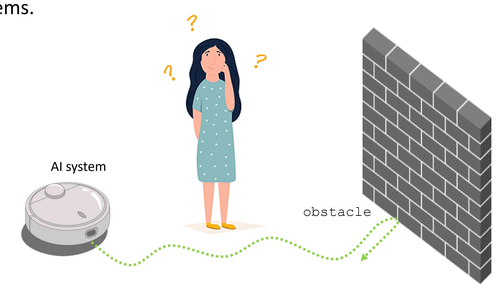
Learning Interpretable Temporal Properties from Positive Examples Only

Rajarshi Roy, Jean-Raphaël Gaglione, Ufuk Topcu, Daniel Neider, Nasim Baharisangari, and Zhe Xu

MAX PLANCK INSTITUTE FOR SOFTWARE SYSTEMS · TEXAS The University of Texas at Austin · dortmund university · UA RUHR CENTER FOR TRUSTWORTHY DATA SCIENCE AND SECURITY RESEARCH ALLIANCE · Arizona State University

Explanation for Complex Systems

Goal: Learn human interpretable models to explain the temporal behavior of AI systems.



Consider the following behavior of an AI-based robot cleaner:

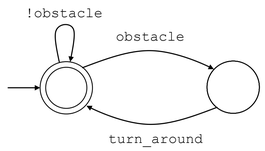
Always, if an `obstacle` *is hit, then* `turn_around` *in the next timestep*

Interpretable Models for Explanation

Deterministic Finite Automata (DFA)

Standard representation for Regular languages

Uses
- atomic propositions
- states
- transitions
- initial state
- final states



Linear Temporal Logic (LTL)

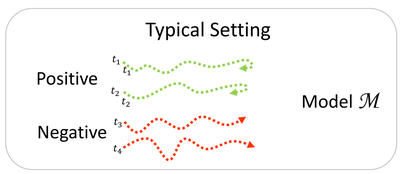
Globally(`obstacle` implies neXt `turn_around`)

Less expressive than Regular languages
Resembles Natural Language

Formulas over
- atomic propositions
- Boolean operators: ∧ (and) ∨ (or) ¬ (not) → (implies)
- temporal operators: G (globally) F (finally) X (next) U (until)

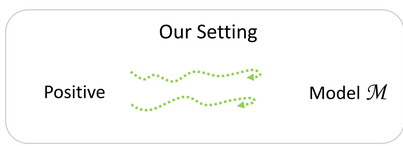
Literature for Learning DFAs and LTL formulas

General Problem: Learn **DFAs** and **LTL formulas** to explain the system trajectories



DFAs
- Biermann and Feldman (IEEE Trans. CS 21, 1972)
- Grinchtein, Leucker, and Piterman (IJCAR, 2006)
- …

LTL formulas
- Neider and Gavran (FMCAD, 2018)
- Camacho and McIlraith (ICAPS, 2019)
- …



DFAs
- Avellaneda and Petrenko (ICGI, 2018)

LTL formulas
No work for the full class of LTL formulas

Why positive examples only?

- Extracting negative examples from a black-box model is often infeasible.
- Generating negative examples in safety critical applications can be risky.

Learning from positive examples is ill-posed

Overgeneralization

The most concise model that accepts positive trajectories is:

DFA

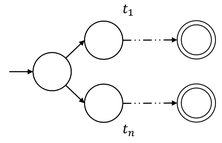
LTL formula $\varphi = \text{true}$

The models are too general!

Overfitting

The strictest formula that accepts the positive trajectories is

DFA



LTL formula $\varphi = \varphi_{t_1} \wedge \cdots \wedge \varphi_{t_n}$

The models are large and too specific!

The learning problem: One Class Classification (OCC)

Positive Trajectories: *Size upper bound:*

Input: P ⋮ K

Problem: Learn a model $\mathcal{M}$ that accepts all trajectories in **P** and

1. has *size* less than **K** (to handle overfitting); and
2. is ***language minimal*** (to handle overgeneralization).

$L(\mathcal{M})$ → the set of accepted trajectories
$\mathcal{M}$ is *language minimal* if for no other $\mathcal{M}'$ that accepts all trajectories in **P** and has size less than **K**, $L(\mathcal{M}') \subset L(\mathcal{M})$

Contributions

For OCC problem for DFAs

Symbolic algorithm
that encodes the search for language minimal DFAs in SAT

For OCC problem for LTL formulas

Counterexample-guided algorithm
that generates negative trajectories to direct the search

Semi-Symbolic algorithm
that combines the symbolic and the counterexample-guided approach

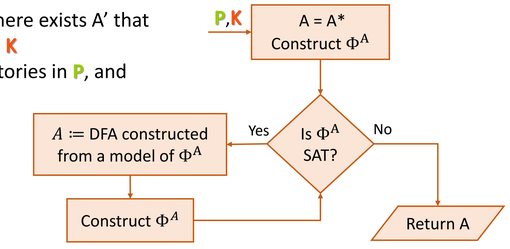
Implemented a **prototype** containing all the algorithms

An **empirical evaluation** showing the ability of our prototype to learn interpretable models

Symbolic Algorithm for DFAs

$L(A^*) \supset L(A_1) \supset L(A_2) \supset \ldots \supset L(A_{sol})$
↑ guess A

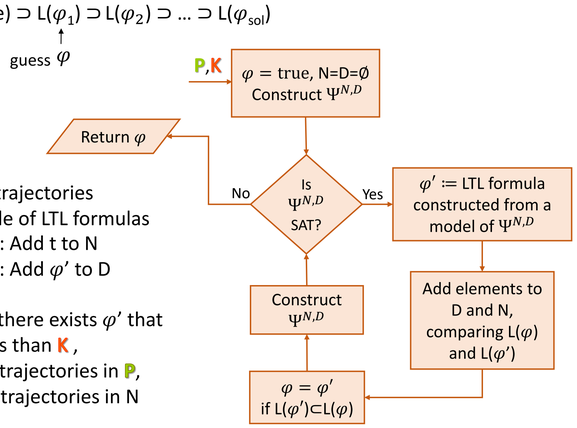
Φ^A is satisfiable iff there exists A' that
- has size less than **K**
- accepts all trajectories in **P**, and
- L(A')⊂L(A)



The symbolic algorithm always terminates and returns a language minimal DFA A

Counterexample-guided Algorithm for LTL formulas

$L(\text{true}) \supset L(\varphi_1) \supset L(\varphi_2) \supset \ldots \supset L(\varphi_{sol})$
↑ guess φ



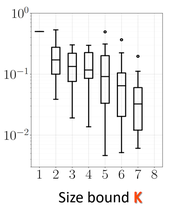
N → negative trajectories
D → discard pile of LTL formulas
- $L(\varphi') \not\subseteq L(\varphi)$: Add t to N
- $L(\varphi') = L(\varphi)$: Add φ' to D

$\Psi^{N,D}$ is SAT iff there exists φ' that
- has size less than **K**,
- accepts all trajectories in **P**,
- rejects the trajectories in N
- is not in D

The counterexample-guided algorithm always terminates and returns a language minimal LTL formula

Evaluation

$\frac{\text{\# of iterations in Symbolic}}{\text{\# of iterations in A\&P algo}}$



Compared against counterexample-guided algorithm for DFAs by Avellaneda and Petrenko (A&P algo)

Obtained at least 10x less number of iterations

More Experiments in the paper!

Conclusion

Considered the **OCC problem for DFAs and LTL formulas.**

Presented **three novel algorithms** for solving OCC problems and implemented them in a prototype