^{\texttt{Goal}}$ satisfies all the Datalog rules in $\mathcal{R}_M$.

$R_{\boldsymbol{N}^+}$**:** Let i, j be such that $\boldsymbol{N}(z_i, z_j) \in D$. Then, by the definition of $D_n^{\boldsymbol{N}}$, we have $0 \leq i < n$ and $j = i + 1$, so $0 \leq i < j \leq n$. Thus, by the definition of $D_n^{\boldsymbol{N}^+}$, $\boldsymbol{N}^+(z_i, z_j) \in D$, so $R_{\boldsymbol{N}^+}$ is satisfied by D.

$R^{\boldsymbol{trans}}_{\boldsymbol{N}^+}$**:** Assume that $\boldsymbol{N}(z_i, z_j)$ and $\boldsymbol{N}^+(z_j, z_k)$ are in D, for some i, j, k. Then, by the definition of $D_n^{\boldsymbol{N}}$, we have $0 \leq i < n$ and $j = i+1$, so $0 \leq i < j \leq n$. In addition, by the definition of $D_n^{\boldsymbol{N}^+}$, we have $0 \leq j < k \leq n$, so $0 \leq i < j < k \leq n$. In particular, we have $0 \leq i < k \leq n$, so $\boldsymbol{N}^+(z_i, z_j) \in D$, and $R^{trans}_{\boldsymbol{N}^+}$ is satisfied by D.

$R^0_{\texttt{B}}$**:** The only term x such that $\boldsymbol{q_0}(x, x) \in D$ is z_0, and $\texttt{B}(z_0, z_0) \in D$, so this rule is not applicable.

$R_{\texttt{B}}$**:** As stated earlier, we have $M(i, t) = \texttt{B}$ for all $t \leq i$. In particular, $M(i, i) = \texttt{B}$ for all $i \leq n$. Thus, by the definition of D_n^Σ, for all $\leq n$, we have $\texttt{B}(z_i, z_i) \in D$, so rule $R_{\texttt{B}}$ is satisfied.

$R_{\texttt{Goal}}$**:** If $\boldsymbol{q_f}(z_i, z_t) \in D$, for some i, t, then by definition of D_n^Q, $q_t = q_f$, and thus $time_\varepsilon(M) = t \leq n$. As such, $D_n^{\texttt{Goal}} = \{\texttt{Goal}\}$, and rule $R_{\texttt{Goal}}$ is satisfied.

$R^0_{q,c}$**:** As stated earlier, the only term x such that $\boldsymbol{q_0}(x, x) \in D$ is z_0. Assume that there are q, c and t such that $\delta(q, c) = (q', c', \leftarrow)$ for some q', c', and $\boldsymbol{q}(z_0, z_t)$ and $\boldsymbol{c}(z_0, z_t)$ are in D. Then, by the definitions of D_n^Q and D_n^Σ, $q_t = q$, $h_t = 0$ and $M(h_t, t) = M(0, t) = c$. Thus, by definition of a run, $time_\varepsilon(M) = t \leq n$, and $D_n^{\texttt{Goal}} = \{\texttt{Goal}\}$, so rule $R^0_{q,c}$ is satisfied.

$R^{\leftarrow}_{q,c}$**:** Let q, q', c, c' and i, t be such that $\delta(q, c) = (q', c', \leftarrow)$, and the atoms $\boldsymbol{q}(z_i, z_t)$, $\boldsymbol{c}(z_i, z_t)$ are in D. Then, by definition of D_n^Q and D_n^Σ, we have $q_t = q$, $h_t = i$ and $M(h_t, t) = c$.

Now, assume that there is a homomorphism from $body(R^{\leftarrow}_{q,c})$ to D that sends p_i to z_i and t_j to z_t, and let j, t be such that p_{i-1} is mapped to z_j and t_{j+1} is mapped to z_t. The atoms $\boldsymbol{N}(z_j, z_i)$ and $\boldsymbol{N}(z_t, z_t)$ must then be present in D, so by definition of $D_n^{\boldsymbol{N}}$, we have $0 \leq j = i-1$ and $t = t+1$. To show that $R^{\leftarrow}_{q,c}$ is satisfied, we then show that $\boldsymbol{q'}(z_{i-1}, z_{t+1})$ and $\boldsymbol{c'}(z_i, z_{t+1})$ are in D. As $0 \leq j = i-1$, we have $i > 0$, so by definition of a run, we have $q_{t+1} = q'$, $h_{t+1} = h_t - 1 = i - 1$ and $M(h_t, t+1) = M(i, t+1) = c'$. Thus, by definition of D_n^Q and D_n^Σ, the atoms $\boldsymbol{q'}(z_{i-1}, z_{t+1})$ and $\boldsymbol{c'}(z_i, z_{t+1})$ are in D, and rule $R^{\leftarrow}_{q,c}$ is satisfied.

$R^{\rightarrow}_{q,c}$**:** Let q, q', c, c' and i, t be such that $\delta(q, c) = (q', c', \leftarrow)$, and the atoms $\boldsymbol{q}(z_i, z_t)$, $\boldsymbol{c}(z_i, z_t)$ are in D. Then, by definition of D^Q_n and D^Σ_n, we have $q_t = q$, $h_t = i$ and $M(h_t, t) = c$, so by definition of a run, we have $q_{t+1} = q'$, $h_{t+1} = h_t + 1 = i + 1$ and $M(h_t, t + 1) = M(i, t + 1) = c'$.

Now, assume that there is a homomorphism from $\mathit{body}(R^{\rightarrow}_{q,c})$ to D that sends p_i to z_i and t_j to z_t. By definition of $D^{\boldsymbol{N}}_n$ (as seen in the previous case), p_{i+1} is mapped to z_{i+1} and t_{j+1} is mapped to z_{t+1}. Then, rule $R^{\rightarrow}_{q,c}$ is satisfied, as the atoms $\boldsymbol{q'}(z_{i+1}, z_{t+1})$ and $\boldsymbol{c'}(z_i, z_{t+1})$ are in D. Indeed, as $q_{t+1} = q'$ and $h_{t+1} = i + 1$, the first one is in D by definition of D^Q_n, and since $M(i, t + 1) = c'$, the second atom is in D by definition of D^Σ_n.

$R^{L}_{q,c}$**:** Assume that the atoms $\boldsymbol{N}^+(z_i, z_k)$, $\boldsymbol{N}(z_t, z_{t+1})$, $\boldsymbol{q}(z_k, z_t)$ and $\boldsymbol{c}(z_i, z_t)$ are in D, for some i, k, t. Then, we have $h_t = k$, $M(i, t) = c$, and $i < k = h_t$ by definitions of D^Q_n, D^Σ_n and $D^{\boldsymbol{N}^+}_n$, respectively. Thus, by definition of a run, $M(i, t + 1) = M(i, t) = c$, so $\boldsymbol{c}(z_i, z_{t+1}) \in D$, and $R^L_{q,c}$ is satisfied.

$R^{R}_{q,c}$**:** Assume that the atoms $\boldsymbol{N}^+(z_i, z_k)$, $\boldsymbol{N}(z_t, z_{t+1})$, $\boldsymbol{q}(z_i, z_t)$ and $\boldsymbol{c}(z_k, z_t)$ are in D, for some i, k, t. Then, we have $h_t = i$, $M(k, t) = c$, and $h_t = i < k$ by definitions of D^Q_n, D^Σ_n and $D^{\boldsymbol{N}^+}_n$, respectively. Thus, by definition of a run, $M(k, t + 1) = M(k, t) = c$, so $\boldsymbol{c}(z_k, z_{t+1}) \in D$, and $R^R_{q,c}$ is satisfied.

$R^{+}_{\boldsymbol{func}}$**:** Assume that $\boldsymbol{N}(z_i, z_j)$ and $\boldsymbol{N}(z_i, z_k)$ are in D, for some i, j, k. Then, by the definition of $D^{\boldsymbol{N}}_n$, we have $j = k = i + 1$. Thus, since $\texttt{Eq}(z_{i+1}, z_{i+1}) \in D$, this rule is satisfied.

$R^{-}_{\boldsymbol{func}}$**:** Assume that $\boldsymbol{N}(z_i, z_k)$ and $\boldsymbol{N}(z_j, z_k)$ are in D, for some i, j, k. Then, by the definition of $D^{\boldsymbol{N}}_n$, we have $i = j = k - 1$. Thus, since $\texttt{Eq}(z_{k-1}, z_{k-1}) \in D$, this rule is satisfied.

$R^{\boldsymbol{cycle}}_{\boldsymbol{N}^+}$**:** By definition of $D^{\boldsymbol{N}^+}_n$, there is no x such that $\boldsymbol{N}^+(x, x) \in D$, so this rule is not applicable.

R_{q_0}**:** By the definition of D^Q_n and a Turing machine, there can only be one term that appears in a $\boldsymbol{q_0}$-atom, which is z_0. Thus, if $\boldsymbol{q_0}(z_i, z_j)$ and $\boldsymbol{q_0}(z_k, z_l)$ are in D, then $i = j = k = l = 0$. Since $\texttt{Eq}(z_0, z_0) \in D$, this rule is satisfied.

R_{-1}**:** Again, the only x such that $\boldsymbol{q_0}(x, x) \in D$ is z_0. Thus, by definition of $\boldsymbol{N}$, there is no i such that $\boldsymbol{N}(z_i, z_0) \in D$, so this rule is not applicable.

$R^{\boldsymbol{step}}_{q_1,q_2}$**:** Assume that $\boldsymbol{q_1}(z_i, z_t)$ and $\boldsymbol{q_2}(z_j, z_t)$ are in D, for some q_1, q_2, i, j, t. By definition of D^Q_n, we then have $i = j = h_t$. Thus, by definition of $D^{\boldsymbol{N}^+}_n$, we do not have $\boldsymbol{N}^+(z_{h_t}, z_{h_t})$, so this rule is never applicable (and thus satisfied).

$R^{\boldsymbol{cell}}_{q_1,q_2}$**:** Assume that $\boldsymbol{q_1}(z_i, z_t)$ and $\boldsymbol{q_2}(z_j, z_t)$ are in D, for some q_1, q_2, i, t. By definition of D^Q_n, we then have $q_1 = q_2 = q_t$, so $R^{cell}_{q_1,q_2}$ is not in $\mathcal{R}_M$ (as it requires $q_1 \neq q_2$). Thus, by contrapositive, no rule $R^{cell}_{q_1,q_2}$ in $\mathcal{R}_M$ is applicable.

$R^{\boldsymbol{cell}}_{c_1,c_2}$**:** Let c_1, c_2, i, t be such that $\boldsymbol{c_1}(z_i, z_t)$ and $\boldsymbol{c_2}(z_j, z_t)$ are in D. By definition of D^{Σ}_n, we then have $c_1 = c_2 = M(i,t)$, so $R^{cell}_{c_1,c_2}$ is not in $\mathcal{R}_M$ (as it requires $c_1 \neq c_2$). Thus, by contrapositive, no rule $R^{cell}_{c_1,c_2}$ in $\mathcal{R}_M$ is applicable.

$R^{\boldsymbol{OoB}}_{c}$**:** Let c, i, t be such that $\boldsymbol{c}(z_i, z_t)$ and $\boldsymbol{N}^+(z_t, z_i)$ are in D. Be definition of $D^{N^+}_n$, we then have $t < i$, and by definition of D^{Σ}_n, we have $i \leq t$, which is a contradiction.

$R_{\texttt{Fail}}$**:** As $\texttt{Fail} \notin D$ this rule is not applicable and thus satisfied.

$\{R^{trans}_{\texttt{Eq}}, R^{sym}_{\texttt{Eq}}, R^{repl}_{P,i}\}$ **:** As all the $\texttt{Eq}$-atoms in D are of the form $\texttt{Eq}(x,x)$, by Lemma 14, no trigger $t = (R, \pi)$ such that $\mathit{body}(R)$ features an $\texttt{Eq}$-atom in its body is applicable on D. As this is the case for these three rules, they are satisfied.

As D satisfies all the Datalog rules in $\mathcal{R}_M$, none of these rules is applicable on D. Since $D \subseteq \mathit{res}(\mathcal{D}'_n)$, we indeed have $\mathit{res}(\mathcal{D}'_n) = D$, which concludes the proof. □

As the result of all derivations from $(\mathcal{R}_M, \emptyset)$ is characterized by this proposition, Theorem 13, which we recall here, is easy to prove.

Theorem 13. *The Turing machine M halts on the empty word if and only if $(\mathcal{R}_M, \emptyset) \models \texttt{Goal}$.*

Proof. Consider a Datalog-first derivation $\mathcal{D} = t_0, \mathcal{D}_0, t_1, \mathcal{D}_1, \ldots$ from $(\mathcal{R}_M, \emptyset)$. If $(\mathcal{R}_M, \emptyset) \models \texttt{Goal}$, then there is $n \in \mathbb{N}$ such that $\texttt{Goal} \in \mathit{res}(\mathcal{D}'_n)$. By Proposition 16, $\mathit{res}(\mathcal{D}'_n)$ contains $\texttt{Goal}$ if and only if $n \geq \mathit{time}_{\varepsilon}(M)$. Thus, $\mathit{time}_{\varepsilon}(M)$ is finite, and M halts. If M halts, then $\mathit{time}_{\varepsilon}(M)$ is finite. Thus, $\mathit{res}(\mathcal{D}_{\mathit{time}_{\varepsilon}(M)})$ contains $\texttt{Goal}$ by Proposition 16, so $(\mathcal{R}_M, \emptyset) \models \texttt{Goal}$. □

3.3 $\mathcal{R}_M$ is Datalog expressible

We now show the following theorem.

Theorem 17. *The pair $(\mathcal{R}_M, \texttt{Goal})$ is Datalog-expressible.*

If M halts on the empty word, then $(\mathcal{R}_M, \emptyset) \models \texttt{Goal}$ by Theorem 13, and since entailment w.r.t. first-order theories is monotonic, $(\mathcal{R}_M, D) \models \texttt{Goal}$ for all databases D. Thus, the query $(\{\rightarrow \texttt{Goal}\}, \texttt{Goal})$ is a rewriting of $\mathcal{R}_M$. Now, if M does not halt, while $\mathcal{R}_M$ does not entail $\texttt{Goal}$ on $\emptyset$, there are databases on which $\mathcal{R}_M$ entails goal, even if we only consider the rules of $\mathcal{R}_M$ up to $R^R_{q,c}$, that were used for the simulation. For instance, if D already contains $\texttt{Goal}$, or a $\boldsymbol{q_f}$-atom, then $(\mathcal{R}_M, \emptyset) \models \texttt{Goal}$, but there are also more interesting (and problematic to find a Datalog rewriting for) databases on which the first part of $\mathcal{R}_M$ would entail $\texttt{Goal}$ too.

We call databases on which the first part of $\mathcal{R}_M$ entails $\texttt{Goal}$ even though M does not halt on the empty word *false starts*. To control the structure of the database and to detect these false starts, we introduce a new predicate $\texttt{Fail}$, and the rules in the second part of $\mathcal{R}_M$ (from R^+_{func} onwards). Then, the argument revolves around the notion of Skolem depth of a term.

3.3.1 Skolem depth

Definition 18. *Consider a database D, a rule set $\mathcal{R}$, and a derivation $\mathcal{D} = D_0, t_1, D_1, \dots$ from $(\mathcal{R}, D)$. For all n, the* Skolem depth *$SkD(x)$ of a term x appearing in D_n is defined inductively as the following:*

- *If x is a constant, then its Skolem depth is 0.*
- *If x is a variable introduced by a trigger t, then its Skolem depth is $\max(\{SkD(y) \mid y \in support(t)\}) + 1$.*

Then, the Skolem depth $SkD(E)$ of a set E of terms is $\max(\{SkD(y) \mid y \in E\})$. The Skolem depth of an atom or a trigger (or a set thereof) is then the Skolem depth of the set of variables appearing in it.

The rough idea behind this definition is that existential rules can be "Skolemized" by replacing their existential quantifiers with Skolem functions, and the Skolem depth of a term is the maximal depth of nested Skolem functions. This notion is interesting here because of Theorem 7 from [4], that we restate here.

Definition 19. *A couple rule set-query has* bounded depth *if there is k such that for all databases D, the following are equivalent:*

- *$(\mathcal{R}, D) \models q$.*
- *There is a finite derivation $\mathcal{D}$ such that $res(\mathcal{D}) \models q$ and all the terms in $res(\mathcal{D})$ are of Skolem depth less than k.*

Theorem 20. *If $(\mathcal{R}, q)$ has bounded depth, then it is Datalog-expressible.*

Thus, in the following, we show that $(\mathcal{R}_M, \texttt{Goal})$ has a bounded depth, and that the bound is one. To do so, we distinguish two cases: the databases that entail `Fail` and those that do not. For the databases that entail `Fail`, we show that they do it fast: we only need to close under Datalog to see if a database entails `Fail`. For the other databases, though, we show that (if `Goal` is not in them), then the body of rules that generate `Goal` cannot contain terms of Skolem depth greater than one. However, this is not enough to conclude: it could be the case that, for instance, there is a derivation that can entail $\boldsymbol{q_f}(x, y)$ with x and y constants, but that this derivation needs terms of Skolem depth greater than one to generate this atom. Thus, we introduce the notion of Skolem-increasing derivation, a restriction of Datalog-first derivations in which after closing under Datalog, we apply the existential trigger that create the variable with the smallest Skolem depth.

Definition 21. *A* Skolem-increasing derivation *is a Datalog-first derivation $\mathcal{D} = \mathcal{D}_{init}, t_0, \mathcal{D}_0, \dots$ such that for all n, there is no existential trigger t applicable on $res(\mathcal{D}_n)$ such that $SkD(t) < SkD(t_{n+1})$.*

Note that Skolem-increasing derivations always exist, as the property that characterizes them is only a criterion for choosing which existential trigger to apply. We do not require for the sequence $(SkD(t_i))_{i \in \mathbb{N}}$ to be increasing, even though we will see that it is indeed the case for our rule set. We also show another important property, which is that to generate an atom featuring terms of Skolem depth less than n, then the sequence of trigger applications that leads to it only features terms of depth less than n.

Proposition 22. *Let D be a database and $\mathcal{D} = \mathcal{D}_0, t_0, \mathcal{D}_1, \dots$ a Datalog-first derivation from $(\mathcal{R}_M, D)$. Then, the following holds for all terms x and y in $res(\mathcal{D})$:*

1. *If $\boldsymbol{N}(x, y) \in res(\mathcal{D})$, then either $SkD(x) = SkD(y) = 0$ or $SkD(y) = SkD(x) + 1$.*
2. *If $\boldsymbol{N}^+(x, y) \in res(\mathcal{D})$, then $SkD(x) \leq SkD(y)$.*
3. *For all states q, if $\boldsymbol{q}(x, y) \in res(\mathcal{D})$, then $SkD(x) \leq SkD(y)$.*
4. *For all triggers t in $\mathcal{D}$ and atoms $A \in output(t)$ not using $\texttt{Eq}$, we have $SkD(support(t)) \leq SkD(A)$.*

To show this, we first prove a few results about the $\texttt{Eq}$ predicate. Let us first describe the role of the rules in which $\texttt{Eq}$ appears.

3.3.2 An axiomatization of equality

First, rules $R^{trans}_{\texttt{Eq}}$, $R^{sym}_{\texttt{Eq}}$ and $R^{repl}_{P,i}$ constitute an axiomatization of non-reflexive equality. Then, rules R^{+}_{func} and R^{-}_{func} ensure that $\boldsymbol{N}$ is both functional and anti-functional, and rule R_{q_0} ensures that there is only one $\boldsymbol{q_0}$-atom. We have already seen in Lemma 14 that making equality reflexive would not yield any new fact (and it is complicated to do properly with existential rules). Thus, we define an equivalence relation $\approx$ which is the reflexive closure of the $\texttt{Eq}$ predicate, and we show that this relation is indeed an axiomatization of equality.

Lemma 23. *Let D be a Datalog closed database w.r.t $\mathcal{R}_M$. The relation $\approx$ defined as:*

$$x \approx y \quad \text{iff} \quad \texttt{Eq}(x, y) \in D \text{ or } x = y$$

is an equivalence relation. In addition, for all Datalog closed instances D, predicates P of arity n and variables $x_1, \dots, x_n, y_1, \dots, y_n$, if $x_i \approx y_i$ for all $i \leq n$, then $P(x_1, \dots, x_n) \in D$ if and only if $P(y_1, \dots, y_n)$.

Proof. Let us first prove that $\approx$ is an equivalence relation.

Symmetry: Let x and y be terms in D such that $x \approx y$. If $x = y$, then $y = x$ so $y \approx x$. Otherwise, it means $\texttt{Eq}(x, y) \in D$. In this case, since D satisfies rule $R^{sym}_{\texttt{Eq}}$, $\texttt{Eq}(y, x) \in D$. Thus, $y \approx x$.

Reflexivity: If x is a term in D, then $x = x$ so $x \approx x$.

Transitivity: Let x, y and z be terms in D such that $x \approx y$ and $y \approx z$. Then:

- If $x = y$, then since $y \approx z$, we have $x \approx z$.
- If $y = z$, then since $x \approx y$, we have $x \approx z$.
- Otherwise, we have both $\texttt{Eq}(x, y) \in D$ and $\texttt{Eq}(y, z) \in D$. Since D is a model of rule $R^{trans}_{\texttt{Eq}}$, it means $\texttt{Eq}(x, z) \in D$, so $x \approx z$.

Thus, $\approx$ is indeed an equivalence relation. Now, for the second part of the lemma, assume that $P(x_1, \dots, x_n) \in D$. We then show by induction on i that $P(y_1, \dots, y_i, x_{i+1}, \dots, x_n) \in D$.

If $i = 0$, by hypothesis we have $P(x_1, \ldots, x_n) \in D$, so there is nothing to prove. Now, assume that $i > 0$ and $P(y_1, \ldots, y_{i-1}, x_i, \ldots, x_n) \in D$. If $x_i = y_i$, then the result follows immediately from the induction hypothesis. Otherwise, we have $\texttt{Eq}(x_i, y_i) \in D$. Thus, since D is a model of rule $R^{repl}_{P,i}$, we have $P(y_1, \ldots, y_i, x_{i+1}, \ldots, x_n) \in D$.

In particular, this result is true for $i = n$, meaning that $P(y_1, \ldots, y_n) \in D$. □

We then consider the quotient of Datalog closed instances, and show that this quotient is also Datalog closed.

Definition 24. *For D a Datalog closed instance w.r.t. $\mathcal{R}_M$, we denote by $[x]$ the* class *of a term x in D for the equivalence relation $\approx$, and define $D/\approx$ the quotient of D by $\approx$ as the database such that:*

- *Its terms are the classes $[x]$ for all $x \in D$.*
- *We have $P([x_1], \ldots, [x_n]) \in D/\approx$ if and only if $P(x_1, \ldots, x_n) \in D$.*

Lemma 25. *If D is a Datalog closed database w.r.t. $\mathcal{R}_M$, then $D/\approx$ satisfies all the Datalog rules in $\mathcal{R}_M$ that do not feature an* Eq*-atom in their heads.*

Proof. Let $R = B(\vec{x}, \vec{y}) \to H(\vec{x})$ be a rule in $\mathcal{R}_M$ not featuring an Eq-atom in its head, and π be a homomorphism from $B(\vec{x}, \vec{y})$ to D_{SC}. As $\pi(B(\vec{x}, \vec{y})) \in D/\approx$, if $\pi(\vec{x}) = \vec{c}$ and $\pi(\vec{y}) = \vec{d}$, then for all $\vec{u} \in \vec{c}$ and $\vec{v} \in \vec{d}$, we have $B(\vec{u}, \vec{v}) \in D$. Thus, since D models R, we also have $H(\vec{u}) \in D$, so $H(\vec{c}) = \pi(H(\vec{x})) \in D_{SC}$, and D_{SC} satisfies R. □

Note that Lemma 23 and Lemma 25 are independent of the rules in $\mathcal{R}_M$ other that our axiomatization, and could be used in any rule set in which the same axiomatization of equality is. Though, since rule $R^{repl}_{P,i}$ can create any atom, if we want to show that Fail cannot appear too late in our context, we need to show that the rules we just described do not equalize too many individuals. This informal intuition calls for the following result, and especially point (4).

Lemma 26. *Let D be a database, and $\mathcal{D} = D_0, t_0, D_1, t_1, D_2, \ldots$ a Datalog-first restricted derivation from $(\mathcal{R}_M, D)$ (with $D_0 = D$), and $\mathcal{D}_0$ the prefix of $\mathcal{D}$ composed only of Datalog triggers. Then, the following claims hold for all n:*

1. *All the atoms $\boldsymbol{N}(v, w)$ in D_n with w constant are in $res(\mathcal{D}_0)$.*
2. *If w is the variable created by an application of rule R_0, there is no v such that $\boldsymbol{N}(v, w) \in D_n$.*
3. *All $\boldsymbol{q_0}$-atoms in D_n either feature only constant and are in $res(\mathcal{D}_0)$, or feature a variable and are created by R_0.*
4. *If $\texttt{Eq}(x, y) \in D_n \setminus res(\mathcal{D}_0)$, then $x = y$.*

Proof. We show the claim by induction on n. The base case is trivial. Indeed, in D there is no variable so (2) holds, and everything that is in D is in $res(\mathcal{D}_0)$, so the other points hold too. We then prove the induction step. Assume that all the points are true at step $n - 1$, and that $t_n = (R_n, \pi_n)$.

First, regarding point (1), assume by contradiction that t_n creates an atom $\boldsymbol{N}(v, w)$ with v any term and w a constant. Then, the rule R_n must be either $R^{repl}_{\boldsymbol{N},1}$ or $R^{repl}_{\boldsymbol{N},2}$, or w would be a variable. If R_n is $R^{repl}_{\boldsymbol{N},1}$, then the atoms $\boldsymbol{N}(u, w)$ and $\texttt{Eq}(u, v)$ must be in D_{n-1}, with u a term such that $u \neq v$. Then, by point (4) of the induction hypothesis, we have $\texttt{Eq}(u, v) \in \mathit{res}(\mathcal{D}_0)$. In addition, $\boldsymbol{N}(u, w)$ is an $\boldsymbol{N}$-atom with a constant on second position in D_{n-1}, so by point (1) of the induction hypothesis, we also have $\boldsymbol{N}(u, w) \in \mathit{res}(\mathcal{D}_0)$. Finally, since $\mathit{res}(\mathcal{D}_0)$ models rule $R^{repl}_{\boldsymbol{N},1}$, we have $\boldsymbol{N}(v, w) \in \mathit{res}(\mathcal{D}_0)$, which contradicts our hypothesis, so rule $R^{repl}_{\boldsymbol{N},1}$ cannot generate $\boldsymbol{N}(v, w)$. An analogous argument proves that rule $R^{repl}_{\boldsymbol{N},2}$ cannot do so either, concluding the proof of this point.

Regarding point (2), assume that w is the variable introduced by an application of rule R_0, and that t_n creates the atom $\boldsymbol{N}(v, w)$. First, rules R_1 and R_{+1} both create an $\boldsymbol{N}$-atom with an existentially quantified variable in its second position. Since w has already been introduced, it cannot be introduced again by these two rules. Rule $R^{repl}_{\boldsymbol{N},1}$ requires an atom $\boldsymbol{N}(u, w)$ to be applicable, which is impossible as $\boldsymbol{N}(v, x)$ is the first with w in second position. Finally, rule $R^{repl}_{\boldsymbol{N},2}$ requires an $\texttt{Eq}$-atom involving w to be applicable, which by induction hypothesis does not exist, which concludes this point.

To show point (3), we first treat the constant case. Assume that t_n creates $\boldsymbol{q_0}(v, w)$ with v and w constants. Then, as it only features constants, it must have been generated by rule $R^{repl}_{\boldsymbol{q_0},1}$ or $R^{repl}_{\boldsymbol{q_0},2}$. If $R_n = R^{repl}_{\boldsymbol{q_0},1}$, then there are $\boldsymbol{q_0}(u, w)$ and $\texttt{Eq}(u, v)$ in D_n, with a term u such that $u \neq v$. Then, by point (4) of the induction hypothesis, we have $\texttt{Eq}(u, v) \in \mathit{res}(\mathcal{D}_0)$. In addition, since $\texttt{Eq}(u, v) \in \mathit{res}(\mathcal{D}_0)$, u is a constant, so $\boldsymbol{q_0}(u, w)$ is a $\boldsymbol{q_0}$-atom over constants. Thus, by point (3) of the induction hypothesis, $\boldsymbol{q_0}(u, w) \in \mathit{res}(\mathcal{D}_0)$. Then, since $\mathit{res}(\mathcal{D}_0)$ is closed under rule $R^{repl}_{\boldsymbol{q_0},1}$, then $\boldsymbol{q_0}(v, w) \in \mathit{res}(\mathcal{D}_0)$. An analogous argument holds if $R_n = R^{repl}_{\boldsymbol{q_0},2}$, which concludes the constant case.

Then, for the variable case of point (3), assume that t_n creates the atom $\boldsymbol{q_0}(v, w)$ with v or w a variable, and that $R_n \neq R_0$. If $R_n = R^{repl}_{\boldsymbol{q_0},1}$, then there must be $\boldsymbol{q_0}(u, w)$ and $\texttt{Eq}(u, v)$ in D_{n-1}, with u a term such that $u \neq v$. Thus, u and v must be constants. Since we assume v or w is a variable, it must be that w is a variable. Thus, by induction hypothesis, the atom $\boldsymbol{q_0}(u, w)$ must have been created by rule R_0, so $u = w$, and then u is both a constant and a variable, which is absurd. An analogous argument concludes this point if $R_n = R^{repl}_{\boldsymbol{q_0},2}$.

Then, to prove point (4), assume by contradiction that t_n is not in $\mathit{res}(\mathcal{D}_0)$, and creates an atom $\texttt{Eq}(x, y)$ with $x \neq y$ two terms. We then do a case distinction on R_n.

If $R_n = R^{+}_{\boldsymbol{func}}$: In this case, there are atoms $\boldsymbol{N}(z, x)$ and $\boldsymbol{N}(z, y)$ in D_{n-1} with z a term, so that t is applicable. If x and y are constant, then both $\boldsymbol{N}(z, x)$ and $\boldsymbol{N}(z, y)$ are in $\mathit{res}(\mathcal{D}_0)$ by point (1) of the induction hypothesis, so by rule R^{+}_{func}, $\texttt{Eq}(x, y) \in \mathit{res}(\mathcal{D}_0)$, contradicting our hypothesis. Then, for the case where x or y is a variable, assume that x is a variable (x and y are symmetric in this case), and let $t_k = (R_k, \pi_k)$ be the trigger that creates x.

If $R_k = R_0$, then by point (2), the atom $\boldsymbol{N}(z, x)$ cannot be in D_{n-1}. Thus, R_k is either R_1 or R_{+1}. In either case, t_k creates the atom $\boldsymbol{N}(v, x)$ along with x. We show that there is no atom $\boldsymbol{N}(w, y)$ in D_{n-1} with $w \in [v]$.

Assume by contradiction that the atom $\boldsymbol{N}(w, y)$ is the first atom of this form created. We then study the rule that introduces $\boldsymbol{N}(w, y)$.

If the rule that creates $\boldsymbol{N}(w, y)$ is R_1 or R_{+1}, then there is l such that t_l introduces $\boldsymbol{N}(w, y)$, and $l \neq k$. If $l < k$, then $res(\mathcal{D}_l) \subseteq res(\mathcal{D}_{k-1})$. As $res(\mathcal{D}_l)$ is Datalog closed, by Lemma 23, $\boldsymbol{N}(v, y) \in res(\mathcal{D}_l)$, so t_k is not applicable, which contradicts our hypothesis. If $l > k$, as $res(\mathcal{D}_k)$ is Datalog closed, by Lemma 23, $\boldsymbol{N}(w, x) \in res(\mathcal{D}_k)$, so t_l is not applicable. Thus, these rules cannot create $\boldsymbol{N}(w, y)$.

Rule $R^{repl}_{\boldsymbol{N},2}$ requires an $\texttt{Eq}$-atom involving y to produce $\boldsymbol{N}(w, y)$, so y has to be a constant. Then, by point (1), the atom $\boldsymbol{N}(v, y)$ is in $res(\mathcal{D}_0)$. Since $res(\mathcal{D}_0)$ is Datalog closed, by Lemma 23, we have $\boldsymbol{N}(v, y) \in res(\mathcal{D}_0)$. Thus, t_k is not applicable, as one can map x to y.

For rule $R^{repl}_{\boldsymbol{N},1}$ to produce $\boldsymbol{N}(w, y)$, there must be $\boldsymbol{N}(u, y)$ and $\texttt{Eq}(u, w)$ in D_{n-1} before it. Thus, $u \in [v]$, so since $\boldsymbol{N}(w, y)$ is the first one, this rule cannot create it either.

Then, there is no atom $\boldsymbol{N}(w, y)$ in D_{n-1} with $w \in [v]$. As $v \in [v]$, the atom $\boldsymbol{N}(v, y)$ is not in D_{n-1} either, so t is not applicable in this case.

We can now show that for any z, the atoms $\boldsymbol{N}(z, x)$ and $\boldsymbol{N}(z, y)$ cannot exist at the same time. If the rule that generates $\boldsymbol{N}(z, x)$ is R_1 or R_{+1}, then $z = v$ and we have shown that for all $w \in [v]$, including v itself, $\boldsymbol{N}(w, y) \notin D_{n-1}$. Then, $\boldsymbol{N}(z, y) \notin D_{n-1}$, so these rules cannot create $\boldsymbol{N}(z, x)$. If this rule is $R^{repl}_{\boldsymbol{N},1}$, then $z \approx v$, and by the same result $\boldsymbol{N}(z, y) \notin D_{n-1}$. Finally, rule $R^{repl}_{\boldsymbol{N},2}$ requires an $\texttt{Eq}$-atom over x, a variable, so it cannot be used either. Thus, the atoms $\boldsymbol{N}(z, x)$ and $\boldsymbol{N}(z, y)$ cannot exist at the same time.

If $R_n = R^-_{\boldsymbol{func}}$: In this case, there are atoms $\boldsymbol{N}(x, z)$ and $\boldsymbol{N}(y, z)$ in D_{n-1} with z a term, so that t is applicable. First notice that if z is a constant, by point (1), $\boldsymbol{N}(x, z)$ and $\boldsymbol{N}(y, z)$ are in $res(\mathcal{D}_0)$. Thus, the closure under rule R^-_{func} entails that $\texttt{Eq}(x, y) \in res(\mathcal{D}_0)$, which contradicts our hypothesis. Thus, z must be a variable, so the atoms $\boldsymbol{N}(x, z)$ and $\boldsymbol{N}(y, z)$ must have been created during the chase. The term z being a variable also means that $R^{repl}_{\boldsymbol{N},2}$ cannot create any atom $\boldsymbol{N}(v, z)$, as in this case z would be a constant. Let $t_k = (R_k, \pi_k)$ be the trigger that created z.

If $R_k = R_0$, then by point (2), the atom $\boldsymbol{N}(x, z)$ cannot be in D_{n-1}. Thus, R_k must be either R_1 or R_{+1}. In either case, t_k creates the atom $\boldsymbol{N}(v, z)$. We show that for all $w \notin [v]$, the atom $\boldsymbol{N}(w, z)$ is not in D_{n-1}. Assume by contradiction that $\boldsymbol{N}(w, z)$ is the first of these atoms to appear in D_{n-1}. As z is a variable created along the atom $\boldsymbol{N}(v, z)$, only rules $R^{repl}_{\boldsymbol{N},1}$ and $R^{repl}_{\boldsymbol{N},2}$ can generate the atom $\boldsymbol{N}(w, z)$. As we saw earlier, rule $R^{repl}_{\boldsymbol{N},2}$ cannot be used, so there is only rule $R^{repl}_{\boldsymbol{N},1}$. This rule requires the atoms $\boldsymbol{N}(u, z)$ and $\texttt{Eq}(u, w)$ to be in D_{n-1} for it to be applicable (for some u). As by hypothesis $\boldsymbol{N}(w, z)$ is the first one with $w \notin [v]$, we have $u \in [v]$ so $u \approx v$, but since we have $\texttt{Eq}(u, w)$, we also have $u \approx w$. Thus, by transitivity, we have $w \approx v$, so $w \in [v]$, which is a contradiction. Thus, the only w such that $\boldsymbol{N}(w, z) \in D_{n-1}$ are in $[v]$. Then, if we have $\boldsymbol{N}(x, z)$ and $\boldsymbol{N}(y, z)$

in D_{n-1}, then $x \approx v \approx y$ so as $x \neq y$, we have $\texttt{Eq}(x, y) \in D_{n-1}$, which contradicts our hypothesis.

If $R_n = R_{q_0}$: In this case, in D_{n-1} there are either the atoms $\boldsymbol{q_0}(x, y)$ and $\boldsymbol{q_0}(v, w)$, or the atoms $\boldsymbol{q_0}(v, x)$ and $\boldsymbol{q_0}(y, w)$ (with v and w terms). In the first case, as $x \neq y$, the terms x and y must both be constants. Then, $\boldsymbol{q_0}(x, y) \in \mathit{res}(\mathcal{D}_0)$ by point (3). Thus, since $\mathit{res}(\mathcal{D}_0)$ is closed under rule R_{q_0}, we have $\texttt{Eq}(x, y) \in \mathit{res}(\mathcal{D}_0)$, contradicting our hypothesis. Thus, there must be atoms $A_x = \boldsymbol{q_0}(v, x)$ and $A_y = \boldsymbol{q_0}(y, w)$ in D_{n-1}. By point (3), there are four cases possible.

- If A_x and A_y both only feature constants, they are both in $\mathit{res}(\mathcal{D}_0)$, so rule R_{q_0} ensures that $\texttt{Eq}(x, y)$ is in $\mathit{res}(\mathcal{D}_0)$ too, which contradicts our hypothesis.
- If A_x features constants and A_y a variable, then by rule R_{q_0}, we have $\texttt{Eq}(v, x) \in \mathit{res}(\mathcal{D}_0)$, and by rule $R^{repl}_{\boldsymbol{q_0},1}$, the atom $\boldsymbol{q_0}(x, x)$ is in $\mathit{res}(\mathcal{D}_0)$. Thus, rule R_0 is not applicable (one can map y to x), and A_y cannot appear.
- The case where A_y features constants and A_x a variable is symmetric to the previous one.
- Finally, if A_x and A_y both feature variables, then they are both created by rule R_0. As it has an empty body, it can be applied at most once, so as $x \neq y$, this case is not possible either. □

3.3.3 Skolem-increasing derivations

We are now equipped with the notions we need to show Proposition 22, which we recall here.

Proposition 22. *Let D be a database and $\mathcal{D} = \mathcal{D}_0, t_0, \mathcal{D}_1, \ldots$ a Datalog-first derivation from $(\mathcal{R}_M, D)$. Then, the following holds for all terms x and y in $\mathit{res}(\mathcal{D})$:*

1. *If $\boldsymbol{N}(x, y) \in \mathit{res}(\mathcal{D})$, then either $\mathit{SkD}(x) = \mathit{SkD}(y) = 0$ or $\mathit{SkD}(y) = \mathit{SkD}(x) + 1$.*
2. *If $\boldsymbol{N}^+(x, y) \in \mathit{res}(\mathcal{D})$, then $\mathit{SkD}(x) \leq \mathit{SkD}(y)$.*
3. *For all states q, if $\boldsymbol{q}(x, y) \in \mathit{res}(\mathcal{D})$, then $\mathit{SkD}(x) \leq \mathit{SkD}(y)$.*
4. *For all triggers t in $\mathcal{D}$ and atoms $A \in \mathit{output}(t)$ not using $\texttt{Eq}$, we have $\mathit{SkD}(\mathit{support}(t)) \leq \mathit{SkD}(A)$.*

Proof. For point (1), assume for a contradiction that there is a first atom $\boldsymbol{N}(x, y) \in \mathit{res}(\mathcal{D})$ such that neither $\mathit{SkD}(x) = \mathit{SkD}(y) = 0$ nor $\mathit{SkD}(y) = \mathit{SkD}(x) + 1$. If y is a constant, then by point (1) of Lemma 26, x must also be a constant so $\mathit{SkD}(x) = \mathit{SkD}(y) = 0$. Otherwise, if y is a variable, then $\boldsymbol{N}(x, y)$ has been generated during the chase. Let $t = (R, \pi)$ be the trigger that generates this atom.

If $R \in \{R_1, R_{+1}\}$: Then t also creates y from x, so $\mathit{SkD}(y) = \mathit{SkD}(x) + 1$.

If $R \in \{R^{repl}_{\boldsymbol{N},1}, R^{repl}_{\boldsymbol{N},2}\}$: Then there is $\boldsymbol{N}(z,t)$ with $z \approx x$ and $t \approx y$. Since y is a variable, it must be that $t = y$ by Lemma 26. As $\boldsymbol{N}(x,y)$ is the first $\boldsymbol{N}$-atom for which the statement does not hold, and $SkD(y) \neq 0$, it must be that $SkD(y) = SkD(z) + 1$. If x is a constant, then z is too so $SkD(z) = 0$ so $SkD(y) = 1$. Thus, as $SkD(x) = 0$, it is indeed the case that $SkD(y) = SkD(x) + 1$. Now, if x is a variable, then $z = x$ so the atom $\boldsymbol{N}(x,y)$ must be used to generate itself, which is a contradiction.

Thus, point (1) holds.

For point (2), assume for a contradiction that there is a first atom $\boldsymbol{N}^+(x,y) \in res(\mathcal{D})$ such that $SkD(x) > SkD(y)$ for some x, y. Then, x must be a variable so $\boldsymbol{N}^+(x,y)$ must have been generated during the chase; let $t = (R, \pi)$ be the corresponding trigger.

If $R = R_{\boldsymbol{N}^+}$: Then the atom $\boldsymbol{N}(x,y)$ is in $res(\mathcal{D})$, so by point (1), $SkD(x) \leq SkD(y)$.

If $R = R^{\boldsymbol{trans}}_{\boldsymbol{N}^+}$: Then there are atoms $\boldsymbol{N}(x,z)$ and $\boldsymbol{N}^+(z,y)$ in $res(\mathcal{D})$. By point (1), we then have $SkD(x) \leq SkD(z)$. In addition, as $\boldsymbol{N}^+(x,y)$ is the first $\boldsymbol{N}^+$-atom such that $SkD(x) > SkD(y)$, we also have $SkD(z) \leq SkD(y)$. Thus, by transitivity, we get $SkD(x) \leq SkD(y)$.

If $R \in \{R^{repl}_{\boldsymbol{N}^+,1}, R^{repl}_{\boldsymbol{N}^+,2}\}$: Then there is an atom $\boldsymbol{N}^+(z,t)$ with $z \approx x$ and $t \approx y$. As x is a variable, $z = t$. Since $\boldsymbol{N}^+(x,y)$ is the first $\boldsymbol{N}^+$-atom such that $SkD(x) > SkD(y)$, $SkD(z) \leq SkD(t)$. Thus, as $x = z$ and x is a variable, t must also be a variable, so $t = y$, which means $\boldsymbol{N}^+(x,y)$ is necessary to generate itself, a contradiction.

Thus, point (2) holds too.

Then, for point (3), assume for a contradiction that there is a first atom $\boldsymbol{q}(x,y) \in res(\mathcal{D})$ such that $SkD(x) > SkD(y)$ for some x, y, q. Then, x must be a variable so $\boldsymbol{q}(x,y)$ must have been generated during the chase; let $t = (R, \pi)$ be the corresponding trigger.

If $R = R_0$: In this case $x = y$ which is not compatible with them having different Skolem depth.

If $R = R^{\leftarrow}_{q,c}$: Then there must be atoms $\boldsymbol{N}(x,z)$ and $\boldsymbol{N}^+(z,y)$ for some term z (among other atoms) in $res(\mathcal{D})$. Thus, by points (1) and (2), $SkD(x) \leq SkD(z) \leq SkD(y)$.

If $R = R^{\rightarrow}_{q,c}$: Then there must be an atom $\boldsymbol{N}^+(x,y)$ in $res(\mathcal{D})$, so by point (2), $SkD(x) \leq SkD(y)$.

If $R = R^{\boldsymbol{max}}_{q,c}$: Then $x = y$ which is not compatible with them having different Skolem depth.

If $R \in \{R^{repl}_{\boldsymbol{q_0},1}, R^{repl}_{\boldsymbol{q_0},2}\}$: Then there must be an atom $\boldsymbol{q}(z,t)$ with $z \approx x$ and $t \approx y$. Note that by Lemma 26, for all terms a and $b \in [a]$, $SkD(a) = SkD(b)$. Thus, as $\boldsymbol{q}(z,t)$ must be in $res(\mathcal{D})$ before $\boldsymbol{q}(x,y)$, $SkD(z) \leq SkD(t)$, so $SkD(x) \leq SkD(y)$.

Thus, such an atom cannot exist, so point (3) holds.

Finally, for point (4), let $t = (R, \pi)$ be a trigger. As rules $R_{\texttt{Goal}}$, $R^0_{q,c}$, $R^{cycle}_{\boldsymbol{N}^+}$, R_{-1}, $R^{step}_{q_1,q_2}$, $R^{cell}_{q_1,q_2}$, $R^{cell}_{c_1,c_2}$, R^{OoB}_c and $R_{\texttt{Fail}}$ do not feature any variable in their heads, if R is among them, point (4) is vacuously true. The same holds for rules R^+_{func}, R^-_{func}, R_{q_0}, $R^{trans}_{\texttt{Eq}}$ and $R^{sym}_{\texttt{Eq}}$, which only introduce `Eq`-atoms. We then show that for all the other rules, point (4) holds by showing that for all the variables in their bodies, there is a variable in each atom of the head that has a greater Skolem depth. We establish the order between the variables' Skolem depths using the previous points.

If $R = R_0$: This rule features no atoms in its body, so the condition is vacuously true.

If $R = R_1$: We have $SkD(w_0) \leq SkD(w_1)$ and w_1 is in the only atom of the head.

If $R = R_{+1}$: We have $SkD(w_0) \leq SkD(w_i) \leq SkD(w_{i+1})$ and w_{i+1} is in the head.

If $R = R_{\boldsymbol{N}^+}$: The same variables appear in the only atoms in the body and the head of this rule.

If $R = R^{\boldsymbol{trans}}_{\boldsymbol{N}^+}$: We have $SkD(w_i) \leq SkD(w_{i+1}) \leq SkD(w_k)$ and w_k is in the head.

If $R = R^0_{\texttt{B}}$: The same variable appear in the only atoms in the body and the head of this rule.

If $R = R_{\texttt{B}}$: We have $SkD(w_0) \leq SkD(w_i)$ and w_i is in the head.

If $R = R^{\leftarrow}_{q,c}$: We have $SkD(p_1) \leq SkD(p_2) \leq SkD(t_1) \leq SkD(t_2)$, and t_2 is in both the atoms in the head.

If $R = R^{\rightarrow}_{q,c}$: We have $SkD(p_1) \leq SkD(p_2) \leq SkD(t_2)$ and $SkD(p_1) \leq SkD(t_1) \leq SkD(t_2)$, so t_2 has the greatest Skolem depth of all these variables, and appears in both atoms in the head.

If $R = R^{\boldsymbol{max}}_{q,c}$: We have $SkD(w_1) \leq SkD(w_2)$, and w_2 appears in the head.

If $R = R^L_{q,c}$: We have $SkD(p_i) \leq SkD(p_k) \leq SkD(t_1) \leq SkD(t_2)$, and t_2 appears in the head.

If $R = R^R_{q,c}$: We have $SkD(p_i) \leq SkD(p_k)$ and $SkD(p_i) \leq SkD(t_1) \leq SkD(t_2)$, and p_k and t_2 both appear in the head.

If $R = R^{repl}_{P,i}$: The variables x_j for $j \neq i$ all appear in the head. Regarding x_i, by point 4, either it is a constant, and y_i is a constant too so they both have a Skolem depth of zero, or it is a variable and $x_i = y_i$. Thus, either way, the head has a greater Skolem depth than the body. □

We are now equipped to show that in Skolem-increasing derivations, all the facts entailed by the knowledge base about terms of depth k are in the database before a term of depth $k + 1$ is produced.

Proposition 27. *Consider a database D, a Skolem-increasing derivation $\mathcal{D} = \mathcal{D}_0, t_1, \mathcal{D}_1$ from $(\mathcal{R}_M, D)$, and an atom A not using* `Eq` *such that $(\mathcal{R}_M, D) \models A$. Then, for all n, if $SkD(A) < SkD(t_{n+1})$, then $A \in \mathcal{D}_n$.*

Proof. Assume for a contradiction that there is an atom A that contradicts the proposition. Then, there must be a sequence of trigger $t^1, \dots, t^k$ appearing in $\mathcal{D}$ such that $t^1 = t_n$, $A \in \mathit{output}(t^k)$, and for all $i < k$, $\mathit{support}(t^{i+1}) \cap \mathit{output}(t^i) \neq \emptyset$. As we saw in the proof of Proposition 22, all the rules either have only one atom in the head or both their atoms feature the variable with the greatest Skolem depth. Thus, for all $i < k$, $SkD(\mathit{output}(t^i)) \leq SkD(\mathit{support}(t^{i+1}))$ by point (4) of Proposition 22, for all $i \leq k$, $SkD(\mathit{support}(t^i)) \leq SkD(\mathit{output}(t^i))$. Thus, both these results together give by transitivity that $SkD(t^1) \leq SkD(t^n)$, which entails that $SkD(t_{n+1}) \leq SkD(A)$, and contradicts our hypothesis. □

Note that, on a side note, this also entails that the sequence $SkD(t_n)$ is increasing. Now that we have this fundamental result, we can go back to our dichotomy about the `Fail` predicate.

3.3.4 Some structure — The `Fail` predicate

First, we describe the rules that generate `Fail`, through a description of what a Datalog closed database without `Fail` looks like. First, rule $R^{cycle}_{\boldsymbol{N}^+}$ ensures that $\boldsymbol{N}$, which is both functional and inverse functional, is without cycles. Thus, the graph formed by the equivalence classes by `Eq` as nodes and $\boldsymbol{N}$-atoms as edges is a disjoint union of what we call chains.

Definition 28. *In a graph $G = (V, E)$, a* chain *C of size n is a sequence of vertices $v_1, \dots, v_n$ such that for all $(v, w) \in E$, either there is i such that $v = v_i$ and $w = v_{i+1}$, or v and w are both not in C.*

Then, rule R_{-1} ensure that the only $\boldsymbol{q_0}$-atom is at the start of a chain. Finally, rules $R^{step}_{q_1,q_2}$, $R^{cell}_{q_1,q_2}$, $R^{cell}_{c_1,c_2}$ and R^{OoB}_c ensure that on each chain, there is only one Turing machine simulation at a step. We formalize this into the following proposition.

Definition 29. *Consider an instance D and a binary predicate P. Then, the P-graph associated with D, denoted with G^P_D, is the graph having as vertices the terms of D, and an edge between x and y if and only if $P(x, y) \in D$.*

Proposition 30. *Let D be a Datalog closed instance that does not contain* `Fail`*. Then:*

- *The graph $G^{\boldsymbol{N}}_{D/\approx}$ is a union of chains.*
- *There is at most one q_0-atom in $D/\approx$, and if there is one, it is of the form $\boldsymbol{q_0}([z_0], [z_0])$ with $[z_0]$ being the first element of a chain. We call this chain the* starting chain*.*

Proof. We first show that $G^{\boldsymbol{N}}_{D/\approx}$ is a union of chain. First, we show that $\boldsymbol{N}$ is functional, that is that for all $c \in D/\approx$, there is at most one $d \in D/\approx$ such that $\boldsymbol{N}(c, d) \in D/\approx$. Indeed, if d_1 and d_2 are such that $\boldsymbol{N}(c, d_1)$ and $\boldsymbol{N}(c, d_2)$ are in $D/\approx$, let $x \in c$, $y_1 \in d_1$ and $y_2 \in d_2$. Then, we have $\boldsymbol{N}(x, y_1)$ and $\boldsymbol{N}(x, y_2)$ in D. Thus, since D satisfies rule R^+_{func}, we have $\mathtt{Eq}(y_1, y_2) \in D$, so $y_1 \approx y_2$,

and $d_1 = d_2$. The same reasoning using rule R^-_{func} shows that $\boldsymbol{N}$ is also inverse functional in $D/\approx$.

Then, to show that $G^{\boldsymbol{N}}_{D/\approx}$ is a union of chains, we must show that it is without cycles. Assume by the absurd that it contains a cycle $c_1, \ldots, c_n$. Let x_i be a representative of c_i for all i. Then, we have $\boldsymbol{N}(x_n, x_0)$ and $\boldsymbol{N}(x_i, x_{i+1})$ in D for all $i < n$. We then show by induction on i that for all i, $\boldsymbol{N}^+(x_{n-i}, x_0) \in D$. For the base case, since D is a model of rule $R_{\boldsymbol{N}^+}$, we have $\boldsymbol{N}^+(x_n, x_0)$ in D. For the induction step, assume that $\boldsymbol{N}^+(x_{n-i+1}, x_0) \in D$. Then, since $\boldsymbol{N}(x_{n-i}, x_{n-i+1}) \in D$ and D models $R^{trans}_{\boldsymbol{N}^+}$, we have $\boldsymbol{N}^+(x_{n-i}, x_0) \in D$, which concludes the inclusion. Thus, in particular for $i = n$, we have $\boldsymbol{N}^+(x_0, x_0) \in D$. Since D models rule $R^{cycle}_{\boldsymbol{N}^+}$, the predicate $\texttt{Fail}$ is in D, which contradicts our initial hypothesis. Thus, $G^{\boldsymbol{N}}_{D/\approx}$ contains no cycle, and is indeed a union of chains.

Now for the second point, if we have $\boldsymbol{q_0}(c_1, c_2) \in D/\approx$ and $\boldsymbol{q_0}(c_3, c_4) \in D/\approx$, let x_i be a representative of c_i for all i, and then we have $\boldsymbol{q_0}(x_1, x_2) \in D$ and $\boldsymbol{q_0}(x_3, x_4) \in D$. As D is a model of R_{q_0}, the atoms $\texttt{Eq}(x_1, x_2)$, $\texttt{Eq}(x_2, x_3)$ and $\texttt{Eq}(x_3, x_4)$ are all in D. Thus, $x_1 \approx x_2 \approx x_3 \approx x_4$, so $c_1 = c_2 = c_3 = c_4$. Thus, there is at most one q_0-atom in $D/\approx$, which is of the form $\boldsymbol{q_0}(c_0, c_0)$. We then show that if it exists, there is no c such that $\boldsymbol{N}(c, c_0) \in D/\approx$. Indeed, if by contradiction we assume there is one, let $x \in c$ and $z_0 \in c_0$. We then have $\boldsymbol{N}(x, z_0) \in D$. Since D models rule R_{-1}, the predicate $\texttt{Fail}$ is entailed, which contradicts our initial hypothesis, and concludes the proof. □

We now are equipped to show that if the Datalog closure of a database D does not contain $\texttt{Fail}$, and $\mathcal{D} = \mathcal{D}_0, t_0, \mathcal{D}_1, \ldots$ is a Datalog-first restricted derivation from $(\mathcal{R}_M, D)$, then $res(\mathcal{D}_n)$ does not contain $\texttt{Fail}$ either. Thus, Proposition 30 applies on $res(\mathcal{D}_n)$ too, for all n. Therefore, this structure is maintained through all derivations from databases that do not entail $\texttt{Fail}$.

Proposition 31. *Consider a database D and a Datalog-first restricted derivation $\mathcal{D} = \mathcal{D}_0, t_1, \mathcal{D}_1, \ldots$. If $\texttt{Fail} \notin \mathcal{D}_0$, then for all n, $\texttt{Fail} \notin \mathcal{D}_n$.*

Proof. We proceed by induction on n. The base case is trivial, so we focus on the induction step. Assume that $\texttt{Fail} \notin \mathcal{D}_{n-1}$. First, as $t_n = (R_n, \pi_n)$ is an existential trigger, R_n must be among R_0, R_1 and R_{+1}. Thus, it follows from its $\mathbb{R}$-applicability that t_n can only either create a starting chain of length one if there is none, or add one element at the end of a chain. Then, by point (4) of Lemma 26 and the fact that no other rule can create $\boldsymbol{q_0}$-atoms or $\boldsymbol{N}$-atoms, this structure cannot change during $\mathcal{D}_n$. Thus, rules $R^{cycle}_{\boldsymbol{N}^+}$ and R_{-1} cannot fire and entail $\texttt{Fail}$.

Regarding the other four rules, they follow the same pattern: if they were applicable on $res(\mathcal{D}_n)$, then their supports must have been introduced at a previous step, which is not possible by induction hypothesis. □

3.3.5 Controlling the chains

We then come back to our result through several lemmas. The main idea is that since Datalog closed databases are composed of chains, we then show that each chain is well-behaved, and that nothing inconvenient happens between chains. We start with the starting chain, by showing that it behaves exactly like the chase on $(\mathcal{R}_M, \emptyset)$.

Lemma 32. *Consider D be a Datalog closed instance that neither contain* `Fail` *nor* `Goal`*, and such that $D/\approx$ has a starting chain. Then, let $\mathcal{D} = t_0, \mathcal{D}_0, t_1, \dots$ be a Datalog-first derivation from $(\mathcal{R}_M, \emptyset)$, and D_{SC} be the instance obtained by removing all the classes in $D/\approx$ that are not on the starting chain, and all the atoms in which such a term appears. Finally, let n be the number of classes on the starting chain. Then, D_{SC} and $res(\mathcal{D}_{n-1})$ are isomorphic.*

Proof. First, by Proposition 30, $G^{\boldsymbol{N}}_{D_{SC}}$ is a chain of size n, and by Proposition 16, $G^{\boldsymbol{N}}_{res(\mathcal{D}_{n-1})}$ is also a chain of size n. We thus denote the elements of $res(\mathcal{D}_{n-1})$ and D_{SC} as z_i and d_i, respectively, where i is z_i and d_i's position in the chain (and ranges from 0 to $n-1$). We then show that the map h that sends z_i to d_i, which is a bijection, is an isomorphism.

Let us start by showing that $h(res(\mathcal{D}_{n-1})) \subseteq D_{SC}$. More precisely, we show by induction on i that for all atoms $A \in res(\mathcal{D}_{n-1})$ featuring only variables among $\{z_j \mid 0 \leq j \leq i\}$, we have $h(A) \in D_{SC}$.

For the base case, the atoms in $res(\mathcal{D}_{n-1})$ featuring only z_0 are $\boldsymbol{q_0}(z_0, z_0)$ and $\mathtt{B}(z_0, z_0)$. Then, $\boldsymbol{q_0}(d_0, d_0) \in D_{SC}$ since D_{SC} is the starting chain in $D/\approx$. Finally, as $\boldsymbol{q_0}(d_0, d_0) \in D_{SC}$, since D_{SC} models rule $R^0_{\mathtt{B}}$ by Lemma 25, we have $\mathtt{B}(d_0, d_0) \in D_{SC}$, which concludes the base case.

For the inductive step, assume that for all atoms $A \in res(\mathcal{D}_{n-1})$ featuring only variables among $\{z_j \mid 0 \leq j \leq i\}$, we have $h(A) \in D_{SC}$, and let A be an atom in $res(\mathcal{D}_{n-1})$ featuring only variables among $\{z_j \mid 0 \leq j \leq i+1\}$. If it only features variables among $\{z_j \mid 0 \leq j \leq i\}$, then the induction hypothesis suffices, so we assume that it features z_{i+1}. As by Proposition 16 we have $res(\mathcal{D}_{n-1})$ characterized, we now exactly which atoms fulfil this specification.

$\boldsymbol{N}(z_i, z_{i+1})$: As we said earlier, the graph $G^{\boldsymbol{N}}_{D_{SC}}$ is a chain, so our naming convention ensures that $\boldsymbol{N}(d_i, d_{i+1}) \in D_{SC}$.

$\boldsymbol{N}^+(z_j, z_{i+1})$ for $j \leq i$: By the previous point, $\boldsymbol{N}(d_i, d_{i+1}) \in D_{SC}$, so by rule $R_{\boldsymbol{N}^+}$, we have $\boldsymbol{N}^+(d_i, d_{i+1}) \in D_{SC}$. Then, if by induction we assume that $\boldsymbol{N}^+(c_{j+1}, d_{i+1}) \in D_{SC}$, as by induction hypothesis $\boldsymbol{N}(c_j, c_{j+1}) \in D_{SC}$, then $\boldsymbol{N}^+(c_j, d_{i+1}) \in D_{SC}$.

$\boldsymbol{q_{i+1}}(z_{h_{i+1}}, z_{i+1})$: As we saw earlier, the atom $N(d_i, d_i + 1)$ is in D_{SC}. Then, as for all t we have $h_t \leq t$, in particular we have $h_i \leq i$, so by induction hypothesis the atoms $q_i(c_{h_i}, d_i)$, $\boldsymbol{N}^+(c_{h_i}, d_{i+1})$ and $\boldsymbol{c}(c_{h_i}, z_i)$ with $c = M(h_i, i)$ are all in D_{SC}. We also have $h_{i+1} \leq i + 1$, so the atom $\boldsymbol{N}(c_{h_{i+1}}, c_{h_i})$ or $\boldsymbol{N}(c_{h_i}, c_{h_{i+1}})$ is in D_{SC}, depending on whether $\delta(q_i, c) = (q_{i+1}, c', \leftarrow)$ or $\delta(q_i, c) = (q_{i+1}, c', \rightarrow)$. Thus, by rules $R^{\leftarrow}_{q,c}$ and $R^{\rightarrow}_{q,c}$, we have $\boldsymbol{q_{i+1}}(c_{h_{i+1}}, d_{i+1})$ and $\boldsymbol{c'}(z_{h_i}, z_{i+1})$ in D_{SC}, with $c' = M(h_i, i + 1)$.

$\boldsymbol{c_j}(z_j, z_{i+1})$ for $j \leq i + 1$, with $c_j = M(j, i + 1)$: By the previous case, if $j = h_i$, this atom is in D_{SC}, so we assume $j \neq h_i$ in the following. The atom $\boldsymbol{N}(z_i, z_i + 1)$ is in D_{SC}, and by induction hypothesis, the atoms $\boldsymbol{q_i}(d_{h_i}, d_i)$ and $\boldsymbol{c_j}(d_j, z_i)$ are in D_{SC}, since for all $j \neq h_i$, $M(j, i) = M(j, i+1)$. Then, if $j < h_i$, we have $\boldsymbol{N}^+(d_j, d_{h_i}) \in D_{SC}$ so by rule $R^L_{q,c}$, the atom $c_j(z_j, z_{i+1})$ is in D_{SC}. If $j > h_i$, we reach the same conclusion using rule $R^R_{q,c}$ instead.

Thus, this concludes the proof that for all i, and all atoms $A \in res(\mathcal{D}_{n-1})$ featuring only variables among $\{z_j \mid 0 \leq j \leq i\}$, we have $h(A) \in D_{SC}$. Thus, in particular, $h(res(\mathcal{D}_{n-1})) \subseteq D_{SC}$.

We then prove that h^{-1} is a homomorphism. Assume by contradiction that A is an atom in D_{SC} such that $h^{-1}(A) \notin \mathit{res}(\mathcal{D}_{n-1})$. First note that we assumed that `Fail` and `Goal` were not in D, so they are not in D_{SC}, and since $G^{\boldsymbol{N}}_{D_{SC}}$ and $G^{\boldsymbol{N}}_{\mathit{res}(\mathcal{D}_{n-1})}$ are both chains, A cannot use the predicate $\boldsymbol{N}$. Then, we show that D_{SC} must contain `Fail`, contradicting our initial hypothesis on D.

If $A = \boldsymbol{N}^+(d_i, d_j)$: As $h^{-1}(A) = \boldsymbol{N}^+(z_i, z_j) \notin \mathit{res}(\mathcal{D}_{n-1})$, it must then be that $j \leq i$. We then show by induction on $i - j$ that if $\boldsymbol{N}^+(z_i, z_j) \in D_{SC}$ with $j \leq i$, then $\texttt{Fail} \in D_{SC}$. For the base case, if $i - j = 0$, then $j = i$. Since D_{SC} models rule $R^{cycle}_{\boldsymbol{N}^+}$, `Fail` must then be in D_{SC}. Otherwise, if $i - j > 0$, then $j < i$, so $\boldsymbol{N}(d_{i-1}, d_i) \in D_{SC}$. Thus, by rule $R^{trans}_{\boldsymbol{N}^+}$, $\boldsymbol{N}(d_j, d_{i-1}) \in D_{SC}$. Then, $(i-1) - j = (i-j) - 1$, so by induction hypothesis, we have $\texttt{Fail} \in D_{SC}$.

If $A = \boldsymbol{q}(d_i, d_j)$: As $\mathit{time}_\varepsilon(M) = \infty$, for all j, we have $\boldsymbol{q_j}(z_{h_j}, z_j) \in \mathit{res}(\mathcal{D}_{n-1})$, so by the previous point, $\boldsymbol{q_j}(d_{h_j}, d_j) \in D_{SC}$. Since $h^{-1}(A) \notin \mathit{res}(\mathcal{D}_{n-1})$, the atom A must be different from $\boldsymbol{q_j}(d_{h_j}, d_j)$. If $h_j \neq i$, then either $\boldsymbol{N}^+(d_{h_j}, d_i)$ or $\boldsymbol{N}^+(d_i, d_{h_j})$ is in D_{SC}. In either case, since D_{SC} models rule $R^{step}_{q_1,q_2}$, `Fail` is in D_{SC}. Otherwise, if $i = h_j$, it must be that $q \neq q_j$. Thus, as D_{SC} models rule $R^{cell}_{q_1,q_2}$, `Fail` is again in D_{SC}.

If $A = \boldsymbol{c}(d_i, d_j)$: First, if $j < i$, then $\boldsymbol{N}^+(z_j, z_i) \in \mathit{res}(\mathcal{D}_{n-1})$, so $\boldsymbol{N}^+(d_j, d_i) \in D_{SC}$. Thus, as D_{SC} models rule R^{OoB}_{c}, `Fail` is in D_{SC}. We then assume that $i \leq j$. As by hypothesis, $\mathit{time}_\varepsilon(M) = \infty$, for all $i \leq j$, the atom $\boldsymbol{c_i}(z_i, z_j)$ is in $\mathit{res}(\mathcal{D}_{n-1})$, with $c_j = M(i, j)$. Thus, by the previous point, $\boldsymbol{c_i}(d_i, d_j) \in D_{SC}$. Since $h^{-1}(A) \notin \mathit{res}(\mathcal{D}_{n-1})$, the atom A must be different from $\boldsymbol{c_i}(d_i, d_j)$, so $c \neq c_i$. Thus, since D_{SC} models rule $R^{cell}_{c_1,c_2}$, `Fail` is in D_{SC}.

Thus, $\mathit{res}(\mathcal{D}_{n-1})$ and D_{SC} are indeed isomorphic. □

Now, since we assume that M does not halt on the empty word, by Theorem 13, we have $(\mathcal{R}_M, \emptyset) \not\models \texttt{Goal}$, so what happens on the starting chain alone cannot entail `Goal`. We then show that the non-starting chains cannot grow much.

Lemma 33. *Let D be a database, and $\mathcal{D} = \mathcal{D}_0, t_0, \mathcal{D}_1, \ldots$ a Datalog-first restricted derivation from $(\mathcal{R}_M, D)$. Then, for all $n \in \mathbb{N}$, if $\mathit{res}(\mathcal{D}_0)$ does not contain* `Fail`*, then:*

1. *If x or y is a variable and $\boldsymbol{N}^+([x], [y]) \in \mathit{res}(\mathcal{D}_n)/\!\approx$, then $[x]$ and $[y]$ are on the same chain.*
2. *All non-starting chains only feature classes of Skolem depth at most one.*

Proof. For point (1), assume for a contradiction that $\boldsymbol{N}^+(x, y)$ is the first $\boldsymbol{N}^+$-atom in $\mathit{res}(\mathcal{D}_n)$ featuring a variable, and such that x and y are two terms on different chains. Then, there is a trigger $t = (R, \pi)$ that created $\boldsymbol{N}^+(x, y)$.

If $R = R_{\boldsymbol{N}^+}$: Then the atom $\boldsymbol{N}(x, y)$ is in $\mathit{res}(\mathcal{D}_n)$, so x and y are on the same chain.

If $R = R^{\boldsymbol{trans}}_{\boldsymbol{N}^+}$: Then the atoms $\boldsymbol{N}(x, z)$ and $\boldsymbol{N}^+(z, y)$ must be in $\mathit{res}(\mathcal{D}_n)$ for some term z. First, since $\boldsymbol{N}(x, z)$ is in $\mathit{res}(\mathcal{D}_n)$, then x and z are on the same chain. It remains to show that z and y are on the same chain.

If z is a variable, then $\boldsymbol{N}^+(z, y)$ is an $\boldsymbol{N}^+$-atom containing a variable created before $\boldsymbol{N}^+(x, y)$, so z and y are on the same chain by hypothesis.

If z is a constant, then by point (1) of Lemma 26, we have $\boldsymbol{N}(x, z) \in \mathit{res}(\mathcal{D}_0)$, so x is a constant. Thus, y has to be a variable, and $\boldsymbol{N}^+(z, y)$ is again an $\boldsymbol{N}^+$-atom containing a variable created before $\boldsymbol{N}^+(x, y)$, so z and y are on the same chain by induction hypothesis.

Thus, in both cases, the result holds.

If $R \in \{R^{\mathit{repl}}_{\boldsymbol{N}^+,1}, R^{\mathit{repl}}_{\boldsymbol{N}^+,2}\}$: Then there must be $\boldsymbol{N}^+(z, t)$ in $\mathit{res}(\mathcal{D}_n)$ with $z \approx x$ and $t \approx y$. Then, if x is a variable, by point (4) of Lemma 26, $[x] = \{x\}$, so $z = x$ and $\boldsymbol{N}^+(z, t)$ is an $\boldsymbol{N}^+$-atom featuring a variable introduced before $\boldsymbol{N}^+(x, y)$, so z and t are on the same chain. The same reasoning applies if y is a variable. Thus, in both cases, z and t are on the same chain, so x and y are on the same chain too.

Thus, no rule can generate such an atom, which then cannot exist, so point (1) holds.

We then prove point (2). Let y be a variable on a non-starting chain, and let $t = (R, \pi)$ be the rule that created it. We then prove that its Skolem depth must be one. As y is on a non-starting chain, we cannot have $\boldsymbol{q_0}(y, y) \in \mathit{res}(\mathcal{D}_n)$, so $R \neq R_0$. In addition, for the same reason, $R \neq R_1$, as otherwise there would be a term x such that $\boldsymbol{q_0}(x, x)$ and $\boldsymbol{N}(x, y)$ are in $\mathit{res}(\mathcal{D}_n)$, so y would be on the starting chain. Thus, it must be that $R = R_{+1}$. Then, there are some terms z_0 and x such that $\boldsymbol{q_0}(z_0, z_0)$, $\boldsymbol{N}^+(z_0, x)$ and $\boldsymbol{N}(x, y)$ are in $\mathit{res}(\mathcal{D}_n)$. As y is on a non-starting chain, z_0 is not on the same chain as y, and thus as x. Thus, by point (1), z_0 and x must both be constants, so $\mathit{SkD}(y) = 1$, concluding the proof. □

Finally, we need to control what happens between chains. What happens between non-starting chains is not an issue as all the terms involved are of Skolem depth at most one by Lemma 33, but if state and character atoms are between a non-starting and a starting chain, there could be a false start that is allowed to grow enough to entail `Goal` using terms of unbounded depth. To prevent this, we show that there can be no atom using a state predicate and terms of Skolem depth greater than one on different chains.

Lemma 34. *Let D be a database, and $\mathcal{D} = \mathcal{D}_0, t_0, \mathcal{D}_1, \ldots$ a Datalog-first restricted derivation from $(\mathcal{R}_M, D)$. Then, for all $n \in \mathbb{N}$, if $\mathit{res}(\mathcal{D}_0)$ does not contain* `Fail`*, and $\boldsymbol{q}(x, y) \in \mathit{res}(\mathcal{D}_n)$ for some state q and terms x, y where x or y is a variable, then x and y are on the same chain.*

Proof. Assume for a contradiction that there is a first atom $\boldsymbol{q}(x, y) \in \mathit{res}(\mathcal{D}_n)$ for some state q and terms x and y on different chains with x or y a variable. This atom must have been generated by some trigger $t = (R, \pi)$, as it features a variable.

If $R = R_0$: Then $x = y$ so x and y are on the same chain.

If $R = R^{\leftarrow}_{q,c}$: Then there are atoms $\boldsymbol{N}(x, z)$ and $\boldsymbol{N}^+(z, y)$ in $res(\mathcal{D}_n)$ for some term z. By point (1) of Lemma 26, if z is a constant, then x has to be a constant too, so y must be a variable. Thus, in all cases, the atom $\boldsymbol{N}^+(z, y)$ features a variable, so z and y must be on the same chain by point (1) of Lemma 33. Thus, as x and z are on the same chain, x and y must be too.

If $R = R^{\rightarrow}_{q,c}$: Then there must be an atom $\boldsymbol{N}^+(x, y)$ in $res(\mathcal{D}_n)$, so x and y must be on the same chain by point (1) of Lemma 33.

If $R = R^{\boldsymbol{max}}_{q,c}$: Then again $x = y$ so x and y are on the same chain.

If $R \in \{R^{repl}_{\boldsymbol{q},1}, R^{repl}_{\boldsymbol{q},2}\}$: Then there must be an atom $\boldsymbol{q}(z, t)$ with $z \approx x$ and $t \approx y$. Since this atom has to come before $\boldsymbol{q}(x, y)$, and x or y is a variable, so either $z = x$ or $t = y$ by (26), z and t must be on the same chain. As being on the same chain is a notion invariant through $\approx$, x and y are on the same chain too.

□

3.3.6 Skolem depth and Datalog-expressibility

We now are fully equipped to show that $(\mathcal{R}_M, \texttt{Goal})$ has bounded depth.

Theorem 35. *If M does not halt on the empty word, then $(\mathcal{R}_M, \texttt{Goal})$ has bounded depth.*

Proof. Let D be a database and $\mathcal{D} = \mathcal{D}_0, t_1, \mathcal{D}_1, \ldots$ any Skolem-increasing derivation. Then, assume that $(\mathcal{R}_M, D) \models \texttt{Goal}$, and let n be the smallest integer such that $\texttt{Goal} \in res(\mathcal{D}_n)$. We then show that all the terms appearing in $res(\mathcal{D}_n)$ are of Skolem depth at most one. Notice that this is enough to show that $(\mathcal{R}_M, \texttt{Goal})$ has bounded depth, as the other direction is trivial.

First, if $n = 0$, as $res(\mathcal{D}_0)$ only contains constants (of Skolem depth zero), the statement holds. Thus, in the following, we assume $\texttt{Goal} \notin res(\mathcal{D}_0)$. As $res(\mathcal{D}_0)$ models rule $R_{\texttt{Fail}}$, this also means that $\texttt{Fail} \notin res(\mathcal{D}_0)$, which by Proposition 31 entails that $(\mathcal{R}_M, D) \not\models \texttt{Fail}$. Let n be the smallest integer such that $\texttt{Goal} \in res(\mathcal{D}_n)$. Then, $\texttt{Goal}$ must have been introduced by some trigger $t = (R, \pi)$ during $res(\mathcal{D}_n)$. As $(\mathcal{R}_M, D) \not\models \texttt{Fail}$, it must be that $R \neq R_{\texttt{Fail}}$.

If $R = R_{\texttt{Goal}}$, then an atom $\boldsymbol{q_f}(x, y)$ must be in $res(\mathcal{D}_n)$. As $\texttt{Goal} \notin res(\mathcal{D}_0)$, by Proposition 27, x or y must be a variable. Thus, by Lemma 34, x and y are on the same chain. In addition, as M does not halt on the empty word, by Lemma 32, x and y cannot be on the starting chain, so they are both on a non-starting chain. By Lemma 33, $SkD(x)$ and $SkD(y)$ are both less than one. Thus, by the contrapositive of Proposition 27, $res(\mathcal{D}_n)$ only contains terms of Skolem depth less than one.

If $R = R^0_{q,c}$, then atoms $\boldsymbol{q_0}(x, x)$, $\boldsymbol{q}(x, y)$ and $\boldsymbol{c}(x, y)$ must be in $res(\mathcal{D}_n)$ (with $\delta(q, c) = (q', c', \leftarrow)$ for some q', c'). As $\texttt{Goal} \notin res(\mathcal{D}_0)$, by Proposition 27, x or y must be a variable. Thus, by Lemma 34, x and y are on the same chain. In addition, as M does not halt on the empty word, by Lemma 32, x and y cannot be on the starting chain, so they are both on a non-starting chain. By Lemma 33, $SkD(x)$ and $SkD(y)$ are both less than one. Thus, by the contrapositive of Proposition 27, $res(\mathcal{D}_n)$ only contains terms of Skolem depth less than one. □

Theorem 17. *The pair* $(\mathcal{R}_M, \texttt{Goal})$ *is Datalog-expressible.*

Proof. If M halts on the empty word, then by Theorem 13, $(\mathcal{R}_M, \emptyset) \models \texttt{Goal}$. Thus, by monotonicity of first-order entailment, for all databases D, $(\mathcal{R}_M, D) \models \texttt{Goal}$, so $(\{\rightarrow \texttt{Goal}\}, \texttt{Goal})$ is a suitable Datalog rewriting.

If M does not halt on the empty word, then by Theorem 35, it has bounded depth, so by Theorem 20, it is Datalog expressible. □

We can finally prove the main result.

Theorem 3. *Query answering over Datalog-expressible rule sets is undecidable.*

Proof. Assume for a contradiction that query answering over Datalog-expressible rule sets is decidable, and consider a Turing machine M. Then, by Theorem 17, the couple $(\mathcal{R}_M, \texttt{Goal})$ is Datalog-expressible. Thus, we can decide if $(\mathcal{R}_M, \emptyset) \models \texttt{Goal}$. By Theorem 13, $(\mathcal{R}_M, \emptyset) \models \texttt{Goal}$ if and only if M halts on the empty word. Thus, we are able to decide the halting problem, which is absurd. □

Bibliography

[1] C. Beeri and M. Vardi. The implication problem for data dependencies. In *Proc. of Automata, Languages and Programming, Eigth Colloquium (ICALP 1981)*, volume 115 of *Lecture Notes in Computer Science*, pages 73–85, 1981.

[2] Alin Deutsch, Alan Nash, and Jeff Remmel. The chase revisited. In *Proceedings of the Twenty-Seventh ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '08, pages 149–158, New York, NY, USA, 2008. Association for Computing Machinery.

[3] Mélanie König, Michel Leclere, Marie-Laure Mugnier, and Michaël Thomazo. Sound, complete and minimal ucq-rewriting for existential rules. *Semantic Web*, 6, 11 2013.

[4] Bruno Marnette. Resolution and Datalog rewriting under value invention and equality constraints. *CoRR*, abs/1212.0254, 2012.