LARS: A Logic-Based Framework for Analyzing Reasoning over Streams

Harald Beck Minh Dao-Tran Thomas Eiter

Institut für Informationssysteme, TU Wien

Stream Reasoning Workshop, Vienna, Nov 09, 2015

- Austrian Science Fund (FWF) grant P26471

Motivation

Different communities looked at different aspects

- **Data Management:**
  - stream processing approach
  - continues queries
  - low-level, high rate input data (cross-joins, pattern matching, etc.)
  - windows for partial data snapshots

Motivation

Different communities looked at different aspects

- **Data Management:**
  - stream processing approach
  - continues queries
  - low-level, high rate input data (cross-joins, pattern matching, etc.)
  - windows for partial data snapshots
- **Knowledge Representation and Reasoning:**
  - stream reasoning
  - higher-level, lower rate (scalability!)
  - changing knowledge bases (ontologies, rule bases)

Motivation

Different communities looked at different aspects

- **Data Management:**
  - stream processing approach
  - continues queries
  - low-level, high rate input data (cross-joins, pattern matching, etc.)
  - windows for partial data snapshots
- **Knowledge Representation and Reasoning:**
  - stream reasoning
  - higher-level, lower rate (scalability!)
  - changing knowledge bases (ontologies, rule bases)
- **Semantic Web:**
  - lifting stream data to a semantic level
  - linked stream data (coupling tuples with timestamps)
  - several extensions of SPARQL

Observations

- *Lack of (unified) formal foundations*

  stream processing:
  - often operational semantics; unpredictable
  - systems may give for same query different results

- *Comparisons / benchmarks unsatisfactory*
  - semantics outcome not / weakly addressed (tuple counting)
  - benchmarks geared towards high-frequency / limits
  - no general methods, no reference semantics

- *Advanced features missing*
  - controlled nondeterminism, incomplete information, negation, model generation

Aims

“Towards a Logic-Based Framework for Analyzing Stream Reasoning”

- Stream Reasoning

Aims

"Towards a Logic-Based Framework for Analyzing Stream Reasoning"

- Stream Reasoning: reasoning services on streaming data

Aims

"Towards a Logic-Based Framework for Analyzing Stream Reasoning"

- Stream Reasoning: reasoning services on streaming data
  - abstraction
  - streams = **tuples** (atoms) associated with **timestamps**
  - essential aspect: **window** functions

Aims

"Towards a Logic-Based Framework for Analyzing Stream Reasoning"

- Stream Reasoning: reasoning services on streaming data
  - abstraction
  - streams = **tuples** (atoms) associated with **timestamps**
  - essential aspect: **window** functions
- Logic-Based

Aims

"Towards a Logic-Based Framework for Analyzing Stream Reasoning"

- Stream Reasoning: reasoning services on streaming data
  - abstraction
  - streams = **tuples** (atoms) associated with **timestamps**
  - essential aspect: **window** functions
- Logic-Based: formal foundations

Aims

"Towards a Logic-Based Framework for Analyzing Stream Reasoning"

- Stream Reasoning: reasoning services on streaming data
  - abstraction
  - streams = **tuples** (atoms) associated with **timestamps**
  - essential aspect: **window** functions
- Logic-Based: formal foundations
- Analysis

Aims

“Towards a Logic-Based Framework for Analyzing Stream Reasoning”

- Stream Reasoning: reasoning services on streaming data
  - abstraction
  - streams = **tuples** (atoms) associated with **timestamps**
  - essential aspect: **window** functions
- Logic-Based: formal foundations
- Analysis: enable prediction, comparison, properties & behaviour assessment

Aims

"Towards a Logic-Based Framework for
Analyzing Stream Reasoning"

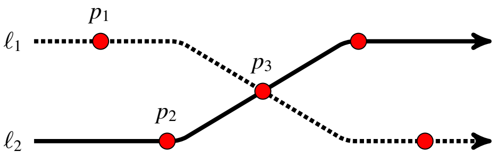
- Stream Reasoning: reasoning services on streaming data
  - abstraction
  - streams = **tuples** (atoms) associated with **timestamps**
  - essential aspect: **window** functions
- Logic-Based: formal foundations
- Analysis: enable prediction, comparison, properties & behaviour assessment

Note: Relates to Complex Event Processing

(e.g., ETALIS [Anicic *et al.*, 2012], RTEC [Artikis *et al.*, 2014])

Example: Public Transportation Monitoring

Samantha with her baby & stroller on public transport, line ℓ_2

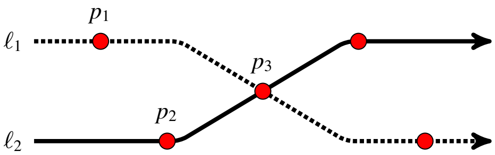


Example: Public Transportation Monitoring

Samantha with her baby & stroller on public transport, line ℓ_2, tram a_2

PLAN			
L	$From$	To	Dur
ℓ_1	p_1	p_3	8
ℓ_2	p_2	p_3	3
...			

LINE	
ID	L
a_1	ℓ_1
a_2	ℓ_2
...	

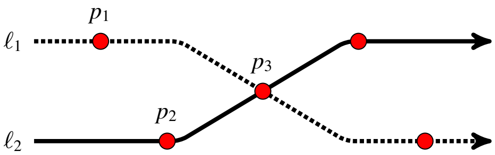


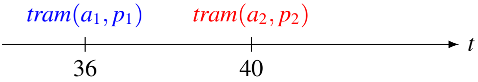
Example: Public Transportation Monitoring

Samantha with her baby & stroller on public transport, line ℓ_2, tram a_2

PLAN			
L	*From*	*To*	*Dur*
ℓ_1	p_1	p_3	8
ℓ_2	p_2	p_3	3
...			

LINE	
ID	*L*
a_1	ℓ_1
a_2	ℓ_2
...	



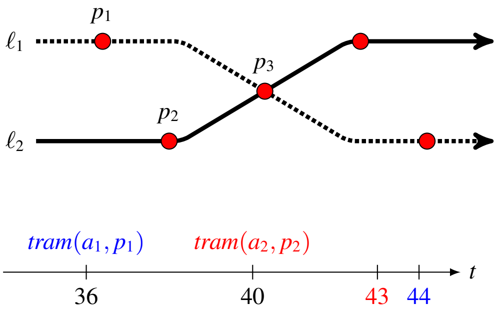


Example: Public Transportation Monitoring

Samantha with her baby & stroller on public transport, line ℓ_2, tram a_2

PLAN			
L	$From$	To	Dur
ℓ_1	p_1	p_3	8
ℓ_2	p_2	p_3	3
...			

LINE	
ID	L
a_1	ℓ_1
a_2	ℓ_2
...	



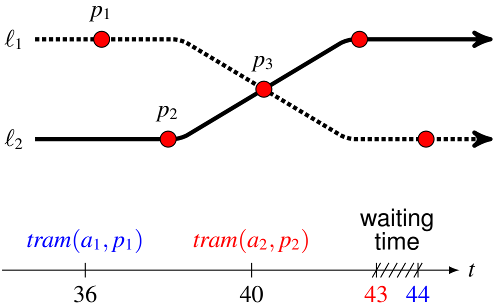
- Report expected arrival times of trams

Example: Public Transportation Monitoring

Samantha with her baby & stroller on public transport, line ℓ_2, tram a_2

PLAN			
L	$From$	To	Dur
ℓ_1	p_1	p_3	8
ℓ_2	p_2	p_3	3
...			

LINE	
ID	L
a_1	ℓ_1
a_2	ℓ_2
...	



- Report expected arrival times of trams
- Report good connections between two lines at a given stop: ≤ 5 mins waiting

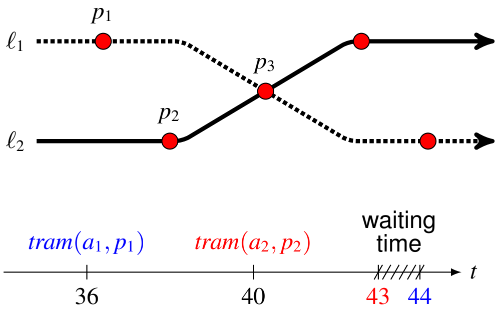
Example: Public Transportation Monitoring

Samantha with her baby & stroller on public transport, line ℓ_2, tram a_2

PLAN			
L	*From*	*To*	*Dur*
ℓ_1	p_1	p_3	8
ℓ_2	p_2	p_3	3
...			

LINE	
ID	*L*
a_1	ℓ_1
a_2	ℓ_2
...	

OLD
a_2
...



- Report expected arrival times of trams
- Report good connections between two lines at a given stop: ≤ 5 mins waiting, no old tram

Example: Public Transportation Monitoring

Samantha with her baby & stroller on public transport, line ℓ_2, tram a_2

PLAN			
L	*From*	*To*	*Dur*
ℓ_1	p_1	p_3	8
ℓ_2	p_2	p_3	3
...			

LINE	
ID	*L*
a_1	ℓ_1
a_2	ℓ_2
...	

OLD
a_2
...

- Report expected arrival times of trams
- Report good connections between two lines at a given stop: ≤ 5 mins waiting, no old tram

Stream Model

- Model streams in a simple way
- Elements:
  - data items: set $\mathcal{A}$ of facts (atoms)
  - timeline: interval $T = [l, u] = \{l, l+1, \ldots, u\} \subseteq 2^{\mathbb{N}_0}$ of integers

Stream Model

- Model streams in a simple way
- Elements:
  - data items: set $\mathcal{A}$ of facts (atoms)
  - timeline: interval $T = [l, u] = \{l, l{+}1, \ldots, u\} \subseteq 2^{\mathbb{N}_0}$ of integers
- A stream is a pair $S = (T, \upsilon)$ of a timeline T and a mapping $\upsilon : T \to 2^{\mathcal{A}}$.

Stream Model

- Model streams in a simple way
- Elements:
  - data items: set $\mathcal{A}$ of facts (atoms)
  - timeline: interval $T = [l, u] = \{l, l+1, \ldots, u\} \subseteq 2^{\mathbb{N}_0}$ of integers
- A stream is a pair $S = (T, \upsilon)$ of a timeline T and a mapping $\upsilon : T \to 2^{\mathcal{A}}$.
- A substream of a stream $S = (T, \upsilon)$ is a stream $S' = (T', \upsilon')$ such that $T' \subseteq T$ and $\upsilon'(t) \subseteq \upsilon(t)$, for all $t \in T'$ (written $S' \subseteq S$).

Stream Model

- Model streams in a simple way
- Elements:
  - data items: set $\mathcal{A}$ of facts (atoms)
  - timeline: interval $T = [l, u] = \{l, l+1, \ldots, u\} \subseteq 2^{\mathbb{N}_0}$ of integers
- A stream is a pair $S = (T, \upsilon)$ of a timeline T and a mapping $\upsilon : T \to 2^{\mathcal{A}}$.
- A substream of a stream $S = (T, \upsilon)$ is a stream $S' = (T', \upsilon')$ such that $T' \subseteq T$ and $\upsilon'(t) \subseteq \upsilon(t)$, for all $t \in T'$ (written $S' \subseteq S$).

Example:

- stream $T = [0, 50]$, $\upsilon = \{36 \mapsto \{tram(a_1, p_1)\}, 40 \mapsto \{tram(a_2, p_2)\}\}$
- substream $T' = [30, 40]$, $\upsilon' = \{36 \mapsto \{tram(a_1, p_1)\}\}$

Window Functions

- Important aspect of stream processing: use only *window* view of data, i.e., limited observability at each point in time
- Different types of windows:
  - time-based windows (within time bounds)
  - tuple-based windows (number of tuples, count)
  - partition-based windows (split input data, process separately)
  - orthogonal, sliding or tumbling (consider atom repeatedly / once)

Window Functions

- Important aspect of stream processing: use only *window* view of data, i.e., limited observability at each point in time
- Different types of windows:
  - time-based windows (within time bounds)
  - tuple-based windows (number of tuples, count)
  - partition-based windows (split input data, process separately)
  - orthogonal, sliding or tumbling (consider atom repeatedly / once)
- Model windows abstractly as functions

$$w_\iota : (S, t, \mathbf{x}) \mapsto S'$$

assigning each stream $S = (T, \upsilon)$ and time point $t \in T$ depending on parameters $\mathbf{x}$ for window type ι a substream S' of S.

LARS: Syntax

$$\alpha \quad ::=$$

LARS: Syntax

$$\alpha \quad ::= \quad a \qquad\qquad \text{where } a \in \mathcal{A} \text{ (prop. atoms)}$$

LARS: Syntax

$$\alpha ::= a \qquad \text{where } a \in \mathcal{A} \text{ (prop. atoms)}$$
$$\mid \neg\alpha \mid \alpha \land \alpha \mid \alpha \lor \alpha \mid \alpha \to \alpha$$

LARS: Syntax

$$\begin{array}{lll} \alpha & ::= & a & \text{where } a \in \mathcal{A} \text{ (prop. atoms)} \\ & & \mid \neg\alpha \mid \alpha \land \alpha \mid \alpha \lor \alpha \mid \alpha \rightarrow \alpha & \\ & & \mid \Diamond\alpha \mid \Box\alpha \mid @_t\alpha & \text{(temporal modalities)} \end{array}$$

- Various ways for time references

LARS: Syntax

$$\begin{array}{lll} \alpha & ::= & a \qquad\qquad\qquad\qquad \text{where } a \in \mathcal{A} \text{ (prop. atoms)} \\ & & \mid \neg\alpha \mid \alpha \wedge \alpha \mid \alpha \vee \alpha \mid \alpha \rightarrow \alpha \\ & & \mid \Diamond\alpha \mid \Box\alpha \mid @_t\alpha \qquad\qquad \text{(temporal modalities)} \\ & & \mid \boxplus^{\mathbf{x}}_{\iota,ch}\alpha \qquad\qquad \text{(window operators)} \end{array}$$

- Various ways for time references
- Nesting of window operators:

$$\boxplus^{60}_{\tau}\Box\,\boxplus^{5}_{\tau}\,\Diamond tramAt(p_1)$$

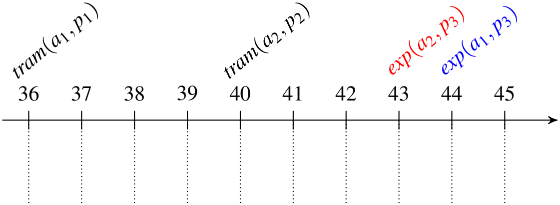
$$\boxplus^{n}_{\#}\,\boxplus^{5}_{\tau}\,\Diamond tramAt(p_1)$$

$$\boxplus^{5}_{\tau}\,\boxplus^{n}_{\#}\,\Diamond tramAt(p_1)$$

Semantics: Evaluation in Stream Context

- Structure: tuple $M = \langle T^\star, \upsilon^\star, W, B \rangle$ where
  - $S^\star = (T^\star, \upsilon^\star)$ is a stream
  - $W = W_1, W_2, \ldots$ are window functions
  - $B \subseteq \mathcal{A}$ is (static, fixed) background knowledge

- Entailment: $M, S^\star, t \Vdash \alpha\,?$ for query α and stream $S^\star$

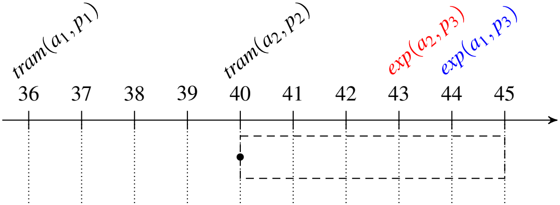


$$M, S^\star, 40 \Vdash \boxplus_\tau^{+5} \Diamond exp(a_1, p_3)\ ?$$

Semantics: Evaluation in Stream Context

- Structure: tuple $M = \langle T^\star, \upsilon^\star, W, B \rangle$ where
  - $S^\star = (T^\star, \upsilon^\star)$ is a stream
  - $W = W_1, W_2, \ldots$ are window functions
  - $B \subseteq \mathcal{A}$ is (static, fixed) background knowledge

- Entailment: $M, S^\star, t \Vdash \alpha\,?$ for query α and stream $S^\star$



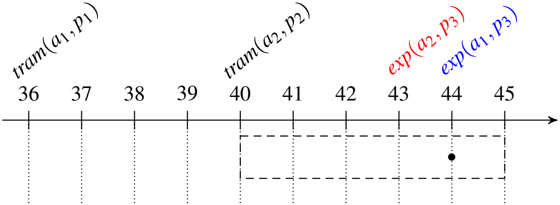
$$M, S^\star, 40 \Vdash \boxplus_\tau^{+5} \Diamond exp(a_1, p_3)\ ?$$

$$M, S, 40 \Vdash \Diamond exp(a_1, p_3)\ ?$$

Semantics: Evaluation in Stream Context

- **Structure**: tuple $M = \langle T^\star, \upsilon^\star, W, B \rangle$ where
  - $S^\star = (T^\star, \upsilon^\star)$ is a stream
  - $W = W_1, W_2, \ldots$ are window functions
  - $B \subseteq \mathcal{A}$ is (static, fixed) background knowledge

- **Entailment**: $M, S^\star, t \Vdash \alpha\,?$ for query α and stream $S^\star$



$$M, S^\star, 40 \Vdash \boxplus_\tau^{+5} \Diamond exp(a_1, p_3)\ ?$$

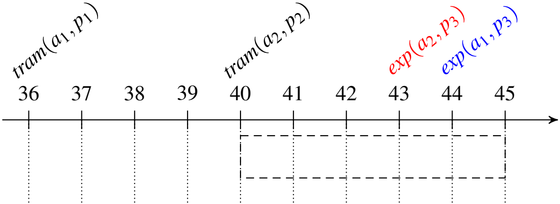
$$M, S, 40 \Vdash \Diamond exp(a_1, p_3)\ ?$$

$$M, S, 44 \Vdash exp(a_1, p_3)$$

Semantics: Evaluation in Stream Context

- Structure: tuple $M = \langle T^\star, \upsilon^\star, W, B\rangle$ where
  - $S^\star = (T^\star, \upsilon^\star)$ is a stream
  - $W = W_1, W_2, \ldots$ are window functions
  - $B \subseteq \mathcal{A}$ is (static, fixed) background knowledge

- Entailment: $M, S^\star, t \Vdash \alpha\,?$ for query α and stream $S^\star$



$$M, S^\star, 40 \Vdash \boxplus_\tau^{+5} \Diamond exp(a_1, p_3)\ ?$$

$$M, S, 40 \Vdash \Diamond exp(a_1, p_3)\ ?$$

$$M, S, 44 \Vdash exp(a_1, p_3)\ \checkmark$$

LARS Rules

Intensional part: define data using rules

- Allows to define auxiliary data, output (result) data
- Use datalog-style rules (i.e., variables and grounding)

$$
\begin{aligned}
tramAt(P) &\leftarrow tram(ID, P). \\
gc(ID_1, ID_2, P) &\leftarrow @_T exp(ID_1, P), \\
&\quad\;\; @_T \boxplus_\tau^{+5} \Diamond exp(ID_2, P), \\
&\quad\;\; \text{not } old(ID_2).
\end{aligned}
$$

- Negation: extend answer set semantics: *answer streams*
- Inherit properties of stable / stratified semantics
  - minimality (pointwise $\subseteq$)
  - supportedness
- Exploit ASP features (nondeterminism, preferences & recursion)

Capturing CQL Queries

- A set Q of CQL queries can be captured via a translation $\Delta(P)$ into a LARS program:

> The result of Q on input streams $\mathbb{S}$ at time t corresponds to the (unique) answer stream of $\Delta(P)$ on stream $\triangledown(\mathbb{S})$ at time t.

- **Translation Idea**
  - stream-to-relation
    - Δ_{SRC} translating source (input streams) to table names, using $\boxplus$ and $\Diamond$ and formulas as names
  - for *relation-to-relation* operators:
    - Δ_ρ translating SQL to relational algebra
    - Δ_δ translating relational algebra to datalog
  - for *relation-to-stream* operators:
    - rule heads produce output streams
  - $\Delta(Q) = \bigcup_{q \in Q} \Delta(q)$, where $\Delta(q) = \Delta_\delta(\Delta_\rho(\Delta_{\text{SRC}}(q)))$

ETALIS Encoding

- ETALIS: rules for complex event processing $a \leftarrow x \text{ SEQ } y$
- interval semantics: $\mathcal{I}(a) = \{\langle t_1, t_4\rangle \mid \langle t_1, t_2\rangle \in \mathcal{I}(x), \langle t_3, t_4\rangle \in \mathcal{I}(y), t_2 < t_3\}$

LARS answer streams can capture ETALIS models for integer timelines, if each atom's intervals do not overlap or meet

Translation idea:

- Encode intervals by serial time points labeled with atom
- Window operator $\boxplus^{[\ell,u]}$: selects substream of interval $[\ell, u]$
- Use syntactic sugar:
  - $[\![\ell, u]\!]$: select interval from ℓ to u
  - $\langle\!\langle \ell, u\rangle\!\rangle\alpha$: holds if $[\ell, u]$ is a *maximal* interval where α holds
- Example: $[\![t_1, t_4]\!]\Box\, a \;\leftarrow\; \langle\!\langle t_1, t_2\rangle\!\rangle\Box\, x, \langle\!\langle t_3, t_4\rangle\!\rangle\Box\, y, t_2 < t_3.$

Further and Ongoing Work

- Complexity analysis
- Modeling of streaming processing / reasoning approaches
  - e.g. when do C-SPARQL (pull) and CQELS (push) coincide?
- Algorithms and implementation
  - answer update (incremental methods)
  - optimization (equivalence)
- Distributed streams (multi-context systems)
- Applications (with KLU, NUI Galway, Siemens)

see `http://www.kr.tuwien.ac.at/research/projects/dhsr/`

References I

Darko Anicic, Paul Fodor, Sebastian Rudolph, and Nenad Stojanovic.
EP-SPARQL: a unified language for event processing and stream reasoning.
In *WWW*, pages 635–644, 2011.

Darko Anicic, Sebastian Rudolph, Paul Fodor, and Nenad Stojanovic.
Stream reasoning and complex event processing in ETALIS.
Semantic Web Journal, 2012.

Alexander Artikis, Marek Sergot, and Georgios Paliouras.
An event calculus for event recognition.
IEEE Trans. Knowl. Data Eng., 2014.

Davide Francesco Barbieri, Daniele Braga, Stefano Ceri, Emanuele Della Valle, and Michael Grossniklaus.
C-SPARQL: a continuous query language for rdf data streams.
Int. J. Semantic Computing, 4(1):3–25, 2010.

References II

- Davide Francesco Barbieri, Daniele Braga, Stefano Ceri, Emanuele Della Valle, and Michael Grossniklaus.
  Incremental reasoning on streams and rich background knowledge.
  In Lora Aroyo, Grigoris Antoniou, Eero Hyvönen, Annette ten Teije, Heiner Stuckenschmidt, Liliana Cabral, and Tania Tudorache, editors, *The Semantic Web: Research and Applications, 7th Extended Semantic Web Conference, ESWC 2010, Heraklion, Crete, Greece, May 30 - June 3, 2010, Proceedings, Part I*, volume 6088 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2010.

- Harald Beck, Minh Dao-Tran, Thomas Eiter, and Michael Fink.
  Towards a logic-based framework for analyzing stream reasoning.
  In Irene Celino, Oscar Corcho, Daniele Dell'Aglio, Emanuele Della Valle, Markus Krötzsch, and Stefan Schlobach, editors, *Proceedings 3rd International Workshop on Ordering and Reasoning (Ordring 2014), October 19-20, 2014 Riva del Garda, Trentino, Italy*, number 1303 in CEUR Workshop Proceedings, pages 11–22. CEUR-WS.org, 2014.
  Online `http://ceur-ws.org/Vol-1303/`.

References III

Harald Beck, Minh Dao-Tran, and Thomas Eiter.
Answer update for rule-based stream reasoning.
In Q. Yang and M. Wooldridge, editors, *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI-15), July 25-31, 2015, Buenos Aires, Argentina*, pages 2741–2747. AAAI Press/IJCAI, 2015.

Harald Beck, Minh Dao-Tran, and Thomas Eiter.
Contrasting RDF stream processing semantics.
In *Proc. 5th Joint International Semantic Technology Conference (JIST 2015), November 11-13, 2015, Yichang, China*, 2015.
To appear.

Harald Beck, Minh Dao-Tran, Thomas Eiter, and Michael Fink.
LARS: A logic-based framework for analyzing reasoning over streams.
In Blai Bonet and Sven Koenig, editors, *Proceedings 29th Conference on Artificial Intelligence (AAAI '15), January 25-30, 2015, Austin, Texas, USA*, pages 1431–1438. AAAI Press, 2015.

References IV

Jean-Paul Calbimonte, Óscar Corcho, and Alasdair J. G. Gray.
Enabling ontology-based access to streaming data sources.
In *ISWC (1)*, pages 96–111, 2010.

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

Martin Gebser, Torsten Grote, Roland Kaminski, and Torsten Schaub.
Reactive answer set programming.
In *LPNMR*, pages 54–66, 2011.

Bernardo Cuenca Grau, Christian Halaschek-Wiener, Yevgeny Kazakov, and Boontawee Suntisrivaraporn.
Incremental classification of description logics ontologies.
J. Autom. Reasoning, 44(4):337–369, 2010.

References V

- Danh Le Phuoc, Minh Dao-Tran, Josiane Xavier Parreira, and Manfred Hauswirth.
  A native and adaptive approach for unified processing of linked streams and linked data.
  In *ISWC (1)*, pages 370–388, 2011.

- Danh Le Phuoc, Josiane Xavier Parreira, and Manfred Hauswirth.
  Linked stream data processing.
  In *Reasoning Web*, 2012.

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

- Raphael Volz, Steffen Staab, and Boris Motik.
  Incrementally maintaining materializations of ontologies stored in logic databases.
  J. Data Semantics, 2:1–34, 2005.

References VI

Carlo Zaniolo.
Logical foundations of continuous query languages for data streams.
In *Datalog*, pages 177–189, 2012.

Related Work

- *SPARQL streaming*
  - CQELS [Phuoc *et al.*, 2011]
  - C-SPARQL [Barbieri *et al.*, 2010a]
  - EP-SPARQL [Anicic *et al.*, 2011]
  - SPARQLstream [Calbimonte *et al.*, 2010]

  snapshot approaches; advanced extensions (non-mononicity, nondeterminism, missing data) difficult

- *ASP / Datalog*
  - periodic ASP solver calls [Do *et al.*, 2011]
  - time-decaying logic programs [Gebser *et al.*, 2011]
  - StreamLog [Zaniolo, 2012]: stratified datalog extension, using "progressive" CWA to "unblock" query evaluation; no windows

- *Ontology Streams:* incremental reasoning
  - materialization [Volz *et al.*, 2005], [Barbieri *et al.*, 2010b]
  - OSMS [Ren and Pan, 2011]: ontology streams with new, obsolete and invariant axioms
  - classification [Grau *et al.*, 2010]