Answer Set Automata: A Learnable Pattern Specification Framework for Complex Event Recognition

Nikos Katzouris
Institute of Informatics, National Center for Scientific Research "Demokritos", Athens, Greece

Georgios Paliouras
Institute of Informatics, National Center for Scientific Research "Demokritos", Athens, Greece

Abstract

Complex Event Recognition (CER) systems detect event occurrences in streaming input using predefined event patterns. Techniques that learn event patterns from data are highly desirable in CER. Since such patterns are typically represented by symbolic automata, we propose a family of such automata where the transition-enabling conditions are defined by Answer Set Programming (ASP) rules, and which, thanks to the strong connections of ASP to symbolic learning, are learnable from data. We present such a learning approach in ASP, capable of jointly learning the structure of an automaton and its transition guards' definitions from building-block predicates, and a scalable, incremental version thereof that progressively revises models learnt from mini-batches using Monte Carlo Tree Search. We evaluate our approach on three CER datasets and empirically demonstrate its efficacy.

2012 ACM Subject Classification Computing methodologies → Logic programming and answer set programming

Keywords and phrases Event Pattern Learning, Answer Set Programming

Digital Object Identifier 10.4230/LIPIcs.TIME.2023.17

Category Extended Abstract

Funding *Nikos Katzouris*: This work is supported by the project EVENFLOW – "Robust Learning and Reasoning for Complex Event Forecasting", which has received funding from the European Union's Horizon research and innovation programme under grant agreement No 101070430.
Georgios Paliouras: This work is supported by the project EVENFLOW – "Robust Learning and Reasoning for Complex Event Forecasting", which has received funding from the European Union's Horizon research and innovation programme under grant agreement No 101070430.

1 Introduction

Complex Event Recognition (CER) systems [3] detect occurrences of *complex events* (CEs) in streaming input, using temporal patterns consisting of *simple events*, e.g. sensor data, or other complex events. CE patterns are typically defined by domain experts in some *event specification language*. Despite the diversity of such languages and the variety of the proposed event processing operators [4], a minimal set of such operators that every ECL should support [3, 4] includes *sequence* and *iteration (Kleene Closure)*, implying respectively that some particular events should succeed one another temporally, or that an event should occur iteratively in a sequence, and the *filtering* operator, which matches input events that satisfy a set of predefined predicates.

Taken together, these three operators point to a computational model for CER based on symbolic finite automata [1] (SFA), i.e., automata where the transition-enabling conditions are predicates than need to be evaluated against the input, rather than mere symbols. As a

© Nikos Katzouris and Georgios Paliouras;
licensed under Creative Commons License CC-BY 4.0

result, in most existing CER systems CE patterns are either defined directly as SFA, or are compiled into such at runtime. Prominent areas of CER research, then, concern the study of trade-offs between ECLs' expressive power and pattern matching complexity, in addition to more practical considerations, such as scaling-up concrete pattern matching algorithms, or dealing with fault tolerance and distributed processing.

CE pattern learning is a less studied CER topic, which, however, is of utmost importance, since CE patterns are not always known in advance, or they frequently need to be revised. A few learning approaches have been proposed, which have several limitations. Some are restricted to sequence-based ESLs that do support operators such as iteration [7, 5, 2], others do not allow for filtering predicates [6], most offer very limited support for reasoning with background knowledge during pattern induction and none supports CE pattern revision.

To address such issues we propose *answer set automata learning*, a framework that allows to specify SFA-based CE patterns in the form of answer set programs (ASP), which, thanks to the strong connections of ASP to symbolic learning, are directly learnable and revisable from data. We begin by encoding CE patterns specified in any ESL that supports the core CER operators of filtering, sequence, iteration, disjunction and conjunction into *answer set automata*, i.e. executable ASP specifications of the SFA that correspond to the initial CE patterns. Our CE patterns-to-ASP programs translation comes with a correctness property, stating that a pattern will be matched against a particular finite piece of input when run with a CER engine, iff its corresponding answer set automaton program satisfies a particular query, when run on the same input with an ASP solver.

The established connection between event pattern matching and logical reasoning allows to learn the ASP program equivalent of a CE pattern from labeled event traces, via abductive learning w.r.t. to an SFA interpreter. Our learning approach, implemented directly on top of an ASP solver, allows to synthesize patterns utilizing the core CER operators by jointly learning the structure of the corresponding SFA and the definitions of its transition guards, consisting of boolean combinations of building-block, background knowledge predicates.

To scale-up the abductive learning core of our SFA synthesis method to large training sets, we propose an incremental learning technique utilizing SFA revision in a Monte Carlo Tree Search (MCTS) that continuously revises programs learnt from mini-batches of the data, in an effort to approximate a global optimum. The revision operators can modify the structure of an automaton, by adding/removing states and transitions, or the structure of the transition guard rules, by adding/removing conditions from such rules' bodies. These revision operators are realized via same abductive learning technique that handles batch learning, using constraints generated from the labeled traces in each mini-batch to guide the search for optimal "local" revisions. MCTS is used to stochastically search in the massive space of SFA structures, while balancing exploitation, i.e. revising already identified, globally-good SFA, in an effort to further improve their quality, with exploration, i.e. revising less promising SFA, in an effort to escape local optima.

We evaluate our SFA learning approach on three CER datasets and empirically demonstrate its efficacy. We also compare our technique to classical automata learning methods on univariate input and show its superiority, both in terms of predictive accuracy and scalability.

References

1 Loris D'Antoni and Margus Veanes. The power of symbolic automata and transducers. In *International Conference on Computer Aided Verification*, pages 47–67. Springer, 2017.

2 Lars George, Bruno Cadonna, and Matthias Weidlich. Il-miner: instance-level discovery of complex event patterns. *Proceedings of the VLDB Endowment*, 10(1):25–36, 2016.

3 Nikos Giatrakos, Elias Alevizos, Alexander Artikis, Antonios Deligiannakis, and Minos N. Garofalakis. Complex event recognition in the big data era: a survey. *VLDB J.*, 29(1):313–352, 2020.

4 Alejandro Grez, Cristian Riveros, Martín Ugarte, and Stijn Vansummeren. A formal framework for complex event recognition. *ACM Transactions on Database Systems (TODS)*, 46(4):1–49, 2021.

5 Sarah Kleest-Meißner, Rebecca Sattler, Markus L Schmid, Nicole Schweikardt, and Matthias Weidlich. Discovering multi-dimensional subsequence queries from traces–from theory to practice. *BTW 2023*, 2023.

6 Yan Li and Tingjian Ge. Imminence monitoring of critical events: A representation learning approach. In *Proceedings of the 2021 International Conference on Management of Data*, pages 1103–1115, 2021.

7 Alessandro Margara, Gianpaolo Cugola, and Giordano Tamburrelli. Learning from the past: automated rule generation for complex event processing. In *Proceedings of the 8th ACM international conference on distributed event-based systems*, pages 47–58, 2014.