24th European Conference on Artificial Intelligence - ECAI 2020
Santiago de Compostela, Spain

Weighted LARS for Quantitative Stream Reasoning

Thomas Eiter[1] and **Rafael Kiesel**[1]

Abstract. We extend LARS, which is a recent stream reasoning framework based on ASP, to weighted LARS (wLARS), where formulae are interpreted as algebraic expressions over semirings. This adds the ability to express quantitative measures of many different natures and to approach respective reasoning problems such as probabilistic reasoning, preferential reasoning and quantitative queries in a uniform manner. Notably, well-known quantitative ASP extensions can be formalized using wLARS, thus lifting them to the streaming setting. We identify a relevant wLARS fragment that is equivalent to weighted automata, which consequently gives us a rule-based language for expressing behaviors of such automata. Furthermore, we analyze evaluating wLARS formulae, showing that brave preferential reasoning is PSPACE- resp. Σ_3^p-complete in relevant settings.

1 Introduction

In recent years, there has been increasing interest in reasoning with non-static data in streams, due to the need for handling data in large quantities that is incrementally available and constantly changing [13], with applications in many domains ranging from traffic monitoring to public health [12]. The LARS framework [3] extends Answer Set Programming (ASP) with means to formulate complex relationships over data streams by rule-based programs, in which data snapshots (windows) are available as language elements. Evaluating LARS programs under stable semantics yields answer streams where rules derive a minimal amount of positive information that ensures supportedness and handles recursion and non-monotonic negation.

However, answer streams may only be an intermediate result in some contexts. Just like in ASP, we are interested in:

- probabilistic reasoning on the answer streams;
- preferential reasoning, i.e. considering only answer streams that are most preferred w.r.t. some measure and ordering;
- quantitative questions ("At how many timepoints does α hold?") rather than only qualitative ones ("Does α hold at some timepoint?").

All of these quantitative problems involve the specification of some measure on the answer streams, which is then normalized, optimized or simply evaluated depending on the problem at hand.

For ASP this fact is not commonly used. Instead the above problems are considered separately, in a multitude of different quantitative extensions for probabilistic reasoning [2, 22], preferential reasoning [23, 8, 7] and quantitative queries [21, 11]. These extensions are rather ad hoc, i.e., designed to solve a specific type of problem, and relationships between them, like the one between LP$^{\text{MLN}}$ [28] and weak constraints [8] are revealed in a posteriori analysis [22]. Also in the context of stream reasoning the formalisms PrASP [25] and MTL with incomplete states [10] are designed to solve a particular type of problem, viz. probabilistic stream reasoning and stream reasoning under incomplete probabilistic information, respectively.

[1] TU Wien, Austria, email: { thomas.eiter, rafael.kiesel }@tuwien.ac.at

We follow a different strategy and propose a general formalism that can be used to specify measures of many different natures on streams, while using homogeneous syntax and semantics. *Semirings* appear to be a suitable means for carrying out calculations that are commonly needed for evaluating measures on streams or interpretations. These are structures $\mathcal{R} = \langle R, \oplus, \otimes, e_\oplus, e_\otimes \rangle$ where intuitively $\oplus$ is addition and $\otimes$ multiplication over a set R with neutral elements $e_\oplus$ and $e_\otimes$, respectively. Calculations are modeled by algebraic expressions. We allow for expressions like $\alpha = \mathrm{Circus} \wedge 20 \vee \mathrm{Restaurant} \wedge 15$ in our LARS extension that use the syntax of formulae and semantics of algebraic expressions to mix qualitative and quantitative information. E.g., over the semiring $(\mathbb{Q}, +, \cdot, 0, 1)$ of the rational numbers, α may specify how much money is spent during an evening. When visiting the Circus but not the $\mathrm{Restaurant}$, the semantics of α is given by $1 \cdot 20 + 0 \cdot 15 = 20$.

Our main contributions are summarized as follows.

- We introduce weighted LARS (wLARS), inspired by weighted logics [14]. Here given a classical interpretation $\mathcal{I}$, a formula is not assigned a truth value but evaluated as an algebraic expression over a semiring $\mathcal{R}$, resulting in a value from it (e.g. a real number, subset of a set). Using wLARS formulae, one can intuitively specify calculations depending on the satisfaction of atoms in $\mathcal{I}$. There is no need to resort to a meta-level, and one can keep using the LARS syntax to specify measures. As the semiring is generic, there is no commitment to a fixed set of operations.
- We show that the weighted formalism is interesting for ASP in its own right by expressing the measures of well-known quantitative extensions of logic programming, viz. (a)ProbLog [11, 21] and LP$^{\text{MLN}}$ [22] in wLARS; for further extensions (e.g., P-Log [2], ASP with weak constraints [8], asprin [7]) this is also possible. As a byproduct, these formalisms are lifted to streams reasoning.
- We identify a fragment of wLARS that is as expressive as a fragment of weighted monadic second order (MSO) logic, and thus by [14] as expressive as weighted finite state automata. This generalizes a similar result for LARS in [3] to the quantitative setting. Thus, wLARS can be seen as a rule-based alternative to weighted automata over words, and similarly to rational formal power series [15].
- While evaluating wLARS formulae can have unbounded complexity, we identify settings where this is feasible in polynomial space resp. tractable. Based on this, we show that preferential reasoning with wLARS is PSPACE- resp. Σ_3^p-complete in relevant settings.

2 Preliminaries

LARS [3] is a stream reasoning framework, in which the observed data may vary between the different time points in the considered, discrete interval. For our purposes the information whether certain data is observed is encoded using propositional variables (*atoms*) from a finite set $\mathcal{A}$. Interpretations of LARS formulae are formalized

by *streams* S, which are pairs (T, v), where $T = \{t, t+1, \ldots, t+n\} \subseteq \mathbb{N}$ is the finite interval of discrete timepoints from t to $t+n$ and $v : T \to 2^{\mathcal{A}}$ expresses that at time $t' \in T$ the atoms $v(t')$ appear in the stream. We say a stream $S' = (T', v')$ is a substream of S, if $T' \subseteq T$ and for all $t' \in T' : v'(t') \subseteq v(t')$, denoted $S' \subseteq S$.

Definition 1 (Window Function)**.** *A* window function w *given a stream S and time point t restricts S to a substream $w(S,t) \subseteq S$.*

Intuitively such window functions filter the data in a stream and give a snapshot of the data, e.g. the data occurring in the last three time points, or the latest 20 atoms.

LARS formulae are defined by the grammar

$$\alpha ::= p \mid \neg\alpha \mid \alpha \wedge \alpha \mid \alpha \vee \alpha \mid \Diamond\alpha \mid \Box\alpha \mid @_t\alpha \mid \boxplus^w\alpha \mid \triangleright\alpha,$$

where $t \in \mathbb{N}$, $p \in \mathcal{A}$ and w is a window function. Here $\triangleright$ is the reset operator, which resets a stream $S \subseteq S^\star$ obtained by applying window functions(s) to the original stream $S^\star$. It allows for more succinct program specifications.

We use the shorthands $\bot := p \wedge \neg p$, $\top := \neg\bot$ and $\alpha \to \beta := \neg\alpha \vee \beta$. A *(pointed) LARS interpretation* is a tuple $\mathcal{I} = (S^\star, S, t)$ of streams $S^\star = (T^\star, v^\star)$, $S = (T, v)$ and a time point $t \in T^\star$, where $S \subseteq S^\star$. *Satisfaction* of a LARS formula α by $\mathcal{I}$, in symbols $\mathcal{I} \models \alpha$, is inductively defined by

$$\begin{aligned}
\mathcal{I} &\models p && \iff && p \in v(t), \text{ for } p \in \mathcal{A}\\
\mathcal{I} &\models \neg\alpha && \iff && \mathcal{I} \not\models \alpha\\
\mathcal{I} &\models \alpha \wedge \beta && \iff && \mathcal{I} \models \alpha \text{ and } \mathcal{I} \models \beta\\
\mathcal{I} &\models \alpha \vee \beta && \iff && \mathcal{I} \models \alpha \text{ or } \mathcal{I} \models \beta\\
\mathcal{I} &\models \Diamond\alpha && \iff && \exists t' \in T : (S^\star, S, t') \models \alpha\\
\mathcal{I} &\models \Box\alpha && \iff && \forall t' \in T : (S^\star, S, t') \models \alpha\\
\mathcal{I} &\models @_{t'}\alpha && \iff && (S^\star, S, t') \models \alpha \text{ and } t' \in T\\
\mathcal{I} &\models \boxplus^w\alpha && \iff && (S^\star, w(S,t), t) \models \alpha\\
\mathcal{I} &\models \triangleright\alpha && \iff && (S^\star, S^\star, t) \models \alpha
\end{aligned}$$

We identify the pair (S,t) with the LARS interpretation (S, S, t).

Definition 2 (LARS Program)**.** *A* LARS program *is a finite set Π of rules r of the form $r = \alpha \leftarrow \beta$, where α, β are LARS formulae. We write α and $\neg\beta$ as a shorthand for $\alpha \leftarrow \top$ and $\bot \leftarrow \beta$ respectively.*

A rule r is satisfied at a time point t by a stream S (written $(S,t) \models r$), if $(S,t) \models \beta \to \alpha$. Programs are seen as the conjunction of their rules; thus a stream S satisfies a program Π at time t (written $(S,t) \models \Pi$), if for all $r \in \Pi$, $(S,t) \models r$ holds.

We distinguish between *extensional* and *intensional* atoms contained in $\mathcal{A}^\epsilon$ and $\mathcal{A}^I$ respectively. Extensional atoms represent the input data (and therefore do not occur in rule heads).

Definition 3 (Data and Interpretation Stream)**.** *A* data stream *is a stream $D = (T, v)$ where v asserts only extensional atoms, i.e. $\forall t \in T : v(t) \subseteq \mathcal{A}^\epsilon$. A stream $S = (T, v')$ is an* interpretation stream *of D, if $D \subseteq S$ and v, v' agree on $\mathcal{A}^\epsilon$.*

Finally, answer sets semantics is defined using the FLP-*reduct* [17], which for a program Π w.r.t. a pair (S,t), is given by $\Pi^{S,t} = \{\alpha \leftarrow \beta \in \Pi \mid (S,t) \models \beta\}$.

Definition 4 (Answer Stream)**.** *An interpretation stream S of D is an* answer stream *for Π at t, if (S,t) satisfies Π and no interpretation stream $S' \subsetneq S$ of D exists s.t. (S',t) satisfies $\Pi^{S,t}$. We denote the set of such streams by $\mathcal{AS}(\Pi, D, t)$.*

We illustrate the basic usage of LARS using a running example.

Example 1 (Traveling)**.** *We know that a friend went on a journey, which satisfies the following constraints:*

- *Travel from S to E while staying within a set of cities $\mathcal{C}$.*
- *Only travel using transportation from the set $\mathcal{T}$.*
- *For the i^{th} trip we use $\overline{\tau}_i$, the i^{th} element of $\overline{\tau} \in \mathcal{T}^n$.*
- *Every trip changes the city.*
- *If there were traffic jams in a city for the last three days our friend does not leave the city using a bus.*

We further know the cost $\text{Cost}(c_1, \tau, c_2)$ of traveling between cities c_1 and c_2 using transport τ, which is ∞ when one cannot travel between two cities using τ. Given these constraints, we want to derive as much information as possible about the journey of our friend.

We can design a LARS program Π whose answer streams correspond to the possible journeys of our friend, using the rules

$$\neg\Diamond \bigvee_{c_1 \neq c_2 \in \mathcal{C}} \text{in}_{c_1} \wedge \text{in}_{c_2}, \qquad \Box \bigvee_{c \in \mathcal{C}} \text{in}_c, \tag{1}$$

$$\neg\Diamond \bigvee_{\text{Cost}(c_1,t,c_2)=\infty} \text{in}_{c_1} \wedge (\boxplus^{\text{next}}\Diamond \text{in}_{c_2}) \wedge \text{travel}_\tau, \tag{2}$$

$$\neg\Diamond \bigvee_{c \in \mathcal{C}} \text{in}_c \wedge (\boxplus^{\text{next}}\Diamond \text{in}_c), \tag{3}$$

$$\Diamond(\boxplus^{\text{next}}\Box\bot \wedge \text{in}_E), \qquad @_0 \text{in}_S, \tag{4}$$

$$\neg\Diamond \bigvee_{c \in \mathcal{C}} (\boxplus^3 \Box \text{traffic\_jam}_c) \wedge \text{in}_c \wedge \text{travel}_{\text{Bus}}. \tag{5}$$

The rules (1) *ensure that our friend is always in exactly one city. Rule* (2) *enforces that one can not travel between certain cities. According to rule* (3) *every trip changes city. The window function* next *in* (3) *gives us the stream restricted to the next timepoint or the empty stream if there is no next time point. Therefore $\boxplus^{\text{next}}\Diamond\alpha$ is similar to a strong next operator - it can only be satisfied when there is a next time point and α holds at it. The rules* (4) *take care of the starting and end city. Here, $\boxplus^{\text{next}}\Box\alpha$ is similar to a weak next operator as it is satisfied when there is no next time point or at the next time point α holds. Rule* (5) *implements the last constraint using the window operator $\boxplus^3$, which restricts the stream to the last three timepoints. The given sequence $\overline{\tau}$, referred to by the program using* travel_τ *is extensional, i.e. input data.*

For more background on LARS and a study of properties see [3].

3 Weighted LARS

In a weighted LARS (wLARS) formula α we allow as atoms besides propositional variables also values from a semiring $\mathcal{R}$. The semantics $[\![\alpha]\!]_{\mathcal{R}}(S,t)$ is not satisfaction but interpretation as an algebraic expression over the semiring $\mathcal{R}$.

Definition 5 (Semiring)**.** *A* semiring $\mathcal{R} = (R, \oplus, \otimes, e_\oplus, e_\otimes)$ *is a nonempty set R equipped with two binary operations $\oplus$ and $\otimes$, called addition and multiplication, such that*

- *$(R, \oplus)$ is a commutative monoid with identity element $e_\oplus$,*
- *$(R, \otimes)$ is a monoid with identity element $e_\otimes$,*
- *multiplication left and right distributes over addition, and*
- *multiplication by $e_\oplus$ annihilates R, i.e. $\forall r \in R : r \otimes e_\oplus = e_\oplus = e_\oplus \otimes r$.*

$\mathcal{R}$ is commutative*, if $(R, \otimes)$ is commutative; if a unary operator*

$$\begin{aligned}
-(\cdot) &: R \to R && \text{such that} && r \oplus -r = e_\oplus\\
\text{resp. } (\cdot)^{-1} &: R \setminus \{e_\oplus\} \to R && \text{such that} && r \otimes r^{-1} = e_\otimes
\end{aligned}$$

exists, then $\oplus$ (resp. $\otimes$) is invertible*.*

Some examples of semirings are

- $\mathbb{F} = (\mathbb{F}, +, \cdot, 0, 1)$, for $\mathbb{F} \in \{\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}\}$ the semiring of the numbers in $\mathbb{F}$ with addition and multiplication,
- $\mathcal{P}(A) = (2^A, \cup, \cap, \emptyset, A)$, the semiring over the powerset of A with union and intersection,
- $\mathbb{B} = (\{\mathbf{t}, \mathbf{f}\}, \vee, \wedge, \mathbf{f}, \mathbf{t})$, the boolean semiring, and
- $\mathcal{R}_{\text{trop}} = (\mathbb{Q} \cup \{\infty\}, \min, +, \infty, 0)$, the tropical semiring.

Semirings have already previously been successfully applied for the definition of uniform semantics in applications ranging from parsing [18] over provenance [19] to argumentation [6]. Inspired by Droste and Gastin's seminal work on weighted logics [14], we define weighted LARS and add two new connectives $\rightarrow_\oplus, \rightarrow_\otimes$.

Definition 6 (Weighted LARS Syntax and Semantics)**.** *A* wLARS formula *over a semiring $\mathcal{R} = (R, \oplus, \otimes, e_\oplus, e_\otimes)$ is of the form*

$$\alpha ::= p \mid \neg\alpha \mid \alpha \wedge \alpha \mid \alpha \vee \alpha \mid \Diamond\alpha \mid \Box\alpha \mid @_t\alpha \mid \boxplus^w\alpha \mid \triangleright\alpha \mid k \mid \alpha \rightarrow_\oplus \alpha \mid \alpha \rightarrow_\otimes \alpha,$$

where $t \in \mathbb{N}, k \in R$, $p \in \mathcal{A}$ and w is a window function. Further $\bot, \top$ and $\rightarrow$ are defined as previously. The connective $\rightarrow_\odot$ requires that $\odot$ is invertible ($\odot \in \{\oplus, \otimes\}$).

Given a wLARS formula α over semiring $\mathcal{R}$ and a LARS interpretation $\mathcal{I} = (S^\star, S, t)$ where $S^\star = (T^\star, v^\star), S = (T, v), t \in T^\star$, the semantics *of α w.r.t. $\mathcal{I}$ is given by*

$$\begin{aligned}
[\![k]\!]_\mathcal{R}(\mathcal{I}) &= k && \text{for } k \in R\\
[\![p]\!]_\mathcal{R}(\mathcal{I}) &= \begin{cases} e_\otimes, & \text{if } p \in v(t),\\ e_\oplus, & \text{otherwise}\end{cases}\\
[\![\neg\alpha]\!]_\mathcal{R}(\mathcal{I}) &= \begin{cases} e_\oplus, & \text{if } [\![\alpha]\!]_\mathcal{R}(\mathcal{I}) \neq e_\oplus,\\ e_\otimes, & \text{otherwise}\end{cases}\\
[\![\alpha \wedge \beta]\!]_\mathcal{R}(\mathcal{I}) &= [\![\alpha]\!]_\mathcal{R}(\mathcal{I}) \otimes [\![\beta]\!]_\mathcal{R}(\mathcal{I})\\
[\![\alpha \vee \beta]\!]_\mathcal{R}(\mathcal{I}) &= [\![\alpha]\!]_\mathcal{R}(\mathcal{I}) \oplus [\![\beta]\!]_\mathcal{R}(\mathcal{I})\\
[\![\Diamond\alpha]\!]_\mathcal{R}(\mathcal{I}) &= \bigoplus_{t' \in T} [\![\alpha]\!]_\mathcal{R}(S^\star, S, t')\\
[\![\Box\alpha]\!]_\mathcal{R}(\mathcal{I}) &= \bigotimes_{t' \in T} [\![\alpha]\!]_\mathcal{R}(S^\star, S, t')\\
[\![@_{t'}\alpha]\!]_\mathcal{R}(\mathcal{I}) &= \begin{cases} [\![\alpha]\!]_\mathcal{R}(S^\star, S, t'), & \text{if } t' \in T,\\ e_\oplus, & \text{otherwise.}\end{cases}\\
[\![\boxplus^w\alpha]\!]_\mathcal{R}(\mathcal{I}) &= [\![\alpha]\!]_\mathcal{R}(S^\star, w(S,t), t)\\
[\![\triangleright\alpha]\!]_\mathcal{R}(\mathcal{I}) &= [\![\alpha]\!]_\mathcal{R}(S^\star, S^\star, t)
\end{aligned}$$

For non-commutative $\otimes$ and $T = \{t, \ldots, t+n\}$ we define

$$\bigotimes_{t' \in T} f(t') = f(t) \otimes f(t+1) \otimes \cdots \otimes f(t+n).$$

If $\rightarrow_\oplus$ or $\rightarrow_\otimes$ are available, we define their semantics as

$$[\![\alpha \rightarrow_\oplus \beta]\!]_\mathcal{R}(\mathcal{I}) = [\![\beta]\!]_\mathcal{R}(\mathcal{I}) \oplus -([\![\alpha]\!]_\mathcal{R}(\mathcal{I})),$$

$$[\![\alpha \rightarrow_\otimes \beta]\!]_\mathcal{R}(\mathcal{I}) = \begin{cases} [\![\beta]\!]_\mathcal{R}(\mathcal{I}) \otimes ([\![\alpha]\!]_\mathcal{R}(\mathcal{I}))^{-1} & [\![\alpha]\!]_\mathcal{R}(\mathcal{I}) \neq e_\oplus\\ e_\oplus & [\![\alpha]\!]_\mathcal{R}(\mathcal{I}) = e_\oplus\end{cases}$$

The semantics of (possibly negated) atoms is given by the neutral elements $e_\oplus, e_\otimes$ of addition and multiplication. This fits naturally, as falsehood $\mathbf{f}$ corresponds to the neutral element of disjunction and truth $\mathbf{t}$ to the neutral element of conjunction.

The usage of $a \rightarrow_\otimes b$ for the formula $b \wedge a^i$ is borrowed from Heyting algebras [1] where the connective $\rightarrow$ has a similar role.

Observe that some equivalences from propositional logic do not hold anymore. The following example shows that $\neg\neg\alpha$ does not necessarily have the same semantics as α:

Example 2 (Traveling cont'd)**.** *Consider the LARS formula $\Diamond\text{in}_\text{P}$ over $\mathbb{N}$ the semiring of natural numbers. Then its semantics is the number of time points that our friend is in Paris:*

$$[\![\Diamond\text{in}_\text{P}]\!]_\mathbb{N}(S, S, t) = \textstyle\sum_{t' \in T, \text{in}_\text{P} \in v(t)} 1 = |\{t \in T \mid \text{in}_\text{P} \in v(t)\}|.$$

If we apply double negation, the semantics $[\![\neg\neg\Diamond\text{in}_\text{P}]\!]_\mathbb{N}(S, S, t)$ is equal to 1 if our friends visits Paris at least once and 0 otherwise.

In general the weighted semantics behaves differently than satisfaction in LARS. However, we see that the weighted semantics is a generalization of the classical LARS semantics: if we choose $\mathcal{R} = \mathbb{B}$ the value w.r.t. the weighted semantics corresponds to satisfaction for unweighted LARS formulas.

Proposition 7. *For every LARS formula α, it holds that*

$$[\![\alpha]\!]_\mathbb{B}(S, t) = \mathbf{t} \iff (S, t) \models \alpha.$$

We define LARS measures as a formal way of specifying quantitative measures using wLARS formulae on answer streams.

Definition 8 (LARS Measure)**.** *A* LARS measure *$\mu = \langle \Pi, \alpha, \mathcal{R} \rangle$ consists of a LARS program Π, a wLARS formula α, and a semiring $\mathcal{R}$. The weight of an answer stream $S \in \mathcal{AS}(\Pi, D, t)$ is*

$$\mu(S, D, t) := [\![\alpha]\!]_\mathcal{R}(S, t).$$

For streams $S \notin \mathcal{AS}(\Pi, D, t)$ we set $\mu(S, D, t) = e_\oplus$.

Example 3 (Traveling cont'd)**.** *We fix an instance as follows:*

- $S = \text{Hamburg}, E = \text{Amsterdam}$
- $\mathcal{C} = \{\text{Amsterdam}, \text{Hamburg}, \text{Paris}\} = \{\text{A}, \text{H}, \text{P}\}$,
- $\mathcal{T} = \{\text{Boat}, \text{Plane}, \text{Train}\} = \{\text{B}, \text{P}, \text{T}\}$, $\overline{\tau} = (\text{T}, \text{B}, \text{P})$
- $\text{Cost}(c_1, \text{B/P/T}, c_2)$ *is given by*

$c_1 \backslash c_2$	A	H	P
A	0/0/0	34/80/47	45/∞/79
H	32/90/53	0/0/0	∞/110/96
P	∞/64/89	∞/113/94	0/0/0

The cost of a specific journey can be expressed by a formula α over the tropical semiring $\mathcal{R}_\text{trop}$ as follows:

$$\alpha = \Box(\textstyle\bigvee_{c_1,c_2 \in \mathcal{C}} \bigvee_{\tau \in \mathcal{T}} \alpha'_{c_1,c_2,\tau} \vee \boxplus^\text{next}\Box\bot), \text{ where}$$

$$\alpha'_{c_1,c_2,\tau} = \text{travel}_\tau \wedge \text{in}_{c_1} \wedge \boxplus^\text{next}\Diamond\text{in}_{c_2} \wedge \text{Cost}(c_1, \tau, c_2)$$

The formula α' can only have a value unequal to $e_\oplus$, if the logical "guard" $\text{travel}_\tau \wedge \text{in}_{c_1} \wedge \boxplus^\text{next}\Diamond\text{in}_{c_2}$ is classically satisfied. This illustrates how one can make algebraic expressions dependent on the truth of atoms w.r.t. interpretations.

The LARS measure defined by $\mu = \langle \Pi, \alpha, \mathcal{R}_\text{trop} \rangle$ yields the money spent by our friend or ∞ if the journey is invalid. E.g. for $S = (T, v)$

$$T = \{0,1,2,3\}, \quad v = \{0 \mapsto \{\text{travel}_\text{T}, \text{in}_\text{H}\}, 1 \mapsto \{\text{travel}_\text{B}, \text{in}_\text{A}\}, 2 \mapsto \{\text{travel}_\text{P}, \text{in}_\text{H}\}, 3 \mapsto \emptyset\},$$

we have $\mu(S, D_{(\text{T},\text{B},\text{P})}, 0) = \infty$ since S does not satisfy Π. Here $D_{\overline{\tau}} = (T, v_{\overline{\tau}})$ is the data stream encoding the sequence $\overline{\tau}$, i.e. $v_{\overline{\tau}}(t) = \{\text{in}_{\overline{\tau}_t}\}$. By changing $v(3)$ to $\{\text{in}_\text{A}\}$, we obtain

$$\begin{aligned}
\mu(S, D_{\overline{\tau}}, 0) &= [\![\Box(\textstyle\bigvee_{c_1,c_2 \in \mathcal{C}} \bigvee_{\tau \in \mathcal{T}} \alpha'_{c_1,c_2,\tau} \vee \boxplus^\text{next}\Box\bot)]\!]_\mathcal{R}(S, 0)\\
&= \textstyle\sum_{t=0}^{2} \min\{[\![\alpha'_{c_1,c_2,\tau}]\!]_\mathcal{R}(S, 0) \mid c_1, c_2 \in \mathcal{C}, \tau \in \mathcal{T}\}\\
&= 53 + 34 + 90 = 177
\end{aligned}$$

Other quantitative questions we could answer are

- q_1: *"How often was our friend in Paris?"*
- q_2: *"What percentage of time was our friend in Paris?"*
- q_3: *"How many cities were visited?"*

using the measures $\mu_1 = \langle \Pi, \Diamond \mathrm{in_P}, \mathbb{N} \rangle$, $\mu_2 = \langle \Pi, \Diamond 1 \rightarrow_{\otimes} \Diamond \mathrm{in_P}, \mathbb{R} \rangle$ *and* $\mu_3 = \langle \Pi, \neg\neg\Diamond \mathrm{in_P} \vee \neg\neg\Diamond \mathrm{in_H} \vee \neg\neg\Diamond \mathrm{in_A}, \mathbb{N} \rangle$ *respectively. The formula* $\Diamond \mathrm{in_P}$ *adds* 1 *for every time point at which* $\mathrm{in_P}$ *holds, while* $\Diamond 1$ *similarly counts the number of time points. Their connection* $\Diamond 1 \rightarrow_{\otimes} \Diamond \mathrm{in_P}$ *in the* μ_2 *thus divides the number of time points in Paris by the total number of time points. The double negations in* μ_3 *are needed to get from the number of times a city was visited a value in* $\{0, 1\}$ *corresponding to whether the city* c *was visited (i.e.,* $[\![\neg\neg\Diamond \mathrm{in_c}]\!]_{\mathbb{N}} = 1$*) or not (i.e.,* $[\![\neg\neg\Diamond \mathrm{in_c}]\!]_{\mathbb{N}} = 0$*).*

We are not only interested in measures on answer streams but also aggregates of such measures.

Definition 9 (LARS Measure for Data Streams)**.** *We extend the definition of LARS measures to a data stream* D *and timepoint* t *via*

$$\mu(D, t) := \bigoplus \{\mu(S, D, t) \mid S \in \mathcal{AS}(\Pi, D, t)\}.$$

We can see LARS measures for data streams as weighted model counting, where each model S has weight $[\![\alpha]\!]_{\mathcal{R}}(S, t)$. For $\alpha = 1$ and $\mathcal{R} = \mathbb{N}$ we get $\mu(D, t) = |\mathcal{AS}(\Pi, D, t)|$.

Example 4 (Traveling cont'd)**.** *Using the LARS measure* μ *in Example 3 on data streams, we can obtain the minimum amount of money spent on any valid journey. For the data stream* $D_{(\mathrm{T,B,P})}$*,*

$$\mu(D_{(\mathrm{T,B,P})}, 0) = \min\{\mu(S, D_{(\mathrm{T,B,P})}, 0) \mid S \in \mathcal{AS}(\Pi, D, 0)\}.$$

Therefore, we obtain the minimum cost of all possible journeys using $\overline{\tau}$*, which is* 177 *for* $\overline{\tau} = (\mathrm{T}, \mathrm{B}, \mathrm{P})$.

3.1 Preferential Reasoning with wLARS

Given a strict partial order $>$ defined on the set R of elements of the semiring $\mathcal{R}$, we can use any quantitative query defined by some LARS measure μ as the objective function w.r.t. which streams are ranked in preference based reasoning.

Definition 10 (Preferred Streams)**.** *We say an answer stream* $S \in \mathcal{AS}(\Pi, D, t)$ *is a* preferred stream *w.r.t. a LARS measure* $\mu = \langle \Pi, \alpha, \mathcal{R} \rangle$*, data stream* D*, timepoint* t*, and strict order* $>$ *on* R*, if*

$$\nexists S' \in \mathcal{AS}(\Pi, D, t) : \mu(S', D, t) > \mu(S, D, t).$$

This allows for a natural generalization of common extensions of ASP with preferences to the specific characteristics of LARS: *asprin* [7] for instance allows for expressions like $w : \alpha$ where formulae are labeled and if the formula α is satisfied by an answer set $\mathcal{I}$, the quality of $\mathcal{I}$ is reduced or enhanced by w. This is sufficient for ASP; however in the context of LARS, new capabilities are needed: we may want a formula α to be satisfied at as many timepoints as possible, which cannot be expressed using the *asprin* formalism. The expressions $w : \Box\alpha$ or $w : \Diamond\alpha$ only approximate the desired preference. Alternatively adding $w : @_t\alpha$ for each time point is not possible, since the number of timepoints may vary. In wLARS we can simply specify the weighted formula $\Diamond(\neg\neg\alpha) \wedge w$, which evaluates to $k \cdot w$ where k is the number of timepoints at which α holds.

Furthermore, preferences w.r.t. multiple objectives are captured in our approach natively:

Proposition 11. *Assume that we have two LARS measures* $\mu_i = \langle \Pi, \alpha_i, \mathcal{R}_i \rangle, i = 1, 2$ *to optimize. Then the LARS measure*

$$\mu_1 \times \mu_2 := \langle \Pi, (e_{\otimes_1}, e_{\oplus_2}) \wedge \alpha_1 \vee (e_{\oplus_1}, e_{\otimes_2}) \wedge \alpha_2, \mathcal{R}_1 \times \mathcal{R}_2 \rangle$$

defines the corresponding multi-objective optimization. Here

$$\mathcal{R}_1 \times \mathcal{R}_2 := (R_1 \times R_2, \oplus, \otimes, (e_{\oplus_1}, e_{\oplus_2}), (e_{\otimes_1}, e_{\otimes_2})),$$
$$(x_1, x_2) \odot (y_1, y_2) := (x_1 \odot_1 y_1, x_2 \odot_2 y_2), \odot \in \{\oplus, \otimes\}.$$

Example 5 (Preferred Traveling)**.** *Given the knowledge that our friend likes Paris in addition to traveling cheaply, we assume that he prefers trips that visit Paris as often as possible, while being as cheap as possible. We therefore are interested in answer streams that are Pareto optimal w.r.t.* $\mu_1 = \langle \Pi, \Diamond \mathrm{in_P}, \mathbb{N} \rangle$ *and the LARS measure* μ_2 *modeling the cost of a journey from Example 3. These are the preferred streams w.r.t.* $\mu_1 \times \mu_2$ *and* $(D_{(T,B,P)}, 0)$*. Both answer streams of* Π*, i.e.* $S = (T, v)$ *from Example 3 and* $S' = (T, v')$ *where*

$$v' = \{0 \mapsto \{\mathrm{travel_T}, \mathrm{in_H}\}, 1 \mapsto \{\mathrm{travel_B}, \mathrm{in_A}\},$$
$$2 \mapsto \{\mathrm{travel_P}, \mathrm{in_P}\}, 3 \mapsto \{\mathrm{in_A}\}\},$$

are preferred since $\mu_1 \times \mu_2(S', D_{(T,B,P)}, 0) = (1, 187)$*, which is incomparable to* $\mu_1 \times \mu_2(S, D_{(T,B,P)}, 0) = (0, 177)$.

4 Lifting other Formalisms

Besides defining native measures for streams, we can use our framework to express extensions of ASP that implicitly use measures. This is possible as ordinary ASP can be seen as a LARS program for a single timepoint, i.e. $T = \{0\}$. Based on this, the extensions are lifted to LARS, i.e., one can use them as known from ASP but in the context of stream reasoning with LARS. Novel quantitative extensions of ASP via semirings can be defined and lifted analogously.

4.1 ProbLog [11] and aProbLog [21]

ProbLog [11] is a prominent rule language for probabilistic query answering, which has been generalized to Algebraic Prolog (aProbLog) [21] to allow for queries over general commutative semirings. We consider only aProbLog in detail, since the lifting of ProbLog follows from the subsumption of ProbLog in aProbLog [21].

In aProbLog over the semiring $\mathcal{R} = (R, \oplus, \otimes, e_{\oplus}, e_{\otimes})$ extensional atoms $f \in F$ and their negation $\neg f$ are labeled with semiring elements $\alpha(f), \alpha(\neg f) \in R$ respectively. The background knowledge BK consists of Horn clauses that are unlabeled. The label of a query q is then given by

$$\mathbf{A}(q) = \bigoplus_{\mathcal{I} \models_{\mathrm{BK}} q} \bigotimes_{f \in \mathcal{I}} \alpha(f) \bigotimes_{f \in F \setminus \mathcal{I}} \alpha(\neg f). \tag{6}$$

Where $\mathcal{I} \models_{\mathrm{BK}} q$ if $\mathcal{I} \subseteq F$ and $\mathcal{I} \cup \mathrm{BK} \models q$. We can model such labeled queries in our framework.

Proposition 12. *Given a set* F *of extensional facts labeled with elements of a semiring* $\mathcal{R}$*, a background knowledge* BK*, and a query* q*, a wLARS formula* α_q *over* $\mathcal{R}$ *and a LARS program* $\Pi_{F,\mathrm{BK}}$ *depending on* F *and* BK *are constructible s.t. the LARS measure* $\mu = \langle \Pi_{F,\mathrm{BK}}, \alpha_q, \mathcal{R} \rangle$ *fulfills* $\mu(D_\emptyset, 0) = A(q)$*, where* $D_\emptyset$ *is the data stream* $(\{0\}, \{0 \mapsto \emptyset\})$.

Proof (sketch). The program $\Pi_{F,\mathrm{BK}}$ contains for each definite clause $a \vee \neg b_1 \vee \cdots \vee \neg b_n \in \mathrm{BK}$ the rule $a \leftarrow b_1, \ldots, b_n$. Furthermore, for each extensional atom $f \in F$ it contains the choice modelled by the rules $f \leftarrow \neg f'$ and $f' \leftarrow \neg f$. We set $\alpha_q = (\neg\neg q) \wedge \bigwedge_{f \in F} (f \wedge \alpha(f) \vee \neg f \wedge \alpha(\neg f))$. Here $\neg\neg q$ sorts out all answer streams that do not satisfy q. The rest of the formula models the value of the given interpretation based on which extensional atoms are satisfied. The sum over all interpretations in (6) is the sum over all interpretation streams S of $D_\emptyset$ in our framework. □

4.2 LP$^{\text{MLN}}$ [28]

LP$^{\text{MLN}}$ [28] assigns interpretations of logic programs a probability. One considers a set of weighted rules

$$\Pi = \{w_1 : R_1, \ldots, w_n : R_n\}$$

where $w_i \in \mathbb{R}$ or $w_i = x$ and R_i is a disjunctive rule. The weight scheme of LP$^{\text{MLN}}$ is the same as that of Markov Logic, i.e. the probability of a rule being correct is log-linear in its weight. If $w_i = x$ the weight is interpreted to be arbitrarily large.

In the context of LP$^{\text{MLN}}$, one considers a different notion of stable models: the reduct $\Pi_{\mathcal{I}}$ of program Π w.r.t. an interpretation $\mathcal{I}$ is defined as $\Pi_{\mathcal{I}} = \{R \mid w : R \in \Pi, \mathcal{I} \models R\}$. The stable models of Π are then the interpretations $\mathcal{I}$ that are (classical) stable models of $\Pi_{\mathcal{I}}$. Furthermore, the weight of a stable model $\mathcal{I}$ is given by

$$W_\Pi(\mathcal{I}) = \textstyle\prod_{w:R\in\Pi_{\mathcal{I}}} \exp(w)\,.$$

(Notably, $\Pi_{\mathcal{I}}$ is easy to express in classical stable models.) Using this definition of stable models also for LARS, one obtains:

Proposition 13. *Given a set of weighted rules Π we can specify a wLARS formula α over the semiring $\mathcal{R} = (\mathbb{R}[x], +, \cdot, 0, 1)$ of the polynomials with coefficients in the reals s.t. for the LARS measure $\mu = \langle \tau(\Pi), \alpha, \mathcal{R}\rangle$, it holds that*

$$\mu(S_{\mathcal{I}}, D_\emptyset, 0) = W_\Pi(\mathcal{I}),$$

where $\tau(\Pi)$ is the logical part of the P-log program Π, $S_{\mathcal{I}} = (\{0\}, \{0 \mapsto \mathcal{I}\})$ and $D_\emptyset = (\{0\}, \{0 \mapsto \emptyset\})$.

Proof (sketch). We choose $\alpha = \bigwedge_{w:R\in\Pi}(\neg\neg R \wedge \exp(w)) \vee \neg R$. The subformula $\neg\neg R$ effects that the value $\exp(w)$ is only included in the product if R is satisfied, otherwise $\neg R$ and the whole term for the given rule have value 1 and thus no effect on the product. □

In LP$^{\text{MLN}}$ one obtains a probability measure P_Π over the set of stable models from the weight function W_Π by normalizing and taking the limit $x \to \infty$. We can simply replace W_Π by μ and normalization by division by the aggregate $\mu(D_\emptyset, 0)$:

$$P_\Pi(\mathcal{I}) = \lim_{x\to\infty} \mu(S_{\mathcal{I}}, D_\emptyset, 0)/\mu(D_\emptyset, 0).$$

4.3 Further ASP Extensions

Some further ASP extensions expressible using wLARS are

P-log [2]: P-log is a well-known probabilistic extension of ASP, where one can assign propositional variables different probabilities depending on conjunctions of literals that are satisfied. Furthermore, one intuitively can specify that a predicate takes a value following a certain probability distribution (e.g. only one of $\mathrm{dice}(1), \ldots, \mathrm{dice}(6)$ each with probability $1/6$). We can also express the measures definable in this ASP extension (omitted due to space restrictions).

Weak Constraints [8]: Buccafurri et al. considered preferred answer sets according to an order on the answer sets defined by constraints of different importance. A strong connection between LP$^{\text{MLN}}$ and weak constraints was shown in [22]; as the measures in LP$^{\text{MLN}}$ are expressible in wLARS, to no surprise we can specify for any program with weak constraints a LARS measure and an order under which the same answer sets are preferred.

***asprin* [7]:** the *asprin* framework allows for the optimization of sets $\{w_i : \alpha_i \mid i = 1, \ldots, n\}$ of labeled formulae w.r.t. arbitrary aggregates of the labels w_i of the satisfied α_i; it subsumes most approaches for preferential reasoning in ASP [7]. We are bound to aggregates expressible by semiring operations; those mentioned in [7], viz. union of sets, count of elements and sum of integers, are all captured by weighted formulae over the semirings $\mathcal{P}(A), \mathbb{N}$, and $\mathbb{Z}$ respectively. Using semirings also has benefits: it allows for complex expressions beyond an aggregation. In *asprin*, one can specify orders and define composite orders for multiple objectives. While wLARS has access to arbitrary definable orders (any strict order is admissible), we do not provide a language for specifying orders.

5 Relation to Weighted MSO and Automata

Weighted (finite state) automata generalize finite state automata, just like the weighted semantics for formulae generalizes the boolean semantics. Instead of just accepting words like finite state automata, weighted automata associate a weight over a semiring to words. Intuitively this defines a function from words to values in a semiring.

Definition 14 (Weighted Automaton)**.** *A* weighted automaton $\mathcal{A}$ *over a finite alphabet A is a tuple $\langle Q, \lambda, \delta, \gamma\rangle$, where (i) Q is a finite set of states, (ii) $\lambda, \gamma : Q \to R$ map states to weights (elements) and (iii) $\delta : A^* \to R^{Q\times Q}$ a monoid homomorphism, for some semiring $\mathcal{R} = (R, \oplus, \otimes, e_\oplus, e_\otimes)$. Its* behavior *is defined as*

$$\|\mathcal{A}\| : A^* \to R, w \mapsto \textstyle\bigoplus_{q,q'\in Q} \lambda(q) \otimes \delta(w)_{q,q'} \otimes \gamma(q').$$

Regarding the expressive power of LARS measures, we identify here a fragment that can express the same functions as weighted automata. To this aim, we do not consider general window functions, but only those definable in monadic second-order logic (MSO).

Definition 15. *A window function w is MSO definable, if there exist MSO formulae $\phi_w(P, P', x), \psi_w(x_s, x_e, x'_s, x'_e, x)$ s.t. for every stream $S = (T, v)$, $w((T, v), t) = (T', v')$ and $t_s = \min T, t_e = \max T, t'_s = \min T', t'_e = \max T'$:*

$$\phi_w(P[v], \sigma(P'), t) \text{ holds iff } \sigma(P') = P[v']$$
$$\psi_w(t_s, t_e, \sigma(x'_s), \sigma(x'_e), t) \text{ holds iff } \sigma(x'_s) = t'_s, \sigma(x'_e) = t'_e$$

where for a valuation v, $P[v]$ is the vector of monadic predicates $P_a[v]$ for $a \in \mathcal{A}$, s.t. $\forall t' \in T' : P_a[v](t') \iff a \in v(t')$.

For example, the window functions used in the examples above are all MSO expressible.

Furthermore, we disallow $\to_\oplus$ and $\to_\otimes$ and restrict negation $\neg$ to only occur in front of atoms $p \in \mathcal{A}$. Since our proofs rely on the commutativity of multiplication we only consider commutative semirings in this section.

Example 6 (Traveling cont.)**.** *We can also model the minimum traveling cost function using a weighted automaton. We choose the alphabet $A = \mathcal{T}$, and use as states Q the cities $\mathcal{C}$. We know the costs $\delta(\tau)_{c_1,c_2} = \mathrm{Cost}(c_1, \tau, c_2)$ of going by τ from c_1 to c_2, which we set to ∞ when $c_1 = c_2$. Then the weighted automaton*

$$\mathcal{A} = \langle Q, \lambda, \delta, \gamma\rangle, \text{ where}$$
$$\lambda(q) = \begin{cases} 0 & \text{if } q = \mathrm{H}, \\ \infty & \text{otherwise.} \end{cases}, \gamma(q) = \begin{cases} 0 & \text{if } q = \mathrm{A}, \\ \infty & \text{otherwise.} \end{cases}$$

models the minimal traveling cost function. To obtain the minimal cost, we use the semiring $\mathcal{R}_{\mathrm{trop}}$ again: then transportation costs are added and the aggregation step with $\bigoplus$ chooses the minimum cost of a journey from Hamburg *to* Amsterdam *with transports $\overline{\tau}$:*

$$\|\mathcal{A}\|(\overline{\tau}) = \textstyle\bigoplus_{q,q'\in Q} \lambda(q) \otimes \delta((\mathrm{T}, \mathrm{B}, \mathrm{P}))_{q,q'} \otimes \gamma(q')$$

$$= \min_{q,q' \in Q} \delta(\mathrm{T})_{\mathrm{H},q} + \delta(\mathrm{B})_{q,q'} + \delta(\mathrm{P})_{q',\mathrm{A}}$$
$$= \delta(\mathrm{T})_{\mathrm{H},\mathrm{A}} + \delta(\mathrm{B})_{\mathrm{A},\mathrm{H}} + \delta(\mathrm{P})_{\mathrm{H},\mathrm{A}} = 177.$$

We have seen that one can model the minimum travel cost calculation both with LARS measures and with weighted automata. In the following we show that generally, the expressivity of weighted automata is equivalent to that of a restricted class of LARS measures.

Theorem 16 (Reduction of Weighted Automata to LARS Measures)**.** *Given a weighted automaton $\mathcal{A}$ over a finite alphabet A and semiring $\mathcal{R}$, there exists a LARS measure $\mu = \langle \Pi, \alpha, \mathcal{R} \rangle$, s.t.*

$$\forall w \in A^* : \|\mathcal{A}\|(w) = \mu \circ \tau(w),$$

where $\tau(w)$ maps words w to pairs of data streams and time points.

The above theorem is shown by constructing a program that has as answer streams all possible paths through the weighted automaton and a wLARS formula, whose semantics for a given answer stream is the weight of the path. By summing over all answer streams, one obtains then the behavior of the automaton on the given word.

We proceed to establish the other direction of encodability, by expressing LARS measures via *restricted* weighted MSO formulae, which were shown to be equivalent to weighted automata [14]. In such formulae, universal quantifiers are restricted to first-order variables. Furthermore, the semantics of the quantified formula may only take finitely many values [14]. We define *restricted* LARS measures analogously as LARS measures $\mu = \langle \Pi, \alpha, \mathcal{R} \rangle$, where $\Box\beta$ can only be a subformula of α if β takes finitely many values.

Theorem 17 (Reduction of LARS Measures to Weighted MSO)**.** *Let $\langle \Pi, \alpha, \mathcal{R} \rangle$ be a restricted LARS measure over a semiring $\mathcal{R}$. Then a restricted weighted MSO formula Ψ over $\mathcal{R}$ is constructible, s.t.*

$$\forall D, t : \mu(D, t) = [\![\Psi]\!]_{\mathcal{R}}(\sigma(D, t)),$$

where $\sigma(D,t) = (\sigma_1(D,t), \sigma_2(D,t))$ s.t. σ_1 is the word that the weighted MSO formula is interpreted over and σ_2 corresponds to the assignments of free variables in Ψ.

Proof (sketch). For the proof, we use the weighted MSO formula $\Psi = \exists \mathbf{A}.T(\Phi_\Pi)(\mathbf{A}) \wedge f(\alpha)$. Intuitively the claim holds because:
$-f(\alpha)$ is a faithful translation of α to weighted MSO,
$-\exists \mathbf{A}$ is the sum over all interpretation streams S of D,
$-\Phi_\Pi(\mathbf{A})$ is a MSO formula, which is satisfied by $\mathbf{A}$ iff $\mathbf{A}$ corresponds to an answer stream. It can be obtained by extending the result in [3] that given a LARS program Π one can construct a MSO formula that is satisfiable iff Π has an answer stream.
$-T(\ .\ )$ is a translation to weighted MSO that preserves the boolean semantics, due to [14]. □

Droste and Gastin [14] showed that restricted weighted MSO and weighted automata are equally expressive; hence, it follows that also restricted LARS measures and weighted automata are equally expressive. Notably, restrictedness is needed: the value $2^{|T|^2}$ of the formula $\alpha = \Box\Box 2$ over $\mathbb{N}$ is inexpressible by weighted automata [14].

6 Computation and Complexity

We first consider evaluating wLARS formulae, by encoding them as weighted automata. This seems promising, as weighted automata have polynomial data complexity when counting the arithmetic operations. Unfortunately, any encoding is non-elementary in general.

Theorem 18. *For every constant k, there is no translation of restricted wLARS formulae of size n to weighted automata s.t. the size of the automaton can be bounded by a function of the form $\exp(k)$, where $\exp(0) = n, \exp(k+1) = 2^{\exp(k)}$.*

This follows from Meyer and Stockmeyer's result [27] that translating MSO to finite automata has non-elementary complexity. However, if we consider windows common in practice like time-based windows and restrict rules to fragments like plain LARS [3], we can obtain a translation from restricted LARS measures to weighted automata that can be bounded double exponentially.

Alternatively, we can evaluate wLARS formulae using a Turing machine. For this we consider as in [3] window functions that can be evaluated in polynomial time. We formally define the evaluation problem as follows:

- **EVAL-wLARS**: given a wLARS formula α over a semiring $\mathcal{R}$, a stream S, a timepoint t in S, compute the value of $[\![\alpha]\!]_{\mathcal{R}}(S,t)$.

The complexity of this problem may grow arbitrarily depending on the semiring and can be undecidable in general (unless $\mathcal{R}$ is explicitly represented, functions for the arithmetic operations are provided in the input). Beck et al. [3] showed that evaluating LARS formulae is PSPACE-complete; thus PSPACE-hardness is a lower bound for any non-trivial semiring. Semirings s.t. EVAL-wLARS is FPSPACE(poly)-complete can be found, where FPSPACE(poly) are the functions with polynomial output size computable by Turing machines in polynomial space. In the next subsection, we present restrictions that allow for efficient evaluation.

6.1 Efficiently Evaluable Fragments

The inefficiency of the evaluation of a weighted formula α under some stream is due to the unrestricted use of quantifiers, i.e. $\Box, \Diamond$ in α. If we bound the nesting depth of quantifiers, denoted $\mathrm{qdepth}(\alpha)$, by some constant k, we obtain that under reasonable restrictions on the semiring, the time needed for evaluation of a formula given some stream is polynomial in the size of the formula and the stream.

Definition 19 (Efficiently Encodable Semiring)**.** *Let $\mathcal{R} = (R, \oplus, \otimes, e_\oplus, e_\otimes)$ be a semiring and $e : R \to \mathbb{N}$ be an injective function s.t. both e and e^{-1} are computable in polynomial time. Then $\mathcal{R}$ is* efficiently encoded *by e iff there exists a constant C s.t. for any $r, r' \in R$ and $\odot \in \{\oplus, \otimes\}$ it holds that*

$$\log_2(e(r \odot r')) \leq \log_2(e(r)) + \log_2(e(r)) + C, \quad (7)$$
$$\log_2(\max(e(-r), e(r^{-1})) \leq \log_2(e(r)) + C. \quad (8)$$

Condition (8) *only applies if $\oplus$ or $\otimes$ are invertible.*

Furthermore, given $e(r), e(r')$ it must be possible to calculate $e(r \odot r'), e(-r)$ and $e(r^{-1})$ in polynomial time in the length of the binary encoding of $e(r), e(r')$.

The restriction on the encoding function is mild in practice, since most practically used semirings, like $\mathbb{Q}, \mathbb{N}, \mathbb{B}, \mathcal{P}(A), \mathcal{R}_{\mathrm{trop}}$ satisfy it.

Theorem 20 (Complexity of Evaluation I)**.** *Let $\mathcal{R}$ be a (fixed) semiring that is efficiently encoded by e, and $k \in \mathbb{N}$ (fixed). Then for any wLARS formula α over $\mathcal{R}$ s.t. $\mathrm{qdepth}(\alpha) \leq k$, we can calculate $[\![\alpha]\!]_{\mathcal{R}}(S,t)$ in polynomial time in the size of α and S.*

Proof (sketch). The proof is by structural induction on the formula α, with the induction hypothesis that the time $t(\alpha)$ needed is in

$\mathcal{O}(N^{n\cdot(k+1)})$, where N is the size of the input, $n \in \mathbb{N}$ is a constant that is not depending on the input and $k = \mathrm{qdepth}(\alpha)$. Furthermore, the size $s(\alpha)$ of the representation of the obtained value, i.e. $\log_2(e([\![\alpha]\!]_{\mathcal{R}}(S,t)))$, is in $\mathcal{O}(N \cdot N^k)$. □

This result is as expected from LARS, where under similar restrictions one can perform model checking in polynomial time [3]. However the FPSPACE(poly)-membership does not seem to generalize to arbitrary formulae. If we consider the formula $\Box^k 2 = \Box\Box^{k-1}2$, which is $2^{|T|^k}$ when evaluated over $\mathbb{N}$, we see that the binary representation of the result has exponential size. Similarly, if we consider the evaluation of LARS measures $\mu(D,t)$ over data streams D, we have to sum up exponentially many values $\mu(S,D,t)$. Thus, efficient encodability is not sufficient for FPSPACE(poly)-membership. Hence, we consider encodings s.t. the summation behaves only slightly worse than the sum over the natural numbers:

Definition 21 (Natural Addition/Multiplication)**.** *Let $\mathcal{R}$ efficiently encoded by e. If some $C \in \mathbb{N}$ exists s.t. for all $r, r' \in R$*

$$e(r \odot r') \leq e(r) + e(r') + C,$$

then e supports natural addition *(*multiplication*) when $\odot = \oplus$ ($\otimes$).*

Having natural addition is a stronger restriction: while $\mathbb{N}$, $\mathbb{B}$, $\mathcal{P}(A)$, $\mathcal{R}_{\mathrm{trop}}$ satisfy it, it is an open question whether an encoding that supports natural addition exists for $\mathbb{Q}$.

Theorem 22 (Complexity of Evaluation II)**.** *Let $\mathcal{R}$ be a (fixed) semiring that is efficiently encoded by e which supports natural addition, and $k \in \mathbb{N}$ (fixed). Then for any LARS measure $\mu = \langle \Pi, \alpha, \mathcal{R}\rangle$ s.t. $\mathrm{qdepth}(\alpha) \leq k$, stream S, data stream D and timepoint t, computing $\mu(D,t)$ and $\mu(S,D,t)$ are both in FPSPACE(poly).*

Proof. Deciding $S \in AS(\Pi, D, t)$ is feasible in PSPACE [3], and Π has at most $2^{|\mathcal{A}^I||T|} \leq 2^{N^2}$ answer streams S for D, where N is the size of the input. For each of them $e(\mu(S,D,t)) \in \mathcal{O}(2^{N^{k+1}})$, thus $e(\mu(D,t)) \in \mathcal{O}(2^{N^{k+3}})$ as e supports natural addition; thus the binary representation of $e(\mu(S,D,t)$ resp. $e(\mu(D,t))$ has size polynomial in N. Furthermore, iterating over all S to compute $e(\mu(D,t))$ is doable in polynomial space. □

The problem is FPSPACE(poly)-complete in general. For restricted classes of programs Π or weighted formulae, we may end up with problems that are complete for classes contained in FPSPACE(poly). For example, when $\alpha = 1$ and $\mathcal{R} = \mathbb{N}$ the evaluation of $\mu(D,t)$ amounts to model counting of programs, which is #P-complete for normal ASP programs and #coNP-complete for disjunctive ASP programs (as follows from results for #SAT resp. #CIRC [16]).

6.2 Preferential Reasoning

Given some LARS measure $\mu = \langle \Pi, \alpha, \mathcal{R}\rangle$ with a strict partial order $>$ on R, we consider the following two problems:

- **Preference Checking (PC)**: Given a stream S, LARS measure μ, data stream D and time t, check whether S is preferred.
- **Brave Preferential Reasoning (BPR)**: Given a LARS measure μ, data stream D, time t and an atomic formulae a check whether there exists a preferred stream S such that $(S,t) \models a$.

We obtain the following results:

Theorem 23. *Let $\mathcal{R}$ be a (fixed) semiring efficiently encoded by e, $>$ be a strict order on R s.t. $r_1 > r_2$ is decidable in polynomial time, and $k \in \mathbb{N}$ (fixed). Then for every LARS measure $\mu = \langle \Pi, \alpha, \mathcal{R}\rangle$ s.t. $\mathrm{qdepth}(\alpha) \leq k$ (i) PC and BPR are PSPACE-complete. If moreover every $\alpha \leftarrow \beta \in \Pi$ fulfills $\max(\mathrm{qdepth}(\alpha), \mathrm{qdepth}(\beta)) \leq k$, then (ii) PC is Π_2^p-complete and (iii) BPR is Σ_3^p-complete.*

Proof (sketch). (i) PSPACE-hardness follows from the fact that already checking whether a stream is an answer stream is PSPACE-hard for general LARS programs [3]. For PSPACE membership we can use that we can iterate over all answer streams in PSPACE.

(ii) Membership in Π_2^p can be shown by guessing S' and checking whether $S' \in \mathcal{AS}(\Pi, D, t)$ and $\mu(S',t) > \mu(S,t)$. Hardness for Π_2^p is shown by reduction from unsatisfiability checking of a disjunctive program using a similar construction as in the proof of (i).

(iii) For membership in Σ_3^p, we can guess an interpretation stream S for D which contains a and use a Π_2^p oracle for (PC). Hardness for Σ_3^p is shown by a reduction from checking whether a QBF formula of the form $\Phi = \exists X \forall Y \exists Z : \phi(X,Y,Z)$ evaluates to true. □

BPR has a lower complexity, namely Δ_3^p, when one considers only pairs $(\mathcal{R}, >)$ such that efficient binary search is possible, as for BPR in the presence of weak constraints [23].

7 Discussion and Conclusion

Our results about the encodability of weighted automata in a fragment of wLARS and vice versa demonstrate not only the expressive power of wLARS but also a convenient way of specifying the calculation of a weighted automaton. In contrast to the transition function δ of a weighted automaton, which needs to encode both which words are accepted and their corresponding weights, we have a clear separation of the weighted and unweighted parts of the calculation. More precisely, obtaining a weight has three steps:

1. specify which interpretations to consider using an unweighted logical expression with answer set semantics, by LARS program Π,
2. assign a value $[\![\alpha]\!]_{\mathcal{R}}(S,t)$ to each of the solutions (S,t), using a weighted logical expression α, and
3. aggregate the solution values $\bigoplus_{S \in \mathcal{AS}(\Pi,D,t)} [\![\alpha]\!]_{\mathcal{R}}(S,t)$.

Regarding the usefulness of wLARS in practical applications, we have seen in our travel scenario that we can answer different meaningful questions using wLARS formulae. This also showed the strengths of the framework: while quite diverse calculations may be done, the flexible choice of the semiring allows for homogeneous syntax to specify the measure. Furthermore, due to the specification using logical connectives one can intuitively mix the parts of the formula representing an algebraic expression and those that guard which calculation is performed, based on the truth of atoms.

Related Work. Both in the context of ASP and stream reasoning there exist quantitative extensions.

• *ASP*: We have seen that well-known quantitative extensions of ASP, viz. LP$^{\mathrm{MLN}}$ [28] and (a)ProbLog [11, 21] can be expressed in our framework. The same can be done for P-log [2], ASP with weak constraints [8] and a considerable fragment of *asprin* [7]. There are however limitations to the quantitative measures we can express. Nilsson-style probabilistic languages like PrASP [25], which fits both in the context of stream reasoning and ASP, specify probabilities in an indirect manner and thus seemingly cannot be captured. General Annotated Logic Programs (GALP) [20] have multi-valued interpretations and allow for the specification of values using arbitrary monotone functions and for limited temporal reasoning (e.g. no window

functions). Therefore, only fragments of wLARS can be expressed in GALP and vice versa. Of the extensions mentioned above, some aim for generality. In particular *asprin* is a very general framework for preferential reasoning; one of LP$^{\text{MLN}}$'s highlighted features is that it is a "good middle ground language" between P-log and weak constraints [22]. The languages used for the specification of measures in these approaches are however more restricted than ours: *asprin* lacks complex expressions and LP$^{\text{MLN}}$ only allows for measures over the real numbers. There are other logic programming extensions that make use of semirings, like Weighted Datalog [4] and Semiring-based Constraint Logic Programming [5]; however they consider non-boolean semantics for programs rather than measures on answer sets of programs.

• *Stream Reasoning*: Quantitative Regular Expressions [24] allow for arbitrary sets of operations in structured stream processing, leading to more involved algebraic expressions in more complex syntax. These measures are more expressive than the ones in our framework but also come with more involved syntax and semantics. de Leng and Heintz [10] concerned themselves with approximate stream reasoning, given probabilistic input data. The conditional probability measures can be modeled as a ratio between two LARS measures. The only quantitative stream reasoning formalisms with answer set semantics we are aware of are PrASP and GALP, mentioned above.

Future Work. Given the lifting of well-known quantitative extensions of ASP to LARS, we consider wLARS to be a promising tool to formally describe quantitative extensions. Especially in the context of preferential reasoning wLARS leads to a natural generalization. We aim to lift more formalisms and to design novel quantitative extensions using wLARS. This includes considering quantitative extensions that do not concern themselves with reasoning from the set of answer streams but also quantitative extensions that change how one can specify which atoms are true in an answer stream (e.g. weight constraints [26]). We further plan to define a weighted variant of an alternative stream reasoning framework,viz. temporal ASP due to [9] and compare it to wLARS. Last but not least, a more detailed complexity analysis of wLARS and LARS measures depending on the type of semirings and an implementation are on our agenda.

ACKNOWLEDGEMENTS

This work has been supported by FWF project W1255-N23 and FFG project 861263. We would like to thank the referees for their comments, which helped to improve the presentation of this paper.

REFERENCES

[1] Raymond Balbes and Alfred Horn, 'Injective and projective Heyting algebras', *Transactions of the American Mathematical Society*, **148**(2), 549–559, (1970).
[2] Chitta Baral, Michael Gelfond, and Nelson Rushton, 'Probabilistic reasoning with answer sets', *Theory and Practice of Logic Programming*, **9**(1), 57–144, (2009).
[3] Harald Beck, Minh Dao-Tran, and Thomas Eiter, 'LARS: A logic-based framework for analytic reasoning over streams', *Artif. Intell.*, **261**, 16–70, (2018).
[4] Stefano Bistarelli, Fabio Martinelli, and Francesco Santini, 'Weighted datalog and levels of trust', in *2008 Third International Conference on Availability, Reliability and Security*, pp. 1128–1134. IEEE, (2008).
[5] Stefano Bistarelli, Ugo Montanari, and Francesca Rossi, 'Semiring-based constraint logic programming', in *IJCAI (1)*, pp. 352–357, (1997).
[6] Stefano Bistarelli and Francesco Santini, 'A common computational framework for semiring-based argumentation systems1, 2', in *ECAI 2010: 19th European Conference on Artificial Intelligence: Proceedings*, volume 215, p. 131, (2010).
[7] Gerhard Brewka, James Delgrande, Javier Romero, and Torsten Schaub, 'asprin: Customizing answer set preferences without a headache', in *Twenty-Ninth AAAI Conference on Artificial Intelligence*, (2015).
[8] Francesco Buccafurri, Nicola Leone, and Pasquale Rullo, 'Enhancing disjunctive datalog by constraints', *IEEE Transactions on Knowledge and Data Engineering*, **12**(5), 845–860, (2000).
[9] Pedro Cabalar, Roland Kaminski, Torsten Schaub, and Anna Schuhmann, 'Temporal answer set programming on finite traces', *Theory and Practice of Logic Programming*, **18**(3-4), 406–420, (2018).
[10] Daniel de Leng and Fredrik Heintz, 'Approximate stream reasoning with metric temporal logic under uncertainty', in *Proceedings of the 33rd AAAI Conference on Artificial Intelligence, Honolulu, HI, USA*, (2019).
[11] Luc De Raedt, Angelika Kimmig, and Hannu Toivonen, 'Problog: A probabilistic prolog and its application in link discovery.', in *IJCAI*, volume 7, pp. 2462–2467. Hyderabad, (2007).
[12] Emanuele Della Valle, Stefano Ceri, Frank Van Harmelen, and Dieter Fensel, 'It's a streaming world! reasoning upon rapidly changing information', *IEEE Intelligent Systems*, **24**(6), 83–89, (2009).
[13] Daniele Dell'Aglio, Emanuele Della Valle, Frank van Harmelen, and Abraham Bernstein, 'Stream reasoning: A survey and outlook', *Data Science*, **1**(1-2), 59–83, (2017).
[14] Manfred Droste and Paul Gastin, 'Weighted automata and weighted logics', *Theoretical Computer Science*, **380**(1), 69, (2007).
[15] Manfred Droste, Werner Kuich, and Heiko Vogler, *Handbook of weighted automata*, Springer Science & Business Media, 2009.
[16] Arnaud Durand, Miki Hermann, and Phokion G. Kolaitis, 'Subtractive reductions and complete problems for counting complexity classes', *Theor. Comput. Sci.*, **340**(3), 496–513, (2005).
[17] Wolfgang Faber, Nicola Leone, and Gerald Pfeifer, 'Recursive aggregates in disjunctive logic programs: Semantics and complexity', in *European Workshop on Logics in Artificial Intelligence*, pp. 200–212. Springer, (2004).
[18] Joshua Goodman, 'Semiring parsing', *Computational Linguistics*, **25**(4), 573–605, (1999).
[19] Todd J Green, Grigoris Karvounarakis, and Val Tannen, 'Provenance semirings', in *Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pp. 31–40. ACM, (2007).
[20] Michael Kifer and VS Subrahmanian, 'Theory of generalized annotated logic programming and its applications', *The Journal of Logic Programming*, **12**(4), 335–367, (1992).
[21] Angelika Kimmig, Guy Van den Broeck, and Luc De Raedt, 'An algebraic prolog for reasoning about possible worlds', in *Twenty-Fifth AAAI Conference on Artificial Intelligence*, (2011).
[22] Joohyung Lee and Zhun Yang, 'LPMLN, weak constraints, and p-log', in *Thirty-First AAAI Conference on Artificial Intelligence*, (2017).
[23] Nicola Leone, Gerald Pfeifer, Wolfgang Faber, Thomas Eiter, Georg Gottlob, Simona Perri, and Francesco Scarcello, 'The dlv system for knowledge representation and reasoning', *ACM Transactions on Computational Logic (TOCL)*, **7**(3), 499–562, (2006).
[24] Konstantinos Mamouras, Mukund Raghothaman, Rajeev Alur, Zachary G Ives, and Sanjeev Khanna, 'StreamQRE: Modular specification and efficient evaluation of quantitative queries over streaming data', in *ACM SIGPLAN Notices*, volume 52(6), pp. 693–708. ACM, (2017).
[25] Matthias Nickles and Alessandra Mileo, 'Web stream reasoning using probabilistic answer set programming', in *International Conference on Web Reasoning and Rule Systems*, pp. 197–205. Springer, (2014).
[26] Ilkka Niemela, Patrik Simons, and Timo Soininen, 'Stable model semantics of weight constraint rules', in *International Conference on Logic Programming and Nonmonotonic Reasoning*, pp. 317–331. Springer, (1999).
[27] Larry J Stockmeyer and Albert R Meyer, 'Word problems requiring exponential time (preliminary report)', in *Proceedings of the fifth annual ACM symposium on Theory of computing*, pp. 1–9. ACM, (1973).
[28] Yi Wang and Joohyung Lee, 'Handling uncertainty in answer set programming', in *Twenty-Ninth AAAI Conference on Artificial Intelligence*, (2015).