Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning (KR 2020)
Main Track

Verifying Strategic Abilities of Neural-symbolic Multi-agent Systems

Michael E. Akintunde, Elena Botoeva, Panagiotis Kouvaros, Alessio Lomuscio
Imperial College London, UK
{michael.akintunde13, e.botoeva, p.kouvaros, a.lomuscio}@imperial.ac.uk

Abstract

We investigate the problem of verifying the strategic properties of multi-agent systems equipped with machine learning-based perception units. We introduce a novel model of agents comprising both a perception system implemented via feed-forward neural networks and an action selection mechanism implemented via traditional control logic. We define the verification problem for these systems against a bounded fragment of alternating-time temporal logic. We translate the verification problem on bounded traces into the feasibility problem of mixed integer linear programs and show the soundness and completeness of the translation. We show that the lower bound of the verification problem is PSPACE and the upper bound is coNEXPTIME. We present a tool implementing the compilation and evaluate the experimental results obtained on a complex scenario of multiple aircraft operating a recently proposed prototype for air-traffic collision avoidance.

1 Introduction

In the area of multi-agent systems (MAS) there is a long-standing interest in analysing the outcomes resulting from the strategic interplay among the agents. A variety of increasingly expressive logics have been put forward over the years to express strategic properties. These range from coalition logic (Pauly 2002), to alternating-time temporal logic (Alur, Henzinger, and Kupferman 2002), coordination logic, strategy logic (Mogavero, Murano, and Vardi 2010; Chatterjee, Henzinger, and Piterman 2007), as well as logics for equilibrium analysis (Gutierrez et al. 2018) and resource-bounded versions (Alechina et al. 2015) of these.

A key object of study in this line of research is the resulting verification problem, normally formulated as the model checking problem, i.e., to decide whether, given a system S and a specification φ, it is the case that $S \vDash \varphi$. In addition to the expressiveness of the specification language, the complexity of the model checking problem also depends on a number of modelling factors, including whether the actors are assumed to have complete or incomplete information and whether or not they have perfect recall. In a MAS setting, agents are normally assumed to have incomplete information, which, even without perfect recall, makes the resulting model checking problem at least EXPTIME for frequently used specification languages such as ATL* or strategy logic.

In all these approaches, the MAS is modelled either as a concurrent game structure (Alur, Henzinger, and Kupferman 2002) or as an interpreted system (Halpern and Fagin 1985), by providing the states, the possible actions, and the transitions for the agents and the system as a whole. These models, also used in knowledge representation (Halpern and Fagin 1985), are well suited to encode traditional MAS where the agents are directly realised by programmers.

An emerging generation of MAS applications, ranging from autonomous systems, swarm robotics and beyond, cannot, however, be modelled via traditional formalisms as above. These are MAS in which the agents' perception system is synthesised from data via machine learning methods and implemented via neural networks (NNs). No formalism, nor verification method exists to conduct an analysis of the possible strategic interplay arising in such systems. The aim of this paper is to make a first contribution in this area.

We introduce a novel formalisation of MAS in which the agents are endowed with a perception mechanism realised via NNs combined with a symbolic action selection mechanism. We call these neural-symbolic agents in line with recent proposals in the area (Lamb et al. 2020). We introduce the verification problem for these systems against a bounded version of alternating-time temporal logic. The usefulness of bounded fragments has previously been shown in the context of knowledge representation and agent-environment systems (Akintunde et al. 2018). We develop and solve the resulting verification problem via a mixed-integer linear programming (MILP) formulation, present a tool developed for this task and evaluate the approach on an avionics advisory system for collision avoidance.

Related Work. There is a rich literature on verification of strategic properties of MAS. In addition the field has released open-source tools (Alur et al. 1998; Dembiński et al. 2003; Lomuscio, Qu, and Raimondi 2017; Hinton et al. 2006; Gutierrez et al. 2018) that can be used to verify MAS prototypes. While these lines adopt similar specification languages to the one here pursued, they do not support MAS in which part of the architecture is realised via NNs.

Over the past three years numerous methods have emerged to verify NNs. Most of these address ReLU-based activation functions, which also equip the agents' perception systems that we study here. Complete approaches differ in the underlying methods, whether based on optimisation (Dvijotham et al. 2018), SAT (Katz et al. 2019), search (Wang et al. 2018), or MILP (Lomuscio and Maganti

2017). Incomplete, abstraction-based approaches have also been put forward (Singh et al. 2019; Gehr et al. 2018). These approaches differ on the precision and scalability of the networks that they can analyse. While the literature on this subject is growing rapidly and is supported by dedicated events, all these approaches differ from ours in that they study NNs in isolation and not as part of a closed-loop system.

The area of computer vision has also seen an increased interest in the development of methods to assess the correctness and resilience of perception systems based on NNs. This has followed the original interest in adversarial attacks (Goodfellow, Shlens, and Szegedy 2014), which exploit the fragility of neural classifiers. Approaches to provide verification methods for convolutional NNs (Kouvaros and Lomuscio 2018) have been proposed and counterexamples have been used for learning (Dreossi et al. 2018). Differently from the present contribution, this body of work focuses on the resilience of the classifier against input noise and does not address the combination of these with decision making or control, nor indeed closed-loop MAS.

More related to the topic of the present investigations are recent contributions in which closed-loop systems with neural network controllers are analysed. (Dutta, Chen, and Sankaranarayanan 2019; Xiang et al. 2018; Sun, Khedr, and Shoukry 2019; Huang et al. 2019) report approaches based on reachable set estimation methods; (Ivanov et al. 2019) presents a method based on hybrid system analysis. These and other similar approaches deal with closed-loop systems in which the controller (the agent in our terminology) is realised entirely via a NN, and so does not combine neural-symbolic elements as we do here. The emphasis of the above works are on safety and not on strategic reasoning, which is our objective. Close to this work is our own previous work on the subject, in which we put forward models for neural agent-environment systems of (Akintunde et al. 2018; 2020a), which are then verified via MILP as we do here. The present article presents several advances over previous material. Firstly, at the modelling level, we here present a novel loosely coupled neural-symbolic architecture which enables us to capture perception systems realised via neural classifiers paired with logic-based decision mechanisms. We are not aware of other proposals in this context. Secondly, we develop the first available treatment for verifying strategic properties of MAS which include neural components. Thirdly, we provide the same lower and upper bounds for the verification problem as those studied for the less expressive specifications (bounded CTL) and agent models presented in (Akintunde et al. 2020a). Lastly, the implementation provided is the first tool for the verification of strategic properties in MAS as we show in a real-life avionics application.

2 Background

In this section we summarise and fix the notation on some of the key notions used later in the paper.

Piecewise-linear functions. A function $f(x_1, \ldots, x_m)$ is said to be *linear* if $f = \sum_i c_i x_i + b$ for $(c_1, \ldots, c_m) \in \mathbb{R}^m$, $b \in \mathbb{R}$. A *piecewise-linear* (PWL) function is a function whose domain can be partitioned into a collection of intervals such that the function is linear on each interval.

Piecewise-linear feed-forward neural networks. A *feed-forward neural network* (FFNN) is a vector-valued function $f{:}\mathbb{R}^m \to \mathbb{R}^n$ defined as follows. The function f is defined by composing a sequence of *layers*, $f_1, \ldots, f_k$, where each layer f_i is the composition of a weighted sum and a non-linear *activation* function. That is, $f_i(x_i) \triangleq act_i(W_i f_{i-1}(x_{i-1}) + b_i)$, where x_i is the input to the i-th layer, act_i is the activation function of the i-th layer and $W_i f_{i-1} + b_i$ is the weighted sum of the previous layer's output for a weight matrix W_i and a bias vector b_i. A FFNN is said to be *piecewise-linear* if it contains only PWL activation functions. One of the most popular PWL activation functions (and activation functions in general) is the Rectified Linear Unit (ReLU), defined as $\mathrm{ReLU}(x) \triangleq \max(0, x)$, which is widely used in supervised learning tasks because of its effectiveness in training (Nair and Hinton 2010).

Reachability problem. Given a PWL FFNN $f{:}\mathbb{R}^m \to \mathbb{R}^n$ and a set $Out \subseteq \mathbb{R}^n$, the *reachability* problem concerns establishing whether there is an input $x \in \mathbb{R}^m$ such that $f(x) \in Out$. In the context of MILP-based verification of FFNNs, the reachability problem is addressed by transforming it into the feasibility problem of a MILP program.

Mixed Integer Linear Programming. An expression $f(x) \square b$ is said to be a *linear constraint* if $f(x)$ is a linear function, $\square \in \{\leq, =, \geq\}$, $b \in \mathbb{R}$ and variables x take values from $\mathbb{R}$. An expression of the form $x \in \mathbb{Z}$ is said to be an *integrality constraint*. In this paper we are only interested in the feasibility problem of linear programs, hence, we define a *linear program* to be simply a finite set of linear constraints[1]. A *mixed integer linear program* (MILP) is a linear program whose set of constraints includes integrality constraints. The *feasibility problem* of a MILP answers to the question whether there exists a *feasible solution* to the program, that is, an assignment to the variables that satisfies all constraints. A set $S \subseteq \mathbb{R}^n$ is said to be *linearly definable* if there exists a linear program C_S such that S is the feasible region (the set of all possible feasible solutions) of C_S.

MILP formulation of the reachability problem. The reachability problem has an exact MILP representation: the corresponding MILP program is feasible iff the answer to the verification problem is *yes* (Lomuscio and Maganti 2017). In the MILP representation, the linear components of the problem are given as linear constraints. Its piecewise-linear components are given as a set of *implication clauses*, one for each linear interval, which are expressed using the big-M method (Griva, Nash, and Sofer 2009). In particular, each implication clause $h(x) \leq 0 \Rightarrow g(x) \leq 0$ (where $h(x)$, $g(x)$ are linear expressions) can be encoded with the following MILP constraints:

$$h(x) \geq -M\delta; \qquad g(x) \leq M(1-\delta).$$

Above δ is a binary variable and M is a sufficiently large constant. So, if $h(x) \leq 0$, then $\delta = 1$ by the first clause, which gives $g(x) \leq 0$ by the second clause. Often the tighter the over-estimation of the actual bounds of $h(x)$ and $g(x)$ by M, the more efficiently the resulting MILP can be

[1] Defined in the standard way, a *linear program* is an optimisation problem whereby a linear objective function is sought to be maximised subject to a set of linear constraints.

solved. Given this state-of-the-art approaches carry out analyses based on interval arithmetic to derive tight bounds for each of the linear functions composing the FFNN in question (Botoeva et al. 2020; Tjeng, Xiao, and Tedrake 2019).

3 Neural Interpreted Systems

We here present a novel semantics for neural MAS called neural interpreted systems (NIS). NIS extend *interpreted systems* (IS) (Fagin et al. 1995), a standard semantics for MAS, previously used to reason about the strategic interplay in MAS (Lomuscio, Qu, and Raimondi 2017). Differently from IS, which assume purely symbolic agents, NIS uniquely combine connectionist with symbolic models; agents are endowed with a neural-based perception unit, used to gather information from the environment, and act upon their observations via traditional symbolic control logic. NIS allow for the natural modelling of hybrid architectures combining traditionally programmed control units with perception units synthesised from data. Autonomous vehicles, for instance, use this architecture to neurally interpret the images from their cameras and symbolically decide the required steering angle of the vehicle on the basis of said interpretation (Dreossi, Donzé, and Seshia 2019). They can be seen as a loosely-coupled neural-symbolic agent model (Garcez, Lamb, and Gabbay 2009).

We begin with a MAS composed of K *neural-symbolic agents* $Agt = \{1, \dots, K\}$ acting in an environment E. We often treat the environment as a special agent thereby considering a MAS as composed of the set $Agt \cup \{E\}$ of agents. Each agent $a \in Agt$ is described by the following.

Definition 1 (Neural-symbolic Agent)**.** *A* neural-symbolic agent*, or agent, is a tuple* $a = (L_a, obs_a, Act_a, prot_a, tr_a)$*, where*

- $L_a = Prv_a \times Per_a$ *is a nonempty (possibly infinite)* set of local states*. A* local state *is a pair* (prv_a, per_a) *of a* private state $prv_a \in Prv_a$ *and a* percept $per_a \in Per_a$ *storing the state of the environment observed by the agent.*
- $obs_a \colon L_a \times L_E \to Per_a$ *is an* observation function *mapping pairs of local states* $L_a \subseteq \mathbb{R}^h \times \mathbb{R}^n$ *and environment states* $L_E \subseteq \mathbb{R}^m$ *to percepts* $Per_a \subseteq \mathbb{R}^n$ *and is implemented via a PWL FFNN*[2] $f_a \colon \mathbb{R}^{h+n+m} \to \mathbb{R}^n$*.*
- Act_a *is a nonempty finite* set of actions*,*
- $prot_a \colon L_a \to 2^{Act_a} \setminus \{\varnothing\}$ *is a* local protocol function *defining the actions the agent can perform in a local state.*
- $tr_a \colon L_a \times Act_1 \times \cdots \times Act_K \times Act_E \to Prv_a$ *is a* local transition function *determining the next private state given a local state and a tuple of actions performed by all agents.*

The environment's description is similar to the agents' descriptions but has no percepts and observation function.

Definition 2 (Environment)**.** *The* environment *is a tuple* $E = (L_E, Act_E, prot_E, tr_E)$*, where* $L_E \subseteq \mathbb{R}^m$ *is a nonempty (possibly infinite) set of local states,* Act_E *is a finite set of actions,* $prot_E : L_E \to 2^{Act_E} \setminus \{\varnothing\}$ *is a protocol function and* $tr_E : L_E \times Act_1 \times \ldots \times Act_K \times Act_E \to L_E$ *is a local transition function.*

[2]Note that in practice the observation function may be implemented via a PWL combination of a number of PWL FFNNs.

Similarly to IS, the agents in a NIS are only aware of their local state. Previous proposals have used perception modules (Wooldridge and Lomuscio 1999) implemented symbolically. Here instead the agents in a NIS observe the state of the environment via the neural observation function described above. As a FFNN is a real vector-valued function, we encode the environment's set of states and the agents' sets of percepts as subsets of $\mathbb{R}^m$ and $\mathbb{R}^n$. The agents' private states and percepts encapsulate all the information accessible to the agents' decision making process; we call a pair of a private state and a percept a *local state*. In particular, given the current local state $\ell_a = (prv_a, per_a)$ for each $a \in Agt$ and the current local state ℓ_E of the environment, the operational cycle of the agents is described as follows. First, every agent and the environment chooses a protocol-compliant action α_a, i.e., $\alpha_a \in prot_a(\ell_a)$ for $a \in Agt \cup \{E\}$. The agents synchronously perform the selected actions. Then, every agent $a \in Agt$ updates the private part of its local state according to its local transition function to $prv'_a = tr_a(\ell_a, \alpha_1, \dots, \alpha_K, \alpha_E)$ producing intermediate local state $\ell'_a = (prv'_a, per_a)$. Similarly, the environment updates its local state to $\ell'_E = tr_E(\ell_E, \alpha_1, \dots, \alpha_K, \alpha_E)$. Finally, every agent $a \in Agt$ observes the updated state of the environment and with the information stored in its updated local state generates a percept $per'_a = obs_a(\ell'_a, \ell'_E)$ with which it updates the perception part obtaining the new local state $\ell''_a = (prv'_a, per'_a)$.

We hereafter assume that the local transition functions can be expressed as PWL functions. Note that this does not prevent NIS from modelling a wide spectrum of real-world use cases: if the transition function has a degree of nonlinearity, it can be approximated by a PWL FFNN to an arbitrary level of precision (D'Ambrosio, Lodi, and Martello 2010).

We now proceed to define the NIS comprising both the agents and the environment, and their associated temporal models. For the definitions we will require the notions of *global states* and *joint actions*. A *global state* $q = (\ell_1, \dots, \ell_K, \ell_E)$ is a tuple of local states for all the agents in the system; it describes the system at a particular instant of time. Given $\ell_a = (prv_a, per_a)$ for $a \in Agt$, a global state $q = (\ell_1, \dots, \ell_K, \ell_E)$ and an agent a, we write $lprv_a(q)$, $lper_a(q)$ and $loc_a(q)$ to denote the private part $lprv_a(q) = prv_a$ and the perception part $lper_a(q) = per_a$ of the local state $loc_a(q) = \ell_a$ of agent a in q. For a global state q and a local protocol $prot_a$ we often write $prot_a(q)$ to mean the set of actions that can be performed by agent a at q, i.e., $prot_a(loc_a(q))$. The set $G = L_1 \times \cdots \times L_K \times L_E$ of possible global states is the Cartesian product of the agents' sets of local states. A *joint action* $\alpha = (\alpha_1, \dots, \alpha_K, \alpha_E)$ is a tuple of local actions for all the agents in the system. The set $ACT = Act_1 \times \cdots \times Act_K \times Act_E$ of all possible joint actions is the Cartesian product of the agents' sets of local actions. The local transition functions induce a *global transition function* $tr : G \times ACT \to G$ where, for $q \in G$, $(\alpha_1, \dots, \alpha_K, \alpha_E) \in ACT$, $tr(q, \alpha_1, \dots, \alpha_K, \alpha_E) = q'$ iff

- $\alpha_E \in prot_E(q)$ and $tr_E(loc_E(q), \alpha_1, \dots, \alpha_K, \alpha_E) = loc_E(q')$; i.e., the environment's action is protocol compliant and its local state is updated as per its local transition function.

- $\alpha_a \in prot_a(q)$, $tr_a\big(loc_a(q), \alpha_1, \ldots, \alpha_K, \alpha_E\big) = lprv_a(q')$ and $obs_a((lprv_a(q'), lper_a(q)), loc_E(q')) = lper_a(q')$ for every agent $a \in Agt$; i.e., the agent's action is protocol compliant, the private part of its local state is updated as per its local transition function and the perception part of its local state is updated as per its observation function.

Definition 3 (Neural Interpreted System)**.** *Given a set of agents Agt, an environment E and a set of atomic propositions AP, a* neural interpreted system *(NIS) is a tuple $\mathcal{S} = (Agt \cup \{E\}, I, V)$, where:*

- *$I \subseteq G$ is a linearly definable set of* initial *global states.*
- *$V\colon AP \to 2^G$ is a labelling function specifying the atomic propositions that are true in each global state.*

The labelling function in a NIS is hereafter assumed to be expressible as a PWL function. In particular, for each $p \in AP$, $V(p)$ returns a description of a subset of G that is expressible as a Boolean PWL function.

Example 1. Consider as a running example two pilots operating two aircraft each equipped with an airborne traffic collision avoidance system (CAS) implemented via PWL FFNNs. Each CAS at every second will issue an advisory to the respective ownship and intruder pilots. The pilot is required to respond to the advisory by applying an acceleration to the aircraft in order to have the aircraft's climbrate compliant with the issued advisory. Each pilot can be modelled as a neural-symbolic agent a, where the pilot's acceleration response is modelled as non-deterministic choice defined in a protocol function $prot_a$, and local states L_a encode information relevant to each pilot – the aircraft's climbrate as its private state and the previous advisory issued by the CAS as its percept. The remaining physical transition dynamics of the system are modelled in an environment agent E. We provide a full formalisation of this example in Section 6.

NIS are (uniquely) associated with temporal models which can be used to interpret our specification language. These are defined as follows.

Definition 4 (Model)**.** *Given a NIS $\mathcal{S} = (Agt \cup \{E\}, I, V)$, the* induced model *of $\mathcal{S}$, or simply the* model*, is a tuple $\mathcal{M}_\mathcal{S} = (G, ACT, T, V)$, where:*

- *G is the set of global states and ACT is the set of joint actions of $\mathcal{S}$.*
- *$T \subseteq G \times ACT \times G$ is the transition relation defined as $(q, \alpha, q') \in T$ iff $tr(q, \alpha) = q'$.*
- *V is a labelling function as in Definition 3.*

A *path* in a model $\mathcal{M}_\mathcal{S} = (G, ACT, T, V)$ is an infinite sequence of global states and joint actions $q^0\alpha^0 q^1\alpha^1 \ldots$ such that for every $i \geq 0$, $(q^i, \alpha^i, q^{i+1}) \in T$.

We verify NIS against a bounded variant of a restricted subset of ATL* (Alur, Henzinger, and Kupferman 2002), drawing inspiration from Real Time Computation Tree Logic (RTCTL) (Emerson et al. 1992). The satisfaction status of the formulae expressible in our language depends only on paths of a bounded length. This has been shown to be practically relevant by enabling the efficient identification of shallow bugs in a system's execution (Biere et al. 2003; Penczek, Woźna, and Zbrzezny 2002). In addition to this practical consideration, our restriction to a bounded fragment of ATL* follows the undecidability of the verification problem for unbounded formulae (Akintunde et al. 2020a).

Definition 5 (Bounded ATL*)**.** *Given a set of agents Agt, an environment E and a set of atomic propositions AP, we define the* specification language *bATL* with the following BNF:*

$$\varphi \ ::= \ p \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \langle\!\langle \Gamma \rangle\!\rangle X^k \varphi \mid [\![\Gamma]\!] X^k \varphi,$$

where $p \in AP$, $k \in \mathbb{N}$ and $\Gamma \subseteq Agt \cup \{E\}$.

The formula $\langle\!\langle \Gamma \rangle\!\rangle X^k \varphi$ is read as "the group of agents Γ can bring about φ at the k-th step". The formula $[\![\Gamma]\!] X^k \varphi$ is read as "irrespective of the actions of the agents in group Γ, φ will hold at the k-th time step." We inductively abbreviate bounded until $\langle\!\langle \Gamma \rangle\!\rangle(\varphi U^k \psi)$ as

$$\langle\!\langle \Gamma \rangle\!\rangle(\varphi U^1 \psi) \triangleq \psi \vee (\varphi \wedge \langle\!\langle \Gamma \rangle\!\rangle X \psi);$$
$$\langle\!\langle \Gamma \rangle\!\rangle(\varphi U^k \psi) \triangleq \psi \vee (\phi \wedge \langle\!\langle \Gamma \rangle\!\rangle X(\langle\!\langle \Gamma \rangle\!\rangle(\phi U^{k-1} \psi))),$$

with the meaning of "the group of agents Γ can bring about ψ at some point in the k following steps and φ holds until then"; the formula $[\![\Gamma]\!](\varphi U^k \psi)$ can be defined analogously.

Example 2. For our collision avoidance example we can specify a property expressing that the ownship agent has a strategy to ensure that the two aircraft are safe from a collision after k steps. Given two neural-symbolic agents $Agt = \{\text{own}, \text{int}\}$ referring to the ownship and intruder defined in Example 1 and taking $\Gamma = \{\text{own}\}$, we can define such a property as $\varphi^k = \langle\!\langle \text{own} \rangle\!\rangle X^k \,\mathsf{safe}$, where safe is an atomic proposition representing a safe region.

We now proceed to define the satisfaction relation for bATL* specifications on NIS-induced models. This uses the notion of a *strategy*.

Definition 6. *A* strategy *for an agent $a \in Agt \cup \{E\}$ is a function $s_a\colon L_a \to 2^{Act_a}$ s.t. if $\alpha \in s_a(\ell_a)$, then $\alpha \in prot_a(\ell_a)$.*

The strategies defined above follow the memoryless, incomplete information and non-uniform setting from (Lomuscio, Qu, and Raimondi 2017) as implemented in the toolkit MCMAS. We can combine a collection of strategies to define a *joint strategy* for a coalition of agents.

Definition 7 (Joint Strategy)**.** *Let $\Gamma \subseteq Agt \cup \{E\}$ be a coalition of agents. A* joint strategy *is a tuple $s_\Gamma = (s_1, s_2, \ldots, s_{|\Gamma|})$ of strategies s_a for each agent $a \in \Gamma$.*

We can now define the evaluation of bATL* formulae on NIS-induced models. Given a set of agents Γ and a global state q, we write q_Γ for the tuple of local states in q restricted to the agents in Γ. Similarly, for a joint action α_Γ, we write α_Γ for the tuple of local actions in α restricted to the agents in Γ. We write $(\mathcal{M}_\mathcal{S}, q) \vDash \varphi$ to mean the bATL* formula φ is satisfied at state q in $\mathcal{M}_\mathcal{S}$; this is often abbreviated $q \vDash \varphi$.

Definition 8 (Satisfaction)**.** *For a model $\mathcal{M}_\mathcal{S} = (G, ACT, T, V)$, a global state q^0, and a bATL* formula φ, the* satisfaction *of φ at q^0, denoted $q^0 \vDash \varphi$, is defined as follows:*

$q^0 \vDash p$ iff $q^0 \in V(p)$, for $p \in AP$;
$q^0 \vDash \varphi \vee \psi$ iff $q^0 \vDash \varphi$ or $q^0 \vDash \psi$;

$q^0 \vDash \varphi \wedge \psi$ *iff* $q^0 \vDash \varphi$ *and* $q^0 \vDash \psi$;

$q^0 \vDash \langle\!\langle \Gamma \rangle\!\rangle X^k \varphi$ *iff there is a joint strategy* s_Γ *and an action* $\alpha_\Gamma^0 \in s_\Gamma(q_\Gamma^0)$ *such that all states* q^1 *with* $(q^0, \alpha^0, q^1) \in T$ *are such that there is an action* $\alpha_\Gamma^1 \in s_\Gamma(q_\Gamma^1)$ *such that all states* q^2 *with* $(q^1, \alpha^1, q^2) \in T$ *are such that, etc., up to state* q^k*, and we have that* $q^k \vDash \varphi$;

$q^0 \vDash [\![\Gamma]\!] X^k \varphi$ *iff for all joint strategies* s_Γ *and actions* $\alpha_\Gamma^0 \in s_\Gamma(q_\Gamma^0)$ *there is a state* q^1 *with* $(q^0, \alpha^0, q^1) \in T$ *such that for all actions* $\alpha_\Gamma^1 \in s_\Gamma(q_\Gamma^1)$ *there is a state* q^2 *with* $(q^1, \alpha^1, q^2) \in T$*, etc., up to* k*, and we have that* $q^k \vDash \varphi$.

A specification φ is said to be *satisfied* by a NIS $\mathcal{S}$, denoted $\mathcal{S} \vDash \varphi$, iff $(\mathcal{M}_\mathcal{S}, q) \vDash \varphi$ for every $q \in I$. This forms the basis of the following verification problem:

Definition 9 (Verification Problem)**.** *Given a NIS* $\mathcal{S}$ *and a specification* $\varphi \in$ *bATL***, determine whether* $\mathcal{S} \vDash \varphi$.

4 Verification of Neural Interpreted Systems

We here develop decision procedures for the verification problem of Definition 9. These extend the procedures devised for a neural (non-symbolic) agent operating on a non-deterministic environment system (NANES) against a bounded fragment of CTL (Akintunde et al. 2020a). As in previous work the approach is based on recasting the verification problem as the feasibility problem of a particular mixed-integer linear program; this is possible due to the fact that NIS are based on PWL functions. Differently from previous work, however, we here need to deal with strategy quantifications, and agent architectures that include neural and symbolic components. We start by observing a logical equivalence that applies to bATL*.

Normal form of specifications. We show that specifications in bATL* can be rewritten using $\langle\!\langle \Gamma \rangle\!\rangle X$ and $[\![\Gamma]\!] X$ only.

Lemma 1. *Given a NIS* $\mathcal{S}$*, its induced model* $\mathcal{M}_\mathcal{S}$*, a state* $q \in G$*, and a formula* $\varphi \in$ *bATL***, we have that:*

$$q \vDash \langle\!\langle \Gamma \rangle\!\rangle X^k \varphi \quad \textit{iff} \quad q \vDash \underbrace{\langle\!\langle \Gamma \rangle\!\rangle X \cdots \langle\!\langle \Gamma \rangle\!\rangle X}_{k \text{ repetitions}} \varphi \tag{1}$$

$$q \vDash [\![\Gamma]\!] X^k \varphi \quad \textit{iff} \quad q \vDash \underbrace{[\![\Gamma]\!] X \cdots [\![\Gamma]\!] X}_{k \text{ repetitions}} \varphi \tag{2}$$

Proof. We prove clause (1), clause (2) can be proved similarly. Let $q^0 = q$. We have that $q^0 \vDash \langle\!\langle \Gamma \rangle\!\rangle X^k \varphi$ iff there is a joint strategy s_Γ and an action $\alpha_\Gamma^0 \in s_\Gamma(q_\Gamma^0)$ such that all states q^1 with $(q^0, \alpha^0, q^1) \in T$ are such that there is an action $\alpha_\Gamma^2 \in s_\Gamma(q_\Gamma^1)$ such that all states q^2 with $(q^1, \alpha^1, q^2) \in T$ are such that, etc., up to state q^k, and we have that $q^k \vDash \varphi$. This is iff there is a joint strategy s_Γ^0 with $s_\Gamma^0(q^0) = \{\alpha^0\}$ such that all states q^1 with $(q^0, \alpha^0, q^1) \in T$ are such that there is a joint strategy $s_\Gamma^1(q^1)$ with $s_\Gamma^1(q^1) = \{\alpha^1\}$ such that all states q^2 with (q^1, α^1, q^2) are such that, etc., up to state q^k, and we have that $q^k \vDash \varphi$. The latter holds iff $q^0 \vDash \langle\!\langle \Gamma \rangle\!\rangle X \cdots \langle\!\langle \Gamma \rangle\!\rangle X \varphi$. □

Due to Lemma 1, in the following we assume that any specification is written by using $\langle\!\langle \Gamma \rangle\!\rangle X$ and $[\![\Gamma]\!] X$ only.

Notation. A *variable assignment* $\mathfrak{a}$ for a MILP π is a function $\mathfrak{a}\colon vars(\pi) \to \mathbb{R}$ that assigns a value to each variable in π. An assignment $\mathfrak{a}$ *satisfies* π, denoted $\mathfrak{a} \vDash \pi$, if $\mathfrak{a}(\delta) \in \{0, 1\}$ for each binary variable δ, $\mathfrak{a}(\iota) \in \mathbb{Z}$ for each integer variable ι, and all constraints in π are satisfied.

When encoding disjunctive cases, it is convenient to use so-called *indicator* constraints. These are constraints of the form $(\delta = v) \Rightarrow c$, where δ is a binary variable and $v \in \{0, 1\}$, stating that whenever the value of δ is v, the linear constraint c should hold. For instance, the two indicator constraints $(\delta_1 = 1) \Rightarrow x = 3$ and $(\delta_2 = 1) \Rightarrow x = 5$ with the constraint $\delta_1 + \delta_2 = 1$ impose that the value of x is either 3 or 5. Given a binary variable δ and a set of MILP constraints π, we write $(\delta = v) \Rightarrow \pi$ for the set of constraints $\{(\delta = v) \Rightarrow c \mid c \in \pi\}$.

Without loss of generality, we assume below that for each agent index $a \in \{1, \ldots, K, E\}$, there is a natural number b_a such that for each global state $q \in G$, the cardinality of $prot_a(q)$ is b_a (in practice, we can take b_a to be the maximum cardinality of $prot_a(q)$ for $q \in G$). We refer to b_a as the *action choice factor* (ACF) for agent a. We may assume that $prot_a$ is given as b_a PWL functions $prot_a^1, \ldots, prot_a^{b_a}$ mapping G to Act_a.

For a natural number n, we hereafter denote by $[n]$ the set $\{1, \ldots, n\}$, and for a finite set S, denote by $|S|$ the cardinality of S. Given a set of agents $\Gamma \subseteq Agt \cup \{E\}$, denote by $\overline{\Gamma} = Agt \cup \{E\} \setminus \Gamma$ the *complement of* Γ *with respect to* $Agt \cup \{E\}$. Further, denote by Ind_Γ the Cartesian product of all $[b_a]$ with $a \in \Gamma$ and $\{0\}$ for $a \in \overline{\Gamma}$. Therefore, each element of Ind_Γ is a tuple $\sigma = (\sigma_1, \ldots, \sigma_K, \sigma_E)$ of *action indices* that are non-negative integers, where $\sigma_a \in [b_a]$ if $a \in \Gamma$ and 0 if $a \in \overline{\Gamma}$. Ind_Γ represents *all distinct joint actions* of the agents in Γ and each element in Ind_Γ encodes one such joint action. As a consequence, $|Ind_\Gamma|$ is the number of all possible joint actions for the agents in Γ. We assume a natural (lexicographic) ordering of the elements of Ind_Γ, and denote by Σ_Γ^i the element number i in Ind_Γ. Given two tuples of integers σ^1 and σ^2 of the same length, denote by $\sigma^1 \cup \sigma^2$ the tuple σ such that $\sigma_a = \max(\sigma_a^1, \sigma_a^2)$.

Boldface letters ($\boldsymbol{x}$, $\boldsymbol{y}$, etc) denote tuples of real-valued MILP variables representing a global state and we call them *state variables*. Given a $(K + 1)$-tuple σ of natural numbers, $C_\sigma(\boldsymbol{x}, \boldsymbol{y})$ denotes the set of MILP constraints encoding the global transition $\boldsymbol{y} = tr(\boldsymbol{x}, prot_1^{\sigma_1}(\boldsymbol{x}), \ldots, prot_K^{\sigma_K}(\boldsymbol{x}), prot_E^{\sigma_E}(\boldsymbol{x}))$; this is admissible since tr and $prot_a^{\sigma_a}$ are PWL. For a literal l, $\mathcal{V}(l, \boldsymbol{x})$ denotes the set of MILP constraints over the state variables $\boldsymbol{x}$ that encode the set of global states at which l holds. For an atomic proposition p, $\mathcal{V}(p, \boldsymbol{x})$ expresses the set $V(p)$ and $\mathcal{V}(\neg p, \boldsymbol{x})$ expresses the set $G \setminus V(p)$ as MILP constraints on $\boldsymbol{x}$. We rely on $C_\sigma(\boldsymbol{x}, \boldsymbol{y})$ as building blocks for encoding the temporal evolution of the system and on $\mathcal{V}(l, \boldsymbol{x})$ for encoding the states of the system at which we stop the exploration. The former is formalised in the lemma below that follows from the fact that PWL functions can be represented by the feasible regions of MILPs.

Lemma 2. *Let* $C_\sigma(\boldsymbol{x}, \boldsymbol{y})$ *be a MILP program corresponding to* $\boldsymbol{y} = tr(\boldsymbol{x}, prot_1^{\sigma_1}(\boldsymbol{x}), \ldots, prot_K^{\sigma_K}(\boldsymbol{x}), prot_E^{\sigma_E}(\boldsymbol{x}))$.

$$\pi_{\mathcal{S},l}(\boldsymbol{x}) = \mathcal{V}(l,\boldsymbol{x})$$
$$\pi_{\mathcal{S},\varphi_1\vee\varphi_2}(\boldsymbol{x}) = (\delta = 1) \Rightarrow \pi_{\mathcal{S},\varphi_1}(\boldsymbol{x}) \;\cup\; (\delta = 0) \Rightarrow \pi_{\mathcal{S},\varphi_2}(\boldsymbol{x})$$
$$\pi_{\mathcal{S},\varphi_1\wedge\varphi_2}(\boldsymbol{x}) = \pi_{\mathcal{S},\varphi_1}(\boldsymbol{x}) \cup \pi_{\mathcal{S},\varphi_2}(\boldsymbol{x})$$
$$\pi_{\mathcal{S},\langle\!\langle\Gamma\rangle\!\rangle X\varphi}(\boldsymbol{x}) = \bigcup_{i=1}^{|Ind_\Gamma|} (\delta_i = 1) \Rightarrow \left(\bigcup_{j=1}^{|Ind_{\overline{\Gamma}}|} C_{\Sigma^i_\Gamma \cup \Sigma^j_{\overline{\Gamma}}}(\boldsymbol{x},\boldsymbol{y}_j) \right) \cup \sum_{i=1}^{|Ind_\Gamma|} \delta_i = 1 \;\cup\; \bigcup_{j=1}^{|Ind_{\overline{\Gamma}}|} \pi_{\mathcal{S},\varphi}(\boldsymbol{y}_j)$$
$$\pi_{\mathcal{S},[\![\Gamma]\!]X\varphi}(\boldsymbol{x}) = \bigcup_{i=1}^{|Ind_\Gamma|} \left(\bigcup_{j=1}^{|Ind_{\overline{\Gamma}}|} (\delta_{ij} = 1) \Rightarrow C_{\Sigma^i_\Gamma \cup \Sigma^j_{\overline{\Gamma}}}(\boldsymbol{x},\boldsymbol{y}_i) \;\cup\; \sum_{j=1}^{|Ind_{\overline{\Gamma}}|} \delta_{ij} = 1 \;\cup\; \pi_{\mathcal{S},\varphi}(\boldsymbol{y}_i) \right)$$

where the binary variables $\delta, \delta_i, \delta_{ij}$, the state variables $\boldsymbol{y_j}, \boldsymbol{y}_i$ and all auxiliary variables in $C_{\Sigma^i_\Gamma \cup \Sigma^j_{\overline{\Gamma}}}$ are fresh

Figure 1: Monolithic encoding $\pi_{\mathcal{S},\varphi}$ for $\varphi \in$ bATL*.

Given two states q and q' in $\mathcal{M}_\mathcal{S}$, we have that $q' = tr(q, prot_1^{\sigma_1}(q), \ldots, prot_K^{\sigma_K}(q), prot_E^{\sigma_E}(q))$ iff there is an assignment $\mathfrak{a}$ to $vars(C_\sigma(\boldsymbol{x},\boldsymbol{y}))$ such that $s = \mathfrak{a}(\boldsymbol{x})$, $s' = \mathfrak{a}(\boldsymbol{y})$, and $\mathfrak{a} \vDash C_\sigma(\boldsymbol{x},\boldsymbol{y})$.

We present the encodings for the verification problem.

Monolithic Encoding. We begin with an encoding which befits a verification procedure where only a single sequence of instructions is executed. Given a NIS $\mathcal{S}$ and a formula $\varphi \in$ bATL*, we construct a MILP $\pi_{\mathcal{S},\varphi}$, whose feasibility corresponds to the satisfaction of φ by a state in $\mathcal{M}_\mathcal{S}$.

Definition 10. *The* monolithic MILP encoding *for a NIS $\mathcal{S}$ and a specification $\varphi \in$ bATL*, denoted $\pi_{\mathcal{S},\varphi}$, is built inductively as the MILP $\pi_{\mathcal{S},\varphi}(\boldsymbol{x})$, for a tuple of fresh state variables $\boldsymbol{x}$, according to the rules in Figure 1.*

The encoding of the literals in Figure 1 is $\mathcal{V}(l,\boldsymbol{x})$. The disjunction and conjunction are encoded similarly to the corresponding cases for bCTL$_{\mathbb{R}^\leq}$ in (Akintunde et al. 2020a). The encoding of the coalitional operators closely follows the semantics of alternating existential and universal quantification of strategies. Existential quantification is encoded via disjunctions, while universal quantifications via conjunctions. N-ary disjunction is captured by using N binary variables $\delta_1, \ldots, \delta_N$ such that their sum amounts to 1 (and hence, only one can be true) and by employing indicator constraints enforcing that when δ_i is true, then the corresponding constraints must be satisfied. N-ary conjunction is captured by enforcing that all corresponding constraints are satisfied at the same time. The encoding of $\langle\!\langle\Gamma\rangle\!\rangle X$ results from generating $|Ind_{\overline{\Gamma}}|$ successor state variables $\boldsymbol{y}_j$ to ensure that for every strategy of the agents in $\overline{\Gamma}$, φ is satisfied at the corresponding successor. Symmetrically, the encoding of $[\![\Gamma]\!]X$ generates $|Ind_\Gamma|$ successor state variables $\boldsymbol{y}_i$.

It can be shown that the encoding is as intended.

Lemma 3. *Given a NIS $\mathcal{S}$, a formula $\varphi \in$ bATL*, and a state q in $\mathcal{M}_\mathcal{S}$ we have that $q \vDash \varphi$ iff there exists an assignment $\mathfrak{a}$ to $vars(\pi_{\mathcal{S},\varphi}(\boldsymbol{x}))$ such that $\mathfrak{a} \vDash \pi_{\mathcal{S},\varphi}(\boldsymbol{x})$ and $q = \mathfrak{a}(\boldsymbol{x})$.*

Having defined the encoding, we can re-use the verification procedure used for NANES and bCTL$_{\mathbb{R}^<}$ in (Akintunde et al. 2020a). Reported in Algorithm 1 with $mono$ set to `True`, it attempts to falsify $\mathcal{S} \vDash \varphi$ by searching for a state $q \in I$ such that $q \nvDash \varphi$ (equivalently, $q \vDash \neg\varphi$). To this purpose, the formula passed to the monolithic encoding is $\neg\varphi \wedge p_I$ in negation normal form, where the negation is pushed down through to the atoms. Here, for simplicity, we assume that

Algorithm 1 The MILP verification procedure.

```
procedure VERIFY(S, φ, mono)
    Input: NIS S = (Agt, E, I, V); formula φ ∈ bATL*
    Output: True/False
    feasible ← False
    φ' ← NNF(¬φ ∧ p_I)
    if mono then
        π_{S,φ'} ← MILP associated with S and φ'
        feasible ← MILP_SOLVER(π_{S,φ'})
    else
        Π_{S,φ'} ← Set of MILPs associated with S and φ'
        for π in Π_{S,φ'} do
            aux ← MILP_SOLVER(π)
            if aux is True then
                feasible ← True
                break
    return ¬feasible
```

p_I is an additional atomic proposition in AP used for labelling the states in I, and that $V(p_I) = I$.

Compositional Encoding. To provide a more scalable verification procedure, we follow (Akintunde et al. 2020a) by devising an encoding which creates separate MILP programs for each disjunctive case encountered in the monolithic encoding. Given a NIS $\mathcal{S}$ and a formula $\varphi \in$ bATL*, we define a set $\Pi_{\mathcal{S},\varphi}$ of MILP programs such that the satisfaction of φ by a state $q \in \mathcal{M}_\mathcal{S}$ corresponds to at least one of the programs $\pi \in \Pi_{\mathcal{S},\varphi}$ being feasible. Due to potentially multiple programs whose feasibility can be checked independently of each other, this encoding is particularly amenable to parallelisation.

We denote by $[C]$ the MILP program for the set C of linear constraints. Given sets $A = \{[A_1], \ldots, [A_p]\}$ and $B = \{[B_1], \ldots, [B_q]\}$ of MILP programs, we denote by $A \times B$ the *product* $\{[A_i \cup B_j] \,|\, i = 1, \ldots, p,\ j = 1, \ldots, q\}$.

Definition 11. *The* compositional MILP encoding *for a NIS $\mathcal{S}$ and a specification $\varphi \in$ bATL*, denoted $\Pi_{\mathcal{S},\varphi}$, is built inductively as the set of MILPs $\Pi_{\mathcal{S},\varphi}(\boldsymbol{x})$, for a tuple of fresh state variables $\boldsymbol{x}$, according to the rules in Figure 2.*

The compositional encoding operates on sets of MILPs. The encoding of the literals is straightforward and generates a singleton set containing the program corresponding to the literal. Disjunctions are dealt with by generating a different set of programs for each of the cases, and hence are

$$\Pi_{\mathcal{S},l}(\boldsymbol{x}) = \{[\mathcal{V}(l,\boldsymbol{x})]\},$$
$$\Pi_{\mathcal{S},\varphi_1\vee\varphi_2}(\boldsymbol{x}) = \Pi_{\mathcal{S},\varphi_1}(\boldsymbol{x}) \cup \Pi_{\mathcal{S},\varphi_2}(\boldsymbol{x}),$$
$$\Pi_{\mathcal{S},\varphi_1\wedge\varphi_2}(\boldsymbol{x}) = \Pi_{\mathcal{S},\varphi_1}(\boldsymbol{x}) \times \Pi_{\mathcal{S},\varphi_2}(\boldsymbol{x}),$$
$$\Pi_{\mathcal{S},\langle\!\langle\Gamma\rangle\!\rangle X\varphi}(\boldsymbol{x}) = \bigcup_{i=1}^{|Ind_\Gamma|}\left(\bigtimes_{j=1}^{|Ind_{\overline{\Gamma}}|}\left\{\left[C_{\Sigma_\Gamma^i\cup\Sigma_{\overline{\Gamma}}^j}(\boldsymbol{x},\boldsymbol{y}_j)\right]\right\}\times\Pi_{\mathcal{S},\varphi}(\boldsymbol{y}_j)\right),$$
$$\Pi_{\mathcal{S},[\![\Gamma]\!]X\varphi}(\boldsymbol{x}) = \bigtimes_{i=1}^{|Ind_\Gamma|}\left(\bigcup_{j=1}^{|Ind_{\overline{\Gamma}}|}\left\{\left[C_{\Sigma_\Gamma^i\cup\Sigma_{\overline{\Gamma}}^j}(\boldsymbol{x},\boldsymbol{y}_i)\right]\right\}\right)\times\Pi_{\mathcal{S},\varphi}(\boldsymbol{y}_i).$$

Figure 2: Compositional encoding $\Pi_{\mathcal{S},\varphi}$ for $\varphi \in$ bATL*.

implemented via the *union* operation over sets of programs. Conversely, conjunctions are encoded by taking all combinations of programs for each of the cases; it is implemented via the *product* operation over sets of programs. Therefore, the patterns of the form $\bigcup_i(\delta_i = 1) \Rightarrow C_i \cup \sum_i \delta_i = 1$ found in the monolithic encoding are replaced by $\bigcup_i\{[C_i]\}$, while "$\cup$" found in the monolithic encoding is replaced by "$\times$".

We can show this encoding is correct.

Lemma 4. *Given a NIS $\mathcal{S}$, a formula $\varphi \in$ bATL*, and a state q in $\mathcal{M}_\mathcal{S}$, we have that $q \vDash \varphi$ iff there is a MILP $\pi(\boldsymbol{x}) \in \Pi_{\mathcal{S},\varphi}(\boldsymbol{x})$ and an assignment $\mathfrak{a}$ to $vars(\pi(\boldsymbol{x}))$ such that $q = \mathfrak{a}(\boldsymbol{x})$ and $\mathfrak{a} \vDash \pi(\boldsymbol{x})$.*

The verification procedure of Algorithm 1 can also use the compositional encoding above, when *mono* is `False`. Notably, checking feasibility of the generated programs (lines 11–15) can be done sequentially, one after another, or in parallel across multiple processors. The theorem below states the correctness of the verification procedure for both the monolithic and compositional encodings.

Theorem 1. *Given a NIS $\mathcal{S}$ and a formula $\varphi \in$ bATL*, Algorithm 1 returns* False *iff $\mathcal{S} \not\models \varphi$.*

5 Complexity of the Verification Problem

In this section we analyse the computational complexity of the verification problem against bATL*. We identify the complexity of the verification problem to lie between PSPACE and coNEXPTIME. The lower bound follows from the lower bound for the verification problem of a neural-symbolic agent operating in non-deterministic environment systems (NANES) against a bounded fragment of CTL (Akintunde et al. 2020a). The upper bound follows from the monolithic version of the verification procedure devised in Section 4.

We start with the PSPACE lower bound. As is known, the CTL modalities EX and AX can be expressed in ATL as $\langle\!\langle Agt\cup\{E\}\rangle\!\rangle X$ and $[\![Agt\cup\{E\}]\!]X$, respectively. Moreover, we can model a neural-symbolic agent operating in a non-deterministic environment as a NIS where Agt is a singleton. Since the validity problem of QBF can be reduced to the verification problem of NANES against a bounded fragment of CTL, we obtain the following result.

Theorem 2. *Verifying NIS against bATL* is* PSPACE*-hard.*

Next, we analyse the upper bound. Recall that the monolithic verification procedure generates one MILP and then checks its (in)feasibility. The feasibility check can be done in non-deterministic polynomial time in the size of the mixed-integer linear program; guess an assignment to the binary variables and then check that the induced linear program is feasible in polynomial time (Papadimitriou and Steiglitz 1982). It remains to estimate the size of the MILP obtained by the monolithic encoding.

First we note that the MILP encodings of PWL FFNNs and of PWL functions are of linear size. We analyse now the encoding of the ATL-specific operators. We have that the size of $\pi_{\mathcal{S},\langle\!\langle\Gamma\rangle\!\rangle X\varphi}$ is $O(|Ind_\Gamma|\cdot|Ind_{\overline{\Gamma}}|\cdot|\mathcal{S}| + |Ind_{\overline{\Gamma}}|\cdot|\pi_{\mathcal{S},\varphi}|)$, while of $\pi_{\mathcal{S},[\![\Gamma]\!]X\varphi}$ is $O(|Ind_\Gamma|\cdot|Ind_{\overline{\Gamma}}|\cdot|\mathcal{S}| + |Ind_\Gamma|\cdot|\pi_{\mathcal{S},\varphi}|)$, where $|\mathcal{S}|$ denotes the number of bits required to represent $\mathcal{S}$. Observe that $|Ind_\Gamma|$ is bound by the total number of successors of a state $|Ind_{Agt\cup\{E\}}| = b_1\cdot\cdots\cdot b_K\cdot b_E$, which in turn is bound by b^{K+1}, where $b = \max(b_1,\ldots,b_K,b_E)$. This gives us the best and the worst case sizes of the monolithic encodings: for $\varphi = \langle\!\langle Agt\cup\{E\}\rangle\!\rangle X^d\neg p$, the size of $\pi_{\mathcal{S},\varphi}$ is $O\left(b^{K+1}(d\cdot|\mathcal{S}| + |V(p)|)\right)$ and for $\varphi = [\![Agt\cup\{E\}]\!]X^d\neg p$, the size of $\pi_{\mathcal{S},\varphi}$ is $O\left(b^{(K+1)\cdot d}(|\mathcal{S}| + |V(p)|)\right)$. When all $b_a = 1$ except for one, the program $\pi_{\mathcal{S},\varphi}$ is linearly large in $|\mathcal{S}|$ (since we can replace b^{K+1} by b), otherwise assuming that $b_a > 1$ for a non-constant number of agents, it is exponentially large in $|\mathcal{S}|$. Moreover, unless φ has only occurrences of temporal modalities of the form $\langle\!\langle Agt\cup\{E\}\rangle\!\rangle X^d$, $\pi_{\mathcal{S},\varphi}$ is exponentially large in the temporal depth of φ.

Given that the verification problem for $\mathcal{S}$ and φ does not hold if and only if the MILP $\pi_{\mathcal{S},\neg\varphi\wedge p_I}$ is feasible, we obtain the following upper bound.

Theorem 3. *Verifying NIS against bATL* is in co*NEXPTIME*.*

We observe that in the worst case the verification problem for NIS against bATL* is not harder than that of NANES against bounded CTL. However, for the latter systems verification against properties of the form AX^dp is coNP-complete, while for a NIS composed of a non-fixed number of agents the monolithic verification procedure does not provide a coNP upper bound for properties of the form $[\![Agt\cup\{E\}]\!]X^dp$. It remains an open question whether, when looking for initial states in I violating the specification, it is sufficient to consider states of polynomial size. The positive answer to this question would mean that the verification problem for NIS against bATL* is PSPACE-complete and against bounded safety properties is coNP-complete.

6 Implementation and Evaluation

In this section we present an implementation of the verification procedures described in Section 4 in a toolkit called VENMAS (Akintunde et al. 2020b) which we evaluated on an air-traffic collision avoidance system.

VENMAS takes as input a NIS $\mathcal{S}$ and a bATL* specification φ. The observation function for each agent is given as linear combination of FFNNs. The local protocol and transition functions for each agent are given as PWL functions. The set I of initial states is given in the form of a hyper-rectangle $[l_1, u_1] \times \cdots \times [l_\mu, u_\mu]$. The user can specify a parameter determining whether the monolithic or compositional procedure with parallel or sequential execution is to be used. The toolkit returns whether or not the specification φ holds on $\mathcal{S}$. We used Python to implement the tool and relied on Gurobi ver. 9.0 (Gu, Rothberg, and Bixby 2020) as a back-end to resolve the feasibility of the generated MILPs.

Two-Agent Aircraft Collision Avoidance Scenario. We validated the toolkit by using a two-agent extension of the scenario introduced in (Julian and Kochenderfer 2019; Julian et al. 2019) and studied in (Akintunde et al. 2020a), VerticalCAS. It was originally a single-agent system composed of *ownship* and *intruder* aircraft where the ownship pilot's aim was to respond to a neural network-controlled collision avoidance system called VerticalCAS. Each second the system issues an *advisory* that together with the current climbrate of the ownship determines a range of accelerations from which the ownship pilot can non-deterministically choose to apply, in order to avoid a *near mid-air collision (NMAC)*, a region where the aircraft are separated by less than 100 ft vertically and 500 ft horizontally. The intruder is assumed to follow a horizontal trajectory throughout.

Instead we here consider both aircraft to be controlled by their own independent VerticalCAS systems, removing the assumption that the intruder follows a horizontal trajectory; the intruder can now climb and descend independently to the ownship. This makes for a more complex scenario with a higher degree of branching in the resulting temporal evolution of the system. For simplicity we hereafter refer to this novel two-agent extension of our construction as VCAS[2].

We describe the global states of the scenario using the set of tuples $G = (h, \dot{h}_{\text{own}}, \dot{h}_{\text{int}}, \tau, ad_{\text{own}}, ad_{\text{int}}) \in [-3000, 3000] \times [-2500, 2500] \times [-2500, 2500] \times [0, 40] \times [9] \times [9]$, where h (ft) is the intruder altitude relative to the ownship, $\dot{h}_{\text{own}}$ (ft/sec) is the ownship's vertical climbrate, $\dot{h}_{\text{int}}$ (ft/sec) is the intruder's vertical climbrate, τ (secs) is the time to loss of horizontal separation of the aircraft, ad_{own} is the previous advisory issued to the ownship, and ad_{int} is the previous advisory issued to the intruder.

Each VerticalCAS system is composed of nine ReLU-FFNNs $F = \{(f_i : \mathbb{R}^4 \to \mathbb{R}^9) : i \in [9]\}$, one for each previously-issued advisory with four inputs, seven hidden layers of 45 nodes and nine outputs representing the score of each possible advisory.

Encoding VCAS[2] as a Neural Interpreted System. Here we model VCAS[2] as a NIS. First, the environment states ℓ_E are tuples $(h, \dot{h}_{\text{own}}, \dot{h}_{\text{int}}, \tau)$, hence $L_E = [-3000, 3000] \times [-2500, 2500] \times [-2500, 2500] \times [0, 40]$. There are two agents $a \in \{\text{own}, \text{int}\}$ defined almost identically. Formally their components are as follows:

- local states ℓ_a are tuples $(\dot{h}_a, ad_a)$, where the current climbrate $\dot{h}_a$ constitutes the private state and the previously issued advisory ad_a corresponds to the percept, so $L_a = [-2500, 2500] \times [9]$,
- actions correspond to accelerations; we set $Act_a = \{0, \pm 3.0, \pm 7.33, \pm 9.33, \pm 9.7, \pm 11.7\}$,
- the protocol function returns the available accelerations, $prot_a(\ell_a) = \mathsf{accs}(\mathsf{compliant}(\ell_a), ad_a)$,
- for accelerations $\ddot{h}_a \in Act_a$, we define the local transition $tr_a((\dot{h}_a, ad_a), \ddot{h}_a, \cdot, \cdot) = \dot{h}_a + \ddot{h}_a \Delta\tau = \dot{h}'_a$,
- the observation function perceives the environment by computing the advisory according to VerticalCAS and information stored in the intermediate agent's local state $\ell'_a = (\dot{h}'_a, ad_a)$ and updated environment's local state ℓ'_E, so $obs_a(\ell'_a, \ell'_E) = ad'_a = \arg\max(\mathsf{apply}_a(f_{ad_a}, \ell'_E))$,

where $\mathsf{compliant}(\ell_a)$ is `True` iff $\dot{h}_a$ is compliant with ad_a; $\mathsf{accs}(\beta, ad_a) \subseteq 2^{Act_a}$ is a set of accelerations such that $\mathsf{accs}(\texttt{True}, \cdot) = \{0\}$ and $\mathsf{accs}(\texttt{False}, ad_a)$ is the set of permitted accelerations for the advisory ad_a; $\Delta\tau = 1$; $\mathsf{apply}_a : F \times L_E \to \mathbb{R}^9$ computes the output of a neural network on an environment state, such that $\mathsf{apply}_{\text{own}}(f, \ell_E) = f(h, \dot{h}_{\text{own}}, \dot{h}_{\text{int}}, \tau)$ and $\mathsf{apply}_{\text{int}}(f, \ell_E) = f(-h, \dot{h}_{\text{int}}, \dot{h}_{\text{own}}, \tau)$; $\arg\max : \mathbb{R}^9 \to [9]$ gives the index of the largest component.

Finally, the environment has one dummy action ε, giving $\alpha_E = \{\varepsilon\}$ and protocol function $prot_E(\ell_E) = \{\varepsilon\}$ for all $\ell_E \in L_E$, and the transition function is defined as

$$tr_E\left(\begin{bmatrix} h \\ \dot{h}_{\text{own}} \\ \dot{h}_{\text{int}} \\ \tau \end{bmatrix}, \ddot{h}_{\text{own}}, \ddot{h}_{\text{int}}, \varepsilon\right) = \begin{bmatrix} h' \\ \dot{h}_{\text{own}} + \ddot{h}_{\text{own}}\Delta\tau \\ \dot{h}_{\text{int}} + \ddot{h}_{\text{int}}\Delta\tau \\ \tau - \Delta\tau \end{bmatrix},$$

where $h' = h - \Delta\tau(\dot{h}_{\text{own}} - \dot{h}_{\text{int}}) - 0.5\Delta\tau^2(\ddot{h}_{\text{own}} - \ddot{h}_{\text{int}})$.

Given $Agt = \{\text{own}, \text{int}\}$ and atomic propositions $AP = \{\mathsf{NMAC}, \mathsf{safe}\}$, define the valuation function $V : AP \to 2^G$ as $V(\mathsf{NMAC}) = \{q \in G \mid |h| \leq 100\}$ and $V(\mathsf{safe}) = \{q \in G \mid h > 100 \text{ or } h < -100\}$ for global states $q = (h, \cdot, \cdot, \tau, \cdot, \cdot)$ with τ never close to zero.

We take the set of initial states $I = [h_I - 2, h_I + 2] \times \{-5.0\} \times \{5.0\} \times \{25.0\} \times \{1\} \times \{1\}$, where $h_I \in \{-30, 10, 50, 90\}$ft. These represent potentially dangerous encounters with the aircraft inside the NMAC region with a descending ownship and an ascending intruder. When $h_I > 0$, the ownship is below the intruder, and when $h_I < 0$, the ownship is above the intruder. We fix an action choice factor of 2.

We can now construct a Neural Interpreted System for VCAS[2] for our two agents $a \in Agt$ as defined above.

Experimental Results. We used VENMAS to evaluate VCAS[2] against the two specifications: $\varphi_1^k = \langle\!\langle \text{own} \rangle\!\rangle X^k \, \mathsf{safe}$, expressing the fact that the ownship has a strategy for the system to be in a safe configuration after k steps, and $\varphi_2^k = [\![\text{int}]\!] X^k \, \mathsf{safe}$, expressing the fact that the intruder has no strategy to avoid the system to enter a safe configuration after k steps (note that φ_1^k implies φ_2^k but not the other way around). We evaluated the formulas with the set of initial states I defined as above. All results were obtained on a machine with an Intel Core i7-7700K CPU with 16GB RAM, running a 64-bit version of Ubuntu 18.04. We denote by MONOLITHIC the results for the monolithic procedure

	COMP-PAR				COMP-SEQ				MONOLITHIC			
h_I	−30	10	50	90	−30	10	50	90	−30	10	50	90
φ_1^1	0.33	0.39	0.34	0.48	0.32	0.34	0.32	0.97	0.04	0.08	0.05	0.04
φ_1^2	1.67	4.07	13.63	13.88	1.29	3.16	41.82	42.64	0.53	4.17	0.16	0.16
φ_1^3	104.29	–	–	–	59.48	–	–	–	303.35	994.45	0.67	0.85
φ_1^4	–	–	–	–	–	–	–	–	–	–	3.20	3.40
φ_1^5	–	–	–	–	–	–	–	–	–	–	9.63	9.22
φ_2^1	0.35	0.37	0.35	0.37	0.32	0.37	0.32	0.57	0.04	0.08	0.05	0.04
φ_2^2	1.23	3.07	2.21	2.47	1.17	2.99	5.66	5.76	0.52	4.25	0.16	0.16
φ_2^3	55.20	564.85	69.38	100.93	22.14	1823.65	213.81	309.56	906.68	933.99	0.66	0.84
φ_2^4	–	–	–	–	–	–	–	–	–	–	3.21	3.38
φ_2^5	–	–	–	–	–	–	–	–	–	–	9.51	9.23

Table 1: Verification times for VCAS[2] against properties φ_1^k and φ_2^k for various k and h_I. Grey cells indicate a `False` result, otherwise a `True` result was obtained. Dashes indicate a one hour timeout.

and by COMP-PAR and COMP-SEQ those for the compositional procedure with parallel and sequential execution respectively.

In Table 1, we report the performance of VENMAS in terms of the amount of time (in seconds) to determine the truth of the specifications φ_1^k and φ_2^k for $k \in \{1, .., 5\}$ with initial relative positions $h_I \in \{-30, 10, 50, 90\}$ for each execution mode. In all cases we used a timeout of one hour. Note that the choice of h_I has an impact on the truth of the various specifications.

Regarding the performance of the different encodings, we find that the monolithic encoding is more efficient than the compositional encoding for $h_I = 50$ and $h_I = 90$. We believe this to be because checking the infeasibility of a single MILP with tight bounds can be determined faster than checking several MILPs. For $h_I = -30$ and $h_I = 10$ the compositional encodings were more efficient. We believe this to be because if a single feasible MILP (from the set of generated MILP instances) is found, then a result can be returned immediately. It remains to be explored which encoding is preferable in general, as the difficulty of solving a MILP is hard to determine in advance. This would open the way to the development of heuristics to guide the choice of the encoding more likely to efficiently solve the problem under analysis.

We are unable to present a comparison with other tools because we are not aware of other tools supporting systems of multiple neural-symbolic agents and strategic properties as we do here. Similarly to (Akintunde et al. 2020a), real values in VENMAS are represented by double-precision floating point numbers and we use the default constraint-satisfaction tolerance level of 10^{-6} in Gurobi.

7 Conclusions

As known, methods for strategic reasoning can be highly advantageous to the development and deployment of MAS. While the literature on the subject over the past twenty years is very considerable, all approaches have so far been limited to agent-based systems whose components are traditionally modelled via various forms of transition systems. But, as we discussed in the introduction, a new generation of MAS is emerging in applications. These are systems in which machine learning elements are used for perception tasks and combined with traditional logic-based elements for decision making and control. The result is a loosely-coupled, neural-symbolic architecture where both machine learning and logic-based elements contribute to the overall agent behaviour. To analyse the resulting behaviour of these systems new agent models and methods need to be devised.

In this paper we put forward the concept of neural interpreted systems to capture agents implementing the above architecture. The proposed architecture closely mirrors several forthcoming applications in which vision is realised via machine learning systems whereas decision making is executed via control logic, e.g., platoons of autonomous vehicles. We have formulated the verification problem against a language representing strategic interplay. Differently from traditional approaches in the literature, we have here opted for a bounded version of ATL with the aim of looking for shallow bugs. This is also in line with considerable recent work on LTL and other languages on finite traces (De Giacomo and Vardi 2016). We have solved the resulting verification problem in terms of MILP and presented an encoding which, as we showed, is amenable to parallelisation.

We reported on a prototype that we developed implementing both encodings and evaluated against a sophisticated model from the avionics literature. The evaluation demonstrated the correctness of the approach and that small systems can feasibly be verified. It also highlighted, however, that a challenge of the present method lies in its scalability, as the MILPs become more difficult to solve as the number of non-linearities encountered in the encodings grow rapidly with the number of steps considered. This problem is exacerbated by the highly branching nature of the models often considered in a strategic setting.

Future work will focus on conquering the scalability of the method, extending the specification language and supporting richer symbolic reasoning abilities in the agents.

Acknowledgements

This work is partly funded by DARPA under the Assured Autonomy programme (FA8750-18-C-0095). The support of the EPSRC Centre for Doctoral Training in High Performance Embedded and Distributed Systems (reference EP/L016796/1) is gratefully acknowledged. Alessio Lomuscio is supported by a Royal Academy of Engineering Chair in Emerging Technologies.

References

Akintunde, M. E.; Lomuscio, A.; Maganti, L.; and Pirovano, E. 2018. Reachability analysis for neural agent-environment systems. In *Proceedings of the 16th International Conference on Principles of Knowledge Representation and Reasoning (KR18)*, 184–193. AAAI Press.

Akintunde, M. E.; Botoeva, E.; Kouvaros, P.; and Lomuscio, A. 2020a. Formal verification of neural agents in non-deterministic environments. In *Proceedings of the 19th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS20)*, 25–33. IFAAMAS.

Akintunde, M. E.; Botoeva, E.; Kouvaros, P.; and Lomuscio, A. 2020b. VENMAS: Verification of neural-symbolic multi-agent systems, https://vas.doc.ic.ac.uk/software/neural/. [Accessed: 2020-07-02].

Alechina, N.; Logan, B.; Nguyen, H. N.; Raimondi, F.; and Mostarda, L. 2015. Symbolic model-checking for resource-bounded ATL. In *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS15)*, 1809–1810.

Alur, R.; Henzinger, T.; Mang, F.; Qadeer, S.; Rajamani, S.; and Tasiran, S. 1998. MOCHA: Modularity in model checking. In *Proceedings of the 10th International Conference on Computer Aided Verification (CAV98)*, volume 1427 of *Lecture Notes in Computer Science*, 521–525. Springer.

Alur, R.; Henzinger, T. A.; and Kupferman, O. 2002. Alternating-time temporal logic. *Journal of the ACM* 49(5):672–713.

Biere, A.; Cimatti, A.; Clarke, E.; Strichman, O.; and Zhu, Y. 2003. Bounded model checking. *Advances in Computers* 58:117–148.

Botoeva, E.; Kouvaros, P.; Kronqvist, J.; Lomuscio, A.; and Misener, R. 2020. Efficient verification of neural networks via dependency analysis. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI20)*, 3291–3299. AAAI Press.

Chatterjee, K.; Henzinger, T.; and Piterman, N. 2007. Strategy logic. In *Proceedings of the 18th International Conference on Concurrency Theory (CONCUR07)*, volume 4703, 59–73.

D'Ambrosio, C.; Lodi, A.; and Martello, S. 2010. Piecewise linear approximation of functions of two variables in MILP models. *Operations Research Letters* 38(1):39–46.

De Giacomo, G., and Vardi, M. Y. 2016. LTL_f and LDL_f synthesis under partial observability. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI16)*, 1044–1050. IJCAI/AAAI Press.

Dembiński, P.; Janowska, A.; Janowski, P.; Penczek, W.; Pólrola, A.; Szreter, M.; Woźna, B.; and Zbrzezny, A. 2003. VERICS: A tool for verifying Timed Automata and Estelle specifications. In *Proceedings of the 9th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS03)*, volume 2619 of *Lecture Notes in Computer Science*, 278–283. Springer.

Dreossi, T.; Ghosh, S.; Yue, X.; Keutzer, K.; Sangiovanni-Vincentelli, A.; and Seshia, S. 2018. Counterexample-guided data augmentation. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence and 23rd European Conference on Artificial Intelligence (IJCAI-ECAI18)*.

Dreossi, T.; Donzé, A.; and Seshia, S. 2019. Compositional falsification of cyber-physical systems with machine learning components. *Journal of Automated Reasoning* 63(4):1031–1053.

Dutta, S.; Chen, X.; and Sankaranarayanan, S. 2019. Reachability analysis for neural feedback systems using regressive polynomial rule inference. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control (HSCC19)*, 157–168. ACM.

Dvijotham, K.; Stanforth, R.; Gowal, S.; Mann, T.; and Kohli, P. 2018. A dual approach to scalable verification of deep networks. In *Proceedings of the 34th Conference on Uncertainty in Artificial Intelligence (UAI18)*, 550–559. AUAI Press.

Emerson, E. A.; Mok, A. K.; Sistla, A. P.; and Srinivasan, J. 1992. Quantitative temporal reasoning. *Real-Time Systems* 4(4):331–352.

Fagin, R.; Halpern, J. Y.; Moses, Y.; and Vardi, M. Y. 1995. *Reasoning about knowledge*. Cambridge: MIT Press.

Garcez, A.; Lamb, L.; and Gabbay, D. 2009. *Neural-symbolic cognitive reasoning*. Cognitive Technologies. Springer.

Gehr, T.; Mirman, M.; Drachsler-Cohen, D.; Tsankov, P.; Chaudhuri, S.; and Vechev, M. 2018. AI^2: Safety and robustness certification of neural networks with abstract interpretation. In *Proceedings of the 2018 IEEE Symposium on Security and Privacy (S&P18)*, 948–963.

Goodfellow, I.; Shlens, J.; and Szegedy, C. 2014. Explaining and harnessing adversarial examples. *CoRR* abs/1412.6572.

Griva, I.; Nash, S.; and Sofer, A. 2009. *Linear and nonlinear optimization*. SIAM, 2nd edition.

Gu, Z.; Rothberg, E.; and Bixby, R. 2020. Gurobi optimizer reference manual. http://www.gurobi.com. [Accessed: 2020-07-02].

Gutierrez, J.; Najib, M.; Perelli, G.; and Wooldridge, M. 2018. Eve: A tool for temporal equilibrium analysis. In *Proceedings of the 16th International Symposium on Automated Technology for Verification and Analysis (ATVA18)*, 551–557. Springer.

Halpern, J. Y., and Fagin, R. 1985. A formal model of knowledge, action, and communication in distributed systems: Preliminary report. In *Proceedings of the Fourth ACM*

Symposium on Principles of Distributed Computing, 224–236.

Hinton, A.; Kwiatkowska, M. Z.; Norman, G.; and Parker, D. 2006. Prism: A tool for automatic verification of probabilistic systems. In *Proceedings of the 12th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS06)*, volume 3920 of *Lecture Notes in Computer Science*, 441–444. Springer.

Huang, C.; Fan, J.; Li, W.; Chen, X.; and Zhu, Q. 2019. ReachNN: Reachability analysis of neural-network controlled systems. *ACM Transactions on Embedded Computing Systems* 18(106):1–22.

Ivanov, R.; Weimer, J.; Alur, R.; Pappas, G. J.; and Lee, I. 2019. Verisig: Verifying safety properties of hybrid systems with neural network controllers. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control (HSCC19)*, 169–178. Association for Computing Machinery.

Julian, K. D., and Kochenderfer, M. J. 2019. A reachability method for verifying dynamical systems with deep neural network controllers. *CoRR* abs/1903.00520.

Julian, K. D.; Sharma, S.; Jeannin, J.-B.; and Kochenderfer, M. J. 2019. Verifying aircraft collision avoidance neural networks through linear approximations of safe regions. *CoRR* abs/1903.00762.

Katz, G.; Huang, D. A.; Ibeling, D.; Julian, K.; Lazarus, C.; Lim, R.; Shah, P.; Thakoor, S.; Wu, H.; Zeljic, A.; Dill, D. L.; Kochenderfer, M. J.; and Barrett, C. W. 2019. The Marabou framework for verification and analysis of deep neural networks. In *Proceedings of the 31st International Conference on Computer Aided Verification (CAV19)*, 443–452.

Kouvaros, P., and Lomuscio, A. 2018. Formal verification of CNN-based perception systems. *CoRR* abs/1811.11373.

Lamb, L.; Garcez, A.; Gori, M.; Prates, M.; Avelar, P.; and Vardi, M. 2020. Graph neural networks meet neural-symbolic computing: A survey and perspective. *CoRR* abs/2003.00330.

Lomuscio, A., and Maganti, L. 2017. An approach to reachability analysis for feed-forward ReLU neural networks. *CoRR* abs/1706.07351.

Lomuscio, A.; Qu, H.; and Raimondi, F. 2017. MCMAS: A model checker for the verification of multi-agent systems. *Software Tools for Technology Transfer* 19(1):9–30.

Mogavero, F.; Murano, A.; and Vardi, M. 2010. Reasoning About Strategies. In *Proceedings of the 30th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS10)*, volume 8, 133–144. Schloss Dagstuhl.

Nair, V., and Hinton, G. E. 2010. Rectified linear units improve restricted Boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML10)*, 807–814. Omnipress.

Papadimitriou, C., and Steiglitz, K. 1982. *Combinatorial optimization: Algorithms and complexity*. USA: Prentice-Hall, Inc.

Pauly, M. 2002. A modal logic for coalitional power in games. *Journal of Logic and Computation* 12(1):149–166.

Penczek, W.; Woźna, B.; and Zbrzezny, A. 2002. Bounded model checking for the universal fragment of CTL. *Fundamenta Informaticae* 51(1-2):135–156.

Singh, G.; Gehr, T.; Puschel, M.; and Vechev, M. 2019. Boosting robustness certification of neural networks. In *Proceedings of the 7th International Conference on Learning Representations (ICLR19)*.

Sun, X.; Khedr, H.; and Shoukry, Y. 2019. Formal verification of neural network controlled autonomous systems. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control (HSCC19)*, 147–156. Association for Computing Machinery.

Tjeng, V.; Xiao, K.; and Tedrake, R. 2019. Evaluating robustness of neural networks with mixed integer programming. In *Proceedings of the 7th International Conference on Learning Representations (ICLR19)*.

Wang, S.; Pei, K.; Whitehouse, J.; Yang, J.; and Jana, S. 2018. Formal security analysis of neural networks using symbolic intervals. In *Proceedings of the 27th USENIX Security Symposium (USENIX18)*.

Wooldridge, M., and Lomuscio, A. 1999. Reasoning about visibility, perception, and knowledge. In *International Workshop on Agent Theories, Architectures, and Languages*, 1–12. Springer.

Xiang, W.; Tran, H.; Rosenfeld, J. A.; and Johnson, T. T. 2018. Reachable set estimation and safety verification for piecewise linear systems with neural network controllers. In *Proceedings of Annual American Control Conference (ACC18)*, 1574–1579. AACC.