KI - Künstliche Intelligenz (2018) 32:193–195
https://doi.org/10.1007/s13218-018-0537-9

SYSTEMS DESCRIPTION

Stream Reasoning with LARS

Harald Beck[1] · Minh Dao-Tran[1] · Thomas Eiter[1] · Christian Folie[1]

Published online: 17 May 2018

Abstract
Stream reasoning is the task of continuously deriving conclusions on streaming data. Different research communities emphasize different aspects such as throughput vs. expressiveness, yet a mathematical model to describe the declarative semantics of such systems has been missing. This motivated the logic-based framework LARS for analytic reasoning over streams. However, it is also attractive for applications by itself.

Keywords Stream reasoning · Incremental reasoning

1 Introduction: Stream Reasoning

Stream reasoning [7, 10] emerged from stream processing [2] to add logic-oriented features on top of processing continuously changing data. The approach of the influential continuous query language (CQL) [1], which can be seen as extension of SQL for streams, is to first obtain *windows*, i.e., recent snapshots of streaming data, and then execute queries as on a database. This principle has been adopted, e.g., in the Semantic Web area to extend static queries for streams [3, 12]. In Knowledge Representation and Reasoning (KR) some recent works (e.g. [9, 11]) have addressed incremental reasoning. However, prior to our work, no language extension of Answer Set Programming (ASP; see this issue) for stream reasoning with windows was proposed.

2 The LARS Framework

In contrast to temporal reasoning as in Linear Temporal Logic (LTL), which deals with reasoning about infinite sequences of states, LARS [5] places finite streams and windows at the conceptual core. We model a *stream* S as a pair (T, v) of a *timeline* T, which is a closed interval in the natural numbers, and an *evaluation function* v that maps each $t \in T$ to a (possibly empty) set of atoms, i.e., expressions of form $p(c_1, \dots, c_n)$ where p is a predicate and $c_1, \dots, c_n$ are constants ($n \geq 0$). A *window function* w takes a stream S and a *time point* $t \in \mathbb{N}$ and returns a substream $w(S, t)$ of S called a *window*. For instance, consider a stream $S = ([0, 500], v)$, where $v(485) = v(490) = \{x\}$ and $v(t) = \emptyset$ for all other time points t. A (sliding) *time-based window* of size 10, applied on S at time 500 yields the window $([490, 500], v)$, which is also obtained by a *tuple-based window* of size 1.

LARS formulas extend propositional logic, in particular by *window operators* $\boxplus^w$, where w is a generic window function. A formula φ is evaluated on a stream S at a time t, where $\boxplus^w \varphi$ means that the evaluation of φ is restricted to the window $w(S, t)$. In contrast to a CQL-style snapshot semantics that drops timestamps of selected data, LARS offers some control: $\Diamond\varphi$ (resp. $\Box\varphi$) holds if φ holds at some (resp. every) time point, and $@_t\varphi$ holds if φ holds at time t in the window.

LARS programs are sets of rules $\varphi_0 \leftarrow \varphi_1, \dots, \varphi_n$, where each φ_i is a formula; variables are viewed as schematic placeholder for concrete (ground) values. The following program P describes a simple cooling system that must keep the

✉ Harald Beck
beck@kr.tuwien.ac.at

Minh Dao-Tran
dao@kr.tuwien.ac.at

Thomas Eiter
eiter@kr.tuwien.ac.at

Christian Folie
christian.folie@outlook.com

[1] Institute of Logic and Computation, TU Wien, Favoritenstraße 9-11, 1040 Vienna, Austria

temperature below 7 degrees, where atom $s(V)$ stands for a temperature with value V.

$$@_T\, high \leftarrow \boxplus^{60sec} @_T\, s(V), V > 7$$

$$cool \leftarrow \boxplus^{60sec} \Box\, high$$

$$warn \leftarrow \boxplus^{\#5} \Diamond s(V), V > 12$$

Rule (1) abstracts away the specific value: for every second T during the last 60, we associate atom *high* with T if there is a temperature signal $s(V)$ with $V > 7$. Rule (2) then triggers cooling if the temperature has always been high in that interval. Finally, Rule (3) infers a warning if one of the last 5 signal values was above 12.

Semantically, LARS programs can be seen as extension of ASP for streams, thus carrying over attractive features like minimal model reasoning, nonmonotonicity and recursion. The closed world assumption (as in databases) allows for reasoning based on the absence of data; in general, the stable model semantics then leads to multiple models. This makes LARS applicable for scenarios which require the generation of alternative solutions like in planning, diagnosis or configuration.

Satisfiability and model checking are in ground LARS PSPACE-complete (for tractable window functions), but in practical settings not harder than in ground ASP. In contrast to LTL, ground LARS programs with time windows yield the full class of regular languages; nonground programs can express also intractable languages.

3 Implementations and Incremental Reasoning

Besides foundational aspects beyond the scope of this note, we developed *Ticker* [6], a prototypical stream reasoning engine that comes with two reasoning modes for LARS programs with the above sliding windows.[1]

The first mode repeatedly calls the ASP solver Clingo [8] based on a translation. The second, more efficient mode is incremental; it uses truth maintenance systems to adjust the model of a dynamically updated ASP encoding. In Ticker syntax, the above program P reads:

```
@T high :- @T s(V) [60 sec], V> 7.
cool :- always high [60 sec].
warn :- s(V) [5 #], V> 12.
```

The *Laser* engine [4] is geared for programs with unique models to achieve high performance, by managing incrementally substitution tables using temporal information. Laser was shown to be significantly faster than C-SPARQL, CQELS or Ticker/Clingo in comparable operations, using Python with PyPy 5.8.[2]

4 Conclusion

In conclusion, LARS is a framework that extends ASP for stream reasoning with an emphasis on generic windows. Algorithmically, frequently changing data calls for incremental reasoning, as exemplified in the prototype engines Ticker and Laser. Further and future work includes the use of LARS in applications like network management and dynamic reconfiguration.

Funding Funding was supported by FWF Grants P26471, P27730, and W1255-N23. Open access funding provided by Austrian Science Fund (FWF).

References

1. Arasu A, Babu S, Widom J (2006) The CQL continuous query language: semantic foundations and query execution. VLDB J. 15(2):121–142
2. Babu S, Widom J (2001) Continuous queries over data streams. SIGMOD Record 3(30):109–120
3. Barbieri D, Braga D, Ceri S, Della Valle E, Grossniklaus M (2010) C-SPARQL: a continuous query language for RDF data streams. Int J Semantic Comput 4(1):3–25
4. Bazoobandi HR, Beck H, Urbani J (2017) Expressive stream reasoning with Laser. In: Proceeding ISWC
5. Beck H, Dao-Tran M, Eiter T, Fink M (2015) LARS: A logic-based framework for analyzing reasoning over streams. In: Proceeding AAAI, pp 1431–1438. AAAI Press
6. Beck H, Eiter T, Folie C (2017) Ticker: a system for incremental ASP-based stream reasoning. TPLP 17(5–6):744–763
7. Della Valle E, Ceri S, van Harmelen F, Fensel D (2009) It's a streaming world! Reasoning upon rapidly changing information. IEEE Intell Syst 24:83–89
8. Gebser M, Kaminski R, Kaufmann B, Schaub T (2014) Clingo = ASP + control: Preliminary report. TPLP, online supplement; also CoRR, abs/1405.3694
9. Gebser M, Kaminski R, Obermeier P, Schaub T (2015) Ricochet robots reloaded: A case-study in multi-shot ASP solving. In: LNCS 9060, Springer , pp. 17–32
10. Mileo A, Dao-Tran M, Eiter T, Fink M (2018) Stream reasoning. In: Liu L, Özsu MT (eds) Encyclopedia of database systems, 2nd (edn), Springer, New York (**ISBN 978-1-4614-8266-6**)

[1] Ticker source: https://github.com/hbeck/ticker.

[2] Laser source: https://github.com/karmaresearch/laser.

11. Motik B, Nenov Y, Piro R, Horrocks I (2015) Incremental update of datalog materialisation: the backward/forward algorithm. In: Proceeding AAAI, AAAI Press , pp 1560–1568
12. Phuoc DL, Dao-Tran M, Parreira JX, Hauswirth M (2011) A native and adaptive approach for unified processing of linked streams and linked data. In: Proceeding ISWC