Distributed Stream Reasoning with LARS and ASP (Extended Abstract*)

Thomas Eiter[1], Paul Ogris[2], and Konstantin Schekotihin[2]

[1] Vienna University of Technology, Austria
eiter@kr.tuwien.ac.at
[2] Alpen-Adria-Universität Klagenfurt, Austria
firstname.lastname@aau.at

Various applications in emerging domains, such as Cyber-Physical Systems (CPS), Industry Digitalization, or Internet of Things, require complex monitoring and decision-making over streams of data. For instance, in CPS quite often sensors provide data about the environment and decision-making components should use it to timely detect environment changes and determine an appropriate reaction of the system. The Logic-based framework for Analytic Reasoning over Streams (LARS) [3] is a *stream reasoning* paradigm that extends Answer Set Programming (ASP) with window operators and time modalities to declaratively specify complex decision problems on streaming data. However, reasoning in presence of continuously changing data is a challenging problem since a model, returned by a stream reasoner for the previous portion of data, must be updated as soon as possible in order to ensure an acceptable latency/throughput of a streaming system. Recently a number of *specific stream reasoners* supporting LARS, e.g., Ticker [4] or Laser [1], and ASP solvers supporting streaming features, e.g., [6], have been proposed and successfully applied in various scenarios.

Existing LARS reasoners employ clever strategies, such as justification-based truth maintenance in Ticker or semi-naive evaluation of Datalog programs in Laser, to efficiently update their models. As a consequence, however, such reasoners can only support a subset of the LARS language. Ticker for example cannot evaluate programs with constraints or odd loops and Laser accepts only the positive fragment of LARS. In this paper, we suggest a novel approach to stream reasoning employing a distributed architecture that *(i)* allows for a better performance than specific reasoners supporting the same LARS subset, and *(ii)* supports all features of the LARS language including integrity constraints.

Interval Semantics. The original LARS semantics is defined over streams in which every time point t is associated with a set of ground atoms that occurred in t [3]. Following this notion, in a distributed system at every t a set of corresponding atoms must be communicated to its components, even if it did not change. Such communication might cause a significant load on the networking infrastructure and thus decrease the overall performance of the system. It is therefore more economical to report only changes of the atom occurrence. Furthermore, occurrences may be reported by different observers and appear in a

* This was previously published in [5] and conducted supported by Austrian Federal Ministry of Transport, Innovation and Technology (BMVIT) *(FFG-PNr.: 861263)*.

single stream. We thus adopt in this work a new view of LARS semantics based on *interval streams* that associates with every ground atom a set of time intervals in which it occurred. This change allows stream reasoners to exchange only changes of atom occurrences, thus, reducing the overall load on the network.

Distributed Stream Reasoning. Distributed reasoning for LARS is based on the fact that the evaluation of streaming atoms over different substreams might be performed sequentially by interconnected stream reasoners. The idea of *stream stratification* was described in [3] and aims at splitting LARS programs along the stream into layers such that computation of window functions in each layer depends only on evaluation results of previous layers. Obviously, this is possible if the input program has no recursion through window operators. Given a set of layers a distributed reasoning system can generate a network of components which can process layer separately and use all possibilities of modern multicore processors and server clusters. Computation on stream strata is based on the *stream dependency graph* (SDG) [3] of a LARS program which allows to detect recursion over window operators and computation of stream strata. In this work, we extend the original concept by including negation as a dependency, intuitively, to ensure that choices are made before (or when) they are used. SDG of a LARS program is used to define a *component graph* (CG), where a stream reasoner is running in every node of the graph. Each reasoner takes inputs from the network as defined by its incoming edges in CG and distributes its answer stream into the network, along its outgoing edges in the graph.

The evaluation results indicate that the distributed system outperforms existing stream reasoners for the same LARS fragment on a realistic *monitoring problem* that was used in the previous work [3]. In addition, the scalability of the novel system was shown by a benchmark representing m chained *n-Queens completion problems* where parts of a solution for a previous problem were provided to the next one. The suggested approach could solve up to 10 instances of 6 chained 18-Queens per second, which is impossible for previous LARS reasoners.

References

1. Bazoobandi, H.R., Beck, H., Urbani, J.: Expressive stream reasoning with laser. In: International Semantic Web Conference (1). Lecture Notes in Computer Science, vol. 10587, pp. 87–103. Springer (2017)
2. Beck, H., Dao-Tran, M., Eiter, T.: Answer update for rule-based stream reasoning. In: IJCAI. pp. 2741–2747. AAAI Press (2015)
3. Beck, H., Dao-Tran, M., Eiter, T.: LARS: A logic-based framework for analytic reasoning over streams. Artif. Intell. **261**, 16–70 (2018)
4. Beck, H., Eiter, T., Folie, C.: Ticker: A system for incremental asp-based stream reasoning. TPLP **17**(5-6), 744–763 (2017)
5. Eiter, T., Ogris, P., Schekotihin, K.: A distributed approach to LARS stream reasoning (system paper). TPLP **19**(5-6), 974–989 (2019)
6. Gebser, M., Kaminski, R., Kaufmann, B., Schaub, T.: Clingo = ASP + control: Preliminary report. CoRR **abs/1405.3694** (2014)