Proceedings of the 19th International Conference on Principles of Knowledge Representation and Reasoning
Main Track

Chasing Streams with Existential Rules

Jacopo Urbani[1] , **Markus Krötzsch**[2] , **Thomas Eiter**[3]
[1]Department of Computer Science, Vrije Universiteit Amsterdam, The Netherlands
[2]Knowledge-Based Systems Group, TU Dresden, Germany
[3]Institute of Logic and Computation, Technische Universität Wien, Austria
jacopo@cs.vu.nl,markus.kroetzsch@tu-dresden.de,eiter@kr.tuwien.ac.at

Abstract

We study reasoning with existential rules to perform query answering over streams of data. On static databases, this problem has been widely studied, but its extension to rapidly changing data has not yet been considered. To bridge this gap, we extend LARS, a well-known framework for rule-based stream reasoning, to support existential rules. For that, we show how to translate LARS with existentials into a semantics-preserving set of existential rules. As query answering with such rules is undecidable in general, we describe how to leverage the temporal nature of streams and present suitable notions of acyclicity that ensure decidability.

1 Introduction

Streaming data arises in many applications, fostered by the need of deriving timely insights from emerging information and the inherent impossibility of storing all available data (Margara et al. 2014). Stream reasoning has become a productive area of KR with many formalisms (Anicic et al. 2011; Le-Phuoc et al. 2011; Barbieri et al. 2010; Tiger and Heintz 2016; Dell'Aglio et al. 2017; Kharlamov et al. 2019; Wałega, Kaminski, and Cuenca Grau 2019). This multiplicity is justified by the breadth of scenarios where stream processing is useful. Many of the approaches are distinguished from classical temporal reasoning, e.g., since data snapshots (*windows*) play an important role to reduce data volumes.

A well-known formalism in this space is LARS (Beck, Dao-Tran, and Eiter 2018), which is a rule-based language for stream reasoning that combines concepts from logic programming with dedicated stream operators to express windows and temporal quantifiers. For example, the LARS rule r_1: $\boxplus^3 \,\Box \mathsf{beltTmp}(X, Y) \wedge \mathsf{high}(Y) \rightarrow \mathsf{warn}(X)$ issues a warning if the temperature on a conveyor belt has been high for all ($\Box$) last three time points ($\boxplus^3$).

Another prominent field in KR are *existential rules*, which are also used as a basis for ontological models, especially in applications with large amounts of data (Baget et al. 2011; Cuenca Grau et al. 2013; Gottlob, Lukasiewicz, and Pieris 2014). Other common names for these rules include *tuple-generating dependencies* (Abiteboul, Hull, and Vianu 1994) and *Datalog*$^\pm$ (Calì et al. 2010). As a simple example, the rule r_2: $\mathsf{belt}(X) \rightarrow \exists Y.\mathsf{beltOperator}(X, Y)$ expresses that every belt has an operator (even if unknown). Existential quantification is central for ontologies and provides high expressivity beyond plain Datalog (Krötzsch, Marx, and Rudolph 2019). While reasoning with existential rules is known to be undecidable in general (Beeri and Vardi 1981), many well-behaved language fragments and practical implementations exist (Benedikt et al. 2017; Urbani et al. 2018; Bellomarini, Sallinger, and Gottlob 2018).

Until now, however, these areas have not been combined, and stream reasoning approaches do not support existential rules. Even for logic-based ontology languages in general, solutions only seem to exist for specific cases where queries are rewritable (Kharlamov et al. 2019; Kalaycı et al. 2019). As a consequence, it is often unclear how existing ontological background knowledge can be used in stream reasoning.

Additionally, the lack of existential quantification prevents useful modelling techniques for stream analysis. In particular, existential quantification can be used to represent temporal events, possibly spanning multiple time points, or to track unknown individuals. For instance, it can be used to create a new incident ID if the temperature on a belt is high for too long, or to track a not-yet-recognized object within a bounding box in a video stream. Notice that while in principle events could be modeled without value invention, i.e., using ad-hoc relations, doing so would put an upper bound to the number of possible events which might be undesirable as the future stream is typically unknown. A similar argument applies to the example above about objects within bounding boxes: it is arguably more natural to introduce new values and treat them as first-class individuals.

With this motivation in mind, we developed an extension of existential rules with LARS-based temporal quantifiers called LARS$^+$. Due to the undecidability of query answering with existential rules, our objective are decidable fragments, with the following contributions:

- We introduce LARS$^+$ as an existential stream reasoning language with a model-theoretic semantics.
- We give a semantics-preserving transformation from LARS$^+$ to existential rules to allow query answering. Doing so allows us to exploit existing decidability results, but these are limited in their use of time. We thus present *time-aware extensions of acyclicity notions* for LARS$^+$ programs.
- Initial experiments suggest that our method is promising.[1]

[1]Source code is at https://github.com/karmaresearch/elars; for a longer version of this paper see (Urbani, Krötzsch, and Eiter 2022).

2 LARS$^+$

Currently, LARS and DatalogMTL (Brandt et al. 2017; Wałega, Kaminski, and Cuenca Grau 2019) are popular for rule-based reasoning on data streams. While we focus on LARS, some of our work may be adapted to DatalogMTL.

To cope with big data volumes, LARS allows one to restrict streams to data snapshots (i.e., substreams) taken by generic window operators $\boxplus$. Typically, windows are used to consider only the knowledge in the most recent past, but this is not enough to avoid a complexity explosion or even undecidability that could arise from reasoning over an indefinite future. To overcome this problem, it is common in this domain to restrict future predictions up to a horizon of interest h, which is moved forward indefinitely.

Our language LARS$^+$ can be viewed as an extension of existential rules with temporal features of LARS. In the choice of data-snapshot operators, we take inspiration from *plain LARS*, which is a LARS fragment that is apt for efficient implementation (Bazoobandi, Beck, and Urbani 2017).

Syntax We consider a two-sorted logic with abstract elements and the natural numbers $\mathbb{N}$ as time points. We assume infinite sets $\mathcal{V}_A$ of *abstract variables*, $\mathcal{V}_T$ of *time variables*, $\mathcal{N}$ of *labelled nulls*, and $\mathcal{C}$ of *constants* that are mutually disjoint and disjoint from $\mathbb{N}$. *Abstract terms* (resp. *time terms*) are elements of $\mathcal{V}_A \cup \mathcal{N} \cup \mathcal{C}$ (resp., $\mathcal{V}_T \cup \mathbb{N}$).

Predicates p are from a set $\mathcal{P}$ of predicates and have arity $\mathsf{ar}(p) \geq 0$, with each position typed (abstract or time sort). A *normal atom* is an expression $p(\mathbf{t})$, $\mathbf{t} = t_1, \ldots, t_{\mathsf{ar}}(p)$, where t_i is a term of proper sort. An *arithmetic atom* has the form $t_1 \leq t_2$ or $t_1 = t_2 + t_3$ for time terms t_1, t_2, t_3. The set of all *atoms* (normal and arithmetic) is denoted $\mathcal{A}$. For an atom α (or any other logical expression introduced below), the *domain* $\mathsf{dom}(\alpha)$ of α is the set of all terms in α; we write $\alpha[\mathbf{x}]$ to state that $\mathbf{x} = \mathsf{dom}(\alpha) \cap (\mathcal{V}_A \cup \mathcal{V}_T)$; and we say that α is *ground* if it contains no variables.

A predicate $p \in \mathcal{P}$ is *simple* if it has no position of time sort, while an atom is *simple* if it normal and has a simple predicate. A *LARS$^+$ atom* α has the form

$$\alpha \coloneqq a \mid b \mid @_T b \mid \boxplus^n @_T b \mid \boxplus^n \Diamond b \mid \boxplus^n \Box b \qquad (1)$$

where a is an arithmetic atom, b is either a null-free simple atom or $\top$ (which holds true at all times), T is a time term, and $n \in \mathbb{N}$. *Window operators* $\boxplus^n$ restrict attention back to n time points in the past, and $@_T$ (resp. $\Box$, $\Diamond$) indicates that a formula holds at time T (resp., *every*, *some* time point).

Arithmetic atoms do not depend on time, whereas atoms $@_T b$ refer to a specific time T. All other LARS$^+$ atoms are interpreted relative to some current time point. Simple atoms b can equivalently be written as $\boxplus^0 \Diamond b$ or as $\boxplus^0 \Box b$.

Definition 1. A *LARS$^+$ rule* is an expression of the form

$$r = \Box \forall \mathbf{x}, \mathbf{y}.(B[\mathbf{x}, \mathbf{y}] \to \exists \mathbf{v}.H[\mathbf{y}, \mathbf{v}]) \qquad (2)$$

where $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{v}$ are mutually disjoint sets of variables, and $\mathbf{v}$ contains only abstract variables; the body $B[\mathbf{x}, \mathbf{y}]$ is a conjunction of LARS$^+$ atoms; and the head $H[\mathbf{y}, \mathbf{v}]$ is a conjunction of atoms of the form b or $@_T b$ in (1). We set $\mathsf{b}(r) \coloneqq B$ and $\mathsf{h}(r) \coloneqq H$, and we usually omit the leading $\Box$ and universal quantifiers when writing rules.

A *LARS$^+$ program* is a finite set of *LARS$^+$ rules*; we denote the set of all such programs by $\mathbf{L}^+$.

Semantics Like for LARS, the semantics of LARS$^+$ is based on streams. Formally, a *stream* $S = (\mathbf{T}, v)$ consists of a timeline $\mathbf{T} = [0, h] \subset \mathbb{N}$ and an *evaluation function* $v : \mathbb{N} \to 2^{\mathcal{A}}$ such that, for all $t \in \mathbb{N}$, $v(t)$ is a set of ground normal atoms and $v(t) = \emptyset$ if $t \notin \mathbf{T}$. We call S a *data stream* if only extensional atoms occur in S, i.e., atoms with designated predicates not occurring in rule heads. Given $n \in \mathbb{N}$ and $t \in \mathbf{T}$, we write $w_n(S, t)$ for the stream $([0, t], v')$ where for any $t' \in \mathbb{N}$, $v'(t') = v(t')$ if $t - n \leq t' \leq t$, and $v'(t) = \emptyset$ otherwise; we call $w_n(S, t)$ a *window* of size n on S at t.

Models of LARS$^+$ are special streams. For a stream $S = (\mathbf{T}, v)$, a simple ground atom b, and $t, t', n \in \mathbb{N}$, we write:

$$\begin{aligned} &S, t \models b \quad \text{if } b \in v(t), \qquad S, t \models @_{t'} b \quad \text{if } S, t' \models b, \\ &S, t \models \Diamond b \,/\, \Box b \quad \text{if } S, t'' \models b \text{ for some / all } t'' \in \mathbf{T}, \\ &S, t \models \boxplus^n \beta \qquad \text{if } w_n(S, t), t \models \beta. \end{aligned}$$

Further, $S, t \models \top$ holds for all $t \in \mathbf{T}$ and $S, t \models a$ for all ground arithmetic atoms a that express a true relation on $\mathbb{N}$. To define satisfaction of rules on a stream S at time point t, we introduce the auxiliary notion of $\mathbf{T}$-*match* σ for a set C of atoms on S and t as a sort-preserving mapping from the variables of C to terms, s.t. (i) each time variable X is mapped to $\mathbf{T}$ ($X\sigma \in \mathbf{T}$) and (ii) $S, t \models \alpha\sigma$ for each $\alpha \in C$.

Definition 2. A LARS$^+$ rule r as in (2) is *satisfied* by a stream $S = (\mathbf{T}, v)$, written $S \models r$, if either (i) $\mathsf{h}(r)$ contains some time point $t \notin \mathbf{T}$ (i.e., ignore inference out of scope), or (ii) for all $t \in \mathbf{T}$, every $\mathbf{T}$-match σ of $\mathsf{b}(r)$ on S and t is extendible to a $\mathbf{T}$-match $\sigma' \supseteq \sigma$ of $\mathsf{b}(r) \cup \mathsf{h}(r)$ on S and t.

A program $P \in \mathbf{L}^+$ is satisfied by S, written $S \models P$, if $S \models r$ for all $r \in P$. A data stream $D = (\mathbf{T}', v')$ is satisfied by S, written $S \models D$, if $\mathbf{T}' \subseteq \mathbf{T}$ and $v'(t) \subseteq v(t)$ for all $t \in \mathbf{T}'$. We then call S a *model* of P resp. D.

Example 1. Consider the data stream $D = ([0, 9], v)$, where $v(t) = \{\mathsf{belt}(b_1), \mathsf{high}(90), \mathsf{beltTmp}(b_1, tmp(t))\}$ for each $t \in [0, 9]$, where $tmp(t) = 90$ if $t \leq 4$ and $tmp(t) = 70$ otherwise. Then any model S of the rules r_1, r_2 in Section 1 and D fulfills $S, 4 \models \mathsf{warn}(b_1) \wedge \mathsf{beltOperator}(b_1, v)$ for some constant or null v. Similarly $S, 5 \models \mathsf{beltOperator}(b_1, v')$ for some constant or null v' while $S, 5 \models \mathsf{warn}(b_1)$ may fail.

3 Query Answering with LARS$^+$

The query answering problem in LARS$^+$ is as follows.

Definition 3. A *LARS$^+$ Boolean Conjunctive Query (BCQ)* q has the form $\exists \mathbf{x}.Q[\mathbf{x}]$, where Q is a conjunction of LARS$^+$ atoms. A stream $S = (\mathbf{T}, v)$ satisfies q at time t, written $S, t \models q$, if some $\mathbf{T}$-match σ of Q on S and t exists. A program $P \in \mathbf{L}^+$ and data stream D *entail* q at time t, written $P, D, t \models q$, if $S, t \models q$ for every model S of P and D.

For instance, a BCQ could be $\exists X. \boxplus^5 \Box \mathsf{warn}(X)$, which asks if there has been a warning over the same belt in the last 5 time points. To solve BCQ answering with LARS$^+$, we propose a consequence-preserving rewriting $\mathsf{rew}(\cdot)$ to existential rules with a time sort. This rewriting is useful because it will allow us to exploit known results for existential rules, e.g., acyclicity notions (Cuenca Grau et al. 2013).

Our proposed rewriting of P into $\mathsf{rew}(P)$ has 5 steps:
(1) Each atom $\boxplus^n \Diamond p(\mathbf{t})$ is replaced by $\boxplus^n @_T\, p(\mathbf{t})$, where T is a fresh variable used only in one atom.
(2) For any simple predicate p, we add auxiliary predicates $[\![\boxplus\Box\, p]\!]$ and $[\![\boxplus @\, p]\!]$ of arity $\mathsf{ar}(p)+2$ resp. $\mathsf{ar}(p)+3$. Intuitively, $[\![\boxplus\Box\, p]\!](\mathbf{t}, n, C)$ and $[\![\boxplus @\, p]\!](\mathbf{t}, n, T, C)$ mean that $\boxplus^n \Box p(\mathbf{t})$ and $\boxplus^n @_T\, p(\mathbf{t})$ hold at time C, respectively.
(3) Using a fresh variable C to represent the current time, we rewrite non-arithmetic atoms α in P (where $\top$ is $\top()$) to

$$\mathsf{rew}(\alpha) = \begin{cases} [\![\boxplus\Box\, p]\!](\mathbf{t}, 0, C) & \text{if } \alpha = p(\mathbf{t}), \\ [\![\boxplus\Box\, p]\!](\mathbf{t}, 0, T) & \text{if } \alpha = @_T\, p(\mathbf{t}), \\ [\![\boxplus\Box\, p]\!](\mathbf{t}, n, C) & \text{if } \alpha = \boxplus^n \Box p(\mathbf{t}), \\ [\![\boxplus @\, p]\!](\mathbf{t}, n, T, C) & \text{if } \alpha = \boxplus^n @_T\, p(\mathbf{t}) \end{cases}$$

(4) We add $[\![\boxplus\Box\, \top]\!](0, C)$ in rule bodies not containing C.
(5) For every predicate p (including $\top$), we add the following rules to P, where $\mathbf{X}$ is a list of variables of length $\mathsf{ar}(p)$ and $m = \max(0, n \mid \boxplus^n \text{ occurs in } P)$:

$$0 \le C \rightarrow [\![\boxplus\Box\, \top]\!](0, C) \tag{3}$$

$$[\![\boxplus\Box\, p]\!](\mathbf{X}, 0, 0) \rightarrow [\![\boxplus\Box\, p]\!](\mathbf{X}, m, 0) \tag{4}$$

$$[\![\boxplus\Box\, p]\!](\mathbf{X}, N', C) \wedge N' {=} N{+}1 \rightarrow [\![\boxplus\Box\, p]\!](\mathbf{X}, N, C) \tag{5}$$

$$\begin{aligned}[\![\boxplus\Box\, p]\!](\mathbf{X}, N, C) \wedge N' {=} N{+}1 \wedge N' {\le} m \wedge C' {=} C{+}1 \\ \wedge\, [\![\boxplus\Box\, p]\!](\mathbf{X}, 0, C') \rightarrow [\![\boxplus\Box\, p]\!](\mathbf{X}, N', C')\end{aligned} \tag{6}$$

$$[\![\boxplus\Box\, p]\!](\mathbf{X}, 0, C) \rightarrow [\![\boxplus @\, p]\!](\mathbf{X}, 0, C, C) \tag{7}$$

$$\begin{aligned}[\![\boxplus @\, p]\!](\mathbf{X}, N, T, C) \wedge N' {\le} m \wedge N' {=} N{+}1 \\ \wedge\, I \le 1 \wedge C' {=} C{+}I \rightarrow [\![\boxplus @\, p]\!](\mathbf{X}, N', T, C')\end{aligned} \tag{8}$$

We rewrite a LARS$^+$ BCQ $\exists \mathbf{x}.Q$ and time point t similarly to $\mathsf{rew}(\exists \mathbf{x}.Q, t) = \exists \mathbf{x}. \bigwedge_{\alpha \in Q} \mathsf{rew}(\alpha) \wedge C {\le} t \wedge t {\le} C$ (treating atoms $\boxplus^n \Diamond p(\mathbf{t})$ as before), and a stream $S = (\mathbf{T}, v)$ to facts $\mathsf{rew}(S) = \{[\![\boxplus\Box\, p]\!](\mathbf{t}, 0, s) \mid p(\mathbf{t}) \in v(t), t \in \mathbf{T}\}$.

Example 2. We illustrate the rewriting on r_1. Step (2) creates predicates $[\![\boxplus\Box \mathsf{beltTmp}]\!]$, $[\![\boxplus\Box \mathsf{high}]\!]$, and $[\![\boxplus\Box \mathsf{warn}]\!]$ and Step (3) the rule $[\![\boxplus\Box \mathsf{high}]\!](X, Y, 3, C) \wedge$ $[\![\boxplus\Box \mathsf{high}]\!](Y, 0, C) \rightarrow [\![\boxplus\Box \mathsf{warn}]\!](X, 0, C)$. Step (5) adds auxiliary rules to implement the semantics; e.g., rule (6) ensures that "$\boxplus\Box$"-facts survive across time points, say if $[\![\boxplus\Box \mathsf{beltTmp}]\!](a, b, 0, 6)$ and $[\![\boxplus\Box \mathsf{beltTmp}]\!](a, b, 2, 5)$ hold, then $[\![\boxplus\Box \mathsf{beltTmp}]\!](a, b, 3, 6)$ should hold as well.

Let us denote by $P' \models_{\mathbf{T}} q'$ entailment of a BCQ q' from existential rules P' with timeline $\mathbf{T}$, which is defined using $\mathbf{T}$-matches as $P, D, t \models q$ but disregarding D and t. Then:

Theorem 1. *For any $P \in \mathbf{L}^+$, BCQ q, data stream D on $\mathbf{T}$, and $t \in \mathbf{T}$ holds $P, D, t \models q$ iff $\mathsf{rew}(P) \cup \mathsf{rew}(D) \models_{\mathbf{T}} \mathsf{rew}(q, t)$.*

Theorem 1 is important as it allows us to implement BCQ answering in LARS$^+$ using existential rule engines, e.g., GLog (Tsamoura et al. 2021); arithmetic atoms over $\mathbf{T}$ can be simulated with regular atoms: simply add the set $\mathsf{rew}(\mathbf{T})$ of all true instances of arithmetic atoms in P over $\mathbf{T}$ and view $\mathsf{rew}(P) \cup \mathsf{rew}(D) \cup \mathsf{rew}(\mathbf{T})$ as a single-sorted theory.

4 Decidability

As BCQ entailment over existential rules is undecidable, we desire that the rewriting $\mathsf{rew}(\cdot)$ falls into a known decidable fragment. Such may be defined by *acyclicity conditions* (Cuenca Grau et al. 2013), which ensure that a suitable *chase*, which is a versatile class of reasoning algorithms for existential rules (Benedikt et al. 2017) based on "applying" rules iteratively, will terminate over a given input. We use a variant of the *skolem chase* (Marnette 2009), using nulls instead of skolem terms (aka *semi-oblivious chase*), extended to the time sort (see (Urbani, Krötzsch, and Eiter 2022)).

Conditions like the canonical *weak acyclicity (WA)* (Fagin et al. 2005) ensure in fact *universal termination*, i.e., chase termination for a given rule set over all sets of input facts. We can thus apply such criteria to $\mathsf{rew}(P)$ (viewed as single-sorted theory) while ignoring $\mathsf{rew}(D)$ and $\mathsf{rew}(\mathbf{T})$. Universal termination may here be seen as an analysis that disregards time. To formalise this, let $\mathsf{strip}(P)$ result from P by deleting all arithmetic atoms, window operators, and temporal quantifiers, and let $\mathbf{CT}$ and $\mathbf{WA}$ be the classes of all rule sets on which the skolem chase universally terminates and of all weakly acyclic rule sets, respectively. Then:

Theorem 2. *For any $P \in \mathbf{L}^+$, we have (i) $\mathsf{strip}(P) \in \mathbf{CT}$ iff $\mathsf{rew}(P) \in \mathbf{CT}$ and (ii) $\mathsf{strip}(P) \in \mathbf{WA}$ iff $\mathsf{rew}(P) \in \mathbf{WA}$.*

Analogous results hold for elaborated acyclicity notions (Cuenca Grau et al. 2013). Notably, we can check acyclicity on the simpler rule set $\mathsf{strip}(P)$. With WA as a representative notion, we let $\mathbf{L}^+_{\mathsf{LWA}} = \{P \in \mathbf{L}^+ \mid \mathsf{strip}(P) \in \mathbf{WA}\}$.

While easy to check, universal termination also considers situations that are impossible on properly encoded streams.

Example 3. Consider $P = \{@_T\, p(X, Y) \wedge T' {=} T{+}1 \rightarrow \exists V. @_{T'}\, p(Y, V)\}$. The skolem chase on $\mathsf{rew}(P) \cup \mathsf{rew}(D) \cup \mathsf{rew}(\mathbf{T})$ terminates on all $\mathbf{T}$ and D, but not universally for non-standard timelines where e.g., $0 = 0 + 1$ holds. That is, reasoning with P always terminates despite $P \notin \mathbf{L}^+_{\mathsf{LWA}}$.

We thus introduce *time-aware acyclicity*, which retains relevant temporal information instead of working with $\mathsf{strip}(P)$ only. First, to simplify P, we fix a fresh time variable N and replace all LARS$^+$ atoms in all rules as follows:

$$p(\mathbf{t}) \mapsto @_N\, p(\mathbf{t}) \quad (9) \qquad \boxplus^n @_T\, p(\mathbf{t}) \mapsto @_T\, p(\mathbf{t}) \quad (11)$$

$$\boxplus^n \Box p(\mathbf{t}) \mapsto @_N\, p(\mathbf{t}) \quad (10) \qquad \boxplus^n \Diamond p(\mathbf{t}) \mapsto @_U\, p(\mathbf{t}) \quad (12)$$

where (9) refers to atoms with no surrounding LARS$^+$ operators and U in (12) is a fresh time variable unique for each replacement; arithmetic atoms are kept unchanged. The resulting program is denoted by $\mathsf{wfree}(P)$ ("*window-free*").

Example 4. Let P consist of the following rules:

$$\boxplus^3 \Box p(X) \rightarrow \exists Y. q(X, Y) \tag{13}$$

$$@_T\, q(X, Y) \wedge U = T + 1 \rightarrow @_U\, p(Y) \tag{14}$$

As in Example 3, the skolem chase on $\mathsf{rew}(P)$ terminates if the given input data encodes a valid timeline, else it may not (indeed, $P \notin \mathbf{L}^+_{\mathsf{LWA}}$). In $\mathsf{wfree}(P)$, (13) is changed to

$$@_N\, p(X) \rightarrow \exists Y. @_N\, q(X, Y) \tag{15}$$

Intuitively, in $\mathsf{wfree}(P)$, N is the time at which rules are evaluated and localises all simple atoms to it; windows are removed and their restrictions relaxed: $\boxplus^n \Box$ ("at all times in window up to now") becomes $@_N$ ("now"); $\boxplus^n @_T$ ("at T if in window") becomes $@_T$; and $\boxplus^n \Diamond$ ("at some time in window") becomes $@_U$ ("at some time"). As this logically weakens rule bodies, $\mathsf{wfree}(P)$ has more logical consequences than P. We obtain the following useful insight:

Theorem 3. *For every $P \in \mathbf{L}^+$ and data stream D, if the skolem chase terminates on* $\mathsf{rew}(\mathsf{wfree}(P))$ *and* $\mathsf{rew}(D)$, *then it also terminates on* $\mathsf{rew}(P)$ *and* $\mathsf{rew}(D)$.

To exploit Theorem 3, we study the chase termination over $\mathsf{rew}(\mathsf{wfree}(P))$ while restricting to actual timelines, which are incorporated by partial grounding.

Definition 4. The *partial grounding* $\mathsf{grnd}_A(P)$ of a program P for a set A of null-free facts over a set $\mathcal{P}_A$ of predicates not occurring in rule heads of P, is the set of all rules $(B \backslash B_A \to \exists \mathbf{z}.H)\sigma$, where B_A are the atoms in B with predicate in $\mathcal{P}_A$, s.t. a rule $B \to \exists \mathbf{z}.H \in P$ and a homomorphism σ between B_A and A exist, i.e., a sort- and constant- preserving mapping $\sigma : \mathsf{dom}(B_A) \to \mathsf{dom}(A)$ s.t. $B_A\sigma \subseteq A$.

As long as A comprises all facts over $\mathcal{P}_A$, $\mathsf{grnd}_A(P)$ has the same models as P and the chase is also preserved. We use this to ground the time sort in LARS$^+$:

Definition 5. Given a program P, the *temporal grounding* of $\mathsf{wfree}(P)$ for a timeline $\mathbf{T}$, denoted $\mathsf{tgrnd}_\mathbf{T}(P)$, is the partial grounding $\mathsf{grnd}_{a(\mathbf{T},P)}(P')$ where

- P' results from $\mathsf{rew}(\mathsf{wfree}(P))$ by adding, for each $T \in \mathcal{V}_T$ in each rule body B, an atom $T \leq T$ to B and
- $a(\mathbf{T}, P)$ is the set of all ground instances of arithmetic atoms in P with values from $\mathbf{T}$ that are true over $\mathbb{N}$.

Example 5. For $\mathsf{wfree}(P)$ from Example 4 and timeline $\mathbf{T} = [0, 1]$, the temporal grounding is as follows (the deleted ground instances of B_A are shown in parentheses):

$$\llbracket \boxplus \Box p \rrbracket(X, 0, 0) \to \exists Y. \llbracket \boxplus \Box q \rrbracket(X, Y, 0, 0) \qquad (0 \leq 0)$$
$$\llbracket \boxplus \Box p \rrbracket(X, 0, 1) \to \exists Y. \llbracket \boxplus \Box q \rrbracket(X, Y, 0, 1) \qquad (1 \leq 1)$$
$$\llbracket \boxplus \Box q \rrbracket(X, Y, 0, 0) \to \llbracket \boxplus \Box p \rrbracket(Y, 0, 1) \qquad (1 = 0 + 1)$$

While universal termination on $\mathsf{tgrnd}_\mathbf{T}(P)$, which can be recognized in Example 5 using e.g. MFA (Cuenca Grau et al. 2013), ensures chase termination on $\mathsf{rew}(P)$ and $\mathsf{rew}(D)$ for all data streams D on $\mathbf{T}$, simpler, position-based notions like WA still fail. We thus encode time into predicate names:

Definition 6. Let P be an existential rules program with atoms of form $\llbracket \boxplus \Box p \rrbracket(\mathbf{s}, 0, t)$ only, where t is a time point. Then $\mathsf{tfree}(P)$ is obtained by replacing each $\llbracket \boxplus \Box p \rrbracket(\mathbf{s}, 0, t)$ with $\llbracket p \rrbracket_t(\mathbf{s})$ for a fresh predicate $\llbracket p \rrbracket_t$ of proper signature.

Let $\mathsf{tfgrnd}_\mathbf{T}(P) := \mathsf{tfree}(\mathsf{tgrnd}_\mathbf{T}(P))$. The following result shows that this is a good basis to check for acyclicity.

Theorem 4. *If* $\mathsf{tfgrnd}_\mathbf{T}(P)$ *is weakly acyclic for* $P \in \mathbf{L}^+$ *and timeline* $\mathbf{T}$, *then the skolem chase terminates on* $\mathsf{rew}(P)$ *and* $\mathsf{rew}(D)$ *for all data streams* D *on* $\mathbf{T}$.

Example 6 (cont'd). As $\mathsf{tfgrnd}_\mathbf{T}(P)$ is WA, by Theorem 4 the skolem chase on $\mathsf{rew}(P)$ and $\mathsf{rew}(D)$ always terminates.

In view of Theorem 4, we call $P \in \mathbf{L}^+$ *temporally weakly acyclic* (TLWA) over $\mathbf{T}$ if $\mathsf{tfgrnd}_\mathbf{T}(P)$ is WA, and denote by $\mathbf{L}^+_{\mathsf{TLWA}}(\mathbf{T})$ the class of all such programs P. We then have:

Theorem 5. $\mathbf{L}^+_{\mathsf{LWA}} \subset \mathbf{L}^+_{\mathsf{TLWA}}(\mathbf{T})$ *holds for all* $\mathbf{T}$ *s.t.* $|\mathbf{T}| \geq 2$.

Regarding complexity, as $\mathsf{rew}(\mathbf{T})$ is polynomial *in the length of* $\mathbf{T}$, it is exponential if $\mathbf{T}$ is encoded in binary. However, a polynomial axiomatisation of time is feasible, following the idea to encode numbers 0,1,...,m using sequences of $\lceil \log_2 m \rceil$ bits and to define predicates on them, cf. (Dantsin et al. 2001), such that for the resulting rewriting $\mathsf{rew}_\mathbf{T}(\cdot)$ instead of $\mathsf{rew}(\cdot)$, Theorems 1 and 2 hold analogously.

S_A: $p_1/p_2/p_3$	*Run*	*Mem*	*Out*	S_B: n	*Run*	*Mem*	*Out*
0.0/0.0/0.0	13.12ms	21.9	10.5k	0	0.6s	45.0	36k
0.3/0.3/0.5	13.34ms	22.7	10.7k	2	1.3s	81.8	64k
0.7/0.7/1.0	13.67ms	22.9	10.7k	4	2.6s	114.4	82k

Table 1: Preliminary experiments for scenario S_A and S_B

BCQ answering for $\mathbf{L}^+_{\mathsf{TLWA}}(\mathbf{T})$ is as for WA rules 2ExpTime-complete in general (on extensional streams, i.e, all $v(t)$, $t \in \mathbf{T}$, are listed). The P-complete data complexity for WA rules carries over to $\mathbf{L}^+_{\mathsf{LWA}}$ but gets 2ExpTime-hard for $\mathbf{L}^+_{\mathsf{TLWA}}(\mathbf{T})$, as hardest WA programs with bounded predicate arities (Calì, Gottlob, and Pieris 2010) can be emulated.

5 Preliminary Evaluation and Conclusion

We implemented an experimental prototype in Python, which is fed with the stream pointwise. At each time point, it computes the LARS$^+$ model with the stream collected up to the last ℓ time points, using the rewriting in Section 3 and the chase implementation of GLog (Tsamoura et al. 2021).

We considered two scenarios S_A and S_B. The first, S_A, is a toy example with conveyor belts and sensors that measure speed and temperature. The program contains 5 simple rules and the stream is parametrized by probability values p_1, p_2, and p_3 that regulate the number of rule executions (higher values lead to more reasoning). Scenario S_B is much more complex than S_A. We considered the dataset *Deep100* from the *ChaseBench* suite (Benedikt et al. 2017), which is a stress test of chase engines. We created a stream by copying all facts on each time point and rewrote the original rules using LARS$^+$ operators and different window sizes n. More details are available at (Urbani, Krötzsch, and Eiter 2022).

Table 1 reports multiple metrics obtained using a laptop, viz. avg. runtime (*Run*), avg. peak use of RAM (in MB, *Mem*), and avg. model size (# facts, *Out*). Notably, a LARS$^+$ model can be computed rather quickly, viz. in $\approx$13ms with an hypothetical input like S_A. This suggests that our approach can be used in scenarios that need fast response times. For "heavier" scenarios like S_B, the runtime increases but still stays within few seconds. Moreover, reasoning used at most 114MB of RAM; thus it may be done on limited hardware, e.g., sensors or edge devices.

Conclusion. Our work shows that combining existential rules with LARS can give rise to a versatile stream reasoning formalism with expressive features which is still decidable. A worthwhile future objective is to develop more efficient algorithms to compute the models. Our translation to existential rules is a good basis, but many optimisations are conceivable. On the theoretical side, a study of further decidability paradigms, especially related to guarded logics, is suggestive. Finally, further extensions towards nonmonotonic reasoning or other issues, like window validity (Ronca et al. 2018), are challenging for existential rules, but would be very useful for stream reasoning.

Acknowledgments

The authors would like to thank Mike Spadaru for his work on earlier versions of the prototype used in this work.

This work was partly supported by *Deutsche Forschungsgemeinschaft* (DFG) in project 389792660 (TRR 248, Center for Perspicuous Systems), by the *Bundesministerium für Bildung und Forschung* (BMBF) in project ScaDS.AI (Center for Scalable Data Analytics and Artificial Intelligence), and by the Center for Advancing Electronics Dresden (cfaed).

References

Abiteboul, S.; Hull, R.; and Vianu, V. 1994. *Foundations of Databases*. Addison Wesley.

Anicic, D.; Fodor, P.; Rudolph, S.; and Stojanovic, N. 2011. EP-SPARQL: A Unified Language for Event Processing and Stream Reasoning. In *WWW*, 635–644.

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

Barbieri, D. F.; Braga, D.; Ceri, S.; Valle, E. D.; and Grossniklaus, M. 2010. C-SPARQL: a continuous query language for rdf data streams. *International Journal of Semantic Computing* 04(01):3–25.

Bazoobandi, H. R.; Beck, H.; and Urbani, J. 2017. Expressive Stream Reasoning with Laser. In *ISWC*, 87–103.

Beck, H.; Dao-Tran, M.; and Eiter, T. 2018. LARS: A Logic-based framework for Analytic Reasoning over Streams. *Artificial Intelligence* 261:16–70.

Beeri, C., and Vardi, M. Y. 1981. The Implication Problem for Data Dependencies. In *ICALP*, 73–85.

Bellomarini, L.; Sallinger, E.; and Gottlob, G. 2018. The Vadalog System: Datalog-based Reasoning for Knowledge Graphs. *PVLDB* 11(9):975–987.

Benedikt, M.; Konstantinidis, G.; Mecca, G.; Motik, B.; Papotti, P.; Santoro, D.; and Tsamoura, E. 2017. Benchmarking the Chase. In *PODS*, 37–52.

Brandt, S.; Kalaycı, E. G.; Kontchakov, R.; Ryzhikov, V.; Xiao, G.; and Zakharyaschev, M. 2017. Ontology-Based Data Access with a Horn Fragment of Metric Temporal Logic. In *AAAI*, 1070–1076.

Calì, A.; Gottlob, G.; Lukasiewicz, T.; Marnette, B.; and Pieris, A. 2010. Datalog+/-: A Family of Logical Knowledge Representation and Query Languages for New Applications. In *LICS*, 228–242.

Calì, A.; Gottlob, G.; and Pieris, A. 2010. Query Answering under Non-guarded Rules in Datalog+/-. In *RR*, 1–17.

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

Dantsin, E.; Eiter, T.; Gottlob, G.; and Voronkov, A. 2001. Complexity and expressive power of logic programming. *ACM Computing Surveys* 33(3):374–425.

Dell'Aglio, D.; Della Valle, E.; van Harmelen, F.; and Bernstein, A. 2017. Stream reasoning: A survey and outlook. *Data Science* 1(1-2):59–83.

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

Gottlob, G.; Lukasiewicz, T.; and Pieris, A. 2014. Datalog+/-: Questions and Answers. In *KR*, 682–685.

Kalaycı, E. G.; Brandt, S.; Calvanese, D.; Ryzhikov, V.; Xiao, G.; and Zakharyaschev, M. 2019. Ontology-based access to temporal data with ontop: A framework proposal. *Applied Mathematics and Computer Science* 29(1):17–30.

Kharlamov, E.; Kotidis, Y.; Mailis, T.; Neuenstadt, C.; Nikolaou, C.; Özçep, Ö. L.; Svingos, C.; Zheleznyakov, D.; Ioannidis, Y. E.; Lamparter, S.; Möller, R.; and Waaler, A. 2019. An ontology-mediated analytics-aware approach to support monitoring and diagnostics of static and streaming data. *Journal of Web Semantics* 56:30–55.

Krötzsch, M.; Marx, M.; and Rudolph, S. 2019. The Power of the Terminating Chase. In *ICDT*, 3:1–3:17.

Le-Phuoc, D.; Dao-Tran, M.; Xavier Parreira, J.; and Hauswirth, M. 2011. A Native and Adaptive Approach for Unified Processing of Linked Streams and Linked Data. In *ISWC*, 370–388.

Margara, A.; Urbani, J.; Van Harmelen, F.; and Bal, H. 2014. Streaming the web: Reasoning over dynamic data. *Journal of Web Semantics* 25:24–44.

Marnette, B. 2009. Generalized Schema-Mappings: from Termination to Tractability. In *PODS*, 13–22.

Ronca, A.; Kaminski, M.; Cuenca Grau, B.; and Horrocks, I. 2018. The Window Validity Problem in Rule-Based Stream Reasoning. In *KR*, 571–581.

Tiger, M., and Heintz, F. 2016. Stream Reasoning Using Temporal Logic and Predictive Probabilistic State Models. In *TIME*, 196–205.

Tsamoura, E.; Carral, D.; Malizia, E.; and Urbani, J. 2021. Materializing knowledge bases via trigger graphs. *PVLDB* 14(6):943–956.

Urbani, J.; Krötzsch, M.; Jacobs, C. J. H.; Dragoste, I.; and Carral, D. 2018. Efficient Model Construction for Horn Logic with VLog - System Description. In *IJCAR*, 680–688.

Urbani, J.; Krötzsch, M.; and Eiter, T. 2022. Chasing Streams with Existential Rules. *arXiv:2205.02220 [cs]*. http://arxiv.org/abs/2205.02220.

Wałega, P. A.; Kaminski, M.; and Cuenca Grau, B. 2019. Reasoning over Streaming Data in Metric Temporal Datalog. In *AAAI*, 3092–3099.