Multi-Models and Multi-Formulas Finite Model Checking for Modal Logic Formulas Induction

Mauro Milella[1], Giovanni Pagliarini[1,2], Andrea Paradiso[1] and Ionel Eduard Stan[1,2,*]

[1]ACLAI Lab., Dept. of Math. and Comp. Sci., University of Ferrara, Italy
[2]Dept. of Math., Phy., and Comp. Sci., University of Parma, Italy

Abstract
Modal symbolic learning is the subfield of artificial intelligence that brings together machine learning and modal logic to design algorithms that extract modal logic theories from data. The generalization of model checking to multi-models and multi-formulas is key for the entire inductive process (with modal logics). We investigate such generalization by, first, pointing out the need for finite model checking in automatic inductive reasoning, and, then, showing how to efficiently solve it. We release an open-source implementation of our simulations.

Keywords
Modal Logic, Machine Learning, Modal Symbolic Learning, Model Checking

1. Introduction

Since the dawn of *artificial intelligence (AI)*, there has been a fundamental separation between two schools of thought: *symbolic AI* and *non-symbolic AI* (e.g., *connectionist AI*). *Machine learning (ML)*, that is, the subfield of AI that designs algorithms to build models from data, encompasses a similar separation: in ML terms, symbolic learning is the process of learning a *logical description* (e.g., a decision tree, or a rule-based classifier) that represents the theory underlying a certain phenomenon, whereas non-symbolic learning is the process of learning a *non-logical description* of the same theory (e.g., a deep neural network, or a naive Bayes classifier).

A recent trend in symbolic ML is *modal symbolic learning*, where the extracted logical descriptions are based on custom-made *modal logic* ($\mathcal{ML}$) formalisms (e.g., temporal or spatial logics, depending on the data at hand). Note that, from the perspective of a logician, symbolic machine learning can be seen as an induction problem, and it often requires checking the truth of many logical formulas, which, in turn, is a well-known problem among $\mathcal{ML}$ theorists referred to as *model checking*. Model checking is the problem of automatically verifying finite state concurrent systems [1] and, usually, model checker algorithms *exhaustively* search through

OVERLAY 2022: 4th Workshop on Artificial Intelligence and Formal Verification, Logic, Automata, and Synthesis, November 28, 2022, Udine, Italy
*Corresponding author.
mauro.milella@edu.unife.it (M. Milella); giovanni.pagliarini@unife.it (G. Pagliarini); andrea.paradiso@edu.unife.it (A. Paradiso); ioneleduard.stan@unife.it (I. E. Stan)
0000-0001-7128-6745 (M. Milella); 0000-0002-8403-3250 (G. Pagliarini); 0000-0002-3614-2487 (A. Paradiso); 0000-0001-9260-102X (I. E. Stan)

the finite state space of the system to assess whether some specification (i.e., property of the system) is true or not.

In this work, we argue that model checking plays a crucial role in the induction of modal formulas, and that an efficient model checking algorithm is essential for modal symbolic learning methods. We show how to extend the typical model checking machinery to check multiple formulas on multiple models, and experimentally show how memoization (i.e. storing model checking results for later reuse) can be leveraged to drastically reduce model checking computational time. Finally, we release an open-source implementation to reproduce the experiments of this work.

2. Model Checking with Memoization for Modal Logic Formulas Induction

Given a set of *propositional letters* $\mathcal{P}$ as the *alphabet*, the well-formed formulas of the *propositional modal logic* ($\mathcal{ML}$) are obtained by the following grammar:

$$\varphi \quad ::= \quad p \mid \neg\varphi \mid \varphi \vee \varphi \mid \Diamond\varphi,$$

where the remaining classic Boolean operators can be obtained as shortcuts. In what follows, we use $\Box\varphi$ to denote $\neg\Diamond\neg\varphi$. The *modality* $\Diamond$ (resp., $\Box$) is usually referred to as *it is possible that* (resp., *it is necessary that*). We refer to $\mathcal{F}$ as the set of formulas produced by the above grammar, and call the *height* of a formula the height of its *syntax tree*.

$\mathcal{ML}$ is paradigmatic of temporal, spatial, and spatio-temporal logics, and it is an extension of *propositional logic* ($\mathcal{PL}$). Its semantics is given in terms of Kripke models. A *Kripke model* $K = (W, R, V)$ is a *Kripke frame* (W, R) composed by a non-empty (possible infinite, but countable) set of *worlds* W (which contains a distinguished world w_0, called *initial world*), a binary *accessibility relation* $R \subseteq W \times W$, and a *valuation function* $V : P \to 2^W$, which associates each world with the set of propositional letters that are true on it. The *truth relation* $K, w \Vdash \varphi$, for a generic (Kripke) model K, a world w (in that model), and a formula φ (to be interpreted on that model), is given by the following clauses:

$$\begin{array}{lll}
K, w \Vdash p & \text{iff} & w \in V(p); \\
K, w \Vdash \neg\psi & \text{iff} & K, w \nVdash \psi \text{ (i.e., it is not the case that } K, w \Vdash \psi); \\
K, w \Vdash \psi_1 \vee \psi_2 & \text{iff} & K, w \Vdash \psi_1 \text{ or } K, w \Vdash \psi_2; \\
K, w \Vdash \Diamond\psi & \text{iff} & \exists w' \text{ s.t. } wRw' \text{ and } K, w' \Vdash \psi.
\end{array}$$

In the following, we use $K \Vdash \varphi$ to denote $K, w_0 \Vdash \varphi$. In modal symbolic learning, Kripke models can be used to represent data such as time series, images, audio, videos, and graphs, which, in the era of big data, accounts for $80\% - 95\%$ of the existing data [2]. This data is commonly referred to as *unstructured*, as it does not have a well-studied *data model*, nor a predefined structure, and it is typically counterposed to *structured*, tabular data, organized in rows and columns, which is found in spreadsheets and relational databases.

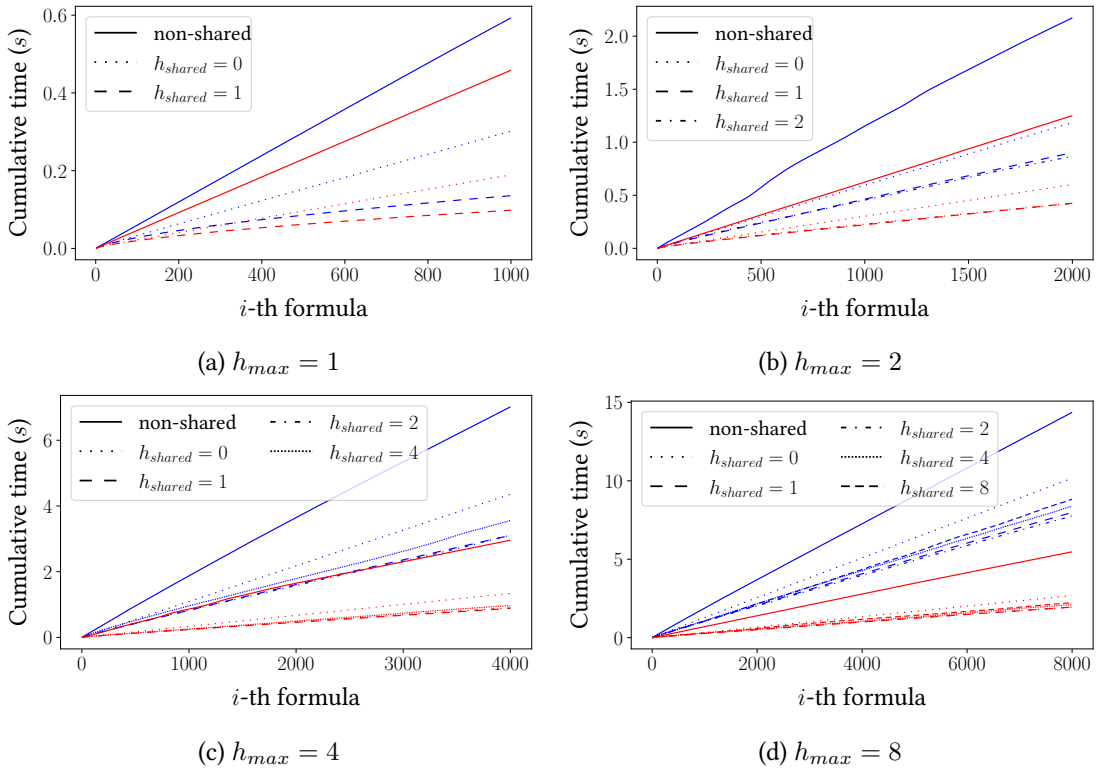
Among the many interesting mathematical problems studied over the years in the field of $\mathcal{ML}$ there is *model checking* [1]. Formally, the model checking problem is the problem of establishing

if $K \Vdash \varphi$, where K is a Kripke model and φ is a formula of $\mathcal{ML}$. Canonically, model checking is the problem of verifying properties of modal temporal logics on *infinite state*, *finitely represented*, abstract models (i.e., Kripke models) of concrete ones (e.g., reactive systems). Depending on the logical formalism, model checking may not be a trivial task [3]. The common denominator of the ML logical approaches (see, e.g., [4, 5, 6, 7, 8, 9, 10, 11, 12]) is that the kind of model checking, which is *key* for the entire learning process, is in fact *finite*, which, to some extent, trivializes the problem itself (in general, it becomes PTIME). Nevertheless, the model checking problem still raises some difficulties, and leaves room for algorithmic optimizations. Finite model checking a single $\mathcal{ML}$ formula on a single model can be achieved by simply adapting the well-known Emerson-Clarke algorithm for checking $\mathcal{CTL}^*$ formulas [13]. This procedure relies on a memoization schema where a structure $\mathcal{H} : \mathcal{F} \to 2^W$ is filled, in a bottom-up fashion, with the truth values of all subformulas on all worlds.

In real-world contexts, it is common to check many formulas on many Kripke models. In fact, modal symbolic learning is an inductive statistical process, that learns a general theory, seen as multiple $\mathcal{ML}$ formulas, from datasets of Kripke models. This sets the stage for the more general problem of finite model checking of multiple models against multiple formulas. Let $\mathcal{K} = \{K_1, \ldots, K_m\}$ be a collection of m Kripke models and $\Phi = \{\varphi_1, \ldots, \varphi_n\}$ be a collection of n $\mathcal{ML}$ formulas, then the *multi-models and multi-formulas finite model checking problem* is the problem of deciding, for all $K \in \mathcal{K}$ and for all $\varphi \in \Phi$, if $K \Vdash \varphi$. The straightforward approach involves calling the single model checking procedure $m \cdot n$ times; however, the memoization results generated by a single call can be reused for a later call on the same model. In fact, the memoization structure for a single model can be shared for checking all formulas; ideally, this reduces the overall computational load, but it requires more memory accesses which, in turn, introduce overhead. This tradeoff can be mitigated by only sharing memoized (sub)formulas with height no greater than a fixed parameter h_{shared}, leveraging the fact that shorter (sub)formulas are more likely to be checked in the future.

We prove and quantify the benefits of a shared memoization structure in an experimental setting. The m Kripke models are generated by fixing the same Kripke frame with 20 worlds, randomly generated using the Fan-in/Fan-out method from [14], and by assigning to each world a random subset of true propositional letters. The n formulas are, first, generated as formulas of a fixed height $h_{\max}$ using a random procedure. On top of this, a pruning strategy is adopted for reducing each formula: in a top-down fashion, each node of the syntax tree is cut with probability pr and substituted with a random propositional letter. As a result, all formulas have maximum height $h_{\max}$, and are generally smaller in size with greater values of pr. We fix an alphabet size of $|\mathcal{P}| = 16$ and $m = 50$ models. Different parametrizations are used for $h_{\max}$ and pr. For each parametrization, $n = 1000 \cdot h_{\max}$ formulas are checked on all models using the *non-shared* memoization approach and the *shared* memoization approach with different values of h_{shared}, with $h_{shared} \leq h_{\max}$. We let $h_{\max} \in \{1, 2, 4, 8\}$, $pr \in \{0.2, 0.5\}$, and $h_{shared} \in \{0, 1, 2, 4, 8\}$; note that when $h_{shared} = 0$, only the results for the propositional letters are shared, and when $h_{shared} = h_{\max}$, those for all subformulas are shared.

The cumulative times of the different approaches are illustrated in Fig. 1. It can be observed that sharing at least the propositional letters is beneficial in all cases; in other words, the shared memoization approach improves over the non-shared one. When comparing different



(a) $h_{max} = 1$ (b) $h_{max} = 2$

(c) $h_{max} = 4$ (d) $h_{max} = 8$

Figure 1: Results of the multi-models and multi-formulas model checking with the non-shared and shared memoization approaches with varying values of $h_{\max}$. Each subfigure shows the cumulative time achieved by the different approaches with $pr = 0.2$ (in blue) and $pr = 0.5$ (in red).

parametrizations of the shared approach, it appears that, within the given experimental setting, $h_{shared} = 1$ and $h_{shared} = 2$ tend to improve over the other values. Overall, the speedup achieved by shared memoization, calculated as the ratio between the total cumulative times of the non-shared and best shared approaches, ranges from 185% to 471%. The open-source implementation of our experiments, together with more results revealing similar trends, is available at https://github.com/aclai-lab/OVERLAY2022.jl.

3. Conclusions

In this work, we considered modal logic as paradigmatic of temporal, spatial, and spatio-temporal logics, and noted how Kripke models can be used for representing data with no predefined structure (which, nowadays, amounts to the vast majority of data). We pointed out the importance of finite model checking in automatic inductive reasoning. We generalized this problem to a multiple models and multiple formulas setting, showing the benefits of a shared memoization approach. This study is part of a bigger investigation on modal symbolic learning, which attempts at learning qualitative patterns from unstructured objects, seen as Kripke models, which is not possible with the limited expressive power of propositional logic.

References

[1] E. Clarke, O. Grumberg, D. Kroening, D. Peled, H. Veith, Model Checking, 2nd ed., MIT Press, 2018.

[2] A. Gandomi, M. Haider, Beyond the hype: Big data concepts, methods, and analytics, International Journal of Information Management 35 (2015) 137–144.

[3] A. Sistla, E. Clarke, The Complexity of Propositional Linear Temporal Logics, Journal of the ACM 32 (1985) 733–749.

[4] E. Bartocci, L. Bortolussi, G. Sanguinetti, Data-driven statistical learning of temporal logic properties, in: Proceedings of the 12th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS), volume 8711 of *Lecture Notes in Computer Science*, Springer, 2014, pp. 23–37.

[5] G. Bombara, C. I. Vasile, F. Penedo, H. Yasuoka, C. Belta, A Decision Tree Approach to Data Classification using Signal Temporal Logic, in: Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control (HSCC), 2016, pp. 1–10.

[6] A. Jones, Z. Kong, C. Belta, Anomaly detection in cyber-physical systems: A formal methods approach, in: Proceedings of the 53rd IEEE Conference on Decision and Control (CDC), 2014, pp. 848–853.

[7] Z. Kong, A. Jones, A. Medina Ayala, E. Aydin Gol, C. Belta, Temporal logic inference for classification and prediction from data, in: Proceedings of the 17th International Conference on Hybrid Systems: Computation and Control (HSCC), 2014, pp. 273–282.

[8] R. Grosu, S. A. Smolka, F. Corradini, A. Wasilewska, E. Entcheva, E. Bartocci, Learning and detecting emergent behavior in networks of cardiac myocytes, Communications of the ACM 52 (2009) 97–105.

[9] A. Brunello, G. Sciavicco, I. E. Stan, Interval Temporal Logic Decision Tree Learning, in: Proceedings of the 16th European Conference on Logics in Artificial Intelligence (JELIA), volume 11468 of *Lecture Notes in Computer Science*, Springer, 2019, pp. 778–793.

[10] G. Sciavicco, I. E. Stan, Knowledge Extraction with Interval Temporal Logic Decision Trees, in: Proceedings of the 27th International Symposium on Temporal Representation and Reasoning (TIME), volume 178 of *LIPIcs*, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020, pp. 9:1–9:16.

[11] E. Lucena-Sánchez, G. Sciavicco, I. E. Stan, Feature and Language Selection in Temporal Symbolic Regression for Interpretable Air Quality Modelling, Algorithms 14 (2021) 76.

[12] G. Pagliarini, G. Sciavicco, Decision Tree Learning with Spatial Modal Logics, in: Proceedings of the 12th International Symposium on Games, Automata, Logics, and Formal Verification (GandALF), volume 346, 2021, pp. 273–290.

[13] E. M. Clarke, E. A. Emerson, Design and synthesis of synchronization skeletons using branching-time temporal logic, in: D. Kozen (Ed.), Logics of Programs, Workshop, Yorktown Heights, New York, USA, May 1981, 1981, pp. 52–71.

[14] D. Cordeiro, G. Mounié, S. Perarnau, D. Trystram, J. Vincent, F. Wagner, Random graph generation for scheduling simulations, in: Proceedings of the 3rd International Conference on Simulation Tools and Techniques (SIMUTools), 2010, p. 60.