Bounded and Unbounded Verification of RNN-Based Agents in Non-deterministic Environments

Extended Abstract

Mehran Hosseini
Imperial College London
London, United Kingdom
m.hosseini@imperial.ac.uk

Alessio Lomuscio
Imperial College London
London, United Kingdom
a.lomuscio@imperial.ac.uk

ABSTRACT

We consider closed-loop Agent-Environment Systems (AESs), where the agent is controlled by a Recurrent Neural Network (RNN) with ReLU activations in a non-deterministic environment. We introduce a new approach based on Mixed-Integer Linear Programming to verify such systems, which allows for more optimised complete and sound verification of bounded temporal properties of such AESs. Using our approach, we additionally, devise a sound algorithm for the unbounded verification of such AESs for the first time.

KEYWORDS

Formal Verification; Verification of Agent-Environment Systems; Verification of Neural Networks; Verification of RNNs; Safe AI

ACM Reference Format:

Mehran Hosseini and Alessio Lomuscio. 2023. Bounded and Unbounded Verification of RNN-Based Agents in Non-deterministic Environments: Extended Abstract. In *Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), London, United Kingdom, May 29 – June 2, 2023*, IFAAMAS, 3 pages.

1 INTRODUCTION

Autonomous agents are being developed and deployed at an increasing pace. Considerable progress has been made to ensure that deployed systems adhere to safety specifications [5, 6, 8–11, 13, 17, 23, 25–27, 31, 39]. A key assumption made in the traditional literature in this area is that agents are traditionally designed and directly programmed via programming languages. A novel generation of multi-agent systems, often referred to as *"neural agents"* [1–3], have emerged that differently from traditional agent-based systems are realised and implemented by *Neural Networks* (*NNs*).

NNs are known to be fragile [7, 35]; therefore, significant attention has been paid to the verification of NN in open loop systems, such as computer vision systems and decision making [12, 18, 21, 29, 30, 33, 34, 36, 38]. Nevertheless, it is crucial to develop methods to verify neural agents against safety specifications in closed loop systems.

Verification of closed-loop neural systems is considerably less studied. Some of the works that consider closed-loop neural systems include [1–4, 15, 24, 32, 37]. Most of these works assume the underlying neural networks are *Feed-Forward Neural Networks* (*FFNNs*), which implies that the agents are stateless. [1] studies the problem of whether an *Agent-Environment System* (*AES*) ever reaches an unwanted state in fixed and arbitrary number of steps. The specifications considered in [1] were extended in [2] to CTL, where the authors showed that verifying FFNN-based AESs against bounded CTL properties is PSpace-hard and in coNExpTime, even though verifying FFNN-based AESs against unbounded CTL properties is *undecidable*. The only proposal that we are aware of in which a memoryful neural agent interacts with an environment is [4], where the agent uses a previously trained RNN [20]. This work only deals with bounded executions and even in this context its scalability is limited, since the solution is based on unrolling the neural model.

Closer to this work is [4], which considers the verification of closed-loop AESs with RNN-based agents against bounded temporal specifications. This work uses unrolling to transform the RNNs, controlling the agents, to FFNNs, and then, uses the approach of [2] to verify the system against bounded LTL specifications. This approach suffers from the blow-up in the size of the resulting FFNN as the number of time-steps increases and does not scale to large networks and large number of time-steps. Some of the other related work that consider verification of RNN, but in open-loop systems, are [22, 28].

In this paper we propose a more scalable approach to the verification problem of memoryful, neural agent-based systems. Specifically, we (1) present a novel recursive approach for the verification of autonomous systems that are composed of systems with an RNN-based agent interacting with an environment and (2) introduce highly optimised methods for the verification of such neural systems against a temporal logic on bounded and unbounded executions of the system.

2 NOTATION & BACKGROUND

Here, we define the concepts and notation used such as RNNs and the agent-environment setup considered, i.e., *Recurrent Neural Agent-Environment Systems* (*RNN-AESs*) adopted from [4]. We use C^* to denote the set of all infinite sequences in C, $\{\boldsymbol{x}^{\langle i \rangle}\}_{i\in\mathbb{N}}$ to indicate an infinite sequence, and $\boldsymbol{x}^{\langle t \rangle}$ to refer to the t-th element in $\{\boldsymbol{x}^{\langle i \rangle}\}_{i\in\mathbb{N}}$. We start by defining recurrent layers and RNNs. For more on other layers please see [16, Chapters 6-10].

Definition 1 (Recurrent Layer). A *recurrent layer* is a function

$$R : \begin{cases} \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}^n, \\ (\boldsymbol{x}^{\langle t \rangle}, \boldsymbol{y}^{\langle t-1 \rangle}) \mapsto \mathrm{ReLU}(\boldsymbol{W}_x \boldsymbol{x}^{\langle t \rangle} + \boldsymbol{W}_h \boldsymbol{y}^{\langle t-1 \rangle} + \boldsymbol{b}), \end{cases}$$

where $\boldsymbol{x}^{\langle t \rangle} \in \mathbb{R}^m$ and $\boldsymbol{y}^{\langle t \rangle} \in \mathbb{R}^n$ are the *input* and *output* of R at time t, and $\boldsymbol{W}_x \in \mathbb{R}^{n\times m}$, $\boldsymbol{W}_h \in \mathbb{R}^{n\times n}$, $\boldsymbol{b} \in \mathbb{R}^n$, and ReLU are the *kernel*, *recurrent kernel*, *bias*, and the *activation function* of R, respectively.

Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), A. Ricci, W. Yeoh, N. Agmon, B. An (eds.), May 29 – June 2, 2023, London, United Kingdom.

Definition 2 (Recurrent Neural Network). An RNN is a function defined by a composition of feed-forward and recurrent layers.

Using the notation of Definition 1, if we show the layer number in a sequential RNN with the superscripts $[i]$, the composition rule for a recurrent layer $R^{[i]}$ of an RNN R can be written as

$$\boldsymbol{y}^{[i]\langle t\rangle} = \mathrm{ReLU}(\boldsymbol{W}_x^{[i]}\boldsymbol{y}^{[i-1]\langle t\rangle} + \boldsymbol{W}_h^{[i]}\boldsymbol{y}^{[i]\langle t-1\rangle} + \boldsymbol{b}^{[i]}).$$

Definition 3 (Environment). An environment is defined by a tuple $E = (\mathcal{S}, O, o, \tau)$, where $\mathcal{S}$ is a *set of states* of the environment, O is a *set of observations* of the environment, $o : \mathcal{S} \to O$ is an environment *observation function* that given an environment state returns an observation of it that agents can access, $\tau : \mathcal{S} \times \mathcal{A} \to 2^{\mathcal{S}}$ is a *transition relation*, which given the current state of the environment $\boldsymbol{s} \in \mathcal{S}$ and an *action* $\boldsymbol{a} \in \mathcal{A}$, performed by the agent, returns the set of possible next states $\tau(\boldsymbol{s}, \boldsymbol{a}) \in \mathcal{S}$. The environment is *deterministic* when the transition relation τ is a function $\tau : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$.

We assume that the observation function and transition relation are linearly-definable, or otherwise, can be linearly approximated to an arbitrary level of precision [1, 14].

Definition 4 (RNN Agent). A *recurrent neural agent*, or an *agent* for short, denoted by Agt_R, acting on an environment E is defined by an *action* function $a : O^* \to \mathcal{A}$. Given a finite sequence of environment observations from $O \subseteq \mathbb{R}^m$, the action function a returns an action from the set $\mathcal{A} = \mathbb{R}^n$ of admissible actions for the agent. The function a is implemented by an RNN $R : \mathbb{R}^m \times \mathbb{R}^\ell \to \mathbb{R}^n$.

Definition 5 (RNN-AES). A Recurrent Neural Agent-Environment System (RNN-AES) is a tuple $AES = (E, Agt_R, \mathcal{I}^{\langle 0\rangle})$, where $E = (\mathcal{S}, O, o, \tau)$ defined by Definition 3, Agt_R is an RNN agent satisfying Definition 4, and $\mathcal{I}^{\langle 0\rangle} \subseteq \mathcal{S}$ is set of initial states of the environment.

Definition 6 (Specifications). For an environment with state space $\mathcal{S} = \mathbb{R}^m$, all specifications are defined by the following BNF.

$$\begin{aligned} \phi &::= \bigcirc^k C \mid C\,U^{\leq k} C \mid \Box C \\ C &::= C \wedge C \mid C \vee C \mid \neg C \mid \boldsymbol{c}^\intercal \cdot \boldsymbol{x} < d \end{aligned}$$

for constants $\boldsymbol{c} \in \mathbb{R}^m$, $d \in \mathbb{R}$, and $k \in \mathbb{N}$ and variable vector $\boldsymbol{x} \in \mathbb{R}^m$.

3 VERIFICATION OF RNN-AES

We introduce a recursive method for reducing the verification of RNN-AESs, which (1) allows verifying RNN-AESs with several recurrent layers, (2) unlike unrolling [4], can handle varying number of time-steps, (3) uses fewer variables in its *Mixed-Integer Linear Programming* (*MILP*) formulation compared to unrolling and, hence, is more efficient. We present our method using a two layer RNN R consisting of an input recurrent layer $R^{[1]} : \mathbb{R}^m \times \mathbb{R}^{n_1} \to \mathbb{R}^{n_1}$ and an output feed-forward layer $R^{[2]} : \mathbb{R}^{n_1} \to \mathbb{R}^{n_2}$. Given the input constraint $C_{inp}^{\langle t\rangle} \subseteq \mathbb{R}^m$ and output constraint $C_{out}^{\langle t\rangle} \subseteq \mathbb{R}^{n_2}$ ($C_{inp}^{\langle t\rangle}$ and $C_{out}^{\langle t\rangle}$ are semi-linear sets and can evolve with time), the MILP problem that is used to verify whether R's output satisfies $C_{out}^{\langle t\rangle}$, when its input $\boldsymbol{x}^{\langle t\rangle} \in C_{out}^{\langle t\rangle}$ is

$$\begin{cases} \text{solve: } \mathrm{ReLU}(\boldsymbol{W}^{[2]}\mathrm{ReLU}(\boldsymbol{W}_x^{[1]}\boldsymbol{x}^{\langle t\rangle} + \boldsymbol{W}_h^{[1]}\boldsymbol{y}^{\langle t-1\rangle} + \boldsymbol{b}^{[1]}) + \boldsymbol{b}^{[2]}) \subseteq C_{out}^{\langle t\rangle}, \\ \text{subject to: } \boldsymbol{x}^{\langle t\rangle} \in C_{inp}^{\langle t\rangle},\ \boldsymbol{y}^{\langle t-1\rangle} \in \mathcal{I}^{[1]\langle t-1\rangle}, \end{cases}$$

where $\mathcal{I}^{[1]\langle t\rangle}$'s are recursively defined as $\mathcal{I}^{[1]\langle t\rangle} = \{\mathrm{ReLU}(\boldsymbol{W}_x\boldsymbol{x}^{\langle t\rangle} + \boldsymbol{W}_h\boldsymbol{y}^{\langle t-1\rangle} + \boldsymbol{b}) : \boldsymbol{x}^{\langle t\rangle} \in C_{inp}^{\langle t\rangle}, \boldsymbol{y}^{\langle t-1\rangle} \in \mathcal{I}^{[1]\langle t-1\rangle}\}$, and $\mathcal{I}^{[1]\langle 0\rangle}$ is the initial hidden state of $R^{[1]}$. Note that the ReLU activation can be encoded in MILP using the "Big-M" method [19]; thus, we can use standard tools for solving MILP to solve the verification problem.

Verifying $\bigcirc^k C$ and $C U^{\leq k} C$. Using the notation above, we can verify a given RNN-AES AES_R against bounded temporal specifications. Algorithm 1 outlines the verification process for $\bigcirc^k C$. The procedure for verifying $C U^{\leq k} C$ is similar.

Algorithm 1: Verifying $\bigcirc^k C$

```
Input  : RNN-AES AES_R and specification φ = ◯^k C.
Output : True/False
states = C
for t ← 1 to k do
    I^[0] = o(C)
    for i ← 1 to N do
        if R^[i] is recurrent then
            I^[i]⟨t⟩ = R^[i](I^[i−1]⟨t⟩, I^[i]⟨t−1⟩)
        else
            I^[i]⟨t⟩ = R^[i](I^[i−1]⟨t⟩)
        end
    end
    states = τ(states, I^[N]⟨t⟩)
end
return SAT_MILP(states ⊆ C)
```

In Algorithm 1, $\mathcal{I}^{[i]\langle t\rangle} = R^{[i]}(\mathcal{I}^{[i-1]\langle t\rangle}, \mathcal{I}^{[i]\langle t-1\rangle})$ and $\mathcal{I}^{[i]\langle t\rangle} = R^{[i]}(\mathcal{I}^{[i-1]\langle t\rangle})$ denote the set of reachable points in the i-th layer of R after t time steps; moreover, they can be linearly encoded using real and binary variables. Finally, since C is also a linearly definable set, we can use MILP solvers to solve $\mathsf{SAT_{MILP}}(states \subseteq C)$ and answer the verification problem in a sound and complete manner.

Verifying $\Box C$. We now introduce Algorithm 2 for verifying RNN-AESs against specifications of the form $\Box C$ using MILP.

Algorithm 2: Verifying $\Box C$

```
Input  : RNN-AES AES_R and specification φ = □C.
Output : True/False
I^[N] = C
for i ← N to 1 do
    if R^[i] is recurrent then
        I^[i−1] = I^[i−1]⟨0⟩ ∪ {y^[i−1] : R^[i](y^[i−1], I^[i]) ⊆ I^[i]}
    else
        I^[i−1] = {y^[i−1] : R^[i](y^[i−1]) ∈ I^[i]}
    end
end
acts = R^[N](··· R^[1](o(C), I^[1]) ···)
return SAT_MILP(τ(C, acts) ⊆ C)
```

It is straightforward to show that the set $acts$ in Algorithm 2 is the set of all possible actions of Agt_R for all input sequences $\{\boldsymbol{x}\}_{i\in\mathbb{N}} \in C^*$ and time steps $t \in \mathbb{N}$. The soundness of Algorithm 2 follows from the fact that if $\tau(C, acts) \subseteq C$, then for all $t \in \mathbb{N}$, we have that $\tau^t(C, acts) = \tau(\ldots(\tau(\tau(C, acts), acts)\ldots), acts) \subseteq C$, and thus, AES_R satisfies $\Box C$.

ACKNOWLEDGMENTS

This work is partially supported by the DARPA Assured Autonomy programme (FA8750-18-C-0095) and the UK Royal Academy of Engineering (CiET17/18-26).

REFERENCES

[1] Michael Akintunde, Alessio Lomuscio, Lalit Maganti, and Edoardo Pirovano. 2018. Reachability Analysis for Neural Agent-Environment Systems. In *Proceedings of the 16th International Conference on Principles of Knowledge Representation and Reasoning, KR*. AAAI Press, 184–193.

[2] Michael E. Akintunde, Elena Botoeva, Panagiotis Kouvaros, and Alessio Lomuscio. 2020. Formal Verification of Neural Agents in Non-deterministic Environments. In *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems, AAMAS*. IFAAMAS, 25–33.

[3] Michael E. Akintunde, Elena Botoeva, Panagiotis Kouvaros, and Alessio Lomuscio. 2020. Verifying Strategic Abilities of Neural-symbolic Multi-agent Systems. In *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning, KR*. AAAI Press, 22–32.

[4] Michael E. Akintunde, Andreea Kevorchian, Alessio Lomuscio, and Edoardo Pirovano. 2019. Verification of RNN-Based Neural Agent-Environment Systems. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence, AAAI*. AAAI Press, 6006–6013.

[5] Natasha Alechina, Mehdi Dastani, Anas Fahad Khan, Brian Logan, and J-J Ch Meyer. 2010. Using Theorem Proving to Verify Properties of Agent Programs. In *Specification and Verification of Multi-Agent Systems*. Springer, 1–33.

[6] Natasha Alechina, Mehdi Dastani, Brian Logan, and John-Jules Ch. Meyer. 2011. Reasoning about Agent Deliberation. *Journal of Autonomous Agents and Multi-Agent Systems* 22, 2 (2011), 1–26.

[7] Battista Biggio and Fabio Roli. 2018. Wild patterns: Ten years after the rise of Adversarial Machine Learning. *Pattern Recognition* 84 (2018), 317–331.

[8] Rafael H. Bordini, Louise A. Dennis, Berndt Farwer, and Michael Fisher. 2008. Automated Verification of Multi-Agent Programs. In *Proceedings of the 23rd IEEE/ACM International Conference on Automated Software Engineering, ASE*. IEEE, 69–78.

[9] Rafael H. Bordini, Michael Fisher, Willem Visser, and Michael J. Wooldridge. 2003. Model Checking Multi-Agent Programs with CASP. In *Proceedings of the 15th International Conference on Computer Aided Verification, CAV (Lecture Notes in Computer Science, Vol. 2725)*. Springer, 110–113.

[10] Rafael H. Bordini, Michael Fisher, Willem Visser, and Michael J. Wooldridge. 2004. Verifiable Multi-Agent Programs. In *Proceedings of the 1st International Workshop on Programming Multiagent Systems, PROMAS (Lecture Notes in Computer Science, Vol. 3067)*. Springer, 72–89.

[11] Rafael H. Bordini, Michael Fisher, Willem Visser, and Michael J. Wooldridge. 2006. Verifying Multi-agent Programs by Model Checking. *Autonomous Agents and Multi-Agent Systems* 12, 2 (2006), 239–256.

[12] Elena Botoeva, Panagiotis Kouvaros, Jan Kronqvist, Alessio Lomuscio, and Ruth Misener. 2020. Efficient Verification of ReLU-Based Neural Networks via Dependency Analysis. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence, AAAI*. AAAI Press, 3291–3299.

[13] Petr Cermák, Alessio Lomuscio, Fabio Mogavero, and Aniello Murano. 2018. Practical Verification of Multi-Agent Systems against SLK Specifications. *Information and Computation* 261, 3 (2018), 588–614.

[14] Claudia D'Ambrosio, Andrea Lodi, and Silvano Martello. 2010. Piecewise Linear Approximation of Functions of Two Variables in MILP Models. *Operations Research Letters* 38, 1 (2010), 39–46.

[15] Jiameng Fan, Chao Huang, Xin Chen, Wenchao Li, and Qi Zhu. 2020. ReachNN*: A Tool for Reachability Analysis of Neural-Network Controlled Systems. In *Automated Technology for Verification and Analysis - 18th International Symposium, ATVA (Lecture Notes in Computer Science, Vol. 12302)*. Springer, 537–542.

[16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press.

[17] Valentin Goranko and Wojciech Jamroga. 2004. Comparing Semantics for Logics of Multi-agent Systems. *Synthese* 139, 2 (2004), 241–280.

[18] Sven Gowal, Krishnamurthy Dvijotham, Robert Stanforth, Rudy Bunel, Chongli Qin, Jonathan Uesato, Relja Arandjelovic, Timothy A. Mann, and Pushmeet Kohli. 2018. On the Effectiveness of Interval Bound Propagation for Training Verifiably Robust Models. arXiv:1810.12715

[19] Igor Griva, Stephen G. Nash, and Ariela Sofer. 2008. *Linear and Nonlinear Optimization* (2nd ed.). SIAM.

[20] Matthew J. Hausknecht and Peter Stone. 2015. Deep Recurrent Q-Learning for Partially Observable MDPs. In *AAAI Fall Symposium on Sequential Decision Making for Intelligent Agents*. AAAI Press, 29–37.

[21] Patrick Henriksen and Alessio R. Lomuscio. 2020. Efficient Neural Network Verification via Adaptive Refinement and Adversarial Search. In *Proceedings of the 24th European Conference on Artificial Intelligence, ECAI*. IOS Press, 2513–2520.

[22] Yuval Jacoby, Clark W. Barrett, and Guy Katz. 2020. Verifying Recurrent Neural Networks using Invariant Inference. In *Automated Technology for Verification and Analysis - 18th International Symposium, ATVA (Lecture Notes in Computer Science, Vol. 12302)*. Springer, 57–74.

[23] Wojciech Jamroga, Artur Meski, and Maciej Szreter. 2013. Modularity and Openness in Modeling Multi-Agent Systems. In *Proceedings of the 4th International Symposium on Games, Automata, Logics and Formal Verification, GandALF13*. EPTCS, 224–239.

[24] Taylor T. Johnson, Diego Manzanas Lopez, Patrick Musau, Hoang-Dung Tran, Elena Botoeva, Francesco Leofante, Amir Maleki, Chelsea Sidrane, Jiameng Fan, and Chao Huang. 2020. ARCH-COMP20 Category Report: Artificial Intelligence and Neural Network Control Systems (AINNCS) for Continuous and Hybrid Systems Plants. In *7th International Workshop on Applied Verification of Continuous and Hybrid Systems, ARCH20*. EasyChair, 107–139.

[25] Magdalena Kacprzak, Alessio Lomuscio, Tomasz Lasica, Wojciech Penczek, and Maciej Szreter. 2004. Verifying Multiagent Systems via Unbounded Model Checking. In *Proceedings of the 3rd NASA Workshop on Formal Approaches to Agent-Based Systems, FAABS (Lecture Notes in Computer Science, Vol. 3228)*. Springer, 189–212.

[26] Magdalena Kacprzak, Alessio Lomuscio, and Wojciech Penczek. 2004. Verification of Epistemic Properties in Multi-Agent Systems by Unbounded Model Checking. In *Proceedings of the 3rd NASA Workshop on Formal Approaches to Agent-Based Systems, FAABS (Lecture Notes in Computer Science, Vol. 3228)*. Springer, 16–28.

[27] Magdalena Kacprzak, Alessio Lomuscio, and Wojciech Penczek. 2004. Verification of Multiagent Systems via Unbounded Model Checking. In *Proceedings of the 3rd International Conference on Autonomous Agents and Multiagent Systems, AAMAS*. ACM, 638–645.

[28] Ching-Yun Ko, Zhaoyang Lyu, Lily Weng, Luca Daniel, Ngai Wong, and Dahua Lin. 2019. POPQORN: Quantifying Robustness of Recurrent Neural Networks. In *Proceedings of the 36th International Conference on Machine Learning, ICML*. PMLR, 3468–3477.

[29] Panagiotis Kouvaros and Alessio Lomuscio. 2021. Towards Scalable Complete Verification of ReLU Neural Networks via Dependency-based Branching. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence, IJCAI*. ijcai.org, 2643–2650.

[30] Alessio Lomuscio and Lalit Maganti. 2017. An Approach to Reachability Analysis for Feed-Forward ReLU Neural Networks. arXiv:1706.07351

[31] Jerzy Pilecki, Marek A. Bednarczyk, and Wojciech Jamroga. 2014. Synthesis and Verification of Uniform Strategies for Multi-agent Systems. In *Proceedings of the 15th International Workshopon Computational Logic in Multi-Agent Systems, CLIMA*. Lecture Notes in Computer Science, Vol. 8624. Springer, 166–182.

[32] Chelsea Sidrane, Amir Maleki, Ahmed Irfan, and Mykel J. Kochenderfer. 2022. OVERT: An Algorithm for Safety Verification of Neural Network Control Policies for Nonlinear Systems. *Journal of Machine Learning Research* 23, 117 (2022), 1–45.

[33] Gagandeep Singh, Timon Gehr, Matthew Mirman, Markus Püschel, and Martin T. Vechev. 2018. Fast and Effective Robustness Certification. In *Advances in Neural Information Processing Systems 31, NeurIPS*. Curran Associates, Inc., 10802–10813.

[34] Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin T. Vechev. 2019. An Abstract Domain for Certifying Neural Networks. *Proceedings of the ACM on Programming Languages* 3, POPL (2019), 41:1–41:30.

[35] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. 2014. Intriguing Properties of Neural Networks. In *2nd International Conference on Learning Representations, ICLR*. OpenReview.net.

[36] Vincent Tjeng, Kai Yuanqing Xiao, and Russ Tedrake. 2019. Evaluating Robustness of Neural Networks with Mixed Integer Programming. In *The 7th International Conference on Learning Representations, ICLR*. OpenReview.net.

[37] Hoang-Dung Tran, Xiaodong Yang, Diego Manzanas Lopez, Patrick Musau, Luan Viet Nguyen, Weiming Xiang, Stanley Bak, and Taylor T. Johnson. 2020. NNV: The Neural Network Verification Tool for Deep Neural Networks and Learning-Enabled Cyber-Physical Systems. In *Proceedings of the 32nd International Conference on Computer Aided Verification, CAV (Lecture Notes in Computer Science, Vol. 12224)*. Springer, 3–17.

[38] Shiqi Wang, Kexin Pei, Justin Whitehouse, Junfeng Yang, and Suman Jana. 2018. Formal Security Analysis of Neural Networks using Symbolic Intervals. In *27th USENIX Security Symposium, USENIX*. USENIX Association, 1599–1614.

[39] Michael J. Wooldridge, Michael Fisher, Marc-Philippe Huget, and Simon Parsons. 2002. Model Checking Multi-Agent Systems with MABLE. In *Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS*. ACM, 952–959.