Poster: Model checking RNNs with modal μ-calculus

Tatsuhiro Aoshima, Toshinori Usui
NTT Secure Platform Laboratories
{tatsuhiro.aoshima.md, toshinori.usui.rt}@hco.ntt.co.jp

***Abstract*—Machine learning models have been applied to many cyber-physical systems such as self-driving cars, robotics, and factory automation. However, it would be difficult to adapt them to more mission-critical systems, such as energy plants, because there is no safety guarantee. This poster presents the security of systems controlled using machine learning models, especially, Recurrent Neural Networks (RNNs). We propose a novel method for checking whether a given RNN satisfies a given specification, as abstractly interpreting the model with the constrained zonotopes. The specification is written in the modal μ-calculus containing many classical temporal logics such as Linear Temporal Logic and Computation Tree Logic.**

I. Introduction

Machine learning models have been applied to many cyber-physical systems such as self-driving cars, robotics, and factory automation. However, there is no guarantee for them to act safely, so it would be difficult to adapt them to more mission-critical systems such as energy plants. Attacks against these systems would seriously disrupt our society.

For example, if some control systems in a nuclear power plant are taken over by an attacker, the attacker could damage the plant and the neighbouring residents. Consider that a control rod in the plant keeps the temperature in the plant nearly constant and is controlled using an RNN. In this case, the developers would try to ensure that, for example, if the temperature x_T is greater than or equal to α (the threshold at which the control rod should start to work), then the position of the control rod o_P must be lower than or equal to γ (the threshold at which the control rod works) at some point in the future. This can be written formally in a modal μ-formula [2] as follows:

$$\nu x.(x_T \geq \alpha \rightarrow (\mu x.o_P \leq \gamma \vee \Box x)) \wedge \Box x$$

The pattern $\nu x.\psi \wedge \Box x$ means ψ is satisfied in any circumstance, and $\mu x.\psi' \vee \Box x$ means ψ' will be satisfied at some point in the future. This μ, ν operator directly corresponds to the corresponding model checking algorithm. Unlike DeepMind's method [3], it can specify over a period of time.

In this case, a model checking algorithm takes an RNN as a checked model and a modal μ-formula as a specification. It checks and outputs whether the given model satisfies the given specification. Then, if not so, it generates a counterexample that is an input to the RNN causing it to violate the specification.

We propose a novel method to make it possible to perform model checking of a given specification written in the modal μ-calculus on a given RNN. First, the algorithm (A) calculates the set of states of the RNN not satisfying the given specification (the semantic set) then (B) checks whether the given initial state is in it. If not so, (C) the RNN is concluded as satisfying the specification because, due to the construction of the semantic set, the initial state satisfies the specification. If so, the RNN is concluded as not satisfying the specification, and the algorithm calculates a counterexample with backpropagation using the calculation process of (A).

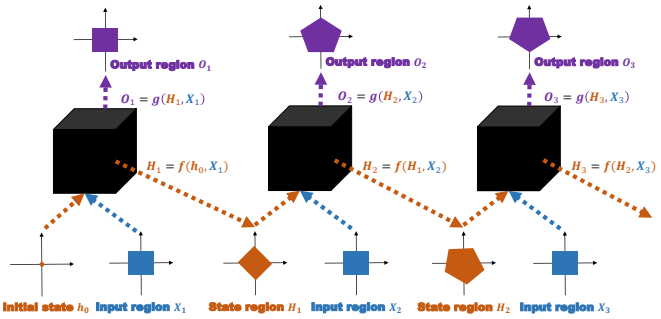


Fig. 1: Approximation of calculation of RNN with constrained zonotopes. Except for initial state $\boldsymbol{h}_0$, all input, state and output vectors are abstractly represented as set expressed as constrained zonotopes. Constrained zonotopes are closed under forward and backward computation of RNN, so set of states not satisfying given specification can be computed along with computation of RNN.

Technically, an RNN is expressed as a non-linear function composed of linear mapping layers and activation functions. Hence, to calculate the semantic set, our method interprets the model abstractly with the constrained zonotopes (see Fig. 1).

II. Technical Background

The safety of machine learning models is becoming an increasing concern. OpenAI released Safety Gym [4] to provide a framework for ensuring that machine learning models respect safety constraints. It can be used only in training new models and cannot be applied to trained models. Furthermore, it is not guaranteed that a model can satisfy the given safety constraints mathematically.

To mathematically guarantee safety, many model checking algorithms have been proposed. These only supports Finite State Machines (FSMs) [2] or PieceWise Affine (PWA) Continuous State Machines (CSMs) [1]. An RNN is a CSM

and is a complex non-linear function constructed of linear mapping layers and activation functions. Hence, the current model checking algorithms cannot be applied to RNNs.

For continuous models, an algorithm [1] using abstraction maintains a set of disjoint regions in a continuous state space. This is suitable for PWA CSMs but not for non-linear CSMs because of complex state representations. Furthermore, automaton-based algorithms are only suitable for FSM and are not infeasible for CSM, because of the infinitely many branches between states.

Modal μ-calculus is a generalization of Linear Temporal Logic and Computation Tree Logic (CTL) and can represent a specification such as a property should be satisfied only at the cyclic positions. Model checking algorithms have a structure similar to that for CTL. Hence, if a specification is easily written in CTL, such an algorithm can check as efficiently as that for CTL can.

III. Proposed Method

An RNN is a pair of functions $(\boldsymbol{f}, \boldsymbol{g})$ and generates an infinite output vector with a given initial vector $\boldsymbol{h}_0$ and a given infinite input vector $(\boldsymbol{x}_t)_{t=1}^{\infty}$ as follows:

$$\boldsymbol{o}_t = \boldsymbol{g}(\boldsymbol{h}_t, \boldsymbol{x}_t),\ \boldsymbol{h}_t = \boldsymbol{f}(\boldsymbol{h}_{t-1}, \boldsymbol{x}_t), \quad t > 0.$$

Our algorithm applies model checking for a given RNN with a specification written in the modal μ-calculus. Refer to the syntax and semantics of the modal μ-calculus in a standard textbook such as [2]. To check whether the given RNN $(\boldsymbol{f}, \boldsymbol{g})$ with the given initial vector $\boldsymbol{h}_0$ satisfies the given specification written as a modal μ-calculus expression φ, we use the semantic set $[\![\varphi]\!]$ of φ. $[\![\varphi]\!]$ is composed of the states of the RNN $(\boldsymbol{f}, \boldsymbol{g})$ satisfying φ.

If the initial state $\boldsymbol{h}_0$ is not in $[\![\neg\varphi]\!]$, then the algorithm concludes that the model does not satisfy the specification, otherwise does. Notice that the negation in φ can be removed from elsewhere before atomic propositions, so there is no need to compute the negation, as assuming that the semantic set of the atomic propositions and those negations are given.

$[\![\varphi]\!]$ can be computed recursively on the structure of φ. For example, the specification $\varphi := \nu x.(\psi \to (\mu x.\psi' \vee \Box x)) \wedge \Box x$ means that, for any path starting from the initial state, if there exists a state satisfying ψ on the path, then there exists a state satisfying ψ' at a future point on any path starting from the state. Hence, using the greatest (least) fixpoint operator $\mathrm{GFP}_X(\cdot), \mathrm{LFP}_X(\cdot)$, $[\![\varphi]\!]$ can be computed as follows:

$$\begin{aligned}&[\![\nu x.(\psi \to (\mu x.\psi' \vee \Box x)) \wedge \Box x]\!] \\ &= \mathrm{GFP}_X\left(([\![\neg\psi]\!] \cup S_1) \cap [\boldsymbol{f}](X)\right).\end{aligned}$$

Notice that $P \to Q$ is $\neg P \vee Q$, and,

$$S_1 := [\![\mu x.\psi' \vee \Box x]\!] = \mathrm{LFP}_X\left([\![\psi']\!] \cup [\boldsymbol{f}](X)\right).$$

Here, S_1 is the set of states from which satisfy ψ' at a future point on any path. Using S_1, $[\![\varphi]\!]$ can be obtained by computing the set of states from which satisfy $\neg\psi$ or having a state in S_1 as a successor on any path.

The termination of the calculation of any semantic set is generally not guaranteed in a continuous state space. For example, consider the case of monotonically increasing state sets in computing fixpoint operators.

Hence, the constrained zonotopes [5] are used for an approximate calculation (see Fig. 1). A constrained zonotope is the intersection of a polytope and the solution set of a linear equation in a real Euclid space. It is closed under addition, matrix application, solving a linear equation in another constrained zonotope, intersection, and bounded monotone element-wise activation functions. Thus, the image and inverse image of any layer in RNNs can be expressed with the contrained zonotopes, because RNNs are compositions of linear transformations and bounded activation functions.

Our algorithm will produce a counterexample if the model does not satisfy the given specification. Hence, in this poster, we consider the specification not including the diamond operator $\diamond$ because, if a counterexample exist, then the number of the branches in the counterexample can be ensured to be only one. To calculate a counterexample, backpropagation is used on the input vectors $(\boldsymbol{x}_t)_{t=1}^{T}$ to satisfy that each state at time t $\boldsymbol{h}_t$ is in $[\![\neg\varphi]\!]^{\widehat{}}_{(T-t)}$ with the computation process $([\![\neg\varphi]\!]^{\widehat{}}_{(t)})_{t=1}^{T}$. For example, except for the given $\boldsymbol{h}_0$, a loss function is designed such as taking 0 if each state at time t satisfies $\boldsymbol{h}_t \in [\![\neg\varphi]\!]^{\widehat{}}_{(T-t)}$; otherwise, a positive number (such as the distance from a set to a given point) is taken.

IV. Future Works

We plan to implement our algorithm to demonstrate its effectiveness. The most numerically cubersome step of the implementation is reducing the generators and the constraints of a constrained zonotope. However, an efficient algorithm for solving this has already been proposed [5]. The constrained zonotopes are not closed under the Hadamard product used by the gates in RNNs, so we are going to develop another mathematical tool closed under the Hadamard product also. Notice that, a Convolutional Neural Network can be seen as a single-step RNN; hence, our algorithm can be applied to them trivially, but it would be difficult to express atomic propositions with the constrained zonotopes (e.g. one representing the set of all panda images).

We hope our algorithm will contribute to adaptation of RNNs to more cyber-physical systems with proven safety.

Acknowledgment

The authors would like to thank members of our laboratory for helpful discussion.

References

[1] Belta, C., B. Yordanov, and E. A. Gol. 2017. *Formal Methods for Discrete-Time Dynamical Systems.* Springer.

[2] Clarke, E. M., T. A. Henzinger, H. Veith, and R. Bloem. 2018. *Handbook of Model Checking.* Springer.

[3] Dvijotham, K., R. Stanforth, S. Gowal, T. A. Mann, and P. Kohli. (2018). A Dual Approach to Scalable Verification of Deep Networks. *UAI*, pp. 550–559.

[4] OpenAI. *Safety Gym*. Retrieved 2019/12/3 from https://openai.com/blog/safety-gym/.

[5] Scott, J. K., D. M. Raimondo, G. R. Marseglia, and R. D. Braatz. (2016). "Constrained zonotopes: A new tool for set-based estimation and fault detection." *Automatica*, vol. 69, pp. 126–136.

Model checking RNNs with modal μ-calculus

Tatsuhiro **Aoshima**, **T**oshinori **Usui** (NTT Secure Platform Laboratories)

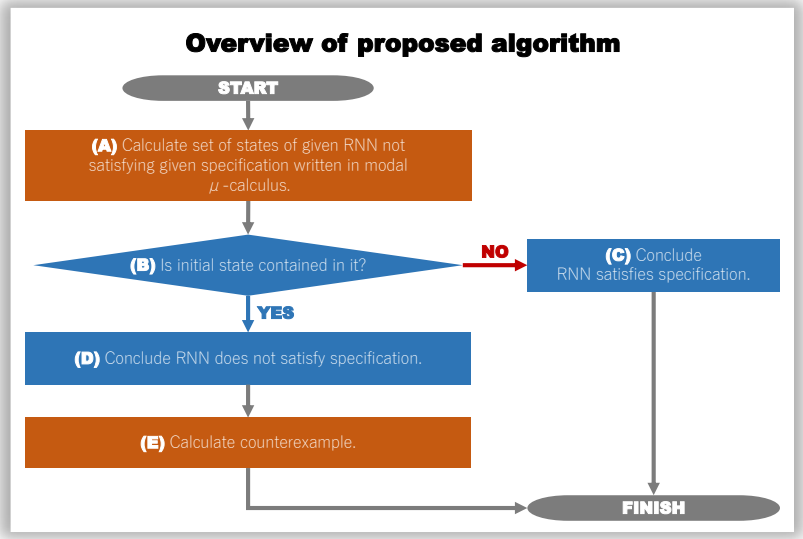
Machine Learning models have been **applied to cyber-physical systems** such as *self-driving cars*, *robotics*, and *factory automation*.

However, there is **no safety guarantee**, so, *attacks would seriously disrupt our society*.

We consider the **security of RNNs** as

(A) *abstractly interpreting* an RNN,
(B),(C),(D) checking whether an RNN satisfies a specification written in *modal μ-calculus*,
(E) generating *a possible attack pattern* if not so,

to check the safety *mathematically*.



Analogy of specification written in modal μ-calculus

Each *formula corresponds to a set of states*:

- $\Box p$ is a set of states satisfying p *at any next state*.
- x is *a set variable in* ***recursive*** *formulae*.
- $\mu x.\, \psi' \vee \Box x \equiv \psi' \vee \Box\psi' \vee \Box(\psi' \vee \Box\psi') \vee \cdots$
  ➡ set of states satisfying ψ' or *at any next state*, or **recursively**, *at any next state*, or ...
- $\nu x.\, (\psi \to (\cdots)) \wedge \Box x$
  $\equiv (\psi \to (\cdots)) \wedge \Box(\psi \to (\cdots)) \wedge \Box\big((\psi \to (\cdots)) \wedge \Box(\psi \to (\cdots))\big) \wedge \cdots$
  ➡ set of states satisfying $(\psi \to (\cdots))$ and *at any next state*, and **recursively**, *at any next state*, and ...

Modal μ-calculus can be used to express many properties. For example,

(1) $\nu x.\, (\psi \to (\mu x.\, \psi' \vee \Box x)) \wedge \Box x$.

Each component represents:

- $\mu x.\, \psi' \vee \Box x$ means ψ' is satisfied *at a future point on any path*.
- $\nu x.\, (\psi \to (\cdots)) \wedge \Box x$ means if ψ is satisfied then $(\cdots)$ is *always* satisfied *on any path*.

Hence, (1) means that, **for any case, if ψ is satisfied, then ψ' is satisfied sometime later**.

NOTICE. Some properties *cannot be expressed in a subset of modal μ-calculus* known as CTL or LTL. However, our algorithm works as efficiently as that for CTL or LTL if the specification can be also written in CTL or LTL.

Abstractly interpreting an RNN

is done by *tracing all states with possible input vectors*:

- Calculate the set of states not satisfying the specification.
- It is represented with a *constrained zonotope closed under* addition, matrix application, *solving a linear equation*, intersection, and *bounded monotone element-wise activation functions*.

A counterexample is generated

with backpropagation: *calculating each input vector to force each state vector to be contained in the set computed in step (A)*. Those inputs *cause an RNN to not satisfy the specification*; hence, it is a possible attack pattern.

NOTICE. Like an RNN having a continuous state space, it is impossible to enumerate any path as a counterexample if it exists, so only formulae not containing $\Diamond$ are handled.