Equivalent Stream Reasoning Programs

Harald Beck       Minh Dao-Tran       Thomas Eiter

Institute of Information Systems, TU Vienna

IJCAI, July 13, 2016

Austrian Science Fund (FWF) grants P26471, W1255-N23
Equivalent Stream Reasoning Programs

Logic-oriented processing of continuously streaming data

“What is true now?”

Various aspects are emphasized, e.g., incremental reasoning (Stream Reasoning Workshop)

Streaming data

• Data comes and goes - only recent data matters

⇒ Limit reasoning to small windows

• Time-based windows

• Tuple-based windows . . .

Stream $S = (T, \nu)$

• Timeline $T$ closed interval in $\mathbb{N}$, $t \in T$ time point

• Evaluation function $\nu: T \to 2^A$ (sets of atoms)
Equivalent Stream Reasoning Programs

- Logic-oriented processing of continuously streaming data
Equivalent Stream Reasoning Programs

- Logic-oriented processing of continuously streaming data
- “What is true now?”
Equivalent Stream Reasoning Programs

- Logic-oriented processing of continuously streaming data
- “What is true now?”
- Various aspects are emphasized, e.g., incremental reasoning (Stream Reasoning Workshop)
Equivalent Stream Reasoning Programs

- Logic-oriented processing of continuously streaming data
- “What is true now?”
- Various aspects are emphasized, e.g., incremental reasoning (Stream Reasoning Workshop)

Streaming data
- Data comes and goes - only recent data matters
- Need to cope with data volume
Equivalent Stream Reasoning Programs

- Logic-oriented processing of continuously streaming data
- “What is true now?”
- Various aspects are emphasized, e.g., incremental reasoning (Stream Reasoning Workshop)

Streaming data
- Data comes and goes - only recent data matters
- Need to cope with data volume

⇒ Limit reasoning to small windows
- Time-based windows
- Tuple-based windows . . .
Equivalent Stream Reasoning Programs

- Logic-oriented processing of continuously streaming data
- "What is true now?"
- Various aspects are emphasized, e.g., incremental reasoning (Stream Reasoning Workshop)
- Streaming data
  - Data comes and goes - only recent data matters
  - Need to cope with data volume
- ⇒ Limit reasoning to small windows
  - Time-based windows
  - Tuple-based windows . . .

Stream $S = (T, \nu)$
- **Timeline** $T$ closed interval in $\mathbb{N}$, $t \in T$ time point
- **Evaluation** function $\nu : T \rightarrow 2^A$ (sets of atoms)
Equivalent Stream Reasoning Programs

- Logic-oriented processing of continuously streaming data
- “What is true now?”
- Various aspects are emphasized, e.g., incremental reasoning (Stream Reasoning Workshop)
- Streaming data
  - Data comes and goes - only recent data matters
  - Need to cope with data volume
- \( \Rightarrow \) Limit reasoning to small **windows**
  - Time-based windows
  - Tuple-based windows . . .
- Stream \( S = (T, \nu) \)
  - **Timeline** \( T \) closed interval in \( \mathbb{N} \), \( t \in T \) time point
  - **Evaluation** function \( \nu : T \rightarrow 2^A \) (sets of atoms)
- **Window function** \( w \) yields window \( w(S, t) \subseteq S \)
Equivalent Stream Reasoning Programs

If you are in a hurry, take a cab, unless, within a window $(\sum)$ of the last 10 minutes, there is a time point $(\exists t)$ where a traffic jam was reported.

```prolog
takeCab ← hurry, not ⊞ 10 ∧ 3 trafficJam
```

"If no train arrived in the last 20 minutes, do not take the subway."

```prolog
@ someTrain ← ⊞ 20 @ T arrivalOfTrain(Id) ¬ takeSub ← ⊞ 20 ∧ 2 ¬ someTrain ¬ takeSub ← not ⊞ 20 ∧ 3 someTrain
```
Equivalent Stream Reasoning Programs

“If you are in a hurry,

\textit{hurry},
Equivalent Stream Reasoning Programs

“If you are in a hurry, take a cab,

\[ \text{takeCab} \leftarrow \text{hurry}, \]

“If no train arrived in the last 20 minutes, do not take the subway.”
Equivalent Stream Reasoning Programs

“If you are in a hurry, take a cab, unless there was a traffic jam in the last 10 minutes.”

\[
\text{takeCab} \leftarrow \text{hurry}, \text{not trafficJamLast10Min}
\]
Equivalent Stream Reasoning LARS Programs

“If you are in a hurry, take a cab, unless,

\[ take\text{Cab} \leftarrow \text{hurry}, \text{not}\]
“If you are in a hurry, take a cab, unless, within a window (⊞) of the last 10 minutes,

\[
\text{take} \text{Cab} \leftarrow \text{hurry}, \text{not} \boxplus^{10}
\]
“If you are in a hurry, take a cab, unless, within a window (□) of the last 10 minutes, there is a time point (◇)

\[ \text{takeCab} \leftarrow \text{hurry}, \text{ not } □^{10} \Diamond \]
Equivalent Stream Reasoning LARS Programs

“If you are in a hurry, take a cab, unless, within a window (□) of the last 10 minutes, there is a time point (◇) where a traffic jam was reported.”

\[
\text{takeCab} \leftarrow \text{hurry}, \ \text{not} \ □^{10} \text{trafficJam}
\]
Equivalent Stream Reasoning LARS Programs

“If you are in a hurry, take a cab, unless, within a window (□) of the last 10 minutes, there is a time point (◇) where a traffic jam was reported.”

\[
\text{takeCab} \leftarrow \text{hurry}, \ \neg □^{10} ◇\text{trafficJam}
\]

“If no train arrived in the last 20 minutes, do not take the subway.”
Equivalent Stream Reasoning LARS Programs

“If you are in a hurry, take a cab, unless, within a window (□) of the last 10 minutes, there is a time point (◊) where a traffic jam was reported.”

\[
\text{takeCab} \leftarrow \text{hurry}, \neg □^{10} \text{trafficJam}
\]

“If no train arrived in the last 20 minutes, do not take the subway.”

\[
\exists_{T} \text{someTrain} \leftarrow □^{20} \exists_{T} \text{arrivalOfTrain}(Id)
\]
“If you are in a hurry, take a cab, unless, within a window (⊞) of the last 10 minutes, there is a time point (◊) where a traffic jam was reported.”

\[
\text{takeCab} \leftarrow \text{hurry}, \ \text{not} \ \ sulph^{10} \diamond \text{trafficJam}
\]

“If no train arrived in the last 20 minutes, do not take the subway.”

\[
\text{@TsomeTrain} \leftarrow \ sulph^{20} \text{@TarrivalOfTrain(}Id\text{)}
\]

\[
\text{not} \ \ sulph^{20}
\]
Equivalent Stream Reasoning LARS Programs

“If you are in a hurry, take a cab, unless, within a window (□) of the last 10 minutes, there is a time point (◇) where a traffic jam was reported.”

\[\text{takeCab} \leftarrow \text{hurry}, \text{not} \ □^{10} \text{trafficJam}\]

“If no train arrived in the last 20 minutes, do not take the subway.”

\[\text{@}_{T}\text{someTrain} \leftarrow □^{20} \text{@}_{T}\text{arrivalOfTrain(Id)}\]

\[\Box^{20}\]
Equivalent Stream Reasoning LARS Programs

“If you are in a hurry, take a cab, unless, within a window (◻) of the last 10 minutes, there is a time point (◇) where a traffic jam was reported.”

\[
\text{takeCab} \leftarrow \text{hurry, not ◻}^{10} \◇ \text{trafficJam}
\]

“If no train arrived in the last 20 minutes, do not take the subway.”

\[
@_T \text{someTrain} \leftarrow ◻^{20} @_T \text{arrivalOfTrain}(Id) \quad ◻^{20} \neg
\]
“If you are in a hurry, take a cab, unless, within a window $[10] \diamondsuit$ of the last 10 minutes, there is a time point $\Box$ where a traffic jam was reported.”

$$\text{takeCab} \leftarrow \text{hurry}, \; \neg [10] \diamondsuit \text{trafficJam}$$

“If no train arrived in the last 20 minutes, do not take the subway.”

$$\exists_{T} \text{someTrain} \leftarrow [[20] \exists_{T} \text{arrivalOfTrain}(Id) \quad \neg[20] \Box \neg \text{someTrain}$$
“If you are in a hurry, take a cab, unless, within a window (⊞) of the last 10 minutes, there is a time point (◊) where a traffic jam was reported.”

\[
\text{takeCab} \leftarrow \text{hurry}, \; \text{not} \; ⊞^{10} ◊ \text{trafficJam}
\]

“If no train arrived in the last 20 minutes, do not take the subway.”

\[
\begin{align*}
\text{@}_{T}\text{someTrain} & \leftarrow \; ⊞^{20} \text{@}_{T}\text{arrivalOfTrain}(Id) \\
\neg \text{takeSub} & \leftarrow \; ⊞^{20} □ \neg \text{someTrain}
\end{align*}
\]
Equivalent Stream Reasoning LARS Programs

“If you are in a hurry, take a cab, unless, within a window $\Box$ of the last 10 minutes, there is a time point $\Diamond$ where a traffic jam was reported.”

\[
\text{takeCab} \leftarrow \text{hurry}, \, \text{not } \Box^{10}\Diamond \text{trafficJam}
\]

“If no train arrived in the last 20 minutes, do not take the subway.”

\[
\begin{align*}
@T\text{someTrain} & \leftarrow \Box^{20}@T\text{arrivalOfTrain}(Id) \\
\neg\text{takeSub} & \leftarrow \Box^{20}\Box\neg\text{someTrain} \\
\Downarrow \\
\neg\text{takeSub} & \leftarrow \text{not } \Box^{20}\Diamond\text{someTrain}
\end{align*}
\]
Equivalent Stream Reasoning Programs

Goal

Towards optimization:

When are two LARS programs equivalent?
Equivalent Stream Reasoning Programs

Goal

- Towards optimization:
  
  When are two LARS programs equivalent?

- Semantic characterizations for suitable notions of equivalence
Equivalent Stream Reasoning Programs

**Goal**

- Towards optimization:
  - When are two LARS programs equivalent?
  - **Semantic** characterizations for suitable notions of equivalence
  - $\Rightarrow$ Do techniques from ASP carry over to LARS?
Strong Equivalence: Recall from ASP
Strong Equivalence: Recall from ASP

- $\mathcal{AS}(P)$: set of answer sets for program $P$
Strong Equivalence: Recall from ASP

- $\mathcal{AS}(P)$: set of answer sets for program $P$
- Ordinary equivalence of $P$ and $Q$: $\mathcal{AS}(P) = \mathcal{AS}(Q)$
Strong Equivalence: Recall from ASP

- $\mathcal{AS}(P)$: set of answer sets for program $P$
- Ordinary equivalence of $P$ and $Q$: $\mathcal{AS}(P) = \mathcal{AS}(Q)$
- Strong Equivalence:

$$P \equiv_s Q \text{ iff for all programs } R, \mathcal{AS}(P \cup R) = \mathcal{AS}(Q \cup R)$$
Strong Equivalence: Recall from ASP

- \( AS(P) \): set of answer sets for program \( P \)
- Ordinary equivalence of \( P \) and \( Q \): \( AS(P) = AS(Q) \)
- Strong Equivalence:
  \[
P \equiv_s Q \text{ iff for all programs } R, \ AS(P \cup R) = AS(Q \cup R)
  \]
- Characterization with logic of Here-and-There models \( HT \):
  \[
P \equiv_s Q \text{ iff } HT(P) = HT(Q)
  \]
Strong Equivalence: Recall from ASP

- $\mathcal{AS}(P)$: set of answer sets for program $P$
- Ordinary equivalence of $P$ and $Q$: $\mathcal{AS}(P) = \mathcal{AS}(Q)$
- Strong Equivalence:

$$P \equiv_s Q \iff \text{for all programs } R, \mathcal{AS}(P \cup R) = \mathcal{AS}(Q \cup R)$$

- Characterization with logic of Here-and-There models $HT$:

$$P \equiv_s Q \iff HT(P) = HT(Q)$$

- Abstraction: Pair $(X, Y)$ of atoms is an SE-model for program $P$, if

$$(i) \ X \subseteq Y \quad (ii) \ Y \models P \quad (iii) \ X \models P^Y$$
Strong Equivalence: Recall from ASP

- $\mathcal{AS}(P)$: set of answer sets for program $P$
- Ordinary equivalence of $P$ and $Q$: $\mathcal{AS}(P) = \mathcal{AS}(Q)$
- Strong Equivalence:

$$P \equiv_s Q \iff \text{for all programs } R, \mathcal{AS}(P \cup R) = \mathcal{AS}(Q \cup R)$$

- Characterization with logic of Here-and-There models $HT$:

$$P \equiv_s Q \iff HT(P) = HT(Q)$$

- Abstraction: Pair $(X, Y)$ of atoms is an SE-model for program $P$, if

$$(i) \ X \subseteq Y \quad (ii) \ Y \models P \quad (iii) \ X \models P^Y$$

- ⇒ Alternative characterization with SE-models $SE$:

$$P \equiv_s Q \iff SE(P) = SE(Q)$$
Strong Equivalence: Recall from ASP

- $\mathcal{AS}(P)$: set of answer sets for program $P$
- Ordinary equivalence of $P$ and $Q$: $\mathcal{AS}(P) = \mathcal{AS}(Q)$
- Strong Equivalence:

$$P \equiv_s Q \iff \text{for all programs } R, \mathcal{AS}(P \cup R) = \mathcal{AS}(Q \cup R)$$

- Characterization with logic of Here-and-There models $HT$:

$$P \equiv_s Q \iff HT(P) = HT(Q)$$

- Abstraction: Pair $(X, Y)$ of atoms is an SE-model for program $P$, if

$(i)\ X \subseteq Y\quad (ii)\ Y \models P\quad (iii)\ X \models P^Y$

- $\Rightarrow$ Alternative characterization with SE-models $SE$:

$$P \equiv_s Q \iff SE(P) = SE(Q)$$

- Answer sets characterized by equilibrium models $(X, X)$, where no smaller $(X', X)$ is an SE-model
Cornerstone: Characterizing Answer Sets/Streams

ASP

- Pair \((X, Y)\) of atoms is an SE-model for program \(P\), if
  
  \(\begin{align*}
  (i) \quad X & \subseteq Y \\
  (ii) \quad Y & \models P \\
  (iii) \quad X & \models P^Y
  \end{align*}\)

- Characterization:

\[ P \equiv_s Q \quad \text{iff} \quad SE(P) = SE(Q) \]
Cornerstone: Characterizing Answer Sets/Streams

ASP

- Pair \((X, Y)\) of atoms is an SE-model for program \(P\), if
  
  \((i)\) \(X \subseteq Y\) \hspace{1cm} \((ii)\) \(Y \models P\) \hspace{1cm} \((iii)\) \(X \models P^Y\)

- Characterization:

\[ P \equiv_s Q \text{ iff } SE(P) = SE(Q) \]

- Can we find similar characterizations for LARS?
Cornerstone: Characterizing Answer Sets/Streams

**ASP**
- Pair \((X, Y)\) of atoms is an SE-model for program \(P\), if
  1. \(X \subseteq Y\)
  2. \(Y \models P\)
  3. \(X \models P^Y\)
- Characterization:
  \[ P \equiv_s Q \quad \text{iff} \quad SE(P) = SE(Q) \]
- Can we find similar characterizations for LARS?

**LARS (Informally)**
- Model of a program \(P\) is (essentially) a stream \(S\) at a time point \(t\)
  \[ S, t \models P \]
Cornerstone: Characterizing Answer Sets/Streams

**ASP**
- Pair \((X, Y)\) of atoms is an SE-model for program \(P\), if
  
  (i) \(X \subseteq Y\)  
  (ii) \(Y \models P\)  
  (iii) \(X \models P^Y\)

- Characterization:
  
  \[
  P \equiv_s Q \iff SE(P) = SE(Q)
  \]

- Can we find similar characterizations for LARS?

**LARS** (Informally)
- Model of a program \(P\) is (essentially) a stream \(S\) at a time point \(t\)
  
  \[
  S, t \models P
  \]

- Answer sets \(\sim\) answer streams
Cornerstone: Characterizing Answer Sets/Streams

**ASP**

- Pair \((X, Y)\) of atoms is an **SE-model** for program \(P\), if
  
  \[(i)\] \(X \subseteq Y\) \hspace{1cm} \[(ii)\] \(Y \models P\) \hspace{1cm} \[(iii)\] \(X \models P^Y\)

- Characterization:

  \[P \equiv_s Q \iff SE(P) = SE(Q)\]

- Can we find similar characterizations for LARS?

**LARS** (Informally)

- Model of a program \(P\) is (essentially) a stream \(S\) at a time point \(t\)

  \[S, t \models P\]

- Answer sets \(\leadsto\) **answer streams**

- Bi-LARS: Evaluate pair of left/right stream \((L, R)\) s.t.

  \[(i)\] \(L \subseteq R\) \hspace{1cm} \[(ii)\] \(R, t \models P\) \hspace{1cm} \[(iii)\] \(L, t \models P^{R, t}\)
Central Challenge: Window behaviour

Bi-LARS: Evaluate pair of left/right stream \((L, R)\) s.t.

\[(i) \ L \subseteq R \quad (ii) \ R, t \models P \quad (iii) \ L, t \models P^{R,t}\]
Central Challenge: Window behaviour

- Bi-LARS: Evaluate pair of left/right stream \((L, R)\) s.t.
  \[(i) \quad L \subseteq R \quad (ii) \quad R, t \models P \quad (iii) \quad L, t \models P^{R,t}\]

- **Time-based** window functions \(w\) are monotone:
  \[L \subseteq R \quad \text{implies} \quad w(L, t) \subseteq w(R, t)\]
Central Challenge: Window behaviour

- Bi-LARS: Evaluate pair of left/right stream \((L, R)\) s.t.
  
  \[
  (i) \ L \subseteq R \quad (ii) \ R, t \models P \quad (iii) \ L, t \models P^{R,t}
  \]

- **Time-based** window functions \(w\) are monotone:
  
  \[L \subseteq R \implies w(L, t) \subseteq w(R, t)\]
Central Challenge: Window behaviour

- **Bi-LARS:** Evaluate pair of left/right stream \((L, R)\) s.t.
  
  1. \(L \subseteq R\)
  2. \(R, t \models P\)
  3. \(L, t \models P^{R,t}\)

- **Time-based** window functions \(w\) are monotone:

  \[L \subseteq R \implies w(L, t) \subseteq w(R, t)\]

\[w_{time}^3(L, 4)\]
Central Challenge: Window behaviour

- Bi-LARS: Evaluate pair of left/right stream \((L, R)\) s.t.
  
  \((i)\) \(L \subseteq R\) \hspace{1cm} \((ii)\) \(R, t \models P\) \hspace{1cm} \((iii)\) \(L, t \models P^{R,t}\)

- **Time-based** window functions \(w\) are monotone:

  \(L \subseteq R\) implies \(w(L, t) \subseteq w(R, t)\)

\[w_{time}^3(L, 4) \quad w_{time}^3(R, 4)\]
Central Challenge: Window behaviour

- **Bi-LARS**: Evaluate pair of left/right stream \((L, R)\) s.t.
  
  (i) \(L \subseteq R\)   
  (ii) \(R, t \models P\)   
  (iii) \(L, t \models P^{R,t}\)

- **Time-based** window functions \(w\) are monotone:

\[ L \subseteq R \implies w(L, t) \subseteq w(R, t) \]

\[ w_{time}^{3}(L, 4) \subseteq w_{time}^{3}(R, 4) \]
Central Challenge: Window behaviour

- **Bi-LARS**: Evaluate pair of left/right stream \((L, R)\) s.t.
  
  \((i)\) \(L \subseteq R\)  \((ii)\) \(R, t \models P\)  \((iii)\) \(L, t \models P^{R,t}\)

- **Time-based** window functions \(w\) are **monotone**:
  
  \(L \subseteq R\) implies \(w(L, t) \subseteq w(R, t)\)

- **Tuple-based** windows are not:

\[
\begin{align*}
0 & \quad 1 & \quad 2 & \quad 3 & \quad 4 \\
\bullet & \quad \bullet & \quad \bullet & \quad \bullet & \quad \bullet
\end{align*}
\]

\[
\begin{align*}
0 & \quad 1 & \quad 2 & \quad 3 & \quad 4 \\
\bullet & \quad \bullet & \quad \bullet & \quad \bullet & \quad \bullet
\end{align*}
\]

\[
w^3_{\text{time}}(L, 4) \subseteq w^3_{\text{time}}(R, 4)
\]
Central Challenge: Window behaviour

- **Bi-LARS**: Evaluate pair of left/right stream \((L, R)\) s.t.
  
  \[(i)\quad L \subseteq R \quad (ii)\quad R, t \models P \quad (iii)\quad L, t \models P^{R,t}\]

- **Time-based** window functions \(w\) are **monotone**:
  
  \(L \subseteq R\) implies \(w(L, t) \subseteq w(R, t)\)

\[w^3_{time}(L, 4) \subseteq w^3_{time}(R, 4)\]

- **Tuple-based** windows are not:

\[\]
Central Challenge: Window behaviour

- **Bi-LARS**: Evaluate pair of left/right stream \((L, R)\) s.t.
  
  \[(i) \quad L \subseteq R \quad (ii) \quad R, t \models P \quad (iii) \quad L, t \models P^{R,t}\]

- **Time-based** window functions \(w\) are **monotone**:  
  
  \[L \subseteq R \quad \text{implies} \quad w(L, t) \subseteq w(R, t)\]

\[
\begin{array}{cccc}
\text{a} & \text{a} & \text{b} & \text{a} \\
0 & 1 & 2 & 3 & 4 \\
\end{array}
\begin{array}{cccc}
\text{a} & \text{a} & \text{b} & \text{a}, \text{c} \\
0 & 1 & 2 & 3 & 4 \\
\end{array}
\]

\[
w_{\text{time}}^3(L, 4) \subseteq w_{\text{time}}^3(R, 4)
\]

- **Tuple-based** windows are not:

\[
\begin{array}{cccc}
\text{a} & \text{a} & \text{b} & \text{a} \\
0 & 1 & 2 & 3 & 4 \\
\end{array}
\begin{array}{cccc}
\text{a} & \text{a} & \text{b} & \text{a}, \text{c} \\
0 & 1 & 2 & 3 & 4 \\
\end{array}
\]

\[
w_{\text{tuple}}^3(L, 4)
\]
Central Challenge: Window behaviour

- **Bi-LARS**: Evaluate pair of left/right stream \((L, R)\) s.t.
  
  \[(i) \ L \subseteq R \quad (ii) \ R, t \models P \quad (iii) \ L, t \models P^{R,t}\]

- **Time-based** window functions \(w\) are **monotone**:
  
  \(L \subseteq R\) implies \(w(L, t) \subseteq w(R, t)\)

\[
w^3_{\text{time}}(L, 4) \subseteq w^3_{\text{time}}(R, 4)
\]

- **Tuple-based** windows are not:

\[
w^3_{\text{tuple}}(L, 4) \not\subseteq w^3_{\text{tuple}}(R, 4)
\]
Central Challenge: Window behaviour

- **Bi-LARS**: Evaluate pair of left/right stream \((L, R)\) s.t.
  
  \((i)\) \(L \subseteq R\) \hspace{1cm} \((ii)\) \(R, t \models P\) \hspace{1cm} \((iii)\) \(L, t \models P^{R,t}\)

- **Time-based** window functions \(w\) are **monotone**: 
  
  \(L \subseteq R\) implies \(w(L, t) \subseteq w(R, t)\)

\[
\begin{array}{cccccc}
0 & 1 & 2 & 3 & 4 \\
\bullet & \bullet & a & b & a \\
\end{array}
\quad \begin{array}{cccccc}
0 & 1 & 2 & 3 & 4 \\
\bullet & \bullet & a & b & a, c \\
\end{array}
\]

\[
w_{\text{time}}^3(L, 4) \subseteq w_{\text{time}}^3(R, 4)
\]

- **Tuple-based** windows are not:

\[
\begin{array}{cccccc}
0 & 1 & 2 & 3 & 4 \\
\bullet & \bullet & \bullet & b & a \\
\end{array}
\quad \begin{array}{cccccc}
0 & 1 & 2 & 3 & 4 \\
\bullet & \bullet & \bullet & b & a, c \\
\end{array}
\]

\[
w_{\text{tuple}}^3(L, 4) \not\subseteq w_{\text{tuple}}^3(R, 4)
\]
Results

- **Semantic characterizations** of equivalences by means of models in Bi-LARS (left/right stream)
  - Monotone fragment: Generalization of logic of Here-and-There
  - Non-monotone fragment more involved
Results

- **Semantic characterizations** of equivalences by means of models in Bi-LARS (left/right stream)
  - Monotone fragment: Generalization of logic of Here-and-There
  - Non-monotone fragment more involved

- **Notions of equivalence** for stream reasoning
  - Strong / Uniform / Data Equivalence
Results

- **Semantic characterizations** of equivalences by means of models in Bi-LARS (left/right stream)
  - Monotone fragment: Generalization of logic of Here-and-There
  - Non-monotone fragment more involved

- **Notions of equivalence** for stream reasoning
  - Strong / Uniform / Data Equivalence

- **Complexity** of deciding eq.: similar to ASP (mostly coNP-c.)