Restricted Chase Termination: You Want More than Fairness

DAVID CARRAL, LIRMM, Inria, University of Montpellier, CNRS, France
LUKAS GERLACH, Knowledge-Based Systems Group, TU Dresden, Germany
LUCAS LARROQUE, Inria, DI ENS, ENS, CNRS, PSL University, France
MICHAËL THOMAZO, Inria, DI ENS, ENS, CNRS, PSL University, France

The chase is a fundamental algorithm with ubiquitous uses in database theory. Given a database and a set of existential rules (aka tuple-generating dependencies), it iteratively extends the database to ensure that the rules are satisfied in a most general way. This process may not terminate, and a major problem is to decide whether it does. This problem has been studied for a large number of chase variants, which differ by the conditions under which a rule is applied to extend the database. Surprisingly, the complexity of the universal termination of the restricted (aka standard) chase is not fully understood. We close this gap by placing universal restricted chase termination in the analytical hierarchy. This higher hardness is due to the fairness condition, and we propose an alternative condition to reduce the hardness of universal termination.

CCS Concepts: • **Theory of computation → Constraint and logic programming**; **Logic and databases**.

Additional Key Words and Phrases: Existential Rules, Tuple Generating Dependencies, Restricted Chase

ACM Reference Format:
David Carral, Lukas Gerlach, Lucas Larroque, and Michaël Thomazo. 2025. Restricted Chase Termination: You Want More than Fairness. *Proc. ACM Manag. Data* 3, 2 (PODS), Article 109 (May 2025), 17 pages. https://doi.org/10.1145/3725246

1 Introduction

The chase is a fundamental algorithm in database theory that is applied to address a wide range of problems. For instance, it is used to check containment of queries under constraints, in data exchange settings, or to solve ontology-based query answering; see the introductions of [11, 13] for more information. Technically speaking, the chase is a bottom-up materialisation procedure that attempts to compute a universal model (a model that can be embedded into all other models via homomorphism) for a knowledge base (KB), consisting of an (existential) rule set[1] and a database.

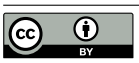
Example 1. *Consider the KB* $\mathcal{K}_1 = \langle \Sigma, D \rangle$ *where* D *is the database* $\{\texttt{Bicycle}(b)\}$ *and* Σ *contains:*

$$\forall x.\texttt{Bicycle}(x) \rightarrow \exists y.\texttt{HasPart}(x,y) \wedge \texttt{Wheel}(y) \qquad \forall x,y.\texttt{HasPart}(x,y) \rightarrow \texttt{IsPartOf}(y,x)$$
$$\forall x.\texttt{Wheel}(x) \rightarrow \exists y.\texttt{IsPartOf}(x,y) \wedge \texttt{Bicycle}(y) \qquad \forall x,y.\texttt{IsPartOf}(x,y) \rightarrow \texttt{HasPart}(y,x)$$

Then, $\{\texttt{Bicycle}(b), \texttt{HasPart}(b,t), \texttt{IsPartOf}(t,b), \texttt{Wheel}(t)\}$ *is a universal model of* $\mathcal{K}$.

[1] Other researchers refer to these first-order formulas as "tuple generating dependencies" or simply as "TGDs".

Authors' Contact Information: David Carral, LIRMM, Inria, University of Montpellier, CNRS, Montpellier, France, david.carral@inria.fr; Lukas Gerlach, Knowledge-Based Systems Group, TU Dresden, Dresden, Germany, lukas.gerlach@tu-dresden.de; Lucas Larroque, Inria, DI ENS, ENS, CNRS, PSL University, Paris, France, lucas.larroque@inria.fr; Michaël Thomazo, Inria, DI ENS, ENS, CNRS, PSL University, Paris, France, michael.thomazo@inria.fr.



ACM 2836-6573/2025/5-ART109
https://doi.org/10.1145/3725246

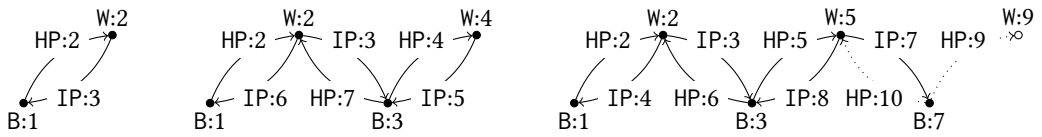


Fig. 1. Three Different Restricted Chase Sequences for the KB $\mathcal{K}_1$ from Example 1

Although there are many variants of the chase, they all implement a similar strategy. Namely, they start with the database and then, in a step-by-step manner, extend this structure with new atoms to satisfy the rules in the input rule set in a most general way. Since none of these variants are guaranteed to terminate (some KBs do not even admit finite universal models), it is only natural to wonder about their respective halting problems [1, 5, 6, 10, 13, 18]. Despite intensive efforts, some results have remained open (until now!). Specifically, prior research has established tight bounds for all classes of chase terminating KBs and rule sets, except for the following:

- The class $\mathsf{CTK}^{rest}_{\forall}$ of all KBs that only admit finite restricted chase sequences.
- The class $\mathsf{CTR}^{rest}_{\forall}$ containing a rule set Σ if $\langle \Sigma, D \rangle \in \mathsf{CTK}^{rest}_{\forall}$ for every database D.

Our main contribution is to show that both classes are Π^1_1-complete, a surprising result given that these are significantly harder than the corresponding classes for other chase variants [13].

The restricted chase differs from other variants in that it introduces new terms to satisfy existential quantifiers in rules only if these are not already satisfied by existing terms. Because of this, the order of rule applications impacts the termination of a chase sequence. For instance, the KB $\mathcal{K}_1$ from Example 1 admits finite and infinite restricted chase sequences; some of these are represented in Fig. 1, where atoms are numbered to denote the sequence step at which they were introduced.

$\mathsf{CTK}^{rest}_{\forall}$ has been claimed to be recursively enumerable (RE) in [13], probably with the following procedure in mind: given an input KB, compute all of its restricted chase sequences in parallel, and halt and accept if all of them are finite. Alas, this strategy does not work as there are terminating input KBs that admit infinitely many finite sequences that are of ever-increasing length.

Example 2. *Consider the KB* $\mathcal{K}_2 = \langle \Sigma, D \rangle$ *where* D *is the database* $\{\mathtt{Real}(a), \mathtt{E}(a,c), \mathtt{E}(c,b), \mathtt{Real}(c), \mathtt{E}(b,b), \mathtt{Brake}(b)\}$ *and* Σ *is the rule set that contains all of the following:*

$$\forall x, y, z.\mathtt{Real}(x) \wedge \mathtt{E}(x,y) \wedge \mathtt{Real}(y) \wedge \mathtt{Brake}(z) \rightarrow \exists v.\mathtt{E}(y,v) \wedge \mathtt{E}(v,z) \wedge \mathtt{Real}(v)$$
$$\forall x.\mathtt{Brake}(x) \rightarrow \mathtt{Real}(x)$$

For any $k \geq 1$*, there is a restricted chase sequence of* $\mathcal{K}_2$ *that yields the (finite) universal model* $D \cup \{\mathtt{E}(c,t_1)\} \cup \{\mathtt{E}(t_i,t_{i+1}) \mid i < k\} \cup \{\mathtt{E}(t_i,b), \mathtt{Real}(t_i) \mid i \leq k\} \cup \{\mathtt{Real}(b)\}$ *of* $\mathcal{K}_2$*. Such a sequence is obtained by applying the first rule* k *consecutive times and then applying the second one once to derive* $\mathtt{Real}(b)$*. After this application, the first rule is satisfied and the restricted chase halts.*

The KB $\mathcal{K}_2$ in the previous example is in $\mathsf{CTK}^{rest}_{\forall}$ because of *fairness*. This is a built-in condition in the definition of all chase variants that guarantees that the chase yields a model of the KB by requiring that, if a rule is applicable at some point during the computation of a sequence, then this rule must be eventually satisfied. Hence, the second rule in $\mathcal{K}_2$ must sooner or later be applied in all restricted chase sequences and thus, all such sequences are finite.

The KB in Example 2 uses a technique called the *emergency brake*, initially proposed by Krötzsch et al. in [16]. The idea is to connect every term in the chase to a special term (the constant b in this example) that is not "Real" and acts as a "Brake". Eventually, this term becomes "Real" because of fairness, all existential restrictions are satisfied, and the restricted chase halts. The emergency brake allows to grow the chase for an arbitrary number of steps whilst guaranteeing its termination.

By activating infinite sequences of emergency brakes, we emulate the eternal recurrence often displayed by Π^1_1-complete problems and thus define the reductions that lead to our main results.

After presenting the necessary preliminaries in Section 2, we discuss related work in Section 3. Then, we show that $\mathsf{CTK}^{rest}_{\forall}$ and $\mathsf{CTR}^{rest}_{\forall}$ are Π^1_1-complete in Sections 4 and 5, respectively. In Section 6, we propose an alternative to fairness for the restricted chase that simplifies its universal termination problem. We conclude with a brief discussion about future work in Section 7.

2 Preliminaries

First-Order Logic. Consider pairwise disjoint, countably infinite sets of *predicates* Preds, *variables* Vars, *constants* Cons, and *nulls* Nulls. Every predicate has an *arity* through $\mathsf{ar} : \mathsf{Preds} \to \mathbb{N} \cup \{0\}$. Elements in $\mathsf{Vars} \cup \mathsf{Cons} \cup \mathsf{Nulls}$ are called *terms*. An *atom* is an expression of the form $\mathsf{P}(\vec{t})$ where $\vec{t}$ a list of terms and P is a $|\vec{t}|$-ary predicate. A *fact* is a variable-free atom. An *(existential) rule* R is a closed first-order formula of the form $\forall \vec{x}, \vec{y}.B[\vec{x}, \vec{y}] \rightarrow \exists \vec{z}.H[\vec{y}, \vec{z}]$ where $\vec{x}$, $\vec{y}$, and $\vec{z}$ are pairwise disjoint lists of variables; B and H are null-free conjunctions of atoms featuring exactly the variables in $\vec{x}, \vec{y}$ and $\vec{y}, \vec{z}$, respectively; and H is non-empty. We write $\mathsf{body}(R)$ and $\mathsf{head}(R)$ to denote B and H, respectively; and refer to the list $\vec{y}$ of variables as the *frontier* of R. We omit universal quantifiers for brevity. A *database* is a finite fact set without nulls. A *knowledge base (KB)* is a pair $\langle \Sigma, D \rangle$ consisting of a finite rule set Σ and a database D.

The Chase. A *substitution* σ is a partial mapping from variables to constants or nulls. For an (arbitrary) expression φ, let $\sigma(\varphi)$ be the expression that results from φ by replacing all occurrences of every variable v in φ by $\sigma(v)$ if the latter is defined. A *trigger* is a pair $\langle R, \sigma \rangle$ consisting of a rule R and a substitution σ that is defined exactly on the universally quantified variables in R. The *support* of a trigger $\langle R, \sigma \rangle$ is $\mathsf{support}(\langle R, \sigma \rangle) = \sigma(\mathsf{body}(R))$. A trigger $\langle R, \sigma \rangle$ is *loaded* for a fact set F if this fact set includes its support; and *obsolete* for F if there exists a substitution σ' that extends σ to the existential variables in R such that $\sigma'(\mathsf{head}(R)) \subseteq F$. The *output* of a trigger $\langle R, \sigma \rangle$ that is not obsolete for F is $\mathsf{output}(\langle R, \sigma \rangle) = \sigma'(\mathsf{head}(R))$, where σ' is some substitution that extends σ by mapping every existential variable in R to a fresh null. A Σ-*trigger* is a trigger with a rule in Σ.

Definition 3. *A* (restricted) chase derivation *for a KB $\langle \Sigma, D \rangle$ is a possibly infinite sequence $F_0, F_1, \ldots$ of fact sets such that (1) $F_0 = D$ and, (2) for each $i \geq 0$, there is some Σ-trigger $\langle R, \sigma \rangle$ that is loaded and not obsolete for F_i such that $F_{i+1} = F_i \cup \mathsf{output}(\langle R, \sigma \rangle)$. Such a chase derivation is a* (restricted) chase sequence *if, (3) for every Σ-trigger λ and every $i \geq 0$ such that λ is loaded for F_i, there is some $j \geq i$ such that λ is obsolete for F_j.*

Condition (3) is known as *fairness*. Note that, if no appropriate trigger according to condition (2) exists for some $i \geq 0$, then the sequence necessarily ends at F_i. The *result* of a chase sequence $\mathcal{F}$ is the union of all fact sets in $\mathcal{F}$. It is well-known that the result F of any chase sequence for a KB $\mathcal{K} = \langle \Sigma, D \rangle$ is a *universal model* for $\mathcal{K}$. That is, every model of $\mathcal{K}$ can be homomorphically embedded into F, which is also a model of this theory. Note that, if we consider infinite sequences, the result of the chase may not be a model of $\mathcal{K}$ if we disregard fairness.

A chase sequence *terminates* if it is finite. A KB *existentially terminates* if it admits a terminating chase sequence; it *universally terminates* if all of its chase sequences terminate. A rule set Σ *existentially terminates* if every KB with Σ existentially terminates; it *universally terminates* if every KB with Σ universally terminates. The classes of knowledge bases that existentially and universally terminate are denoted by $\mathsf{CTK}^{rest}_{\exists}$ and $\mathsf{CTK}^{rest}_{\forall}$, respectively. The classes of rule sets that existentially and universally terminate are denoted by $\mathsf{CTR}^{rest}_{\exists}$ and $\mathsf{CTR}^{rest}_{\forall}$, respectively. We also consider similar classes for the oblivious and core chase variants, which we denoted in the obvious manner. For instance, $\mathsf{CTR}^{obl}_{\exists}$ is the set of all rule sets that existentially terminate for the oblivious chase.

Turing Machines. As per our definition, all machines reuse the same *initial* state. Moreover, machines do not write blanks and cannot access accepting or rejecting states; these are not relevant in our context because we only consider halting problems.

Definition 4. *A* (non-deterministic Turing) machine *is a tuple* $\langle Q, \Gamma, \delta \rangle$ *where* Q *is a set of states that contains the* initial state q_0, Γ *is a tape alphabet with* $\Gamma \supseteq \{0, 1\}$ *and* $\mathsf{B} \notin \Gamma$, *and* δ *is a transition function for* Q. *That is,* δ *is a function that maps from* $Q \times \Gamma \cup \{\mathsf{B}\}$ *to* $\mathcal{P}(Q \times \Gamma \times \{\leftarrow, \rightarrow\})$.

Definition 5. *A* configuration *for a machine* $\langle Q, \Gamma, \delta \rangle$ *is a tuple* $\langle n, t, p, q \rangle$ *where* n *is a natural number;* $t : \{1, \ldots, n\} \to \Gamma \cup \{\mathsf{B}\}$ *is a function such that* $t(n) = \mathsf{B}$, *and* $t(i+1) = \mathsf{B}$ *if* $t(i) = \mathsf{B}$ *for some* $1 \leq i < n$; p *is a number in* $\{1, \ldots, n\}$; *and* q *is a state in* Q. *The* starting configuration *on some* word $w_1, \ldots, w_n \in \{0, 1\}^*$ *is the tuple* $\langle n+1, t, 1, q_0 \rangle$ *where* t *is the function that maps* 1 *to* w_1, 2 *to* w_2, …, n *to* w_n, *and* $n+1$ *to* B.

For a configuration $\langle n, t, p, q \rangle$, we use t to encode the contents of the tape at each position; moreover, we use p and q to encode the position of the head and the state of the machine, respectively. Note that elements of the tape alphabet Γ may not occur after a blank symbol in such a configuration.

Definition 6. *Consider a machine* $M = \langle Q, \Gamma, \delta \rangle$ *and a configuration* $\rho = \langle n, t, p, q \rangle$ *with* $q \in Q$. *Then, let* $Next_M(\rho)$ *be the smallest set that, for every* $\langle r, a, \leftrightarrow \rangle \in \delta(t(p), q)$ *with* $\leftrightarrow = \rightarrow$ *or* $p \geq 2$, *contains the configuration* $\langle n+1, t', p', r \rangle$ *where:*

- *Let* $t'(p) = a$, *let* $t'(n+1) = \mathsf{B}$, *and let* $t'(i) = t(i)$ *for every* $1 \leq i \leq n$ *with* $i \neq p$.
- *If* $\leftrightarrow = \leftarrow$, *then* $p' = p - 1$; *otherwise,* $p' = p + 1$.

As described above, any given machine defines a function that maps configurations to sets of configurations. An exhaustive traversal through a path in this possibly infinite tree of configurations that begins with a starting configuration yields a run:

Definition 7. *A* run *of a machine* M *on a configuration* ρ_1 *is a possibly infinite sequence* $S = \rho_1, \rho_2, \ldots$ *of configurations such that* ρ_{i+1} *is in* $Next_M(\rho_i)$ *for every* $1 \leq i < |S|$, *and* $Next_M(\rho_{|S|}) = \emptyset$ *if* S *is finite. A* partial run *of* M *on* ρ_1 *is a sequence of configurations that can be extended into a run of* M *on* ρ_1. *A* (partial) run *of* M *on a word* $\vec{w}$ *is a (partial) run on the starting configuration of* $\vec{w}$.

Computability Theory. The *arithmetical hierarchy* consists of classes of formal languages Σ^0_i with $i \geq 1$ where Σ^0_1 is the class of all semi-decidable languages and Σ^0_{i+1} is obtained from Σ^0_i with a Turing jump [19]. The co-classes are denoted by Π^0_i. Equivalently, these classes can be viewed as the sets of natural numbers definable by first-order logic formulas with bounded quantifier alternation. That is, Σ^0_i is the class of sets of natural numbers definable with a formula of the form $\exists \vec{x}_1 \forall \vec{x}_2 \ldots Q_i \vec{x}_i . \phi[x, \vec{x}_1, \ldots, \vec{x}_i]$ where ϕ is a quantifier-free formula and Q_i is $\exists$ if i is odd or $\forall$ otherwise. For Π^0_i, the alternation starts with $\forall$. We also the first level of the *analytical hierarchy*; that is, Σ^1_1 and Π^1_1 [19]. The analytical hierarchy can analogously be defined using second-order formulae with bounded second-order quantifier alternation. In the following, we introduce complete problems for these classes that we later use in our reductions. Consider a machine M and a state q_r.

- The machine M is *non-recurring through* q_r on some word $\vec{w}$ if every run of M on $\vec{w}$ features q_r finitely many times.
- It is *universally non-recurring through* q_r if it is non-recurring through q_r on all words.
- It is *robust non-recurring through* q_r if every run of M on any configuration features q_r finitely many times.

We obtain Π^1_1-completeness of the first problem by adjusting a proof from the literature [15] and for the latter two using simple reductions that we define in **??**.

	KB		Rule Set	
	Sometimes	**Always**	**Sometimes**	**Always**
Oblivious	RE-complete [6]		RE-complete [10, 18]	
Restricted	RE-complete [6]	Π^1_1-complete	Π^0_2-complete [13]	Π^1_1-complete
Core	RE-complete [6]		Π^0_2-complete [13]	

Table 1. Undecidability status of the main decision problems related to chase termination; the results presented without citations refer to the main contributions of this article

3 Related Work

Novel Notation. The notation introduced in Section 2 to refer to classes of terminating KBs and rule sets differs from previous literature [13]; for instance, we write $\mathsf{CTR}^{rest}_{\forall}$ instead of $\mathsf{CT}^{rest}_{\forall\forall}$. Moreover, given some database D, we do not consider a class such as $\mathsf{CT}^{rest}_{D\forall}$ [13], which contains a rule set Σ if $\langle \Sigma, D\rangle$ universally terminates for the restricted chase. For our purposes, it is clearer to consider a single class of terminating KBs (such as $\mathsf{CTK}^{rest}_{\forall}$) instead of one class of terminating rule sets for every possible database because of the following result.

Proposition 8. *For a database D', a quantifier $Q \in \{\forall, \exists\}$, and a chase variant $var \in \{obl, rest, core\}$; there is a many-one reduction from CTK^{var}_{Q} to $\mathsf{CT}^{var}_{D'Q}$ and vice-versa.*

Proof. There is a many-one reduction $\mathsf{CT}^{var}_{D'Q}$ to CTK^{var}_{Q} since, for a rule set Σ, we have that $\Sigma \in \mathsf{CT}^{var}_{D'Q}$ if and only if $\langle \Sigma, D'\rangle \in \mathsf{CTK}^{var}_{Q}$. To show that there is a many-one reduction in the other direction we describe a computable function that maps a KB $\mathcal{K} = \langle \Sigma, D\rangle$ into the rule set Σ' such that $\mathcal{K} \in \mathsf{CTK}^{var}_{Q}$ if and only if $\Sigma' \in \mathsf{CT}^{var}_{D'Q}$. Namely, let Σ' be the rule set that results from applying the following modifications to Σ: (i) replace all occurrences of every predicate P with a fresh predicate P', (ii) add the conjunction $\bigwedge_{P(\vec{c})\in D} P'(\vec{c})$ to the body of every rule, and (iii) add the rule $\rightarrow \bigwedge_{P(\vec{c})\in D} P'(\vec{c})$. The reduction is correct because one can easily establish a one-to-one correspondence between the sequences of $\mathcal{K}$ and those of $\langle \Sigma', D'\rangle$ once we ignore the single trigger with $\rightarrow \bigwedge_{P(\vec{c})\in D} P'(\vec{c})$ at the beginning of every sequence of the latter KB. Note that the sets of facts produced at subsequent steps of these corresponding sequences are identical modulo replacement of all occurrences of every predicate P by P'. □

Chase Termination in the General Case. All decision problems related to chase termination are undecidable. However, these are complete for different classes within the arithmetical and analytical hierarchies, as summarised in Table 1. In the following paragraphs, we discuss some simple proofs as well as the relevant references to understand all of the results in this table.

One can readily show via induction that, if a fact occurs in some oblivious chase sequence of some KB, then it also occurs in all oblivious chase sequences of this KB. Hence, all such chase sequences of a KB yield the same result, and thus we conclude that $\mathsf{CTK}^{obl}_{\exists} = \mathsf{CTK}^{obl}_{\forall}$ and $\mathsf{CTR}^{obl}_{\exists} = \mathsf{CTR}^{obl}_{\forall}$.

Deutsch et al. proved that, if a KB admits a finite universal model, then all of its core chase sequences yield precisely this model and thus all of these sequences are finite; see Theorem 7 in [6]. Regardless of the variant, all terminating chase sequences yield a (not necessarily minimal) finite universal model; hence, if a KB does not admit a finite universal model, then it does not admit any finite chase sequence. Therefore, we have that either all core chase sequences of a KB are finite or all of them are infinite. Because of this, we conclude that $\mathsf{CTK}^{core}_{\exists} = \mathsf{CTK}^{core}_{\forall}$ and $\mathsf{CTR}^{core}_{\exists} = \mathsf{CTR}^{core}_{\forall}$.

To understand why $\mathsf{CTK}^{obl}_{\exists}$ (resp. $\mathsf{CTK}^{rest}_{\exists}$ or $\mathsf{CTK}^{core}_{\exists}$) is recursively enumerable (RE), consider the following procedure: given some input KB, compute all of its oblivious (resp. restricted or core)

chase sequences in parallel and accept as soon as you find a finite one. Deutsch et al. proved that $\mathsf{CTK}^{rest}_{\exists}$ is RE-hard. More precisely, they defined a reduction that takes a machine M as input and produces a KB $\mathcal{K}$ as output such that M halts on the empty word if and only $\mathcal{K}$ is in $\mathsf{CTK}^{rest}_{\exists}$; see Theorem 1 in [6]. This reduction works because all restricted chase sequences of $\mathcal{K}$ yield the same result, which encodes the computation of M on the empty word with a grid-like structure (as we ourselves do in later sections). One can use the same reduction to show that $\mathsf{CTK}^{obl}_{\exists}$ is also RE-hard.

Deutsch et al. also proved that $\mathsf{CTK}^{core}_{\exists}$ is RE-hard. More precisely, they showed that checking if a KB admits a universal model is undecidable; see Theorem 6 in [6]. Moreover, they proved that the core chase is a procedure that halts and yields a finite universal model for an input KB if this theory admits one; see Theorem 7 of the same paper. Therefore, the core chase can be applied as a semi-decision procedure for checking if a KB admits a finite universal model.

In Section 4, we argue that $\mathsf{CTK}^{rest}_{\forall}$ is Π^1_1-complete. This contradicts Theorem 5.1 in [13], which states that $\mathsf{CTK}^{rest}_{\forall}$ is RE-complete. Specifically, it is claimed that this theorem follows from results in [6], but the authors of that paper only demonstrate that $\mathsf{CTK}^{rest}_{\forall}$ is undecidable without proving that it is in RE. Before our completeness result, the tightest lower bound was proven by Carral et al., who proved that this class is Π^0_2-hard; see Proposition 42 in [5].

Marnette proved that $\mathsf{CTR}^{obl}_{\exists}$ is in RE. More precisely, he showed that a rule set Σ is in $\mathsf{CTR}^{obl}_{\exists}$ if and only if the KB $\langle \Sigma, D^{\star}_{\Sigma} \rangle$ is in $\mathsf{CTK}^{obl}_{\exists}$ where $D^{\star}_{\Sigma} = \{\mathsf{P}(\star, \ldots, \star) \mid \mathsf{P} \in \mathsf{Preds}(\Sigma)\}$ is the *critical instance* and $\star$ is a special fresh constant; see Theorem 2 in [18]. This result follows because one can show that, for any database D, the (only) result of the oblivious chase of $\langle \Sigma, D^{\star}_{\Sigma} \rangle$ includes the (only) result of the oblivious chase of $\langle \Sigma, D \rangle$ if we replace all syntactic occurrences of constants in the latter with $\star$. Since $\mathsf{CTK}^{obl}_{\exists}$ is in RE, we conclude that $\mathsf{CTR}^{obl}_{\exists}$ is also in this class.

Gogacz and Marcinkowski proved that $\mathsf{CTR}^{obl}_{\exists}$ is RE-hard. More precisely, they presented a reduction that takes a 3-counter machine M as input and produces a rule set Σ such that M halts on ε if and only if $\langle \Sigma, D^{\star}_{\Sigma} \rangle$ is in $\mathsf{CTK}^{obl}_{\exists}$; see Lemma 6 in [10].[2] Hence, M halts on the ε and only if Σ is in $\mathsf{CTR}^{obl}_{\exists}$ by Theorem 2 in [18]. Furthermore, Bednarczyk et al. showed that this hardness result holds even when we consider single-head binary rule sets; see Theorem 1.1 in [1].

To understand why $\mathsf{CTR}^{rest}_{\exists}$ is in Π^0_2, consider the following semi-decision procedure that can access an oracle that decides the RE-complete class $\mathsf{CTK}^{rest}_{\exists}$: given some input rule set Σ; iterate through every database D, use the oracle to decide if $\langle \Sigma, D \rangle$ is in $\mathsf{CTK}^{rest}_{\exists}$, and accept if this is not the case. Consider an analogous procedure to understand why $\mathsf{CTR}^{core}_{\exists}$ is in Π^0_2.

Grahne and Onet proved that $\mathsf{CTR}^{rest}_{\exists}$ is Π^0_2-hard. To show this, they defined two reductions that take a word rewriting system R and a word $\vec{w}$ as input, and produce a rule set Σ_R and a database $D_{\vec{w}}$, respectively. Then, they proved that R terminates on $\vec{w}$ if and only if the KB $\langle \Sigma_R, D_{\vec{w}} \rangle$ is in $\mathsf{CTK}^{rest}_{\exists}$; this claim holds because $\langle \Sigma_R, D_{\vec{w}} \rangle$ only admits a single restricted chase result, which encodes all branches of computation of R on $\vec{w}$ in an implicit tree-like structure. Therefore, R is uniformly terminating if Σ_R is in $\mathsf{CTR}^{rest}_{\exists}$. To ensure that Σ_R is in $\mathsf{CTR}^{rest}_{\exists}$ if R is uniformly terminating, Grahne and Onet make use of "flooding", a technique used in earlier work dealing with datalog boundedness [7]. For a comprehensive presentation of this technique and its applications, see Section 2 of [11]. Using the very same reduction, Grahne and Onet also proved that $\mathsf{CTR}^{core}_{\exists}$ is Π^0_2-hard.

In Section 5, we show that $\mathsf{CTR}^{rest}_{\forall}$ is Π^1_1-complete. This contradicts Theorem 5.16 in [13], where it is stated that this class is Π^0_2-complete. The error in the upper-bound of this theorem arose from the assumption that $\mathsf{CTK}^{rest}_{\forall}$ is in RE, which, as previously discussed, is not the case. Regarding the lower bound, they consider an extended version of this class of rule sets where they allow the inclusion of a single "denial constraint"; that is, an implication with an empty head that halts the

[2] We do not think that it is possible to intuitively explain this reduction in a couple of lines. Go read this paper, it's worth it!

chase if the body is satisfied during the computation of a chase sequence. They prove that the always restricted halting problem for rule sets is Π^0_2-hard if one such constraint is allowed. Our results imply that we do not need to consider such an extension to obtain a higher lower bound.

Chase Termination of Syntactic Fragments. Undeterred by the undecidability results discussed above, researchers have proven we can decide chase termination if we consider syntactic fragments of existential rules for which query entailment is decidable [2, 3, 12, 17]. Another way of checking termination in practice is to develop acyclicity and cyclicity notions; that is, sufficient conditions for termination and non-termination of the chase. Indeed, experiments show that we can determine chase termination for a large proportion of real-world rule sets with these checks [4, 8, 9, 14].

4 Knowledge base termination

Theorem 9. *The class* $\mathsf{CTK}^{rest}_{\forall}$ *is* Π^1_1*-complete.*

The theorem immediately follows from the upcoming Lemma 12 and Lemma 13.

For the membership part, we define a non-deterministic Turing machine that loops on q_r if and only if there is a non-terminating chase sequence for a given rule set.

Definition 10. *Consider a rule set* Σ*. For a fact set* F*, let* $active(F)$ *be the set of all triggers with a rule in* Σ *that are loaded and not obsolete for* F*. Let* $\mathcal{M}_\Sigma$ *be a non-deterministic Turing machine with start state* q_0 *and a designated state* q_r *that executes the following procedure.*

1. *Check if the input tape contains a valid encoding of a database. If not, halt.*
2. *Initialize two counters* $i = j = 0$ *and a set of facts* F_0 *containing exactly the encoded database.*
3. *If* $active(F_i)$ *is empty, halt.*
4. *Non-deterministically pick a trigger* $\langle R, \sigma\rangle$ *from* $active(F_i)$ *and let* $F_{i+1} = F_i \cup \sigma'(\mathsf{head}(R))$ *where* σ' *extends* σ *by mapping existential variables in* R *to fresh nulls (not occurring in* F_i*).*
5. *If all triggers in* $active(F_j)$ *are obsolete for* F_i*, then increment* j *and visit* q_r *once.*
6. *Increment* i *and go to 3.*

Lemma 11. *For every database* D *and rule set* Σ*, there is a run of* $\mathcal{M}_\Sigma$ *on the encoding of* D *that visits* q_r *infinitely often if and only if there is a non-terminating chase sequence for* $\langle \Sigma, D\rangle$*.*

Proof. Assume that there is a run of $\mathcal{M}_\Sigma$ on the encoding of D that visits q_r infinitely many times. Then, the sequence $F_0, F_1, \ldots$ constructed by $\mathcal{M}_\Sigma$ is an infinite restricted chase derivation for $\langle \Sigma, D\rangle$ by construction. Since q_r is visited infinitely many times, j grows towards infinity. Therefore, every trigger that is loaded for some F_j with $j \geq 0$ is obsolete for some $i \geq j$; which is exactly *fairness*. Hence, the infinite derivation is a proper chase sequence.

Assume that there is an infinite chase sequence $F_0, F_1, \ldots$ for $\langle \Sigma, D\rangle$. By definition, for each $i \geq 0$, there is a trigger $\lambda \in active(F_i)$ that yields F_{i+1}. Hence, there is a run of $\mathcal{M}_\Sigma$ that non-deterministically picks these triggers. Because of fairness, for every trigger λ in $active(F_j)$ with $j \geq 0$, there is $i \geq j$ such that λ is obsolete for F_i. Hence, the run of $\mathcal{M}_\Sigma$ visits q_r infinitely often. □

Lemma 12. *Deciding membership in* $\mathsf{CTK}^{rest}_{\forall}$ *is in* Π^1_1*.*

Proof. We show a reduction to non-recurrence through q_r on the empty word. For a given rule set Σ, let $\mathcal{M}^D_\Sigma$ be a non-deterministic Turing machine that results from $\mathcal{M}_\Sigma$ by adding an initial step that replaces the initial tape content by an encoding of D. Then, by Lemma 11, Σ is in $\mathsf{CTK}^{rest}_{\forall}$ if and only if no run of $\mathcal{M}^D_\Sigma$ on the empty input visits q_r infinitely many times. □

Lemma 13. *The class* $\mathsf{CTK}^{rest}_{\forall}$ *is* Π^1_1*-hard.*

To prove hardness, we reduce non-recurrence through q_r on the empty word to knowledge base termination. In other words, to a Turing machine M, we will associate a database D_ε and a rule set Σ_M such that there exists a run of M on the empty word reaching q_r infinitely often if and only if the restricted chase of Σ_M on D_ε does not halt.

A perhaps surprising feature of this reduction is that the restricted chase must halt for rule sets generated from Turing machines that do not halt on the empty word, as long as they reach q_r only finitely often. As we cannot get any computable bound on the number of steps required to reach q_r, we must simulate any finite run of the Turing machine in a terminating way. This calls for the use of *emergency brakes* as presented in the introduction. We "stack" such brakes, each one being responsible to prevent the non-termination for runs that do not go through q_r.

Schema. We will make use of the following predicates. Note that the last position usually holds an emergency brake. We introduce: For each letter a in the Turing machine alphabet or equal to the blank B, a binary predicate a. For each state q of the Turing machine, a binary predicate q. Two ternary predicates F and R, that encode the successor relation for time and for cells. Two binary predicates $\mathsf{C_L}$ and $\mathsf{C_R}$, used to copy tapes content. A unary predicate $\mathtt{Real}$ and a binary predicate $\mathtt{NextBr}$, used for the machinery of emergency brakes. Two unary predicates $\mathtt{Brake}$ and $\mathtt{End}$ to identify terms used as emergency brakes and the last element of a configuration, respectively.

Each time a new term is created during the chase, we link it in a specific way to the relevant brake. To simplify the subsequent presentation, we denote by $\mathtt{brSet}(x, w)$ the set of atoms $\{\mathsf{F}(x, w, w), \mathsf{R}(x, w, w), \mathtt{Real}(x), \mathtt{Brake}(w)\}$. The remainder of this section is devoted to the reduction from the "non-recurrence through q_r" problem to knowledge base restricted chase termination. We first present the reduction, and then focus on the main ideas required to show correctness.

The Reduction. Each configuration ρ of a Turing machine is encoded by a database as follows.

Definition 14. *The database D_ρ encoding a configuration $\rho = \langle n, t, p, q\rangle$ is*

$$D_\rho = \{\mathsf{R}(c_i, c_{i+1}, w_1), \mathsf{a_i}(c_i, w_1) \mid 1 \leq i \leq n; \mathsf{a_i} = t(i)\} \cup \left\{\mathsf{q}(c_p, w_1), \mathsf{B}(c_{n+1}, w_1), \mathtt{End}(c_{n+1}, w_1)\right\} \cup \bigcup_{1 \leq i \leq n+1} \mathtt{brSet}(c_i, w_1)$$

For a word w, we denote by D_w the database D_{ρ_w}, where ρ_w is the initial configuration of M on w.

Given a Turing machine M with states Q and tape alphabet Γ, we build Σ_M composed of the following rules. We first have a set of rules required for setting up emergency brakes.

$$\mathtt{Brake}(w) \to \bigwedge_{\mathsf{a} \in \Gamma \cup \{\mathsf{B}\}} \mathsf{a}(w, w), \bigwedge_{\mathsf{q} \in Q} \mathsf{q}(w, w), \mathsf{F}(w, w, w), \mathsf{R}(w, w, w), \mathsf{C_L}(w, w), \mathsf{C_R}(w, w), \mathtt{Real}(w), \mathtt{nextBr}(w, w) \quad (R_{\mathtt{Brake}})$$

$$\mathtt{brSet}(x, w), \mathtt{nextBr}(w, w') \to \mathtt{brSet}(x, w') \quad (R_{\mathtt{nextBr}})$$

The next four rules are responsible of simulating the moves of the head of the Turing machine. The first two rules deal with the case where the machine is not in q_r, and the head moves to the right (resp. to the left). The important feature of these rules is the presence in both the body and the head of the same brake w.

For all $q \neq q_r, q' \in Q$ and $a, b, c \in \Gamma \cup \{\mathtt{B}\}$ such that $(q', b, \rightarrow) \in \delta(q, a)$:

$$\begin{aligned} &\mathsf{q}(x,w), \mathsf{a}(x,w), \mathsf{R}(x,y,w), \mathsf{c}(y,w), \mathtt{brSet}(x,w), \mathtt{brSet}(y,w) \\ &\rightarrow \exists x', y'\ \mathsf{q}'(y',w), \mathsf{c}(y',w), \mathsf{b}(x',w), \mathsf{C_L}(x',w), \mathsf{C_R}(y',w), \\ &\quad \mathsf{R}(x',y',w), \mathsf{F}(x,x',w), \mathsf{F}(y,y',w), \mathtt{brSet}(x',w), \mathtt{brSet}(y',w) \end{aligned} \tag{$R^{\rightarrow}_{\neg q_r}$}$$

For all $q \neq q_r, q' \in Q$ and $a, b, c \in \Gamma \cup \{\mathtt{B}\}$ such that $(q', b, \leftarrow) \in \delta(q, a)$:

$$\begin{aligned} &\mathsf{q}(x,w), \mathsf{a}(x,w), \mathsf{R}(y,x,w), \mathsf{c}(y,w), \mathtt{brSet}(x,w), \mathtt{brSet}(y,w) \\ &\rightarrow \exists x', y'\ \mathsf{q}'(y',w), \mathsf{c}(y',w), \mathsf{b}(x',w), \mathsf{C_L}(y',w), \mathsf{C_R}(x',w), \\ &\quad \mathsf{R}(y',x',w), \mathsf{F}(x,x',w), \mathsf{F}(y,y',w), \mathtt{brSet}(x',w), \mathtt{brSet}(y',w) \end{aligned} \tag{$R^{\leftarrow}_{\neg q_r}$}$$

The following two rules treat the case where the transition is from q_r. The only difference with the two above rules is the introduction of a new brake w' in the head of the rules. This permits non-terminating restricted chase sequences in the presence of specific runs.

For all $q' \in Q$ and $a, b, c \in \Gamma \cup \{\mathtt{B}\}$ such that $(q', b, \rightarrow) \in \delta(q_r, a)$:

$$\begin{aligned} &\mathsf{q}_r(x,w), \mathsf{R}(x,y,w), \mathsf{a}(x,w), \mathsf{c}(y,w), \mathtt{brSet}(x,w), \mathtt{brSet}(y,w) \\ &\rightarrow \exists x', y', w',\ \mathsf{q}'(y',w'), \mathsf{c}(y',w'), \mathsf{b}(x',w'), \mathsf{R}(x',y',w'), \\ &\quad \mathsf{F}(x,x',w'), \mathsf{F}(y,y',w'), \mathsf{C_L}(x',w'), \mathsf{C_R}(y',w'), \\ &\quad \mathtt{brSet}(x',w'), \mathtt{brSet}(y',w'), \mathtt{nextBr}(w,w') \end{aligned} \tag{$R^{\rightarrow}_{q_r}$}$$

For all $q' \in Q$ and $a, b, c \in \Gamma \cup \{\mathtt{B}\}$ such that $(q', b, \leftarrow) \in \delta(q_r, a)$:

$$\begin{aligned} &\mathsf{q}_r(x,w), \mathsf{R}(y,x,w), \mathsf{a}(x,w), \mathsf{c}(y,w), \mathtt{brSet}(x,w), \mathtt{brSet}(y,w) \\ &\rightarrow \exists x', y', w',\ \mathsf{q}'(y',w'), \mathsf{c}(y',w'), \mathsf{b}(x',w'), \mathsf{R}(y',x',w'), \\ &\quad \mathsf{F}(x,x',w'), \mathsf{F}(y,y',w'), \mathsf{C_L}(y',w'), \mathsf{C_R}(x',w'), \\ &\quad \mathtt{brSet}(x',w'), \mathtt{brSet}(y',w'), \mathtt{nextBr}(w,w') \end{aligned} \tag{$R^{\leftarrow}_{q_r}$}$$

The following rules copy the content of unchanged cells to the right and the left of the head from one configuration to the next. We instantiate one of each rule for each $a \in \Gamma \cup \{\mathtt{B}\}$.

$$\begin{aligned} &\mathsf{C_R}(x',w'), \mathsf{F}(x,x',w'), \mathsf{R}(x,y,w), \mathsf{a}(y,w), \mathtt{brSet}(x,w), \mathtt{brSet}(x',w'), \mathtt{brSet}(y,w) \\ &\quad \rightarrow \exists y'\ \ \mathsf{F}(y,y',w'), \mathsf{R}(x',y',w'), \mathsf{a}(y',w'), \mathsf{C_R}(y',w'), \mathtt{brSet}(y',w') \end{aligned} \tag{$R_{\mathsf{C_R}}$}$$

$$\begin{aligned} &\mathsf{C_L}(x',w'), \mathsf{F}(x,x',w'), \mathsf{R}(y,x,w), \mathsf{a}(y,w), \mathtt{brSet}(x,w), \mathtt{brSet}(x',w'), \mathtt{brSet}(y,w) \\ &\quad \rightarrow \exists y'\ \ \mathsf{F}(y,y',w'), \mathsf{R}(y',x',w'), \mathsf{a}(y',w'), \mathsf{C_L}(y',w'), \mathtt{brSet}(y',w') \end{aligned} \tag{$R_{\mathsf{C_L}}$}$$

Finally, we extend the represented part of the configuration by one cell at each step, as coherent with our definition of Turing machine runs:

$$\begin{aligned} &\mathsf{C_R}(x',w'), \mathsf{F}(x,x',w'), \mathtt{End}(x,w), \mathtt{brSet}(x,w), \mathtt{brSet}(x',w') \\ &\rightarrow \exists y',\ \mathsf{R}(x',y',w'), \mathsf{B}(y',w'), \mathtt{End}(y',w'), \mathtt{brSet}(y',w') \end{aligned} \tag{R_{End}}$$

Example 15. *Consider a machine $M = \langle \{q_0, q_r\}, \{0,1\}, \delta \rangle$ where δ is a transition function that maps $\langle q_0, 0\rangle$ to $\{\langle q_r, 1, \rightarrow\rangle\}$, $\langle q_r, \mathsf{B}\rangle$ to $\{\langle q_0, 1, \leftarrow\rangle\}$, $\langle q_0, 1\rangle$ to $\{\langle q_r, 1, \rightarrow\rangle\}$, and $\langle q_r, 1\rangle$ to $\{\langle q_0, 1, \leftarrow\rangle\}$; note how the (only) run of M on the word 0 contains infinitely many configurations with the state q_r. In this representation, every label on an edge or a term represents several facts in the chase. For the sake*

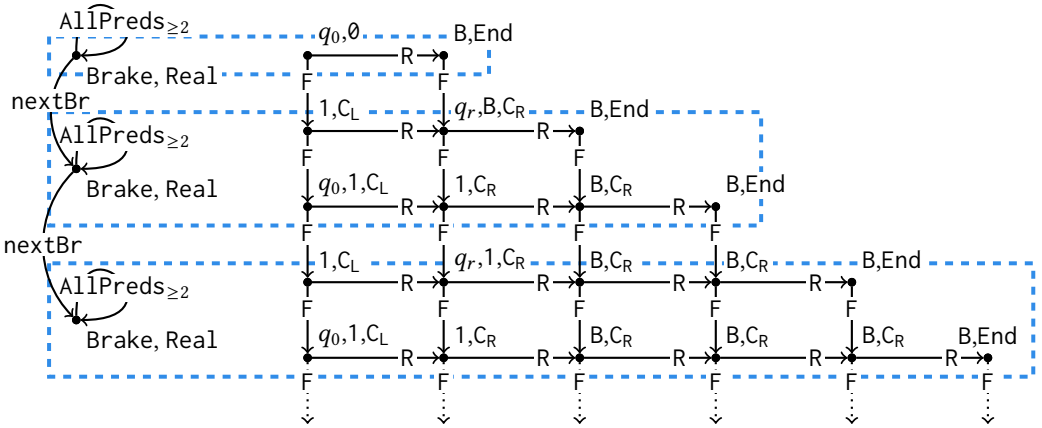


Fig. 2. An Infinite Restricted Chase Sequence of $\langle \Sigma_M, D_0 \rangle$ where M is the machine from Example 15, and $\mathtt{AllPreds}_{\geq 2}$ above is a shortcuts for "$\mathtt{F}$, $\mathtt{R}$, q_0, q_r, $\mathtt{0}$, $\mathtt{1}$, $\mathtt{B}$, $\mathtt{C_R}$, $\mathtt{C_L}$, $\mathtt{nextBr}$".

of clarity, these labels can be extended with another argument, which should be some "$\mathtt{Brake}$*" term in the same dashed or later dashed box.*

Correctness proof of the reduction. The reduction is now fully described, and we claim that:

Proposition 16. *Σ_M universally halts for the restricted chase on D_ρ if and only if there exists no run of M on ρ that goes infinitely often through q_r.*

We first prove that if there exists a run of M going through q_r infinitely often, then there exists a non-terminating chase sequence. To that purpose, we identify interesting subsets of databases.

Definition 17 (Wild Frontier of Configuration ρ)**.** *A set of atoms F has a* wild frontier *of configuration $\rho = \langle n, t, p, q \rangle$ overseen by $w \in \mathsf{terms}(F)$ if there exists $x_1, \ldots, x_{n+1} \in \mathsf{terms}(F)$ such that:*

- $\mathtt{Real}(w) \notin F$;
- $\{\mathsf{R}(x_i, x_{i+1}, w), \mathsf{a_i}(x_i, w)\} \subseteq F$ *for all* $i \in \{1, \ldots, n\}$, $\mathsf{a_i} = t(i)$;
- $\mathsf{q}(x_p, w), \mathsf{End}(x_{n+1}, w), \mathsf{B}(x_{n+1}, w) \in F$;
- $\mathtt{brSet}(x_i, w) \in F$ *for all* $i \in \{1, \ldots, n+1\}$;
- *any other atom of F having x_i as first argument has w as second.*

A wild frontier has three important features *(i)* it contains the necessary atoms to simulate the run of a Turing machine on that configuration; *(ii)* it is correctly connected to a (not yet real) brake w; *(iii)* it does not contain atoms preventing the above run to be simulated through a restricted derivation. By comparing Definition 14 and Definition 17, it is clear that D_ε has a wild frontier of the configuration of M on the empty word, overseen by w_1. The construction of an infinite restricted derivation is made by inductively using the following key proposition.

Proposition 18. *If F has a wild frontier of ρ overseen by w, and ρ' is reachable in one step by a transition of M, then there exists a restricted derivation $\mathcal{D}_{\rho \to \rho'} = F, \ldots, F'$ such that F' has a wild frontier of ρ' overseen by w', where $w' \neq w$ is a fresh existential if ρ is in q_r, and $w' = w$ otherwise.*

Concatenating the infinite sequence of derivations built in Proposition 18 does not however provide a fair sequence of derivations, because of Rules $R_{\mathtt{Brake}}$, $R_{\mathtt{nextBr}}$ and of the non-determinism

of M. Fairness is enforced by applying $R_{\texttt{Brake}}$ and $R_{\texttt{nextBr}}$ "late enough" to ensure that none of the triggers involved in the proof of Proposition 18 are made obsolete. This is possible because the run of M going infinitely many often through q_r, infinitely many brakes are created. Details are provided in the appendix.

Lemma 19. *Let $(\rho_i)_{i\in\mathbb{N}}$ be a run of M on the empty word that visits q_r infinitely often. There exists an infinite restricted chase sequence for $\langle \Sigma_M, D_\varepsilon\rangle$.*

To show the converse, we fix an infinite restricted chase sequence $\mathcal{D}$ as $(F_i)_{i\in\mathbb{N}}$, where $F_0 = D_\varepsilon$. We build from $\mathcal{D}$ an infinite run that visits q_r infinitely often by identifying a substructure of the chase, consisting of *state atoms*. We then prove that a run can be built from these states atoms (and other elements of the chase), which fulfills the required conditions.

Definition 20. *A* state atom of F *is an atom of F of the form $\mathsf{q}(x, w)$ where $q \in Q$ and x is not a brake. A state atom A* precedes A' *if there is a trigger t such that $A \in \mathsf{support}(t)$ and $A' \in \mathsf{output}(t)$. In this case, we write $A \prec A'$.*

It is worth noticing that in the chase of $\langle \Sigma_M, D_\varepsilon\rangle$, state atoms are organised as a tree structure rooted in the unique state atom belonging to D_ε, and such that A is a parent of A' if and only if $A \prec A'$. Intuitively, we can assign with each of these state atoms a configuration such that the configuration associated with A' is reachable in one transition of M from the configuration associated with its parent A. The key property is that in an infinite restricted chase, there exists an infinite sequence $(A_n)_{n\in\mathbb{N}}$ with good properties.

Lemma 21. *For all databases D, and all infinite chase sequences from $\langle \Sigma_M, D\rangle$ with result F, there is an infinite sequence $(A_n)_{n\in\mathbb{N}}$ of state atoms of F such that:*

- $A_0 \in D$;
- $A_n \prec A_{n+1}$ *for all* $n \in \mathbb{N}$;
- *for infinitely many* $i \in \mathbb{N}$, A_i *is of the shape* $\mathsf{q}_r(t_i, w_i)$.

Proof sketch. Since the rules that introduce state atoms (Rules $R^{\leftarrow}_{q_r}$, $R^{\rightarrow}_{q_r}$, $R^{\leftarrow}_{\neg q_r}$ and $R^{\rightarrow}_{\neg q_r}$) feature a state atom in their body, $\prec$ defines a forest structure over state atoms, where the root of each tree is an atom of the database. There is thus a finite amount of trees. We can prove by induction that there is a finite amount of atoms that feature a given brake. Thus, there is an infinite amount of brakes in F. Then, since the rules that introduce new brakes (Rules $R^{\rightarrow}_{q_r}$ and $R^{\leftarrow}_{q_r}$) introduce a state atom too, there is an infinite number of state atoms. Thus, one of the trees must be infinite, and since branching can be proven to be finite, there must be an infinite branch by König's lemma. □

Lemma 22. *For every configuration ρ, if the restricted chase does not terminate on $\langle \Sigma_M, D_\rho\rangle$ then there exists a run of M on ρ which visits q_r infinitely many times.*

Lemmas 19 and 22 directly imply Proposition 16, and hence the correctness of the reduction.

5 Rule set termination

Theorem 23. $\mathsf{CTR}^{rest}_{\forall}$ *is Π^1_1-complete.*

The theorem immediately follows from the upcoming Lemma 24 and Lemma 25.

Lemma 24. *Deciding membership in $\mathsf{CTR}^{rest}_{\forall}$ is in Π^1_1.*

Proof. We reduce to universal non-recurrence through q_r. More precisely, we show that a rule set Σ is in $\mathsf{CTR}^{rest}_{\forall}$ if and only if $\mathcal{M}_\Sigma$ from Definition 10 is universally non-recurring through q_r.

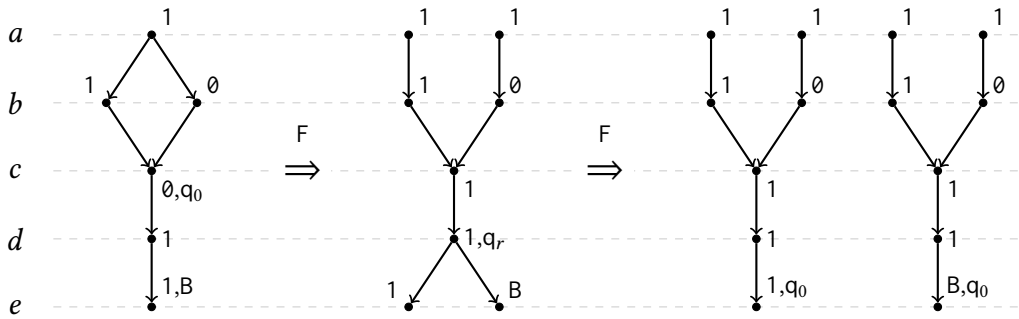


Fig. 3. First three steps of the restricted chase from $\langle \mathcal{R}_M, D \rangle$ as defined in Example 26. The predicate F and the brakes are not represented for the sake of readability, but terms are connected through the future predicate to an element on the same line at the previous step. Unlabeled arrows represent R-atoms.

If Σ is in $\mathsf{CTR}^{rest}_{\forall}$, then $\langle \Sigma, D \rangle$ is in $\mathsf{CTK}^{rest}_{\forall}$ for each D. Hence, by Lemma 11, $\mathcal{M}_\Sigma$ is non-recurring on every input that is the encoding of some database D. On inputs that are not encodings of databases, $\mathcal{M}_\Sigma$ halts immediately by Definition 10. Therefore, $\mathcal{M}_\Sigma$ is universally non-recurring.

If $\mathcal{M}_\Sigma$ is universally non-recurring through q_r, then, in particular, $\mathcal{M}_\Sigma$ is non-recurring through q_r on every input that is the encoding of a database. Hence, by Lemma 11, each restricted chase sequence for each knowledge base with Σ is finite. Therefore, Σ is in $\mathsf{CTR}^{rest}_{\forall}$. □

Lemma 25. $\mathsf{CTR}^{rest}_{\forall}$ *membership is* Π^1_1*-hard.*

The rest of the section is dedicated to proving this lemma, by reducing robust non-recurrence through q_r to rule set termination. In fact, the reduction is very similar to the one we use for knowledge base termination: to a machine M, we associate the rule set Σ_M, which will belong to $\mathsf{CTR}^{rest}_{\forall}$ if and only if M is robust non-recurring through q_r. The direct implication follows from Lemma 22 by contrapositive: if a Turing machine M is *not* robust non-recurring through q_r, then there is a configuration ρ such that M visits q_r infinitely many times from ρ. Then, by Lemma 22, the restricted chase does not terminate on $\langle \Sigma_M, D_\rho \rangle$, and thus $\Sigma_M \notin \mathsf{CTR}^{rest}_{\forall}$. The other direction requires more work. Consider a Turing machine M, and assume that there is some database D such that the restricted chase does not terminate on $\langle \Sigma_M, D \rangle$. We then show that M is not robust non-recurring through q_r.

Since the restricted chase does not terminate on $\langle \Sigma_M, D \rangle$, there is an infinite chase sequence from this knowledge base. We use F to denote its result. As in Section 4, by Lemma 21, F contains an infinite sequence of state atoms $\mathcal{A} = (A_n)_{n \in \mathbb{N}}$ such that $A_0 \in D$, $A_n \prec A_{n+1}$ for all $n \in \mathbb{N}$, and there are infinitely many integers i such that A_i is a q_r-atom.

In the knowledge base case, we had control over the database as part of the knowledge base, which meant that we could start from a "well-formed" database (in the sense that it encodes a single start configuration). This allowed us to extract the *unique* configuration associated with a state atom. However, in the rule set case, the database D leading to non-termination is arbitrary and can contain any kind of structure, as highlighted by the following example.

Example 26. *Consider a Turing machine M that moves to the right in every step, writing* 1 *regardless of the symbol it reads. It alternates between its start state* q_0 *and the designated state* q_r*. Now, consider the database depicted on the left side of Fig. 3, which contains the atoms* $\mathsf{R}(a, b_1, w)$, $\mathsf{R}(a, b_2, w)$, $\mathsf{R}(b_1, c, w)$, $\mathsf{R}(b_2, c, w)$, $\mathsf{R}(c, d, w)$, $\mathsf{R}(d, e, w)$, $\mathsf{q}_0(c, w)$, $\mathsf{1}(a, w)$, $\mathsf{1}(b_1, w)$, $\mathsf{0}(b_2, w)$, $\mathsf{0}(c, w)$, $\mathsf{1}(d, w)$, $\mathsf{1}(e, w)$, $\mathsf{B}(e, w)$, *and* $\mathtt{brSet}(x, w)$ *for all* $x \in \{a, b_1, b_2, c, d, e\}$*. This database represents four different configurations,*

each with a tape of size 5, the start state q_0*, and the head positioned at the third cell. These configurations correspond to tapes with contents* 11011, 10011, 1101B, *and* 1001B.

As the simulation progresses, these configurations evolve simultaneously, creating new structures shown in the middle and right of Fig. 3. Notice how term e has two successors through the F *predicate, one for each symbol atom it belongs to. Furthermore, when the head encounters a branching structure, it splits into two, as observed in the third step of the simulation. In a sense, if the machine simulation is able to perform steps on the database at all, then it will gradually "heal" the structure step by step towards proper encodings of machine configurations.*

As highlighted by this example, the structure of the set of atoms connected to a state atom not present in the database is specific: it is the union of two trees rooted in the state atom. The first has arrows going towards the state atom, and the second one has arrows going away from the state atom. In fact, this structure represent the set of paths in the initial database (after the appropriate number of steps of simulation), which we coin a bow tie, due to its shape.

Definition 27. *The inverse* E^- *of a binary relation* E *is the relation defined by* $(x, y) \in E^-$ *if and only if* $(y, x) \in E$*. In a directed graph* $G = (V, E)$ *we denote with* V_x^{-y} *the connected component*[3] *of* x *in the subgraph induced by* $V \setminus \{y\}$ *on* G*, for any two vertices* x *and* y*. A* bow tie *is a graph* (V, E) *with two distinguished vertices* x *and* y *that has the following properties:*

(1) $(x, y) \in E$*;*
(2) *The subgraph induced by* V_x^{-y} *on* (V, E^-) *is a directed tree rooted in* x*;*
(3) *The subgraph induced by* V_y^{-x} *on* (V, E) *is a directed tree rooted in* y*;*
(4) *The sets* V_x^{-y} *and* V_y^{-x} *form a partition of* V*; that is, they are disjoint and* $V = V_x^{-y} \cup V_y^{-x}$*.*

The edge (x, y) *is called the* center *of the bow tie, and the sets* V_x^{-y} *and* V_y^{-x} *are called the* left *and* right *parts of the bow tie, respectively.*

In the following, we denote with $\mathsf{semterms}(F)$ (for semantically meaningful terms) the set of all the terms in F, except the brakes (which appear in the last position of atoms). We also define E_R as the relation over $\mathsf{semterms}(F)$ such that $(x, y) \in E_R$ if and only if there is w such that $R(x, y, w) \in F$. For all state atoms $A = \mathsf{q}(x, w)$ generated during the chase, we denote the connected component of x in the graph $(\mathsf{semterms}(F), E_R)$ with $\mathsf{bowtie}(A)$. The following lemma explains how this bow tie structure is generated at each step.

Lemma 28. *For all database* D*, and every* F *result of a chase sequence for* $\langle \Sigma_M, D \rangle$*, the graph* $\mathsf{bowtie}(A)$ *is a finite bow tie for all state atoms* $A \in F \setminus D$*. In addition:*

- *The center of the bow tie is the atom generated along with* A*, by rule* $R^{\leftarrow}_{\neg q_r}$*,* $R^{\rightarrow}_{\neg q_r}$*,* $R^{\leftarrow}_{q_r}$ *or* $R^{\rightarrow}_{q_r}$*;*
- *all the atoms in the left part of the bow tie are generated by rule* $R_{\mathsf{C_L}}$*;*
- *all the atoms in the right part of the bow tie are generated by rule* $R_{\mathsf{C_R}}$*, except possibly the end of a maximal path, which may have been generated by rule* R_{End}*.*

Proof sketch. This proof relies on an analysis of how R-atoms are generated during the chase. All the rules that generate R-atoms (over non-brake terms) generate R-atoms containing at least one existentially quantified variable. Three cases occur:

- Rules $R^{\leftarrow}_{\neg q_r}$, $R^{\rightarrow}_{\neg q_r}$, $R^{\leftarrow}_{q_r}$ and $R^{\rightarrow}_{q_r}$ generate an R-atom $\mathsf{R}(u, v, w)$ where u and v both existentially quantified.
- Rule $R_{\mathsf{C_L}}$ generates an R-atom $\mathsf{R}(u, v, w)$ where u is existentially quantified and v is a frontier variable.

[3]We consider here weakly connected components; a weakly connected component in a directed graph is a maximal subgraph such that there is an undirected path between any two vertices.

- Rules $R_{\mathsf{C_R}}$ and R_{End} generate an R-atom $\mathsf{R}(u, v, w)$ where u is a frontier variable and v is existentially quantified.

Thus, all connected components are generated by a rule of the first kind, and then extended to the left by a rule of the second kind, and to the right by a rule of the third kind. Since no rule can generate an atom $\mathsf{R}(u, v, w)$ where u and v are both frontier variable (assuming u and v are not brakes), this yields the wanted structure. Finiteness is guaranteed by the emergency brakes. □

We now have a bit of structure to work with. Let us give a bit of intuition before concluding the proof. We have considered an infinite sequence $\mathcal{A} = (A_n)_{n\in\mathbb{N}}$ of state atoms, with $A_0 \in D$ and $A_n \prec A_{n+1}$ for all $n \in \mathbb{N}$, and we have just shown that to each state atom (not in D) is attached a bow tie structure. As mentioned before, the bow tie $\mathsf{bowtie}(A_n)$ consists in a set of (non-disjoint) paths that represent configurations that can be obtained from a configuration containing A_0 in the database, after n steps of simulation. In addition, Lemma 28 shows how each of these paths is constructed using a path from $\mathsf{bowtie}(A_{n-1})$. We also have seen in Example 26 that a bow tie can get split. From these two facts we get that the number of configurations represented by $\mathsf{bowtie}(A_n)$ decreases as n grows. Since this number is an integer, and each bow tie represents at least one configuration, this sequence will be stationnary at some point N. At this point, we know that each of the configurations represented by $\mathsf{bowtie}(A_N)$ visits q_r infinitely many time. Thus, we pick such a configuration ρ, and we show that the restricted chase does not terminate on $\langle \Sigma_M, D_\rho \rangle$, which is enough to conclude the proof by Lemma 22. We then formalize this argument.

Definition 29. *The* set of configurations $\mathsf{configs}(A_n)$ associated to a state atom $A_n = q(x, w) \in \mathcal{A}$, *with* $n > 0$, *is the set whose elements are the sets*

$$\{A_n\} \cup \bigcup_{i\leq m} \mathsf{brSet}(x_i, w) \cup \{\mathsf{P}(y_1, \ldots, y_k, w) \in F \mid P \in \{\mathsf{R}, \mathsf{0}, \mathsf{1}, \mathsf{B}, \mathsf{End}\} \text{ and } \forall i,\ y_i \in \{x_1, \ldots, x_m\}\}$$

for all maximal paths $(x_1, \ldots, x_m)$ *in* $\mathsf{bowtie}(A_n)$.

Lemma 30. *For all* $n > 0$, $\mathsf{configs}(A_n)$ *is finite, non-empty, and each of its elements homomorphically embeds into* D_ρ *for some configuration* ρ. *Also, there is an injective function* pred_n *from* $\mathsf{configs}(A_{n+1})$ *to* $\mathsf{configs}(A_n)$ *such that* $S \in \mathsf{configs}(A_{n+1})$ *can be generated using only atoms in* $\mathsf{pred}_n(S)$.

Proof sketch. To each set $S \in \mathsf{configs}(A_{n+1})$ we can associate a configuration ρ and a path p in $\mathsf{bowtie}(A_{n+1})$. We then define $\mathsf{pred}_n(S)$ as the set of atoms that was used to generate it, which is not hard: its associated configuration is an extension of a configuration that yields ρ, and its associated path is connected through the F-predicate to p. To show injectivity of pred_n, we then rely on a lemma stating that if $\mathsf{F}(x, z, w)$ and $\mathsf{F}(y, z, w)$ are both in F, then $x = y$. □

Since for all n, there is an injective function from $\mathsf{configs}(A_{n+1})$ to $\mathsf{configs}(A_n)$, the sequence $(|\mathsf{configs}(A_n)|)_{n\in\mathbb{N}_{>0}}$ is a decreasing sequence of natural numbers, as mentioned before. Thus, there must be some $N \in \mathbb{N}$ such that for all $n \geq N$, $|\mathsf{configs}(A_n)| = |\mathsf{configs}(A_N)| > 0$. We pick S_0 in $\mathsf{configs}(A_N)$, and let ρ be a configuration such that S_0 homomorphically embeds into D_ρ.

Lemma 31. *The restricted chase does not terminate on* $\langle \Sigma_M, D_\rho \rangle$.

Proof sketch. Since for all $n \geq N$, $|\mathsf{configs}(A_n)| = |\mathsf{configs}(A_N)|$, pred_n is actually a bijection. We thus define S_{n+1} as $\mathsf{pred}^{-1}_{N+n}(S_n)$. Intuitively, the sequence $(S_n)_{n\in\mathbb{N}}$ encodes the run of M that visits q_r infinitely many times from ρ. We then construct an infinite chase sequence from $\langle \Sigma_M, D_\rho \rangle$ such that S_n homomorphically embed in it for all n. □

By Lemma 22, this means that there is a run of M which visits q_r infinitely many times, and thus that M is not robust non-recurring through q_r, concluding the reduction.

6 An Alternative to Fairness to Simplify Restricted Chase Termination

All chase variants can be applied to semi-decide Boolean conjunctive query (BCQ) entailment. This is the case because, if a KB $\mathcal{K}$ entails a BCQ Q under standard first-order semantics, then every chase sequence of $\mathcal{K}$ features a fact set that entails Q. Consequently, it suffices to compute an (arbitrarily large) finite prefix of any (arbitrarily chosen) chase sequence of $\mathcal{K}$ to semi-decide whether $\mathcal{K}$ entails Q. Note that semi-decidability of BCQ entailment breaks down if we remove the fairness condition from the definition of a chase sequence. Unfortunately, this condition complicates the problem of universal termination for the restricted chase (see Theorems 9 and 23). To address this situation, we propose an alternative to fairness in the following definition that retains semi-decidability while simplifying the termination problem of the chase (see Theorem 34).

Definition 32. *A* breadth-first chase sequence *for a KB $\langle \Sigma, D\rangle$ is a chase derivation $F_0, F_1, \ldots$ such that, (†) if some Σ-trigger λ is loaded for some F_i, then there is some $j \in \{i, \ldots, i+n\}$ such that λ is obsolete for F_j and n is the (finite) number of Σ-triggers that are loaded and not obsolete for F_i.*

Note that, since (†) implies fairness as introduced in Definition 3, every breadth-first chase sequence is also a chase sequence and we preserve semi-decidability of BCQ entailment.

Definition 33. *Let $\mathsf{CTK}^{b1r}_\forall$ be the class of all KBs that only admit finite breadth-first chase sequences. Let $\mathsf{CTR}^{b1r}_\forall$ be the class containing a rule set if $\mathsf{CTK}^{b1r}_\forall$ contains all KBs with this rule set.*

Theorem 34. *The class $\mathsf{CTK}^{b1r}_\forall$ is in* RE*, and the class $\mathsf{CTR}^{b1r}_\forall$ is in Π^0_2.*

Proof. To show that $\mathsf{CTK}^{b1r}_\forall$ is in RE, we define a semi-decision procedure, which executes the following instructions on a given input KB $\mathcal{K} = \langle \Sigma, D\rangle$:

(1) Initialise the set $\mathcal{P}_1$ of lists of facts that contains the (unary) list D, and a counter $i := 2$.
(2) Compute the set C_i of all chase derivations of length i of $\mathcal{K}$ that can be obtained by extending a chase derivation in $\mathcal{P}_{i-1}$ with one fact set. Intuitively, C_i includes all lists of length i that can be extended into breadth-first chase sequences for $\mathcal{K}$.
(3) Compute the maximal subset $\mathcal{P}_i$ of C_i that does not contain a chase derivation $F_1, \ldots, F_i \in C_i$ if there is some $1 \leq k \leq i$ and some Σ-trigger λ such that λ is loaded for F_k, the trigger λ is not obsolete for F_i, and $i - k$ is larger than the number of Σ-triggers that are loaded and not obsolete for F_k. Intuitively, $\mathcal{P}_i$ filters out prefixes in C_i that already violate (†).
(4) If $\mathcal{P}_i$ is empty, *accept*. Otherwise, increment $i := i + 1$ and go to 2.

If the procedure accepts, then $\mathcal{P}_i$ is empty for some i and all breadth-first chase sequences of $\mathcal{K}$ are of length at most $i - 1$. If the procedure loops, then there is an infinite chase derivation $F_0, F_1, \ldots$ of $\mathcal{K}$ such that $F_0, \ldots, F_{i-1} \in \mathcal{P}_i$ for every $i \geq 1$, which is a breadth-first derivation for $\mathcal{K}$.

The class $\mathsf{CTR}^{b1r}_\forall$ is in Π^0_2 because we can semi-decide if a rule set Σ is not in $\mathsf{CTR}^{b1r}_\forall$ using an oracle that solves $\mathsf{CTK}^{b1r}_\forall$. We simply enumerate every database D, use the oracle to check if the KB $\langle \Sigma, D\rangle$ is in $\mathsf{CTK}^{b1r}_\forall$, and *accept* if this is not the case. □

The previous result holds because the condition (†) is *finitely verifiable*; that is, every infinite chase derivation that does not satisfy this condition has a finite prefix that witnesses this violation. Note that fairness does not have this property since any finite prefix of any chase derivation can be extended into a (fair) chase sequence. In fact, we can readily show a version of Theorem 34 for any other alternative condition if it is finitely verifiable. For an example of one such trigger application strategy, consider the one from [20], which is a bit more complex to define than (†) but nevertheless results in a very efficient implementation of the restricted chase.

7 Open Problems

After settling the general case regarding restricted chase termination and proposing an alternative fairness condition, there are still open challenges. Namely, what is the undecidability status of the classes $\mathsf{CTK}_\forall^{rest}$ and $\mathsf{CTR}_\forall^{rest}$ if we only consider single-head rules or only guarded (multi-head) rules? Note that, with guarded rules, it is not obvious how to simulate a Turing machine. For single-head rules, we cannot implement the emergency brake and thus our proofs do not apply. Moreover, if we only consider single-head rule sets, we can ignore fairness when determining restricted chase termination because of the "fairness theorem" [12]: a single-head KB admits an infinite (possibly unfair) chase derivation if and only if admits an infinite (fair) chase sequence. We think that answers to these problems will help to develop a better understanding for the (restricted) chase overall.

Acknowledgments

On TU Dresden side, this work is partly supported by Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) in project number 389792660 (TRR 248, Center for Perspicuous Systems), by the Bundesministerium für Bildung und Forschung (BMBF, Federal Ministry of Education and Research) in the Center for Scalable Data Analytics and Artificial Intelligence (project SCADS25B, ScaDS.AI), and by Bundesministerium für Bildung und Forschung (BMBF, Federal Ministry of Education and Research) and Deutscher Akademischer Austauschdienst (DAAD, German Academic Exchange Service) in project 57616814 (SECAI, School of Embedded and Composite AI).

References

[1] Bartosz Bednarczyk, Robert Ferens, and Piotr Ostropolski-Nalewaja. 2020. All-Instances Oblivious Chase Termination is Undecidable for Single-Head Binary TGDs. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, Christian Bessiere (Ed.). ijcai.org, 1719–1725. https://doi.org/10.24963/IJCAI.2020/238

[2] Marco Calautti, Georg Gottlob, and Andreas Pieris. 2015. Chase Termination for Guarded Existential Rules. In *Proceedings of the 34th ACM Symposium on Principles of Database Systems, PODS 2015, Melbourne, Victoria, Australia, May 31 - June 4, 2015*, Tova Milo and Diego Calvanese (Eds.). ACM, 91–103. https://doi.org/10.1145/2745754.2745773

[3] Marco Calautti and Andreas Pieris. 2021. Semi-Oblivious Chase Termination: The Sticky Case. *Theory Comput. Syst.* 65, 1 (2021), 84–121. https://doi.org/10.1007/S00224-020-09994-5

[4] David Carral, Irina Dragoste, and Markus Krötzsch. 2017. Restricted Chase (Non)Termination for Existential Rules with Disjunctions. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, Carles Sierra (Ed.). ijcai.org, 922–928. https://doi.org/10.24963/IJCAI.2017/128

[5] David Carral, Lucas Larroque, Marie-Laure Mugnier, and Michaël Thomazo. 2022. Normalisations of Existential Rules: Not so Innocuous!. In *Proceedings of the 19th International Conference on Principles of Knowledge Representation and Reasoning, KR 2022, Haifa, Israel, July 31 - August 5, 2022*, Gabriele Kern-Isberner, Gerhard Lakemeyer, and Thomas Meyer (Eds.). https://proceedings.kr.org/2022/11/

[6] Alin Deutsch, Alan Nash, and Jeffrey B. Remmel. 2008. The chase revisited. In *Proceedings of the Twenty-Seventh ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2008, June 9-11, 2008, Vancouver, BC, Canada*, Maurizio Lenzerini and Domenico Lembo (Eds.). ACM, 149–158. https://doi.org/10.1145/1376916.1376938

[7] Haim Gaifman, Harry G. Mairson, Yehoshua Sagiv, and Moshe Y. Vardi. 1993. Undecidable Optimization Problems for Database Logic Programs. *J. ACM* 40, 3 (1993), 683–713. https://doi.org/10.1145/174130.174142

[8] Lukas Gerlach and David Carral. 2023. Do Repeat Yourself: Understanding Sufficient Conditions for Restricted Chase Non-Termination. In *Proceedings of the 20th International Conference on Principles of Knowledge Representation and Reasoning, KR 2023, Rhodes, Greece, September 2-8, 2023*, Pierre Marquis, Tran Cao Son, and Gabriele Kern-Isberner (Eds.). 301–310. https://doi.org/10.24963/KR.2023/30

[9] Lukas Gerlach and David Carral. 2023. General Acyclicity and Cyclicity Notions for the Disjunctive Skolem Chase. In *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*, Brian Williams, Yiling Chen, and Jennifer Neville (Eds.). AAAI Press, 6372–6379. https://doi.org/10.1609/AAAI.V37I5.25784

[10] Tomasz Gogacz and Jerzy Marcinkowski. 2014. All-Instances Termination of Chase is Undecidable. In *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014,*

Proceedings, Part II (Lecture Notes in Computer Science, Vol. 8573), Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias (Eds.). Springer, 293–304. https://doi.org/10.1007/978-3-662-43951-7_25

[11] Tomasz Gogacz and Jerzy Marcinkowski. 2014. Termination of oblivious chase is undecidable. *CoRR* abs/1401.4840 (2014). arXiv:1401.4840 http://arxiv.org/abs/1401.4840

[12] Tomasz Gogacz, Jerzy Marcinkowski, and Andreas Pieris. 2023. Uniform Restricted Chase Termination. *SIAM J. Comput.* 52, 3 (2023), 641–683. https://doi.org/10.1137/20M1377035

[13] Gösta Grahne and Adrian Onet. 2018. Anatomy of the Chase. *Fundam. Informaticae* 157, 3 (2018), 221–270. https://doi.org/10.3233/FI-2018-1627

[14] Bernardo Cuenca Grau, Ian Horrocks, Markus Krötzsch, Clemens Kupke, Despoina Magka, Boris Motik, and Zhe Wang. 2013. Acyclicity Notions for Existential Rules and Their Application to Query Answering in Ontologies. *J. Artif. Intell. Res.* 47 (2013), 741–808. https://doi.org/10.1613/JAIR.3949

[15] David Harel. 1986. Effective transformations on infinite trees, with applications to high undecidability, dominoes, and fairness. *J. ACM* 33, 1 (jan 1986), 224–248. https://doi.org/10.1145/4904.4993

[16] Markus Krötzsch, Maximilian Marx, and Sebastian Rudolph. 2019. The Power of the Terminating Chase (Invited Talk). In *22nd International Conference on Database Theory, ICDT 2019, March 26-28, 2019, Lisbon, Portugal (LIPIcs, Vol. 127)*, Pablo Barceló and Marco Calautti (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 3:1–3:17. https://doi.org/10.4230/LIPICS.ICDT.2019.3

[17] Michel Leclère, Marie-Laure Mugnier, Michaël Thomazo, and Federico Ulliana. 2019. A Single Approach to Decide Chase Termination on Linear Existential Rules. In *22nd International Conference on Database Theory, ICDT 2019, March 26-28, 2019, Lisbon, Portugal (LIPIcs, Vol. 127)*, Pablo Barceló and Marco Calautti (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 18:1–18:19. https://doi.org/10.4230/LIPICS.ICDT.2019.18

[18] Bruno Marnette. 2009. Generalized schema-mappings: from termination to tractability. In *Proceedings of the Twenty-Eigth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2009, June 19 - July 1, 2009, Providence, Rhode Island, USA*, Jan Paredaens and Jianwen Su (Eds.). ACM, 13–22. https://doi.org/10.1145/1559795.1559799

[19] Hartley Rogers, Jr. 1987. *Theory of recursive functions and effective computability (Reprint from 1967).* MIT Press. http://mitpress.mit.edu/catalog/item/default.asp?ttype=2&tid=3182

[20] Jacopo Urbani, Markus Krötzsch, Ceriel J. H. Jacobs, Irina Dragoste, and David Carral. 2018. Efficient Model Construction for Horn Logic with VLog - System Description. In *Automated Reasoning - 9th International Joint Conference, IJCAR 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14-17, 2018, Proceedings (Lecture Notes in Computer Science, Vol. 10900)*, Didier Galmiche, Stephan Schulz, and Roberto Sebastiani (Eds.). Springer, 680–688. https://doi.org/10.1007/978-3-319-94205-6_44

Received December 2024; revised February 2025; accepted March 2025