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.*, 2015b], 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., 2007b]. 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 *et al.*, 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., 2015a] 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.

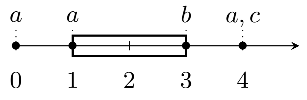

Figure 1: Time-based window of size 2 at $t = 3$

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.*, 2015b], 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 *et al.*, 2011]; for equivalence relations between ASP programs, we refer to [Lifschitz *et al.*, 2001; Eiter *et al.*, 2007b].

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

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 $v : \mathbb{N} \mapsto 2^{\mathcal{A}}$.

Intuitively, a stream S 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 $v = \{0 \mapsto \{a\}, 1 \mapsto \{a\}, 3 \mapsto \{b\}, 4 \mapsto \{a, c\}\}$, 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 $S' = ([1, 3], \{1 \mapsto \{a\}, 3 \mapsto \{b\}\})$. ■

Window operators $\boxplus$. Window functions can be accessed in formulas by window operators. For every window function w, employing an expression $\boxplus^w \alpha$ has the effect that α 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.*

$$\alpha ::= a \mid \neg\alpha \mid \alpha \wedge \alpha \mid \alpha \rightarrow \alpha \mid \Diamond\alpha \mid \Box\alpha \mid @_t\alpha \mid \boxplus^w\alpha$$

The precedence of operators $\Diamond, \Box, @_t$ and $\boxplus^w$ is as for $\neg$; e.g., $\neg \boxplus^w \Diamond a \wedge @_t b \rightarrow c = ((\neg \boxplus^w \Diamond a) \wedge (@_t b)) \rightarrow c$.

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

Definition 5 (LARS Entailment) *Let $M = \langle S, W, \mathcal{B}\rangle$ be a structure, $S = (T, v)$ and $t \in T$. The* LARS-entailment *relation $\Vdash$ between (M, t) and formulas is defined as follows. Let $a \in \mathcal{A}$ be an atom, and let $\alpha, \beta \in \mathcal{F}$ be formulas. Then,*

$M, t \Vdash a$	*iff*	$a \in v(t)$ *or* $a \in \mathcal{B}$,
$M, t \Vdash \neg\alpha$	*iff*	$M, t \nVdash \alpha$,
$M, t \Vdash \alpha \wedge \beta$	*iff*	$M, t \Vdash \alpha$ *and* $M, t \Vdash \beta$,
$M, t \Vdash \alpha \rightarrow \beta$	*iff*	$M, t \nVdash \alpha$ *or* $M, t \Vdash \beta$,
$M, t \Vdash \Diamond\alpha$	*iff*	$M, t' \Vdash \alpha$ *for some* $t' \in T$,
$M, t \Vdash \Box\alpha$	*iff*	$M, t' \Vdash \alpha$ *for all* $t' \in T$,
$M, t \Vdash @_{t'}\alpha$	*iff*	$M, t' \Vdash \alpha$ *and* $t' \in T$,
$M, t \Vdash \boxplus^w\alpha$	*iff*	$M', t \Vdash \alpha$, *where* $M' = \langle S', W, \mathcal{B}\rangle$ *such that* $S' = w(S, t)$.

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

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

Example 2 Let $S = ([0, 3], \{0 \mapsto \{a\}, 1 \mapsto \{a\}, 3 \mapsto \{b, x\}\})$ be a stream and let $\varphi = \boxplus^2 \Diamond a \wedge \neg y \rightarrow x$ be a LARS formula. Consider the structure $M = \langle S, \{\boxplus^2\}, \emptyset\rangle$. We have that $M, 3 \Vdash \boxplus^2 \Diamond a \wedge \neg y$. Indeed, since $y \notin v(3)$, it follows that $M, 3 \Vdash \neg y$. Furthermore, $M, 3 \Vdash \boxplus^2 \Diamond a$ since $M', 1 \Vdash a$, where M' is obtained by replacing S with $S' = ([1, 3], \{1 \mapsto \{a\}, 3 \mapsto \{b, x\}\})$, i.e., the result of applying the time-based window of size 2 on S at time point 3. Furthermore, $M, 3 \Vdash x$ as $x \in v(3)$; thus $M, 3 \Vdash \varphi$ holds. ■

We now turn to LARS programs that build on formulas.

LARS Programs

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

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

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

For instance, rule $x \leftarrow \boxplus^2 \Diamond a, \text{not}\, y$ amounts to φ of Ex. 2.

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

$$\bar{\beta}(r) = \beta_1 \wedge \cdots \wedge \beta_m \wedge \neg\beta_{m+1} \cdots \wedge \neg\beta_n\,. \tag{2}$$

Let $t \in T$. We say M *satisfies* rule r at t, denoted by $M, t \models r$, if $M, t \Vdash \bar{\beta}(r) \rightarrow \alpha$. 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 $M' = \langle I', W, \mathcal{B}\rangle$ 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 $M = \langle I, W, \mathcal{B}\rangle$ is a minimal model of the (FLP)-*reduct

$$P^{M,t} = \{r \in P \mid M, t \models \bar{\beta}(r)\}\,;$$

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

Example 3 (cont'd) Consider the stream $D = ([0,3], \upsilon)$, where $\upsilon = \{0 \mapsto \{a\}, 1 \mapsto \{a\}, 3 \mapsto \{b\}\}$ and a program P, given by the following rules: $x \leftarrow \boxplus^2 \Diamond a, \mathrm{not}\, y$ and $y \leftarrow \boxplus^2 \Diamond a, \mathrm{not}\, x$. By taking $W = \{\boxplus^2\}$ and $\mathcal{B} = \emptyset$, we get at $t = 3$ two answer streams I^x and I^y which augment D by adding x and y to $\upsilon(3)$, respectively. For instance, the reduct $P^{M,3}$, where $M = \langle I^x, W, \mathcal{B} \rangle$, contains only the first rule. We have $M, 3 \models P^{M,3}$, since $M, 3 \Vdash \boxplus^2 \Diamond a \wedge \neg y \rightarrow x$. 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 $(S_{\mathrm{L}}, S_{\mathrm{R}})$ at the same time. We will later consider a substream relation $S_{\mathrm{L}} \subseteq S_{\mathrm{R}}$ on 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 $S_{\mathrm{L}} = (T, \upsilon_{\mathrm{L}})$ and $S_{\mathrm{R}} = (T, \upsilon_{\mathrm{R}})$, we call $\mathbf{S} = (S_{\mathrm{L}}, S_{\mathrm{R}})$ a *bi-stream* and $\mathbf{M} = \langle \mathbf{S}, W, \mathcal{B} \rangle$ a *bi-structure*, where W are window functions and $\mathcal{B}$ is background data as in LARS. We call S_{L} the *left-stream* and S_{R} the *right-stream*. Moreover, $\mathrm{L} = \langle S_{\mathrm{L}}, W, \mathcal{B} \rangle$ and $\mathrm{R} = \langle S_{\mathrm{R}}, W, \mathcal{B} \rangle$ denote the underlying *LARS structures of* $\mathbf{M}$; the *left-structure* and the *right-structure*, respectively.

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

$\mathbf{M}, t, \mathsf{w} \Vdash a$	*iff*	$a \in \upsilon_{\mathsf{w}}(t)$ *or* $a \in \mathcal{B}$, *for* $a \in \mathcal{A}$,
$\mathbf{M}, t, \mathsf{w} \Vdash \alpha \wedge \beta$	*iff*	$\mathbf{M}, t, \mathsf{w} \Vdash \alpha$ *and* $\mathbf{M}, t, \mathsf{w} \Vdash \beta$,
$\mathbf{M}, t, \mathsf{w} \Vdash \Diamond \alpha$	*iff*	$\mathbf{M}, t', \mathsf{w} \Vdash \alpha$ *for some* $t' \in T$,
$\mathbf{M}, t, \mathsf{w} \Vdash \Box \alpha$	*iff*	$\mathbf{M}, t', \mathsf{w} \Vdash \alpha$ *for all* $t' \in T$,
$\mathbf{M}, t, \mathsf{w} \Vdash @_{t'} \alpha$	*iff*	$\mathbf{M}, t', \mathsf{w} \Vdash \alpha$ *and* $t' \in T$,
$\mathbf{M}, t, \mathsf{L} \Vdash \alpha \rightarrow \beta$	*iff*	$\mathbf{M}, t, \mathsf{L} \nVdash \alpha$ *or* $\mathbf{M}, t, \mathsf{L} \Vdash \beta$, *and* $\mathbf{M}, t, \mathsf{R} \Vdash \alpha \rightarrow \beta$
$\mathbf{M}, t, \mathsf{R} \Vdash \alpha \rightarrow \beta$	*iff*	$\mathbf{M}, t, \mathsf{R} \nVdash \alpha$ *or* $\mathbf{M}, t, \mathsf{R} \Vdash \beta$,
$\mathbf{M}, t, \mathsf{L} \Vdash \neg \alpha$	*iff*	$\mathrm{L}, t \Vdash \neg \alpha$ *and* $\mathrm{R}, t \Vdash \neg \alpha$
$\mathbf{M}, t, \mathsf{R} \Vdash \neg \alpha$	*iff*	$\mathrm{R}, t \Vdash \neg \alpha$
$\mathbf{M}, t, \mathsf{L} \Vdash \boxplus^w \alpha$	*iff*	$\mathrm{L}, t \Vdash \boxplus^w \alpha$ *and* $\mathrm{R}, t \Vdash \boxplus^w \alpha$,
$\mathbf{M}, t, \mathsf{R} \Vdash \boxplus^w \alpha$	*iff*	$\mathrm{R}, t \Vdash \boxplus^w \alpha$.

Moreover, we define $\mathbf{M}, t \Vdash \alpha$ *iff* $\mathbf{M}, t, \mathsf{L} \Vdash \alpha$.

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

Observe that for the connectives/operators $\wedge$, $\Diamond$, $\Box$, $@_{t'}$ and $\rightarrow$, Bi-LARS has a recursive definition, while the evaluation for $\neg$ and $\boxplus$ 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 $\mathbf{S} = (S_{\mathrm{L}}, S_{\mathrm{R}})$, where $S_{\mathrm{L}} = ([0,0], \{0 \mapsto \{a\}\})$ and $S_{\mathrm{R}} = ([0,0], \{0 \mapsto \emptyset\})$, and take $\alpha = a \rightarrow ((a \rightarrow a) \rightarrow a)$. We have $\mathrm{L}, 0 \Vdash \alpha$ and $\mathrm{R}, 0 \Vdash \alpha$, but $\mathbf{M}, 0 \nVdash \alpha$.

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 $\mathbf{M}, t, \mathsf{R} \Vdash \alpha$ *iff* $\mathrm{R}, t \Vdash \alpha$.

We call a bi-stream $(S_{\mathrm{L}}, S_{\mathrm{R}})$ *total*, if $S_{\mathrm{L}} = S_{\mathrm{R}}$. Restricting the study to total interpretations, Bi-LARS-satisfaction clearly collapses to LARS-satisfaction.

Proposition 1 *Let* $M = \langle S, W, \mathcal{B} \rangle$ *be a structure, where* $S = (T, \upsilon)$ *and* $\mathbf{M} = \langle \mathbf{S}, W, \mathcal{B} \rangle$, *where* $\mathbf{S} = (S, S)$, $t \in T$ *and* α *be a formula. Then,* $\mathbf{M}, t \Vdash \alpha$ *iff* $M, t \Vdash \alpha$.

Bi-LARS Semantics for LARS programs. Entailment in Bi-LARS is extended from formulas to programs analogously as for LARS. Let $D = (T, \upsilon)$ be a data stream, $t \in T$ and let P be a program. We say a bi-structure $\mathbf{M}$ *satisfies* a rule r of form (1) at t, denoted by $\mathbf{M}, t \models r$, if $\mathbf{M}, t \Vdash \bar{\beta}(r) \rightarrow \alpha$. In this case, $\mathbf{M}$ is a *(bi-)model* of r (for D at t). Similarly as for LARS, $\mathbf{M}, t \models P$, if $\mathbf{M}, t \models r$ for all $r \in P$. We then call $\mathbf{M}$ a *(bi-)model* of P (for D at t).

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

$$a \mid @_t a \mid \boxplus @_t a \mid \boxplus \Diamond a \mid \boxplus \Box a \,. \tag{3}$$

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

Definition 8 ($\mathcal{F}_{\mathrm{bi}}$, LARS$_{\mathrm{bi}}$) *By* $\mathcal{F}_{\mathrm{bi}}$ *we denote the class of LARS formulas without* $\rightarrow$, *where* $\Diamond$ *only occurs in the scope of* $\neg$ *or* $\boxplus$. *Moreover, LARS*$_{\mathrm{bi}}$ *is the class of LARS programs where all formulas* α, β_i *in rules (1) are in* $\mathcal{F}_{\mathrm{bi}}$.

In the sequel, programs are tacitly assumed to be in LARS$_{\mathrm{bi}}$.

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 $\boxplus$ into separate LARS evaluations for left and right, due to the following central property.

Proposition 2 *Let* $\mathbf{M} = \langle \mathbf{S}, W, \mathcal{B} \rangle$ *be such that* $S_{\mathrm{L}} \subseteq S_{\mathrm{R}}$ *and let* $\alpha \in \mathcal{F}_{\mathrm{bi}}$. *Then,* $\mathbf{M}, t \Vdash \alpha$ *iff* $S_{\mathrm{L}}, t \Vdash \alpha$ *and* $S_{\mathrm{R}}, t \Vdash \alpha$.

The proof is by induction on the structure of formulas. The relation $S_{\mathrm{L}} \subseteq S_{\mathrm{R}}$ 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 $P^{M,t}$. It establishes a semantic connection between left and right, which can be exploited to conclude that $\mathbf{M}, t \Vdash \alpha$ implies $S_{\mathrm{L}}, t \Vdash \alpha$; for arbitrary formulas, this fails. E.g., consider $S_{\mathrm{L}} = ([0,0], \{0 \mapsto \emptyset\})$, $S_{\mathrm{R}} = ([0,0], \{0 \mapsto \{a\}\})$, and $\alpha = \neg a \rightarrow b$; then $\mathbf{M}, 0 \Vdash \alpha$ but $\mathrm{L}, 0 \nVdash \alpha$. 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* $\mathbf{M}{=}\langle \mathbf{S}, W, \mathcal{B}\rangle$ *s.t.* $S_{\mathrm{L}} \subseteq S_{\mathrm{R}}$*, time t and program P, we have* $\mathbf{M}, t \models P$ *iff* $\mathrm{R}, t \models P$ *and* $\mathrm{L}, t \models P^{\mathrm{R},t}$.

We are now going to characterize answer streams similarly as in [Lifschitz et al., 2001] and [Turner, 2001], by capturing the minimality condition in terms of bi-equilibrium models.

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

(i) $\mathbf{M}, t \models P$*, and*

(ii) $\mathbf{M}', t \not\models P$*, for each* $\mathbf{M}' = \langle (I', I), W, \mathcal{B}\rangle$ *such that* $D \subseteq I' \subset I$ *and* $I' = (T, v')$.

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

Theorem 2 *Let* $M = \langle I, W, \mathcal{B}\rangle$ *be a structure s.t. I is an interpretation stream for D at t. Then,* $I \in \mathcal{AS}(P, D, t)$ *iff* $\mathbf{M} = \langle (I, I), W, \mathcal{B}\rangle$ *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 *@-atoms*, i.e., extended atoms of form $@_t a$, have *time references in* T, if $\{t \mid @_t a \in A\} \subseteq T$. This notion carries over naturally for programs P, i.e., if $\{t \mid @_t a \text{ occurs in } P\} \subseteq T$.

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 @-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 [Lifschitz et al., 2001; Eiter et al., 2007b] 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 $\mathbf{M} = \langle \mathbf{S}, W, \mathcal{B}\rangle$ such that $S_{\mathrm{L}} \subseteq S_{\mathrm{R}}$. By $\mathrm{bi}(P)$, we denote the set of all respective bi-models of a program P (where D and t are implicit).

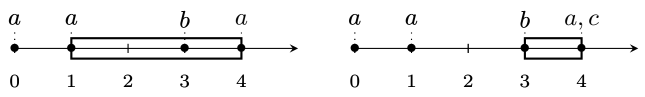

Figure 2: Tuple-based windows of size 3 at $t = 4$.

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

Furthermore, we also characterize uniform equivalence in terms of bi-entailment. Let $\mathbf{M}_{\mathsf{LR}} = \langle (S_{\mathrm{L}}, S_{\mathrm{R}}), W, \mathcal{B}\rangle$ and $\mathbf{M}_{\mathsf{RR}} = \langle (S_{\mathrm{R}}, S_{\mathrm{R}}), W, \mathcal{B}\rangle$.

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

(i) *for each* $\mathbf{M}_{\mathsf{RR}}$*,* $\mathbf{M}_{\mathsf{RR}} \in \mathrm{bi}(P)$ *iff* $\mathbf{M}_{\mathsf{RR}} \in \mathrm{bi}(Q)$*, and*

(ii) *for each* $\mathbf{M}_{\mathsf{LR}}$*, where* $S_{\mathrm{L}} \subset S_{\mathrm{R}}$*,* $\mathbf{M}_{\mathsf{LR}} \in \mathrm{bi}(P)$ *(resp.* $\mathbf{M}_{\mathsf{LR}} \in \mathrm{bi}(Q)$*) iff* $\mathbf{M} \in \mathrm{bi}(Q)$ *(resp.* $\mathbf{M} \in \mathrm{bi}(P)$*) for some* $\mathbf{M} = \langle (S, S_{\mathrm{R}}), W, \mathcal{B}\rangle$ *s.t.* $S_{\mathrm{L}} \subseteq S \subset S_{\mathrm{R}}$.

The proofs of Theorems 3 and 4 are abstracted from those for answer set programs (cf. [Lifschitz et al., 2001], resp. [Eiter and Fink, 2003]) 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 [Woltran, 2004], denoted by $P \equiv_u^A Q$, 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 @-atoms. Notably, data equivalence amounts to a special case of relativized uniform equivalence: Consider for $D = (T, v)$ the set $A = \{@_t a \mid a \in \mathcal{A}^{\mathcal{E}} \text{ and } t \in T\}$. Then,

$$P \equiv_d Q \quad \text{iff} \quad P \equiv_u^A Q.$$

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

LARS Here-&-There and Monotone Windows

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

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

$$\mathbf{M}, t, \mathsf{w} \Vdash \boxplus^w \alpha \quad \textit{iff} \quad \mathbf{M}', t, \mathsf{w} \Vdash \alpha\,,$$

where $\mathbf{M}' = \langle (S'_{\mathrm{L}}, S'_{\mathrm{R}}), W, \mathcal{B}\rangle$ *and* $S'_{\mathsf{w}} = w(S_{\mathsf{w}}, t)$.

This recursive variant may in general break the connection between left and right, i.e., the relation $S_{\mathrm{L}} \subseteq S_{\mathrm{R}}$.

Example 4 Consider streams $S_{\mathrm{L}} = ([0, 4], v_{\mathrm{L}})$ and $S_{\mathrm{R}} = ([0, 4], v_{\mathrm{R}})$ as depicted in Figure 2, where $S_{\mathrm{L}} \subset S_{\mathrm{R}}$. Applying a tuple-based window operator with size 3 at $t = 4$ returns $S'_{\mathrm{L}} = ([1, 4], \{1 \mapsto \{a\}, 3 \mapsto \{b\}, 4 \mapsto \{a\}\})$ as substream of S_{L}, and $S'_{\mathrm{R}} = ([3, 4], \{3 \mapsto \{b\}, 4 \mapsto \{a, c\}\}$ for S_{R}. We observe that $S'_{\mathrm{L}} \not\subseteq S'_{\mathrm{R}}$, 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 $\boxplus$ semantics may be equally used.

Definition 12 *We call a window function w* monotone*, if for every streams $S_1 = (T_1, \upsilon_1)$ and $S_2 = (T_2, \upsilon_2)$ it holds that*

$$S_1 \subseteq S_2 \text{ implies } w(S_1, t) \subseteq w(S_2, t) \text{ for all } t \in T_1,$$

i.e., w preserves substreams. If $T_1 = T_2$, this extends to timelines, i.e., $w(S_i, t) = (T'_i, \upsilon'_i)$ implies $T'_1 = T'_2$ 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., for the following class of formulas/programs:

Definition 14 ($\mathcal{F}_{\mathrm{HT}}$, LARS$_{\mathrm{HT}}$) *By $\mathcal{F}_{\mathrm{HT}}$ we denote the class of LARS formulas where (i) each $\boxplus$ is monotone, (ii) each subformula $\alpha \rightarrow \beta$ expresses negation (i.e., $\neg\alpha$, as $\beta = \bot$), and (iii) no negation occurs within the scope of $\Diamond$ or another negation. By LARS$_{\mathrm{HT}}$ we denote the class of LARS programs where all formulas α, β_i in rules (1) are in $\mathcal{F}_{\mathrm{HT}}$.*

Note that nested negation must be excluded, as e.g. the rule $a \leftarrow \neg\neg a$ has the equilibrium models $(\emptyset, \emptyset)$ and $(\{a\}, \{a\})$. Only the first one amounts to an FLP-answer set.

With an inductive argument, one can show that the central property of Prop. 2 carries over to formulas in $\mathcal{F}_{\mathrm{HT}}$ under HT-entailment. This allows one to establish the characterization in Theorem 1 for this setting. Thus, for LARS$_{\mathrm{HT}}$ programs, FLP-based answers streams and HT-equilibrium models coincide, and the equivalence notions can equally be characterized by HT-entailment.

Theorem 5 (LARS Here-and-There) *Theorems 1-4 also hold for LARS$_{\mathrm{HT}}$ programs under* HT*-entailment.*

Note that LARS$_{\mathrm{HT}}$ includes plain LARS programs with monotone windows.

Computational Complexity

As regards the complexity of equivalence checking, different scenarios emerge. We focus here on deciding $P \equiv_e Q$ for an equivalence notion e, where the data stream D, the programs P, Q and a (query) time point t are given.

More general is to request equivalence at multiple or each time point t over D, and/or to consider evolutions of the data stream D. For a small (polynomial) horizon around t, i.e., an interval $[t_0, t_1]$ such that $t_0 \leq t \leq t_1$ and $|t_1 - t_0|$ is polynomially bounded, this essentially reduces to polynomially many such questions. This is in line with the view that stream reasoning may lose information, i.e., query results over the full history and the reduced data may be different.

Setup. We adopt the assumptions in [Beck *et al.*, 2015b], in particular that relevant atoms are confined to $\mathcal{A}$, the background $\mathcal{B}$ and the window functions W are fixed and can be polynomially evaluated. Then, both model checking and satisfiability testing for arbitrary LARS formulas is PSPACE-complete [Beck *et al.*, 2015b], as answer stream checking and deciding answer stream existence. However, we note:

Lemma 2 (cf. [Beck *et al.*, 2015b]) *For LARS formulas α without nested windows, deciding $M, t \Vdash \alpha$ is polynomial and deciding satisfiability of α wrt. a timeline T and time point t is* NP*-complete. The same holds for window nesting depth bounded by a constant k.*

In practice, bounded nesting of windows will apply; thus we adopt this assumption. As an easy consequence, we get:

Corollary 1 *Given a bi-structure $\mathbf{M}$, a time point t, and a (window-bounded) LARS formula α, deciding $\mathbf{M}, t \Vdash \alpha$ is feasible in polynomial time.*

Besides plain LARS, we study here the following fragment.

Stratified LARS programs extend the usual notion of stratified logic programs by allowing constraints and building the dependency graph as follows. Formulas of the form $@_{t'}a$ and $\boxplus\alpha$ are the nodes, where $a \in \mathcal{A}$, t' is a time point, and $\boxplus\alpha$ occurs only in a rule body as β_i. Relative to D and t, atoms a are identified with $@_t a$, and edges are added in the graph as usual, where also an (negation-style) edge from $\boxplus\alpha$ to every $@_{t'}a$ is added such that the value of $\boxplus\alpha$ at t depends on the one of a at t'. As this depends on the semantics of the window operator, for a simple syntactic criterion we assume here that this is only the atom a in α; e.g. time-based windows satisfy this property. Stratified LARS programs are stream-stratified LARS programs in [Beck et al., 2015a] and can be evaluated bottom up using an iterative fixed-point computation to obtain an answer stream (which exists if no constraint is violated).

Complexity results. Our results are compactly summarized in Table 1. Besides the program classes, we distinguish between only monotonic and possible nonmonotonic windows.

As it appears, the complexities of the various problems ranges from P to Π_2^p. In most of the cases, the upper bound is an immediate or easy consequence of Lemma 2 and the characterizations of answer streams and equivalences from above. In particular, deciding strong equivalence is always in coNP, while deciding answer stream existence resp. refuting uniform or data equivalence can be done in nondeterministic polynomial time using an NP oracle to verify a guess for an answer stream resp. counterexample to equivalence.

For stratified programs, the fixed-point computation of the unique answer stream is feasible in polynomial time; this explains the P-entries. Fixed-point computation is also feasible on the reduct $P^{M,t}$ of a plain LARS program with monotone windows to check minimality of M, as here negative literals

$\mathcal{AS}(P,D,t)\neq\emptyset$ / $P \equiv_o Q$ $P \equiv_e Q,\ \ e=s\,/\,u\,/\,d$	mon $\boxplus$	nonmon $\boxplus$
plain LARS	NP / coNP	Σ_2^p / Π_2^p
	coNP/ coNP / coNP	coNP/ Π_2^p / Π_2^p
stratified	P / P	P / P
	coNP / coNP / coNP	coNP / coNP / coNP

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

can be dropped in $P^{M,t}$. This explains why answer stream existence / ordinary equivalence is in NP/ coNP.

As for the lower bounds, deciding $P \equiv_u Q$ is reducible to deciding $P \equiv_d Q$. Indeed, given ordinary logic programs P and Q on $\mathcal{A}$, 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 $@_t a \leftarrow @_t a_d$ for the time points t. Thus, for the hardness results, it is 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., 2007b]. 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., 2007a]).

What remains are the Σ_2^p-hardness of answer stream existence and Π_2^p-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 $M,t \Vdash \boxplus_{\neg a} @_t d_a$ iff $M,t \Vdash \neg a$. ■

Similarly we can define window operators $\boxplus_\Phi$ that check whether a model M satisfies at t a polynomial-time property Φ; 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 Φ may also be deciding $M,t \Vdash \boxplus\varphi$ where the window nesting in $\boxplus\varphi$ 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. [2007b] 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 $(S_{\mathrm{L}}, S_{\mathrm{R}})$ 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., 2015a]. 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.*, 2015b], 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 Pearce'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

[Aguado *et al.*, 2008] Felicidad Aguado, Pedro Cabalar, Gilberto Pérez, and Concepción Vidal. Strongly equivalent temporal logic programs. In *JELIA*, 2008.

[Arasu *et al.*, 2006] Arvind Arasu, Shivnath Babu, and Jennifer Widom. The CQL continuous query language: semantic foundations and query execution. *VLDB J.*, 15(2):121–142, 2006.

[Barbieri *et al.*, 2010a] Davide Francesco Barbieri, Daniele Braga, Stefano Ceri, Emanuele Della Valle, and Michael Grossniklaus. Incremental reasoning on streams and rich background knowledge. In *ESWC 2010*, pages 1–15, 2010.

[Barbieri *et al.*, 2010b] Davide Francesco Barbieri, Daniele Braga, Stefano Ceri, Emanuele Della Valle, and Michael Grossniklaus. Querying RDF streams with C-SPARQL. *SIGMOD Record*, 39(1):20–26, 2010.

[Beck *et al.*, 2015a] Harald Beck, Minh Dao-Tran, and Thomas Eiter. Answer update for rule-based stream reasoning. In *IJCAI*, 2015.

[Beck *et al.*, 2015b] Harald Beck, Minh Dao-Tran, Thomas Eiter, and Michael Fink. LARS: A logic-based framework for analyzing reasoning over streams. In *AAAI*, 2015.

[Bozzelli and Pearce, 2015] Laura Bozzelli and David Pearce. On the complexity of temporal equilibrium logic. In *30th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2015, Kyoto, Japan, July 6-10, 2015*, pages 645–656. IEEE, 2015.

[Brewka *et al.*, 2011] Gerd Brewka, Thomas Eiter, and Miroslaw Truszczyński. Answer set programming at a glance. *Communications of the ACM*, 54(12):92–103, 2011.

[Cabalar and Diéguez, 2014] Pedro Cabalar and Martín Diéguez. Strong equivalence of non-monotonic temporal theories. In *KR*, 2014.

[Cabalar and Vega, 2007] Pedro Cabalar and Gilberto Pérez Vega. Temporal equilibrium logic: A first approach. In *Computer Aided Systems Theory - EUROCAST 2007*, pages 241–248, 2007.

[Do *et al.*, 2011] Thang M. Do, Seng Wai Loke, and Fei Liu. Answer set programming for stream reasoning. In *AI*, pages 104–109, 2011.

[Eiter and Fink, 2003] Thomas Eiter and Michael Fink. Uniform equivalence of logic programs under the stable model semantics. In *ICLP*, 2003.

[Eiter and Gottlob, 1995] T. Eiter and G. Gottlob. On the Computational Cost of Disjunctive Logic Programming: Propositional Case. *Annals of Mathematics and Artificial Intelligence*, 15(3/4):289–323, 1995.

[Eiter *et al.*, 2007a] T. Eiter, M. Fink, H. Tompits, and S. Woltran. Complexity results for checking equivalence of stratified logic programs. In *IJCAI*, 2007.

[Eiter *et al.*, 2007b] Thomas Eiter, Michael Fink, and Stefan Woltran. Semantical characterizations and complexity of equivalences in answer set programming. *ACM Transactions on Computational Logic*, 8(3), July 2007.

[Faber *et al.*, 2004] Wolfgang Faber, Nicola Leone, and Gerald Pfeifer. Recursive aggregates in disjunctive logic programs: Semantics and complexity. In *JELIA*, 2004.

[Gebser *et al.*, 2011] Martin Gebser, Orkunt Sabuncu, and Torsten Schaub. An incremental answer set programming based system for finite model computation. *AI Commun.*, 24(2):195–212, 2011.

[Gebser *et al.*, 2012] Martin Gebser, Torsten Grote, Roland Kaminski, Philipp Obermeier, Orkunt Sabuncu, and Torsten Schaub. Stream reasoning with answer set programming. Preliminary report. In *KR*, 2012.

[Gelfond and Lifschitz, 1991] M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9:365–385, 1991.

[Heyting, 1930] Arend Heyting. Die formalen Regeln der intuitionistischen Logik. In *Sitzungsberichte der preußischen Akademie der Wissenschaften. phys.-math. Klasse*, pages 42–65, 57–71, 158–169, 1930.

[Kuhtz and Finkbeiner, 2009] Lars Kuhtz and Bernd Finkbeiner. LTL path checking is efficiently parallelizable. In *ICALP*, 2009.

[Lifschitz *et al.*, 2001] Vladimir Lifschitz, David Pearce, and Agustín Valverde. Strongly equivalent logic programs. *ACM Trans. Comput. Log.*, 2(4):526–541, 2001.

[Motik *et al.*, 2015] Boris Motik, Yavor Nenov, Robert Piro, and Ian Horrocks. Incremental Update of Datalog Materialisation: The Backward/Forward Algorithm. In *AAAI*, 2015.

[Pearce, 2006] David Pearce. Equilibrium logic. *Annals of Mathematics and Artificial Intelligence*, 47(1-2):3–41, 2006.

[Ren and Pan, 2011] Yuan Ren and Jeff Z. Pan. Optimising ontology stream reasoning with truth maintenance system. In *CIKM*, pages 831–836, 2011.

[Turner, 2001] Hudson Turner. Strong equivalence for logic programs and default theories (made easy). In *LPNMR*, 2001.

[Woltran, 2004] Stefan Woltran. Characterizations for relativized notions of equivalence in answer set programming. In *JELIA*, 2004.