Equivalent Stream Reasoning Programs*

Harald Beck and Minh Dao-Tran and Thomas Eiter
Institute of Information Systems, Vienna University of Technology
Favoritenstraße 9-11, A-1040 Vienna, Austria
{beck,dao,eiter}@kr.tuwien.ac.at

Abstract

The emerging research field of stream reasoning faces the challenging trade-off between expressiveness of query programs and data throughput. For optimizing programs methods are needed to tell whether two programs are equivalent. Towards providing practical reasoning techniques on streams, we consider LARS programs, which is a powerful extension of Answer Set Programming (ASP) for stream reasoning that supports windows on streams for discarding information. We define different notions of equivalence between such programs and give semantic characterizations in terms of models. We show how a practically relevant fragment can be alternatively captured using Here-and-There models, yielding an extension of equilibrium semantics of ASP to this class of programs. Finally, we characterize the computational complexity of deciding the considered equivalence relations.

Introduction

Stream reasoning is an emerging research field that aims at providing logic-oriented techniques on top of stream processing. High throughput of data is a central challenge for stream processing methods, which usually focus on low-level computations such as filtering and aggregation. Central to modern stream processing technologies are window mechanisms that limit the input to snapshots of recent data. Window operators can be utilized either to define which data is still relevant, or as a practical means to cope with the volume of data. Also declarative methods and logic-oriented languages for reasoning over data streams have been considered [Do et al., 2011; Gebser et al., 2012], in particular, ones that allow to express such windows [Barbieri et al., 2010b]. A recent expressive formalism is the logic-based LARS framework [Beck et al., 2015], which allows for generic window functions and temporal operators in formulas. On top of this, LARS provides rule-based semantics that can be seen as an extension of Answer Set Programming (ASP) for data streams.

Declarative and logic-oriented approaches to stream reasoning typically aim for more expressiveness, which makes efficient evaluation even harder to achieve. One way to mitigate this problem is to optimize queries or programs by simplifying them using equivalence preserving transformations. However, this requires support for checking when two programs are equivalent in the first place.

Various notions of query or program equivalence have been studied in the literature, e.g., in database research and for answer set programs [Lifschitz et al., 2001; Eiter and Fink, 2003; Eiter et al., 2007]. However, equivalence relations between declarative programs for stream reasoning have not been considered so far. Towards optimization of such programs, we are thus interested in techniques that allow us to tell when two programs produce the same results. Characterizing equivalence between LARS programs in purely logical terms is challenging due to a non-structural definition of the FLP-semantics [Faber, Leone, and Pfeifer, 2004] defined for them, which imposes some limitations. Yet another difficulty arises from the generic definition of window operators.

We tackle this issue with the following contributions:

- We develop practically relevant notions of equivalence for LARS programs that extend well-known equivalence relations for logic programs and introduce data equivalence for streams.
- We define a novel logic called Bi-LARS to capture the FLP-based semantics of a large fragment of LARS programs, including the practical one introduced in [Beck et al., 2015] that we call plain LARS.
- We lift model-theoretic characterizations of strong/uniform/relativized uniform equivalence from ASP to the stream setting to characterize the defined equivalence relations.
- We introduce the notion of monotone windows and show how a variant of Bi-LARS leads to an extension of the logic of Here-and-There for our setting. We thus get a link to equilibrium logic [Lifschitz et al., 2001] for a class of programs.
- Finally, we investigate the complexity of checking the considered equivalence relations. Under benign window operators, the complexity is not worse than for logic programs; only in some cases the complexity does increase.

To the best of our knowledge, no similar work on equivalence notions in stream reasoning exists to date. Our results

---

*This research has been supported by the Austrian Science Fund (FWF) projects P24090, P26471, and W1255-N23.
thus give an entry point towards optimization of expressive rule-based programs for reasoning over data streams.

**LARS**

We will gradually introduce the main concepts of LARS [Beck et al., 2015], a recent logic-based framework for analyzing and expressing reasoning over streams. Where appropriate, we give only informal descriptions. We assume that the reader is familiar with ASP [Gelfond and Lifschitz, 1991; Brewka, Eiter, and Truszczyński, 2011]; for equivalence relations between ASP programs, we refer to [Lifschitz, Pearce, and Valverde, 2001; Eiter, Fink, and Woltran, 2007].

We consider propositional (ground) LARS. Throughout, $\mathcal{A}$ denotes the set of atoms, which is partitioned into extensional (input) atoms $A^e$ and intensional (derived) atoms $A^2$.

**Definition 1 (Stream)** A stream $S = (T, v)$ consists of a timeline $T$, which is a closed interval $T \subseteq \mathbb{N}$ of integers called time points, and an evaluation function $v : \mathbb{N} \rightarrow 2^A$.

Intuitively, a stream associates with each time point a set of atoms. We call $S$ a data stream, if it contains only extensional atoms. To cope with the amount of data, one usually considers only recent atoms. Let $S = (T, v)$ and $S' = (T', v')$ be two streams such that $S' \subseteq S$, i.e., $T' \subseteq T$ and $v'(t') \subseteq v(t')$ for all $t' \in T'$. Then $S'$ is called a substream or a window of $S$.

**Definition 2 (Window function)** Any (computable) function $w$ that returns, given a stream $S = (T, v)$ and a time point $t \in T$, a substream $S'$ of $S$ is called a window function.

Important are time-based window functions, which select all atoms appearing in last $n$ time points, and tuple-based window functions, which select a fixed number of latest tuples.

**Example 1** Figure 1 depicts a stream $S = ([0,4], v)$ where $v = \{0 \rightarrow \{a\}, 1 \rightarrow \{a\}, 3 \rightarrow \{b\}, 4 \rightarrow \{a,c\}\}$, and the application of the time-based window function $w$ on $S$ that looks back two time units from time point $t = 3$. This returns the substream $S' = ([1,3], \{1 \rightarrow \{a\}, 3 \rightarrow \{b\}\})$.

**Window operators $\square$.** Window functions can be accessed in formulas by window operators. For every window function $w$, employing an expression $\square w \alpha$ has the effect that $\alpha$ is evaluated on the substream of the data stream delivered by $w$.

**Definition 3 (Formulas)** The set $\mathcal{F}$ of formulas is given as follows, where $a \in \mathcal{A}$ is an atom, $t \in \mathbb{N}$, $w$ a window function.

$$\alpha ::= a \mid \neg \alpha \mid \alpha \land \alpha \mid \alpha \rightarrow \alpha \mid \alpha \lor \alpha \mid \square \alpha \mid \square w \alpha$$

The precedence of operators $\circ, \square, \square_w$ and $\square w$ is as for $\land$; e.g., $\neg \square w \alpha \land \square_w b \rightarrow c = ((\neg \square w \alpha) \land (\square_w b)) \rightarrow c$.

**Definition 4 (Structure)** A structure is any tuple $M = (S, W, B)$ where $S$ is a stream, $W$ is a set of window functions and $B \subseteq A$ is the background data of $M$.

**Definition 5 (LARS Entailment)** Let $M = (S, W, B)$ be a structure, $S = (T, v)$ and $t \in T$. The LARS-entailment relation $\models$ between $(M, t)$ and formulas is defined as follows.

Let $a \in A$ be an atom, and let $\alpha, \beta \in \mathcal{F}$ be formulas. Then,

$$M, t \models a \iff a \in v(t) \lor a \in B,$$

$$M, t \models \neg \alpha \iff M, t \not\models \alpha,$$

$$M, t \models \alpha \land \beta \iff M, t \models \alpha \land M, t \models \beta,$$

$$M, t \models \alpha \rightarrow \beta \iff M, t \not\models \alpha \lor M, t \models \beta,$$

$$M, t \models \square \alpha \iff M, t' \models \alpha \text{ for some } t' \in T,$$

$$M, t \models \square_w \alpha \iff M, t' \models \alpha \text{ for all } t' \in T,$$

$$M, t \models \square w \alpha \iff M', t \models \alpha, \text{ where } M' = (S', W, B) \text{ such that } S' = w(S, t).$$

Throughout, we assume that $A$ contains two special atoms $\top, \bot$ that are true/false in every structure.

In the sequel, $\mathbb{E}^n$ stands for a time-based window operator that takes the last $n$ time points and all data there.

**Example 2** Let $S = ([0,3], \{0 \rightarrow \{a\}, 1 \rightarrow \{a\}, 3 \rightarrow \{b, x\}\})$ be a stream and let $\varphi = \square^2 \alpha \land \neg \gamma \rightarrow x$ be a LARS formula. Consider the structure $M = (S, \{\mathbb{E}^2\}, \emptyset)$. We have that $M, 3 \not\models \square^2 \alpha \land \neg \gamma$. Indeed, since $y \notin v(3)$, it follows that $M, 3 \not\models \neg \gamma$. Furthermore, $M, 3 \models \square^2 \alpha$ since $M', 1 \models \alpha$, where $M'$ is obtained by replacing $S$ with $S' = ([1,3], \{1 \rightarrow \{a\}, 3 \rightarrow \{b, x\}\})$, i.e., the result of applying the time-based window of size 2 on $S$ at point 3. Furthermore, $M, 3 \models x$ as $x \in v(3)$; thus, $M, 3 \models \varphi$ holds.

We now turn to LARS programs that build on formulas.

**LARS Programs**

**Syntax.** LARS programs [Beck et al., 2015] are sets of rules

$$\alpha \leftarrow \beta_1, \ldots, \beta_n, \not\beta_{m+1}, \ldots, \not\beta_n,$$

where $\alpha, \beta_1, \ldots, \beta_n \in \mathcal{F}$ are formulas, and $\alpha$ contains only intensional atoms; $B(r) = \{\beta_i \mid 1 \leq i \leq n\}$ is the body of $r$.

For instance, rule $x \leftarrow \square^2 \alpha, \not\gamma$ amounts to $\varphi$ of Ex. 2.

**Semantics.** Let $D = (T, v_D)$ be a data stream. We call a stream $I = (T, v)$ a data stream for $D$, if it coincides with $D$ on $A^2$, i.e., for every $a \in A^2$ and $t \in I$, $a \in v(t)$ only if $a \in v_D(t)$; the structure $M = (I, W, B)$ is then an interpretation for $D$. Throughout, we assume $W$ (implicit by the considered programs) and $B$ are fixed, and are thus also omitted. For a rule $r$ of the form (1), we define

$$\bar{\beta}(r) = \beta_1 \land \cdots \land \beta_n \land \neg \beta_{m+1} \cdots \land \neg \beta_n.$$

Let $t \in T$. We say $M$ satisfies rule $r$ at $t$, denoted by $M, t \models r$, if $M, t \models \alpha \rightarrow \alpha$. In this case, $M$ is a model of $r$ (for $D$) at $t$. The notions of satisfaction and models carry over to programs as usual, i.e., $M, t \models P$, if $M, t \models r$ for all $r \in P$. Moreover, $M$ is a minimal model of $P$ (for $D$ at $t$) if there does not exist an $M' = (I', W', B)$ s.t. $M', t \models P$ where $I' = (T, v') \subset I$. Note that smaller models must have the same timeline.

**Definition 6 (Answer Stream)** Let $P$ be a program, $D = (T, v)$ be a data stream and $t \in T$. An interpretation stream $I$ for $D$ is an answer stream of $P$ for $D$ at $t$, if $M = (I, W, B)$ is a minimal model of the (FLP)-reduce $P^{M, t} = \{r \in P \mid M, t \models \bar{\beta}(r)\}$.

$\text{AS}(P, D, t)$ denotes the set of all such answer streams $I$. 

---

**Figure 1:** Time-based window of size 2 at $t = 3$
Example 3 (cont’d) Consider the stream $D = ([0, 3], v)$, where $v = \{0 \mapsto \{a\}, 1 \mapsto \{a\}, 3 \mapsto \{b\}\}$ and a program $P$, given by the following rules: $x \leftarrow \Box^2 \diamond a, \not y$ and $y \leftarrow \Box^2 \diamond a, \not x$. By taking $W = \{\Box^2\}$ and $B = \emptyset$, we get at $t = 3$ two answer streams $I^x$ and $I^y$ which augment $D$ by adding $x$ and $y$ to $v(3)$, respectively. For instance, the reduct $P^{M, 3}$, where $M = (I^x, W, B)$, contains only the first rule. We have $M, 3 \models P^{M, 3}$, since $M, 3 \models \Box^2 \diamond a \land \neg y \to x$. Towards a smaller model, we can not remove $a$, as $a$ is extensional (i.e., in the data stream), nor $x$ because this would invalidate the implication. The argument for $I^y$ is analogous. ■

Bi-Structural LARS Evaluation

We now define an extended variant of LARS semantics, where formulas (resp. programs) are evaluated on a pair of streams $(S_L, S_R)$ at the same time. We will later consider a substream relation $S_L \subseteq S_R$ according models similar to the logic of Here-and-There [Heyting, 1930] which was extensively studied in relation to equivalence notions for ASP.

In the sequel, we use the following notation. Given streams $S_L = (T, u_L)$ and $S_R = (T, u_R)$, we call $S = (S_L, S_R)$ a bi-stream and $M = (S, W, B)$ a bi-structure, where $W$ are window functions and $B$ is background data as in LARS. We call $S_L$ the left-stream and $S_R$ the right-stream. Moreover, $L = (S_L, W, B)$ and $R = (S_R, W, B)$ denote the underlying LARS structures of $M$; the left-stream and right-stream, respectively.

**Definition 7 (Bi-LARS Entailment)** Let $M = (S, W, B)$ be as above and let $t \in T$. The Bi-LARS entailment relation $\models$ between $(M, t, w)$ for worlds $w \in \{L, R\}$ and formulas is defined as follows (where $\alpha, \beta \in F$ are formulas):

$$
\begin{align*}
M, t, w \models \alpha & \iff \alpha \in v_w(t) \lor \alpha \in B, \text{ for } \alpha \in A, \\
M, t, w \models \alpha \land \beta & \iff M, t, w \models \alpha \land M, t, w \models \beta, \\
M, t, w \models \Box \alpha & \iff M, t', w \models \alpha \text{ for some } t' \in T, \\
M, t, w \models \Box_t \alpha & \iff M, t', w \models \alpha \text{ for all } t' \in T, \\
M, t, L \models \alpha \to \beta & \iff M, t, L \not\models \alpha \lor M, t, L \models \beta, \\
& \quad \text{and } M, t, R \models \alpha \to \beta, \\
M, t, R \models \alpha \to \beta & \iff M, t, R \not\models \alpha \lor M, t, R \models \beta, \\
M, t, L \models \neg \alpha & \iff L, t \models \neg \alpha \lor R, t \models \neg \alpha, \\
M, t, R \models \neg \alpha & \iff R, t \models \neg \alpha, \\
M, t, L \models \Box^w \alpha & \iff L, t \models \Box^w \alpha \lor R, t \models \Box^w \alpha, \\
M, t, R \models \Box^w \alpha & \iff R, t \models \Box^w \alpha.
\end{align*}
$$

Moreover, we define $M, t \models \alpha$ iff $M, t, L \models \alpha$.

Similarly as for LARS, $M, t \models \top \lor \bot$ always/never holds. If $M, t \models \alpha$ holds, we say that $M$ *entails* $\alpha$ at time $t$ and we then call $M$ a *bi-model* of $\alpha$ at time $t$. Entailment and the notion of a model are extended to sets of formulas as usual.

Observe that for the connectives/operators $\land, \lor, \Box, \Box_t$, and $\to$, Bi-LARS has a recursive definition, while the evaluation for $\Box$ and $\Box_t$ branches into separate evaluation in the underlying LARS structures. This is required with the aim to provide semantic characterizations of equivalences for a large class of LARS programs. However, we will also later examine when a recursive definition is possible. Note that in general, entailment in both LARS structures does not imply entailment in the bi-structure. For instance, consider the bi-stream $S = (S_L, S_R)$, where $S_L = ([0, 0], \{0 \mapsto \{a\}\}$ and $S_R = ([0, 0], \{0 \mapsto \emptyset\}$, and take $\alpha = a \rightarrow ((a \rightarrow a) \rightarrow a)$. We have $L, 0 \models \alpha$ and $R, 0 \not\models \alpha$, but $M, 0 \not\models \alpha$.

The following lemma, which is immediate from Def. 7, intuitively states that evaluation for the right-stream is independent of the left-stream.

**Lemma 1** $M, t, R \models \alpha$ iff $R, t \models \alpha$.

We call a bi-stream $(S_L, S_R)$ total, if $S_L = S_R$. Restricting the study to total interpretations, Bi-LARS-satisfaction clearly collapses to LARS-satisfaction.

**Proposition 1** Let $M = (S, W, B)$ be a structure, where $S = (T, v)$ and $M = (S, W, B)$, where $S = (S, S)$, $t \in T$ and $\alpha$ be a formula. Then, $M, t \models \alpha$ iff $M, t \models \alpha$.

**Bi-LARS Semantics for LARS programs.** Entailment in Bi-LARS is extended from formulas to programs analogously as for LARS. Let $D = (T, v)$ be a data stream, $t \in T$ and $P$ be a program. We say a bi-structure $M$ satisfies a rule $r$ of form (1) at $t$, denoted by $M, t \models r$, if $M, t \not\models \beta(r) \to \alpha$.

In this case, $M$ is a (bi-)model of $r$ (for $D$ at $t$). Similarly as for LARS, $M, t \models P$, if $M, t \models r$ for all $r \in P$. We then call $M$ a (bi-)model of $P$ (for $D$ at $t$).

**Plain LARS and LARSbi.** In [Beck et al., 2015], a practical fragment of LARS was introduced. We call this fragment *plain LARS*, which restricts the rule head to be of form $a$ or $\Box_t a$, where $a$ is an atom, and body formulas to be extended atoms, i.e., expressions of the grammar

$$a \mid \Box_t a \mid \Box a \mid \Box a \mid \Box a \mid \Box a . \quad (3)$$

While plain LARS serves as guiding fragment, we will obtain our results for the following broad class of LARS programs.

**Definition 8 (Fbi, LARSbi)** By $F_{bi}$ we denote the class of LARS formulas without $\neg$, where $\diamond$ only occurs in the scope of $\neg$ or $\Box$. Moreover, LARS_{bi} is the class of LARS programs where all formulas $\alpha, \beta_i$ in rules (1) are in $F_{bi}$.

In the sequel, programs are tacitly assumed to be in LARS_{bi}.

Note that the FLP-semantics of answer streams (Def. 6) is defined non-recursively. Still, we can capture it by branching in Bi-LARS evaluation of $\neg$ and $\Box_t$ into separate LARS evaluations for left and right, due to the following central property.

**Proposition 2** Let $M = (S, W, B)$ be such that $S_L \subseteq S_R$ and let $\alpha \in F_{bi}$. Then, $M, t \models \alpha$ iff $S_L, t \models \alpha$ and $S_R, t \models \alpha$.

The proof is by induction on the structure of formulas. The relation $S_L \subseteq S_R$ naturally arises with minimality checking of models, where intuitively $S_R$ is a model $M$ of a program $P$ at time $t$ and $S_L$ is a candidate model of the reduct $P^{M, 3}$. It establishes a semantic connection between left and right, which can be exploited to conclude that $M, t \models \alpha$ implies $S_L, t \models \alpha$; for arbitrary formulas, this fails. E.g., consider $S_L = ([0, 0], \{0 \mapsto \emptyset\}$, $S_R = ([0, 0], \{0 \mapsto \{a\}\}$, and $\alpha = a \rightarrow b$; then $M, 0 \models \alpha$ but $L, 0 \not\models \alpha$. Similarly, excluding $\Diamond \varphi$ is necessary to ensure that the only-if direction of Proposition 2 holds.

The following result now captures the essence of the reduct-based semantics: the left-structure must satisfy each rule whose body is true in the model given by the right-structure. The proof is based on [Lifschitz et al., 2001].
Theorem 1 For any \( M = (S, W, B) \) s.t. \( S_L \subseteq S_R \), time \( t \) and program \( P \), we have \( M, t \models P \iff R, t \models P \) and \( L, t \models P_{R,t} \).

We are now going to characterize answer streams similarly as in \cite{Lifschitz2001,Turner2001}, by capturing the minimality condition in terms of bi-equilibrium models.

**Definition 9 (bi-Equilibrium Model)** Let \( M = (I, W, B) \) be a structure. We say \( M = (I, I, W, B) \) is a bi-equilibrium model of a program \( P \) for data stream \( D \) at time \( t \), if

(i) \( M, t \models P \), and

(ii) \( M', t \not\models P \), for each \( M' = (I', I, W, B) \) such that \( D \subseteq I' \subseteq I \) and \( I' = (T, v') \).

We obtain the next theorem from Def. 6, Prop. 1 and Thm. 1.

**Theorem 2** Let \( M = (I, W, B) \) be a structure s.t. \( I \) is an interpretation stream for \( D \) at \( t \). Then, \( I \in \mathcal{AS}(P, D, t) \) iff \( M = (I, I, W, B) \) is a bi-equilibrium model.

This allows us to characterize program equivalences which include non-trivial window operators, as will be shown next.

**Equivalence**

We now introduce equivalence notions for stream reasoning in our setting. Given a timeline \( T \), we say a set \( A \) of \( \triangleright \) atoms, i.e., extended atoms of form \( \triangleright \alpha \), have time references in \( T \), if \( \{ t \mid \triangleright \alpha, \alpha \in A \} \subseteq T \). This notion carries over naturally for programs \( P \), i.e., if \( \{ t \mid \triangleright \alpha, \alpha \in P \} \). \( \subseteq T \).

**Definition 10 (Equivalence Notions)** Let \( T \) be a timeline, \( D = (T, v) \) be a data stream, and let \( t \in T \) be a time point. We say two LARS programs \( P \) and \( Q \) are

(i) (ordinary) equivalent (for \( D \) at \( t \)), denoted by \( P \equiv Q \), if \( \mathcal{AS}(P, D, t) = \mathcal{AS}(Q, D, t) \);

(ii) strongly equivalent (for \( D \) at \( t \)), denoted by \( P \equiv_s Q \), if \( \mathcal{AS}(P \cup R, D, t) = \mathcal{AS}(Q \cup R, D, t) \) for all LARS programs \( R \) with time references in \( T \);

(iii) uniform equivalent (for \( D \) at \( t \)), denoted by \( P \equiv_u Q \), if \( \mathcal{AS}(P \cup F, D, t) = \mathcal{AS}(Q \cup F, D, t) \) for sets \( F \) of \( \triangleright \) atoms with time references in \( T \);

(iv) data equivalent (for \( D \) at \( t \)), denoted by \( P \equiv_d Q \), if \( \mathcal{AS}(P, D \cup S, t) = \mathcal{AS}(Q, D \cup S, t) \) for all data streams \( S \) with timeline \( T \).

Intuitively, (i)-(iii) can be seen as extensions of corresponding notions in ASP \cite{Lifschitz2001,Eiter2007} to the LARS setting. In particular, these notions emerge if the programs \( P \) and \( Q \) are ordinary logic programs (without windows and temporal operators) and we consider a void data stream \( D = (\{0, 0\}, v) \) at time point 0. On the other hand, data equivalence is well-known in the database area and plays an important role in stream reasoning, as the possibility to drop data is crucial to gain performance. The addition of all rules resp. facts accounts for the nonmonotonic nature of answer streams, as replacing ordinary equivalent programs \( P \) and \( Q \) in the context of other rules \( R \) might change answer streams.

We now characterize strong and uniform equivalence by means of bi-models. To this end, from now we tacitly restrict to bi-structures \( M = (S, W, B) \) such that \( S_L \subseteq S_R \). By \( \text{bi}(P) \), we denote the set of all respective bi-models of a program \( P \) (where \( D \) and \( t \) are implicit).

**Theorem 3 (Strong Equivalence)** Let \( D = (T, v) \) be a data stream, \( t \in T \), and let \( P \) and \( Q \) be \( \text{LARS}_{bi} \) programs. Then, \( P \equiv_s Q \) (for \( D \) at \( t \)).

Furthermore, we also characterize uniform equivalence in terms of bi-entailment. Let \( M_{LR} = (\langle S_L, S_R \rangle, W, B) \) and \( M_{RR} = (\langle S_L, S_R \rangle, W, B) \).

**Theorem 4 (Uniform Equivalence)** \( P \equiv_u Q \) iff

(i) for each \( M_{RR} \), \( M_{RR} \in \text{bi}(P) \) iff \( M_{RR} \in \text{bi}(Q) \), and

(ii) for each \( M_{LR} \), where \( S_L \subseteq S_R \), \( M_{LR} \in \text{bi}(P) \) (resp. \( M_{LR} \in \text{bi}(Q) \)) iff \( M \in \text{bi}(Q) \) (resp. \( M \in \text{bi}(P) \)) for some \( M = (\langle S_L, S_R \rangle, W, B) \) s.t. \( S_L \subseteq S \subseteq S_R \).

The proofs of Theorems 3 and 4 are abstracted from those for answer set programs (cf. \cite{Lifschitz2001,Eiter2003}) and exploit the following key properties: (1) the reduct of the union of two programs is the union of their reducts (2) the reduct of a set of atoms \( F \) is \( F \) itself, (3) an atom evaluates to true iff it is in the interpretation stream, and (4) a structure entails the union of two programs iff it entails each program separately.

In a similar way, also relativised uniform equivalence \cite{Woltran2004}, denoted by \( P \equiv_{uA} Q \), can be characterized, i.e., the condition that \( \mathcal{AS}(P \cup F, D, t) = \mathcal{AS}(Q \cup F, D, t) \) for all facts \( F \subseteq A \), where \( A \) is a set of \( \triangleright \) atoms. Notably, data equivalence amounts to a special case of relativized uniform equivalence: Consider for \( D = (T, v) \) the set \( A = \{ \triangleright \alpha \mid \alpha \in A \} \) and \( t \in T \). Then,

\[ P \equiv_u Q \iff P \equiv_{uA} Q. \]

Recall that plain LARS allows only intensional atoms and \( \triangleright \) atoms with intensional atoms in rule heads. However, this is not the case for all \( \text{LARS}_{bi} \) programs, which also allow window operators in rule heads.

**LARS Here-&-There and Monotone Windows**

In Definition 7, the semantics of the window operator \( \bowtie \) was defined in Bi-LARS by straight branching into separate evaluation of the left and the right stream. Consider the following alternative \( \bowtie \) semantics.

**Definition 11 (Recursive \( \bowtie \))** We define the following alternative \( \bowtie \) semantics for Bi-LARS. Let \( w \in \{L, R\} \).

\[ M, t, w \vdash \bowtie^\alpha \alpha \iff M', t, w \vdash \alpha, \]

where \( M' = (\langle S'_L, S'_R \rangle, W, B) \) and \( S'_w = w(S_w, t) \).

This recursive variant may in general break the connection between left and right, i.e., the relation \( S_L \subseteq S_R \).

**Example 4** Consider streams \( S_L = (\{0, 4\}, v_L) \) and \( S_R = (\{0, 4\}, v_R) \) as depicted in Figure 2, where \( S_L \subseteq S_R \). Applying a tuple-based window operator with size 3 at \( t = 4 \) returns \( S'_L = (\{1, 4\}, \{1 \mapsto \{a\}, 3 \mapsto \{b\}, 4 \mapsto \{a\}\}) \) as substream of \( S_L \), and \( S'_R = (\{3, 4\}, \{3 \mapsto \{b\}, 4 \mapsto \{a, c\}\}) \) for \( S_R \). We observe that \( S'_L \not\subseteq S'_R \), i.e. the substream relation breaks.

In Example 4, the window is nonmonotonic in the sense that by increasing the input stream, atoms may disappear from the output. When excluding such nonmonotonic windows, the recursive version for bi semantics may be equally used.
Definition 12 We call a window function \( w \) monotone, if for every streams \( S_1 = (T_1, v_1) \) and \( S_2 = (T_2, v_2) \) it holds that
\[
S_1 \subseteq S_2 \implies w(S_1, t) \subseteq w(S_2, t) \text{ for all } t \in T_1,
\]
\( i.e., \) \( w \) preserves substreams. If \( T_1 = T_2 \), this extends to timelines, \( i.e., \) \( w(S_1, t) = (T'_1, v'_1) \implies T'_1 = T'_2 \text{ for all } t \in T_1. \)

Likewise, we call a window operator \( \boxplus^w \) monotone if the underlying window function \( w \) is monotone. \( E.g., \) time-based window operators just filter data and thus have this property.

Proposition 3 For plain LARS with monotone windows, the window semantics of Def. 7 and Def. 11 coincide.

Using Def. 11 for a variant of Bi-LARS on monotone windows can be seen as an extension of Here-and-There [Heyting, 1930] underlying Equilibrium Logic [Pearce, 2006].

Definition 13 (HT-entailment) HT-entailment is defined as variant of Bi-LARS entailment (Def. 7), using instead Def. 11 for the \( \boxplus \) semantics and \( \neg \alpha := \alpha \rightarrow \bot \) for negation.

Based on HT-entailment, we obtain a conservative extension of Pearce’s Equilibrium Logic for LARS with monotone windows, which treats nested implications intuitionistically, and thus different from FLP-based semantics. Under limited nesting of negation, the two semantics actually coincide; \( E.g., \) time-based window operators just filter data and thus have this property.

Proposition 3 For plain LARS with monotone windows, the window semantics of Def. 7 and Def. 11 coincide.

Using Def. 11 for a variant of Bi-LARS on monotone windows can be seen as an extension of Here-and-There [Heyting, 1930] underlying Equilibrium Logic [Pearce, 2006].

Definition 13 (HT-entailment) HT-entailment is defined as variant of Bi-LARS entailment (Def. 7), using instead Def. 11 for the \( \boxplus \) semantics and \( \neg \alpha := \alpha \rightarrow \bot \) for negation.

Based on HT-entailment, we obtain a conservative extension of Pearce’s Equilibrium Logic for LARS with monotone windows, which treats nested implications intuitionistically, and thus different from FLP-based semantics. Under limited nesting of negation, the two semantics actually coincide; \( E.g., \) time-based window operators just filter data and thus have this property.

Proposition 3 For plain LARS with monotone windows, the window semantics of Def. 7 and Def. 11 coincide.

Using Def. 11 for a variant of Bi-LARS on monotone windows can be seen as an extension of Here-and-There [Heyting, 1930] underlying Equilibrium Logic [Pearce, 2006].

Definition 13 (HT-entailment) HT-entailment is defined as variant of Bi-LARS entailment (Def. 7), using instead Def. 11 for the \( \boxplus \) semantics and \( \neg \alpha := \alpha \rightarrow \bot \) for negation.

Based on HT-entailment, we obtain a conservative extension of Pearce’s Equilibrium Logic for LARS with monotone windows, which treats nested implications intuitionistically, and thus different from FLP-based semantics. Under limited nesting of negation, the two semantics actually coincide; \( E.g., \) time-based window operators just filter data and thus have this property.
\[ AS(P, D, t) \neq \emptyset / P \equiv_0 Q \]
\[ P \equiv_0 Q, \; e = s / u / d \]  

<table>
<thead>
<tr>
<th></th>
<th>mon</th>
<th>nonmon</th>
</tr>
</thead>
<tbody>
<tr>
<td>plain LARS</td>
<td>NP / coNP</td>
<td>coNP / coNP / coNP</td>
</tr>
<tr>
<td>stratified</td>
<td>P / P</td>
<td>coNP / coNP / coNP</td>
</tr>
</tbody>
</table>

Table 1: Complexity of consistency and equivalence checking for \( D \) at \( t \) (entries denote completeness results)

and \( Q \) on \( A \), let \( A_d \) be a copy of \( A \) and let \( P_d = P \cup R_d \) and \( Q_d = Q \cup R_d \), where \( R_d = \{ a \leftarrow a_d \mid a \in A \} \). Then, \( P \) and \( Q \) are uniformly equivalent (wrt. \( A \)) iff \( P_d \) and \( Q_d \) are data equivalent wrt. \( A_d \). This can be extended to LARS programs (where window operators possibly must be adapted), using rules \( \circlearrowleft_t a \leftarrow \circlearrowleft_t a_d \) for the time points \( t \). Thus, for the hardness results, it remains to consider uniform equivalence.

Many of the lower bounds are then inherited from the complexity of the respective notions for ordinary logic programs [Eiter et al., 2007]. In particular, plain LARS programs subsume normal logic programs, for which deciding answer stream existence is NP-complete, while deciding uniform resp. strong equivalence is coNP-complete (even for acyclic, i.e., recursion-free programs [Eiter et al., 2007]).

What remains are the \( \Sigma^p_2 \)-hardness of answer stream existence and \( \Pi^p_2 \)-hardness of ordinary equivalence and uniform equivalence, respectively, for plain LARS programs with nonmonotonic windows. These results are shown by reductions of evaluating QBFs of the form \( \exists X \forall Y E(X, Y) \), resp. \( \forall X \exists Y E(X, Y) \). Actually, the proofs establish them for plain programs without negation (i.e., for Horn programs); they hinge on techniques in [Eiter and Gottlob, 1995] and [Eiter and Fink, 2003] for the respective problems on ordinary disjunctive logic programs, but must compensate the lack of negation and of disjunction. We can emulate negation using nonmonotonic windows by the following construction.

**Example 5** To emulate \( \neg a \) for an atom \( a \), we create a window operator \( \boxplus_{\neg a} \) with an associated window function \( w_{\neg a}(S, t) \) that removes \( d_a \) from \( v(t) \), if \( a \in v(t) \), where \( d_a \) is a fresh atom, and otherwise leaves \( S \) unchanged. If \( d_a \) is a fact in a program \( P \) and thus true in every model \( M \) of \( P \) at \( t \), we have \( M, t \models \boxplus_{\neg a} @_t a_d \) iff \( M, t \models \neg a \).

Similarly we can define window operators \( \boxplus_\Phi \) that check whether a model \( M \) satisfies at \( t \) a polynomial-time property \( \Phi \); e.g., whether a truth assignment given in \( v(t) \) satisfies a Boolean formula. We use this to evaluate \( E(X, Y) \) using window atoms. However, for space reasons, we must omit details.

Notably, the property \( \Phi \) may also be deciding \( M, t \models \boxplus_\Phi \) where the window nesting in \( \boxplus_\Phi \) is bounded (cf. Lemma 2); the latter can be used to compile complex formulas inside window operators away, and shows that plain LARS with nonmonotonic windows is quite powerful.

**Related Work and Conclusion**

Lifschitz et al. [2001] introduced strong equivalence of logic programs under ASP semantics and showed that it coincides with equivalence in Here-and-There logic. Inspired by this, Eiter et al. [2007] characterized uniform and relativized notions of equivalence in ASP in terms of HT-interpretations (\( H, T \)). We generalize this to the LARS framework with bi-structures \( (S_L, S_R) \) containing pairs of streams.

As regards optimization in stream reasoning, to our knowledge not much foundational work exists. Typically, the interest is concentrated on dealing with evolving data, and to develop incremental evaluation techniques, e.g., [Motik et al., 2015; Ren and Pan, 2011; Gebser et al., 2011; Barbieri et al., 2010a; Beck et al., 2015]. In the context of data processing, [Arasu et al., 2006] studied query equivalence for the Continuous Query Language (CQL), but at a very elementary level. As in essence CQL can be captured by LARS programs [Beck et al., 2015], results on LARS equivalence may be fruitfully applied in this context as well.

Naturally, stream reasoning relates to temporal reasoning; in [Cabalar and Vega, 2007], nonmonotonic Linear Temporal Equilibrium Logic (TEL) was presented as an extension of Pace’s Equilibrium Logic [2006] to linear time logic (LTL), defining temporal stable models over infinite structures. Notably, strong equivalence for TEL theories amounts to equivalence in the underlying Temporal Here-and-There logic [Aguado et al., 2008; Cabalar and Diéguez, 2014]. However, TEL differs from LARS in several respects. LARS aims primarily at finite (single-path) structures, and the notion of window, which requires to go beyond the HT setting, has no counterpart in TEL. Furthermore, the temporal operators in LARS are more geared to access of data in windows in practice. Also the complexity of TEL, extensively studied in [Bozzelli and Pearce, 2015], and LARS is different: model checking for (unbounded) LARS formulas resp. programs is PSPACE-complete, while for LTL underlying TEL it is polynomial in our finite path setting (in fact, efficiently parallelizable [Kuhtz and Finkbeiner, 2009]).

**Outlook.** In future work, our studies can be extended in several directions. Apart from other notions of equivalence, additional program classes are of practical relevance. In particular, by confining to widely used time-based and tuple-based window operators one might exploit more specific properties than by the distinction of monotone vs. nonmonotone ones. Related to this is identifying maximal (relative to some criteria) fragments where Here-and-There semantics coincides with Bi-LARS, which is tailored for FLP. Furthermore, one might introduce window operators to the more expressive temporal equilibrium logic. Apart from potential extensions of Bi-LARS to capture according semantics, combining nonmonotonic temporal reasoning with features to drop data is an intriguing issue. Moreover, besides semantic also syntactic criteria for program equivalence are practically important. In particular, finding normal forms in relation to prominent window operators will directly support the optimization of programs for reasoning over streaming data.

**Acknowledgments.** We thank Andreas Humenberger for his assistance in the early phase of this work.
References


