Accelerating Verification for Neuro-symbolic Multi-agent Systems via Compact MILP encodings

Author Name
Affiliation
email@example.com

Abstract

We consider the problem of verifying neuro-symbolic multi-agent systems against temporal formulae expressed in Linear Temporal Logic (LTL), where verification amounts to bounded model-checking to regain decidability. Existing decision procedures are limited to syntactically constrained fragments of LTL in which only formulae of bounded temporal depth can be expressed. Crucially, employing these approaches requires a rewriting step that may introduce a worst-case exponential blow-up in the size of the unconstrained original LTL formula. In this paper, we overcome this problem and propose a novel Mixed-Integer Linear Programming encoding that is linear in the size of the formula and temporal depth in question. Our encoding is based on bounded semantics, rather than bounded syntax, and allows us to handle the full LTL syntax succinctly. Our approach offers complementary advantages to existing methods, especially when dealing with formulae that contain nested temporal operators (such as until or eventually), which may lead to memory and time-out issues in current verification tools. Experiments using existing benchmarks, as well as new ones we introduce here, highlight the advantages of our approach in terms of efficiency and scalability.

1 Introduction

Recent advances in deep learning [Goodfellow *et al.*, 2016] have enabled a new generation of neuro-symbolic artificial agents, i.e., agents whose reasoning capabilities combine symbolic and data-driven components [Akintunde *et al.*, 2018]. While such neuro-symbolic agents can offer significant improvements in performance and generalisation, the black-box nature of their neural components can hinder their deployment in safety-critical settings, as the safety of their outputs cannot be ascertained.

Formal verification techniques can help mitigate these concerns. Previous research has addressed the automatic verification of systems that include AI components [Bogomolov *et al.*, 2019; Huang *et al.*, 2022; Ivanov *et al.*, 2019; Lopez *et al.*, 2023], including MAS comprising agents with neural components [Akintunde *et al.*, 2019, 2022, 2020]. While the verification problem for neuro-symbolic MAS against unbounded formulae is generally undecidable [Akintunde *et al.*, 2022], methods have been proposed to ascertain the correct behaviour of these systems against specifications expressed, e.g. in bounded fragments of Linear Temporal Logic (LTL) [Pnueli, 1977]. However, state-of-the-art verifiers for neuro-symbolic MAS, such as [Akintunde *et al.*, 2022], only accept a bounded fragment of LTL that consists of arbitrary disjunctions, conjunctions over atomic propositions, and the next-time temporal operator X. While this fragment is expressive enough to encode the satisfaction of arbitrary LTL formulae up to a bounded temporal depth, its practical applicability is limited by the need to first rewrite any LTL formula of interest into this supported fragment. More specifically, given an arbitrary LTL formula φ and a temporal bound d, rewriting may result in a formula of size $O(d^{|\varphi|})$ in the worst case. This exponential blow-up in the size of the original formula φ may directly affect the runtime of existing verifiers, limiting their practical applicability.

In this paper we overcome this challenge by proposing a novel Mixed-integer Linear Programming (MILP) treatment for temporal formulae that, similarly to SAT encodings for bounded model checking, is based on bounded semantics rather than bounded syntax. This allows us to consider the entirety of LTL syntax, rather than constrained fragments previous work was limited to. Crucially, our encoding is always linear in the size of φ and d. This succinctness translates to improved reasoning times for complex formulae, as we show in our experiments across a range of benchmarks.

Related work. Several approaches have been proposed for the verification of neuro-symbolic MAS [Akintunde *et al.*, 2019, 2022, 2020]. These works presented MILP-based encodings to reason about temporal properties expressed in bounded fragments of LTL, CTL and ATL. However, differently from previous encodings, our approach can handle unbounded LTL specifications and can do so while also resulting in encodings of smaller size.

More broadly, this line of work is related to existing efforts focusing on closed-loop systems [Bogomolov *et al.*, 2019; Huang *et al.*, 2022; Ivanov *et al.*, 2019]. Differently from these works, however, we consider systems where multiple agents interact with each other (and the environment).

Finally, our approach is related to complete approaches for the verification of standalone neural networks [Gehr *et al.*, 2018; Tran *et al.*, 2020; Katz *et al.*, 2017; Henriksen and Lomuscio, 2020; Zhang *et al.*, 2022]. These works, however, do not target closed-loop, temporally extended behaviours arising from interactions with the environment and are thus not applicable here.

Structure of the paper. We introduce the necessary background on neuro-symbolic MAS, LTL and MILP in Section 2. Then, in Section 3, we formalise the verification problem for neuro-symbolic MAS and present our novel MILP encoding to solve it. Section 4 presents our experimental analysis, demonstrating the improved scalability of our approach. Finally, we conclude in Section 5.

2 Background

Mixed Integer Linear Programming (MILP). 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*. A *linear program* is an optimisation problem where the objective function is linear and the constraints on the variables of the function are linear. A *MILP* is a linear program whose set of constraints include integrality constraints. The *feasibility problem* of a MILP is the question of whether or not there exists a feasible solution to the program, that is, an assignment to the variables that satisfies all constraints. An *ε-feasible solution* is a solution that allows for constraint violations up to a small tolerance ε. Practical applications are typically associated with ε-feasible solutions because of the finite precision in which MILP solvers operate.

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 (i.e, the set of all possible solutions) of C_S.

MILP allows for the expression of *indicator constraints* of the form $(\delta = v) \Rightarrow c$, for a binary variable δ, binary value $v \in \{0, 1\}$ and a linear constraint c, meaning that whenever the value of δ is v, the constraint c should hold. In this work we use indicator constraints to encode disjunctions.

Neural Interpreted Systems. We recall Neural Interpreted Systems (NIS), a formal semantics for neural MAS [Akintunde *et al.*, 2020] that extends *interpreted systems* (IS) [Fagin *et al.*, 1995], a standard semantics for MAS. Differently from IS, which assume purely symbolic agents, NIS uniquely combine connectionist with symbolic models. In particular, the 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. 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 that interpretation [Dreossi *et al.*, 2019].

We begin with a MAS composed of K *neuro-symbolic agents* $Agt = \{1, \ldots, K\}$ 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 (Neuro-symbolic Agent)**.** *A* neuro-symbolic agent*, or* agent*, is a tuple* $a = (L_a, \mathsf{obs}_a, Act_a, \mathsf{prot}_a, \mathsf{tr}_a)$*, where:*

- $L_a \subseteq \mathbb{R}^{n_a}$, $n_a \geq 1$*, is a nonempty (possibly infinite)* set of local states.
- $\mathsf{obs}_a \colon L_E \to Per_a$ *is an* observation function *mapping environment states* L_E *to percepts* Per_a*.*
- Act_a *is a nonempty finite* set of actions.
- $\mathsf{prot}_a \colon L_a \to 2^{Act_a} \setminus \{\emptyset\}$ *is a* local protocol function *defining the available actions in each local state.*
- $\mathsf{tr}_a \colon L_a \times Per_a \times Act_1 \times \cdots \times Act_K \times Act_E \to L_a$ *is a* local transition function *determining the next local state given the current local state, a percept, and a tuple of actions performed by all agents.*

In the following we assume obs_a, prot_a and tr_a to be piecewise-linear functions, meaning that the domain of each function can be partitioned so that the function is affine on each partition. The environment is defined similarly to an agent but without percepts or observation function.

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

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* $q = (\ell_1, \ldots, \ell_K, \ell_E)$ is a tuple of local states for all the agents in the system; it describes the system at a particular instant of time. Given a global state q and an agent a, we write $loc_a(q)$ to denote the local state of agent a in q and $\mathsf{prot}_a(q)$ to denote the set of actions that can be performed by agent a at q, i.e., $\mathsf{prot}_a(loc_a(q))$. 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* $\alpha = (\alpha_1, \ldots, \alpha_K, \alpha_E)$ 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* $\mathsf{tr} : G \times ACT \to G$, where, for $q \in G$, $(\alpha_1, \ldots, \alpha_K, \alpha_E) \in ACT$, $\mathsf{tr}(q, \alpha_1, \ldots, \alpha_K, \alpha_E) = q'$ iff

- $\alpha_E \in \mathsf{prot}_E(q)$ and $\mathsf{tr}_E(loc_E(q), \alpha_1, \ldots, \alpha_K, \alpha_E) = loc_E(q')$; i.e., the environment's action is protocol compliant and its local state is updated as per its local transition function.
- for every agent $a \in Agt$ we have that $\alpha_a \in \mathsf{prot}_a(q)$ and $\mathsf{tr}_a\big(loc_a(q), \mathsf{obs}_a(loc_E(q)), \alpha_1, \ldots, \alpha_K, \alpha_E\big) = loc_a(q')$; i.e., the agent's action is protocol compliant and its local state is updated as per its local transition function given its current local state, its perception of the environment, and the performed joint action.

Definition 3 (Neural Interpreted System)**.** *A* neural interpreted system *(NIS) is a tuple* $\mathcal{S} = (Agt \cup \{E\}, I)$*, where* Agt *is a set of agents,* E *is an environment, and* $I \subseteq G$ *is a linearly definable set of* initial *global states.*

NIS are (uniquely) associated with temporal models capturing their temporal evolution. These are defined as follows.

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

- *G is the set of global states of $\mathcal{S}$.*
- *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 $\mathsf{tr}(q, \alpha) = q'$.*

A *path* π in a model $\mathcal{M}_{\mathcal{S}} = (G, ACT, T)$ is an infinite sequence of global states $q^0 q^1 \ldots$ such that for every $i \geq 0$, $(q^i, \alpha^i, q^{i+1}) \in T$ for some joint action α^i. We use $\pi(i)$ to denote the global state q^i and $\pi[i]$ to denote the suffix $q^i q^{i+1} \ldots$ of π.

Linear Temporal Logic. We express *specifications* in Linear Temporal Logic [Pnueli, 1977] over linear inequalities on the agents' local states.

Definition 5 (LTL)**.** *Given sets of agents' and environment local states $L_1 \subseteq \mathbb{R}^{n_1}, \ldots, L_K \subseteq \mathbb{R}^{n_K}$, $L_E \subseteq \mathbb{R}^{n_E}$,* LTL formulae φ over linear inequalities *are defined as follows:*

$$\begin{aligned} \varphi &::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathsf{X}\varphi \mid \varphi \,\mathsf{U}\, \varphi, \\ p &::= c_1(a_1, d_1) + \cdots + c_l(a_l, d_l) \; op \; c, \end{aligned}$$

where $c_i, c \in \mathbb{R}$, $a_i \in Agt \cup \{E\}$, $d_i \in \{1, \ldots, n_{a_i}\}$, $op \in \{\leq, \geq\}$.

Here atomic propositions p are linear constraints on the components of the agents' local states. For instance, the atomic proposition $(a_1, d_1) + (a_2, d_2) \leq 2$ states that "the sum of the d_1-st and d_2-nd components of the a_1-st and a_2-nd agents, respectively, is less than or equal 2." The formula $\mathsf{X}\varphi$ is read as "φ holds at the *next* step", $\varphi_1 \,\mathsf{U}\, \varphi_2$ is interpreted as "φ_2 holds at some point and *until* that point φ_1 holds". We assume customary abbreviations for LTL: *disjunction* $\varphi_1 \vee \varphi_2$ for $\neg(\neg\varphi_1 \wedge \neg\varphi_2)$, *implication* $\varphi_1 \rightarrow \varphi_2$ for $\neg\varphi_1 \vee \varphi_2$, *eventually* $\mathsf{F}\varphi$ for $true \,\mathsf{U}\, \varphi$, and *globally* $\mathsf{G}\varphi$ for $\neg\mathsf{F}\neg\varphi$.

We interpret LTL formulae on temporal models associated with NIS as in Definition 4. Following the undecidability of verification for infinite paths [Akintunde *et al.*, 2022], we consider the interpretation of LTL formulae over path prefixes of bounded length [Biere *et al.*, 2003].

Definition 6 (Satisfaction)**.** *For a model $\mathcal{M}_{\mathcal{S}}$ associated with a NIS $\mathcal{S}$ and an LTL formula φ, the* satisfaction of φ in a path π of bound $d \geq 0$ *in $\mathcal{M}_{\mathcal{S}}$, denoted $\pi \models_d \varphi$, is defined as $\pi \models_d \varphi$ iff $\pi \models_d^0 \varphi$, where*

$\pi \models_d^i p$ *iff* $c_1 loc_{a_1}\big(\pi(i)\big)_{d_1} + \cdots + c_l loc_{a_l}\big(\pi(i)\big)_{d_l} \; op \; c$, *where* $p = c_1(a_1, d_1) + \cdots + c_l(a_l, d_l) \; op \; c$;

$\pi \models_d^i \neg\varphi$ *iff* $\pi \not\models_d^i \varphi$;

$\pi \models_d^i \varphi_1 \wedge \varphi_2$ *iff* $\pi \models_d^i \varphi_1$ *and* $\pi \models_d^i \varphi_2$;

$\pi \models_d^i \mathsf{X}\varphi$ *iff* $i < d$ *and* $\pi \models_d^{i+1} \varphi$;

$\pi \models_d^i \varphi_1 \mathsf{U} \varphi_2$ *iff there exists* j, $i \leq j \leq d$, *s.t.* $\pi \models_d^j \varphi_2$; *and* $\pi \models_d^h \varphi_1$ *for all* $i \leq h < j$.

When π is a finite path, we write $\pi \models \varphi$ as an abbreviation for $\pi \models_{|\pi|} \varphi$. An LTL property φ is said to be satisfied in d-bounded paths in a model, denoted $\mathcal{M}_{\mathcal{S}} \models_d \varphi$, if we have that $\pi \models_d \varphi$ for every path π in $\mathcal{M}_{\mathcal{S}}$. This notion of satisfaction concerns the main decision problem that we study here.

Definition 7 (Verification Problem)**.** *Given a NIS $\mathcal{S}$, an LTL property φ, and $d \geq 0$, decide whether $\mathcal{M}_{\mathcal{S}} \models \varphi$.*

We use $(\mathcal{S}, \varphi, d)$ to denote a verification problem with respect to NIS $\mathcal{S}$, LTL formula φ and $d \geq 0$.

3 Verification Procedure

This section introduces a novel MILP-based procedure to solve the verification problem introduced in the previous section. The procedure is inspired by SAT encodings for the bounded model checking of traditional systems [Biere *et al.*, 1999], where the satisfaction of LTL formulae by temporal paths is encoded as a Boolean satisfiability problem. Here, we utilise this encoding as an intermediate syntax that facilitates the MILP encoding of LTL formulae in NIS paths. We begin by fixing a NIS path and giving a MILP translation of the SAT-based LTL satisfiability check on that specific path. We then extend this encoding to represent the satisfaction of LTL formulae on NIS models.

3.1 MILP formulation of the LTL Boolean satisfiability problem

We fix a NIS path π of length $d \geq 0$ and an LTL formula φ, and describe a novel MILP formulation for whether $\pi \models \varphi$.

We start by introducing a propositional variable s_ψ^i, for each point in time $i \in \{0, \ldots, d+1\}$, and each sub-formula ψ of φ. Each s_ψ^i has the meaning "ψ is satisfied by the suffix $\pi[i]$ of π". We use $f_1 \equiv f_2$ as a shortcut for $\{f_1 \rightarrow f_2\} \cup \{f_2 \rightarrow f_1\}$, and write $\mathsf{subf}(\varphi)$ and $\mathsf{atom}(\varphi)$ for the sets of all sub-formulae and atomic propositions of φ, respectively. With this preliminary notation, we now define the SAT encoding of the satisfaction of φ on π.

Definition 8 (SAT Encoding)**.** *Let π be a NIS path of length $d \geq 0$ and φ an LTL formula. The SAT encoding $[\![\varphi, \pi]\!]_d$ of φ with respect to π is the set of Boolean formulae*

$$[\![\varphi, \pi]\!]_d = \mathsf{Subf} \cup \mathsf{Atom},$$

where:

$$\begin{aligned} \mathsf{Subf} = & \bigcup_{\neg\psi \in \mathsf{subf}(\varphi)} \bigcup_{i=0}^{d} s_{\neg\psi}^i \equiv \neg s_\psi^i \\ & \bigcup_{\psi_1 \wedge \psi_2 \in \mathsf{subf}(\varphi)} \bigcup_{i=0}^{d} s_{\psi_1 \wedge \psi_2}^i \equiv s_{\psi_1}^i \wedge s_{\psi_2}^i \\ & \bigcup_{\mathsf{X}\psi \in \mathsf{subf}(\varphi)} \bigcup_{i=0}^{d} s_{\mathsf{X}\psi}^i \equiv s_\psi^{i+1} \\ & \bigcup_{\psi_1 \,\mathsf{U}\, \psi_2 \in \mathsf{subf}(\varphi)} \bigcup_{i=0}^{d} s_{\psi_1 \,\mathsf{U}\, \psi_2}^i \equiv s_{\psi_2}^i \vee (s_{\psi_1}^i \wedge s_{\psi_1 \,\mathsf{U}\, \psi_2}^{i+1}) \\ & \cup \left\{ s_\varphi^0 \right\} \cup \left\{ \neg s_\psi^{d+1} \mid \psi \in \mathsf{subf}(\varphi) \right\}, \text{ and} \end{aligned}$$

$$\mathsf{Atom} = \bigcup_{p \in \mathsf{atom}(\varphi)} \bigcup_{i=0}^{d} \left\{ s_p^i \;\middle|\; \pi(i) \models p \right\} \cup \left\{ \neg s_p^i \;\middle|\; \pi(i) \not\models p \right\}.$$

Note that the formulae in Atom assert the truth values of the propositions s_p^i depending on whether $\pi(i) \models p$ as per Definition 6.

Example 1. Suppose $\varphi = a \wedge Xb$. Consider the following path π, where a holds at time step 0, and b and c at time step 1:

$$\pi : \quad \overset{0}{\underset{a}{\bullet}} \text{———} \overset{1}{\underset{b,c}{\bullet}}$$

Then, $[\![\varphi, \pi]\!]_1 = \mathsf{Subf} \cup \mathsf{Atom}$, where

$$\begin{aligned}\mathsf{Subf} = \{\ & s^0_{a\wedge Xb} \equiv s^0_a \wedge s^0_{Xb},\ s^0_{Xb} \equiv s^1_b,\\ & s^1_{a\wedge Xb} \equiv s^1_a \wedge s^1_{Xb},\ s^1_{Xb} \equiv s^2_b,\\ & s^0_{a\wedge Xb},\\ & \neg s^2_a,\ \neg s^2_b,\ \neg s^2_{Xb},\ \neg s^2_{a\wedge Xb}\ \},\ \text{and}\\ \mathsf{Atom} = \{\ & s^0_a,\ \neg s^0_b,\ \neg s^1_a,\ s^1_b\ \}.\end{aligned}$$

The satisfiability of the Boolean formulae $[\![\varphi, \pi]\!]_d$ can be used to decide the satisfiability of the underlying LTL formula on the given path.

Lemma 1. *Consider an LTL formula φ and a NIS path π. Then, $[\![\varphi, \pi]\!]_d$ is satisfiable iff $\pi \models \varphi$.*

Proof. The result follows from the SAT encoding of the bounded model checking of LTL [Biere *et al.*, 1999]. □

Following the above Lemma, we now provide a translation $\{\!\{\varphi, \pi\}\!\}_d$ of the Boolean encoding $[\![\varphi, \pi]\!]_d$ into a MILP program. As we show below, the translation ensures that the MILP program has a feasible solution iff the Boolean program is satisfiable. The construction of the MILP program exploits the observation that every formula in Atom and Subf is either a literal or equivalent to Boolean formulae of either one of the following *canonical* forms:

$$\begin{array}{ll}\neg s_1 \vee s_2 & \neg s_1 \vee (s_2 \wedge s_3)\\ \neg s_1 \vee s_2 \vee (s_3 \wedge s_4) & \neg s_1 \vee \neg s_2 \vee s_3,\end{array}$$

where each s_i is an atom. We thus provide a MILP translation for Boolean formulae of precisely this canonical form.

Definition 9. *Let π be a NIS path of length $d \geq 0$ and φ be an LTL formula. The MILP encoding $\{\!\{\varphi, \pi\}\!\}_d$ of $[\![\varphi, \pi]\!]_d$ is*

$$\{\!\{\varphi, \pi\}\!\}_d \triangleq \{\langle f \rangle \geq 1 \mid f \in [\![\varphi, \pi]\!]_d\},$$

where $\langle f \rangle$ is defined recursively on the structure of f:

$$\begin{aligned}&\langle f_1 \vee f_2 \rangle = \langle f_1 \rangle + \langle f_2 \rangle\\ &\langle f_1 \wedge f_2 \rangle = 0.5 * (\langle f_1 \rangle + \langle f_2 \rangle)\\ &\langle \neg f \rangle = 1 - \langle f \rangle\\ &\langle s^i_\psi \rangle = t^i_\psi,\end{aligned}$$

with t^i_ψ being a MILP binary variable.

Example 2. Consider the LTL formula φ and path π from Example 1. Then, $\{\!\{\varphi, \pi\}\!\}_d$ is as follows:

$$\begin{gathered}1 - t^0_{a\wedge Xb} + 0.5(t^0_a + t^0_{Xb}) \geq 1\\ 1 - t^0_a + 1 - t^0_{Xb} + t^0_{a\wedge Xb} \geq 1\\ 1 - t^0_{Xb} + t^1_b \geq 1 \qquad 1 - t^1_b + t^0_{Xb} \geq 1\\ \cdots\\ t^0_{a\wedge Xb} \geq 1\\ 1 - t^2_a \geq 1 \quad 1 - t^2_b \geq 1 \quad 1 - t^2_{Xb} \geq 1 \quad 1 - t^2_{a\wedge Xb} \geq 1\\ t^0_a \geq 1 \qquad 1 - t^0_b \geq 1 \qquad 1 - t^1_a \geq 1 \qquad t^1_b \geq 1,\end{gathered}$$

where we omit the constraints for the formulae in the second row of Subf in Example 1.

The MILP encoding $\{\!\{\varphi, \pi\}\!\}_d$ of φ and π is equivalent to the SAT encoding $[\![\varphi, \pi]\!]_d$ of φ and π.

Lemma 2. *Consider an LTL formula φ and a NIS path π. Then, $\{\!\{\varphi, \pi\}\!\}_d$ is feasible iff $[\![\varphi, \pi]\!]_d$ is satisfiable.*

Proof Sketch. The result can be shown by proving that we can obtain a satisfying assignment to $[\![\varphi, \pi]\!]_d$ from a solution to $\{\!\{\varphi, \pi\}\!\}_d$ and vice-versa. □

As we describe below, the above result forms the basis of the MILP-based verification of NIS against LTL formulae. The encoding differs significantly from previous MILP treatments of bounded semantics for temporal formulae, where the specification in question has to be first reformulated to include only the next-time temporal operator, and then translated into MILP using indicator constraints [Akintunde *et al.*, 2022, 2020]. Crucially, the resulting MILP programs grow exponentially in the size of the formula considered. In contrast, we here present an encoding that is linear with respect to both factors. As we show in the next section, this may lead to markedly improved execution times.

3.2 MILP-based verification of NIS

We formalise NIS models as MILP programs and couple the MILP variables representing the global states with the LTL encoding given above. This allows us to derive a procedure for verifying NIS against LTL formulae.

Let $\mathcal{S}$ be a NIS and $d \geq 0$. We start by constructing a MILP program $\{\!\{\mathcal{S}\}\!\}_d$, where each solution represents a *valid path of length d* in $\mathcal{M}_\mathcal{S}$. The construction, based on Akintunde *et al.* [2022, 2020], uses the following objects:

- A *vector $\boldsymbol{y}_i$ of real-valued MILP variables*, for each global state q_i that the system is in in step $0 \leq i \leq d$. In an analogous notation to global states, we use $loc_a(\boldsymbol{y}_i)$ to denote the MILP variables associated with the local state of agent a in q_i.
- Assuming an ordering $\alpha_1, \ldots, \alpha_m$ of all global actions, a *binary variable* ac^i_j, for each $i \in \{0, \ldots, d-1\}$ and global action α_j, $j \in \{1, \ldots, m\}$. The meaning of ac^i_j is "at time step i, the global action α_j is taken".

Given the above basic objects, we now define $\{\!\{\mathcal{S}\}\!\}_d$ that captures the set I of initial global states and each valid global transition up to d time steps. Since, by assumption, the set I of initial global states is linearly definable, it can be expressed as a set $C_I(\boldsymbol{y}_0)$ of MILP constraints over the variables $\boldsymbol{y}_0$. As, also by assumption, the local protocol, observation and transition functions are piecewise-linear functions, a global transition $\mathsf{tr}(q_i, \alpha_j) = q_{i+1}$ for a global action α_j can be described as a set $C_j(\boldsymbol{y}_i, \boldsymbol{y}_{i+1})$ of piecewise-linear constraints. These account for the compliance of α_j to the agents' protocols and observation functions, and the alignment of the local states of the agents in q_i and q_{i+1} according to the agents local transition functions. Given $C_j(\boldsymbol{y}_i, \boldsymbol{y}_{i+1})$, a global transition from state q_i to state q_{i+1} is represented by the following set

$C_{T,i}(\boldsymbol{y}_i, \boldsymbol{y}_{i+1})$:

$$\Big\{ \sum_{j=1}^{m} ac_j^i = 1 \Big\} \cup \bigcup_{j=1}^{m} (ac_j^i = 1) \Rightarrow C_j(\boldsymbol{y}_i, \boldsymbol{y}_{i+1}),$$

where the expression $(\delta = v) \Rightarrow C$, for a set C of constraints, denotes the set of constraints $\{(\delta = v) \Rightarrow c \mid c \in C\}$. Finally, the encoding for the system's evolution is given by

$$\{\!\{\mathcal{S}\}\!\}_d = C_I(\boldsymbol{y}_0) \cup \bigcup_{i=0}^{d-1} C_{T,i}(\boldsymbol{y}_i, \boldsymbol{y}_{i+1}).$$

The set of MILP constraints $\{\!\{\mathcal{S}\}\!\}_d$ precisely captures the temporal evolution of $\mathcal{S}$.

Lemma 3. *Let $\mathcal{S}$ be a NIS, φ an LTL formula and $d \geq 0$. Then, the following hold:*

- *For any assignment $\mathcal{A}$ to the variables $\boldsymbol{y}_0, \ldots, \boldsymbol{y}_d$ that satisfies all constraints in $\{\!\{\mathcal{S}\}\!\}_d$, we have that there is a path $\pi_{\mathcal{A}}$ in $\mathcal{M}_{\mathcal{S}}$ such that $\mathcal{A}(\boldsymbol{y})_i = \pi_{\mathcal{A}}(i)$ for each i with $1 \leq i \leq d$.*
- *For any path π of length d in $\mathcal{M}_{\mathcal{S}}$, we have that there is an assignment $\mathcal{A}_\pi$ to the variables $\boldsymbol{y}_0, \ldots, \boldsymbol{y}_d$, which satisfies all constraints in $\{\!\{\mathcal{S}\}\!\}_d$, and for which $\mathcal{A}_\pi(\boldsymbol{y})_i = \pi(i)$ for each i with $1 \leq i \leq d$.*

We now extend $\{\!\{\mathcal{S}\}\!\}_d$ with constraints capturing the satisfaction of φ by $\mathcal{S}$. This requires the bridging of two different domains: the domain of symbols (atomic propositions) and the domain of continuous values (taken by state variables). In particular, in the MILP encoding of φ given above, the truth value of a binary variable t_p^i should be determined based on the values of the real-valued variables $\boldsymbol{y}_i$ representing the global state $\pi(i)$. To this end, for each $p \in \mathsf{atom}(\varphi)$ and $0 \leq i \leq d$, we define $\{\!\{i, p\}\!\}$ to be the set of MILP constraints

$$\begin{aligned} \{ \; & (t_p^i = 1) \;\Rightarrow\; c_1 loc_{a_1}(\boldsymbol{y}_i)_{d_1} + \cdots + c_l loc_{a_l}(\boldsymbol{y}_i)_{d_l} \; op \; c, \\ & (t_p^i = 0) \;\Rightarrow\; c' \;\; op \;\; c_1 loc_{a_1}(\boldsymbol{y}_i)_{d_1} + \cdots + c_l loc_{a_l}(\boldsymbol{y}_i)_{d_l} \; \}, \end{aligned}$$

where $p = c_1(a_1, d_1) + \cdots + c_l(a_l, d_l) \; op \; c$, and $c' = c + \varepsilon$ if op is $\leq$, otherwise $c' = c - \varepsilon$, for a small constant ε. The slack variable ε used in the representation of falsity of atomic propositions circumvents the inability of MILP to express strict inequalities. The constraints $\{\!\{i, p\}\!\}$ enforce an implication rule between the values of each t_p^i and the satisfaction of proposition p in the global state $\boldsymbol{y}_i$, which in turn enforces the inclusion of s_p^i as either a positive or a negative literal in the SAT encoding of φ.

Lemma 4. *Consider an assignment $\mathcal{A}$ to the variables $\boldsymbol{y}_0, \ldots, \boldsymbol{y}_d$ and t_p^i that satisfies all constraints in $\{\!\{\mathcal{S}\}\!\}_d \cup \{\!\{i, p\}\!\}$. Then, $\mathcal{A}(t_p^i) = 1$ (resp., $\mathcal{A}(t_p^i) = 0$) implies that there is a path $\pi_{\mathcal{A}}$ in $\mathcal{M}_{\mathcal{S}}$ such that $s_p^i \in [\![p, \pi_{\mathcal{A}}]\!]_d$, the SAT encoding of p with respect to $\pi_{\mathcal{A}}$ (resp., $\neg s_p^i \in [\![p, \pi_{\mathcal{A}}]\!]_d$).*

Proof. We show "$\mathcal{A}(t_p^i) = 1$ implies that there is a path $\pi_{\mathcal{A}}$ in $\mathcal{M}_{\mathcal{S}}$ with $\pi_{\mathcal{A}}(i) \models p$". The case of $\mathcal{A}(t_p^i) = 0$ is analogous.

Let $p = c_1(a_1, d_1) + \cdots + c_l(a_l, d_l) \; op \; c$. Since $\mathcal{A}$ satisfies the constraints in $\{\!\{\mathcal{S}\}\!\}_d$, we have that the variables $\boldsymbol{y}_0, \ldots, \boldsymbol{y}_d$ represent a path $\pi_{\mathcal{A}}$ of length d in $\mathcal{M}_{\mathcal{S}}$. By the definition of $\{\!\{i, p\}\!\}$, we have that $\mathcal{A} \models c_1 loc_{a_1}(\boldsymbol{y}_i)_{d_1} + \cdots + c_l loc_{a_l}(\boldsymbol{y}_i)_{d_l} \; op \; c$. So, $\pi_{\mathcal{A}}(i) \models p$. Hence, by the definition of Atom in $[\![p, \pi_{\mathcal{A}}]\!]_d$, we have that $s_p^i \in \mathsf{Atom}$. □

We now have all the necessary ingredients to define the MILP associated with a given verification problem $(\mathcal{S}, \varphi, d)$.

Definition 10 (MILP encoding)**.** *Given a verification problem $(\mathcal{S}, \varphi, d)$, its associated MILP $\{\!\{\mathcal{S}, \varphi\}\!\}_d$ is:*

$$\begin{aligned} \{\!\{\mathcal{S}, \varphi\}\!\}_d \;\; = \;\; & \{\!\{\mathcal{S}\}\!\}_d \cup \{\!\{\varphi, \; \boldsymbol{y}_0 \cdots \boldsymbol{y}_d\}\!\}_d \cup \\ & \Big\{ \; \{\!\{i, p\}\!\} \mid p \in \mathsf{atom}(\varphi), \; 0 \leq i \leq d \; \Big\}. \end{aligned}$$

The following theorem states the correctness of the MILP encoding.

Theorem 1. *Let $(\mathcal{S}, \varphi, d)$ be a verification problem. The MILP program $\{\!\{\mathcal{S}, \varphi\}\!\}_d$ has an ε-feasible solution iff there is a path π of length d in $\mathcal{M}_{\mathcal{S}}$ with $\pi \models \varphi$.*

Proof. Assume a solution to the constraints $\{\!\{\mathcal{S}, \varphi\}\!\}_d$. By satisfying the constraints $\{\!\{S\}\!\}_d$, the solution is a valid path π of length d in $\mathcal{M}_{\mathcal{S}}$. By satisfying the constraints $\{\!\{\varphi, \; \boldsymbol{y}_0 \cdots \boldsymbol{y}_d\}\!\}_d$, Lemmas 2 and 4 imply that the solution is also a satisfying assignment to the SAT encoding $[\![\varphi, \pi]\!]_d$ of φ with respect to π. Thus, by Lemma 1, the solution represents a path π of length d in $\mathcal{M}_{\mathcal{S}}$ with $\pi \models \varphi$. □

The above theorem provides a constructive procedure for solving the verification problem. Given a verification problem $(\mathcal{S}, \varphi, d)$, we construct the MILP $\{\!\{\mathcal{S}, \neg\varphi\}\!\}_d$ for the negation of φ as per Definition 10. The MILP is then passed to a MILP solver. If the solver outputs a solution, then the solution corresponds to a counterexample, i.e., a path π that satisfies $\neg\varphi$, as enforced by the constraints $\{\!\{S\}\!\}_d \cup \{\!\{\neg\varphi, \; \boldsymbol{y}_0 \cdots \boldsymbol{y}_d\}\!\}_d$. If a solution cannot be found, then the satisfaction of φ by $\mathcal{S}$ can be concluded.

Remark 1. *Owing to the real-valued nature of the NIS state space, the procedure is sound only up to a numerical tolerance ε, where ε are the slack variables used in the modelling of the atomic propositions. This reflects an inherent practical limitation of MILP solvers, which operate in finite-precision arithmetic and can therefore guarantee feasibility only within specified tolerances.*

4 Evaluation

In this section we present an empirical evaluation of our verification procedure over three sets of experiments. The first two are based on an Aircraft Collision Avoidance System (VCAS) from the literature [Akintunde *et al.*, 2022, 2020]. The third set of experiments concerns the verification of an Automated Cruise Control (ACC) system based on [Sugiyama *et al.*, 2008]. This is a new benchmark we introduce in this paper to test the scalability of verification approaches in domains where the number of neuro-symbolic agents can be increased arbitrarily. We report results obtained using a Python implementation of our approach. The implementation is based on and extends the VENMAS toolkit [Akintunde *et al.*, 2024] for the verification of neuro-symbolic MAS. VENMAS is best optimised for verifying NIS against

properties in the $(\mathsf{X}, \wedge, \vee)$ fragment of LTL, and leverages rewriting steps to translate general LTL specifications into the supported fragment. As rewriting might result in an exponential blow-up in the size of the formula being verified, we call this encoding *exponential*. For our experiments, we extended VENMAS by adding our succinct encoding of the LTL semantics and compared the runtimes of both encodings. Further details on our experimental setup can be found in the appendix.

LTL specifications. We evaluated our approach on verification tasks against the following specifications:

- $\varphi_1^k = \mathsf{G}\ \mathsf{safe}$
- $\varphi_2^k = \mathsf{G}(\mathsf{unsafe} \rightarrow \mathsf{X}\ \mathsf{safe})$
- $\varphi_3^k = \mathsf{G}(\mathsf{unsafe} \rightarrow \mathsf{F}\ \mathsf{safe})$
- $\varphi_4^k = \mathsf{G}(\mathsf{F}\ \mathsf{safe})$
- $\varphi_5^k = \mathsf{G}(\mathsf{unsafe} \rightarrow \mathsf{F}\ \mathsf{G}\ \mathsf{safe})$
- $\varphi_6^k = \mathsf{G}(\mathsf{unsafe} \rightarrow \mathsf{F}\ \mathsf{F}\ \mathsf{safe})$
- $\varphi_7^k = \mathsf{G}(\mathsf{unsafe} \rightarrow \mathsf{F}\ \mathsf{F}\ \mathsf{F}\ \mathsf{safe})$

for various values of k. The first five formulae capture standard properties, namely safety, immediate recovery and liveness ensuring eventual safety. The last two specifications capture again eventual safety; however, these formulae feature an arbitrary nesting of the temporal operator F. We use these formulae to conduct an ablation study evaluating the scalability of our approach against the existing rewriting-based procedure. The precise meaning of the proposition safe is defined for each scenario in the following.

4.1 Vertical Collision Avoidance System (VCAS)

We consider two variations of the collision avoidance scenario introduced in [Julian *et al.*, 2019] and studied in [Akintunde *et al.*, 2022] and [Akintunde *et al.*, 2020]. The original VCAS proposal considers a system whereby the pilot of an *ownship* aircraft receives control commands based on a neural network-controlled collision avoidance system so as to avoid colliding with an *intruder* aircraft. More specifically, the system issues one of nine advisories that together with the current climbrate of the ownship determines a range of accelerations from which the pilot can choose, in order to avoid a near mid-air collision (NMAC) with the intruder (i.e., a region where the aircraft are separated by less than 100 ft vertically and 500 ft horizontally).

In our first set of experiments, and following the formalisation proposed in [Akintunde *et al.*, 2020], we implement the VCAS scenario as a single-agent problem, where the intruder agent in assumed to follow a horizontal trajectory throughout. We denote this setting as VCAS-1. We then consider a more complex extension, denoted as VCAS-2, were both aircraft are assumed to be controlled by neuro-symbolic agents and the assumption that the intruder follows a horizontal trajectory is dropped. The VCAS system relies on nine ReLU-based feed-forward neural networks, one for each advisory, with three inputs for VCAS-1 and four inputs for VCAS-2, seven hidden layers of 45 nodes and nine outputs representing the score of each possible advisory. We refer to [Akintunde *et al.*, 2020] for further details on how the VCAS problem can be represented as a NIS and encoded in MILP. Further details on the experimental settings considered for these experiments can be found in the appendix.

Experimental results. Figure 1 shows the results obtained for the VCAS-1 scenario with branching factor 2. Additional results for branching factor 1 are reported in the appendix. The core of our results highlights a complementary relationship between the exponential and succinct encodings, contingent upon the structure of the target LTL formula. For properties which involve no nesting of the until operator (recall that G and F are expressed using U), the exponential approach showed a better or comparable performance to our approach in terms of verification runtime. This suggests that in simpler cases where the necessary rewriting for the exponential approach does not yield a substantial formula blow-up, the exponential encoding is to be preferred. Conversely, for properties which feature increasing depth of the until operator nesting, the succinct approach proved significantly more effective. As the nesting depth increases, the explicit rewriting of the LTL formula into an equivalent, but much larger, formula for the exponential encoding leads to an explosion in formula size. Our method, by leveraging its compact representation, mitigates this explosion, resulting in substantially reduced runtimes. A similar pattern was also observed for the VCAS-2 scenario, as shown in Figure 2. For VCAS-1, out of 1120 individual verification queries, exponential encoding timed out 132 times, and on top of that, run out of memory 45 times. Conversely, our encoding timed out 32 times and had no memory issues. For VCAS-2, out of 560 individual verification queries, exponential encoding timed out 25 times and run out of memory 33 times. Conversely, our encoding timed out 54 times and had no memory issues. While both encodings struggled on the same instances, our succinct encoding never exceeded the memory limit and was never terminated by the operating system.

4.2 Automated Cruise Control (ACC)

We consider a traffic jam scenario based on [Sugiyama *et al.*, 2008]. In this scenario, K agents are initially uniformly spaced on a circular track of length Λ and are all moving with the same speed. The track has Λ varying values of friction F against motion at each location. Each agent knows its own speed and remaining fuel, and is equipped with an ACC system that suggests the acceleration to the agent. The ACC is implemented by means of a feed forward ReLU network and is trained using Reinforcement Learning (more details about training and implementation are available in the appendix). The environment tracks the location and speed of each agent, so knows the friction of the track at their location. The goal of the agents is to go as far as possible without crashing into each other. A full formalisation of this problem into the MAS setting, as well as further details on the parameters used for these experiments are provided in the appendix.

Experimental results. Figure 3 shows the results obtained for the ACC scenario across 7 specifications and increasing number of agents (2-6). These experiments were conducted with a branching factor of 1 to enable a focused evaluation of the effect of increasing the number of agents, as well as the

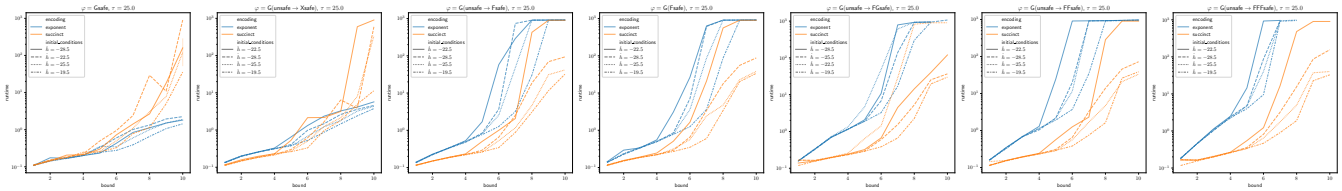

Figure 1: Runtimes for our encoding (succinct) and SOTA (exponent) on VCAS-1 across formulae φ^k_{1-7} for $k \in [1, 10]$, branching factor 2.

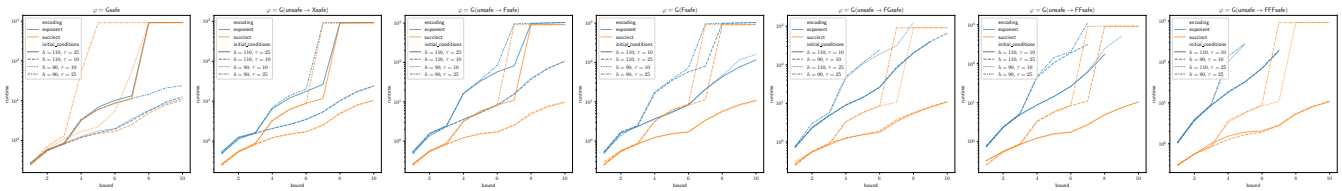

Figure 2: Runtimes for our encoding (succinct) and SOTA (exponent) on VCAS-2 across formulae φ^k_{1-7} for $k \in [1, 10]$, branching factor 2.

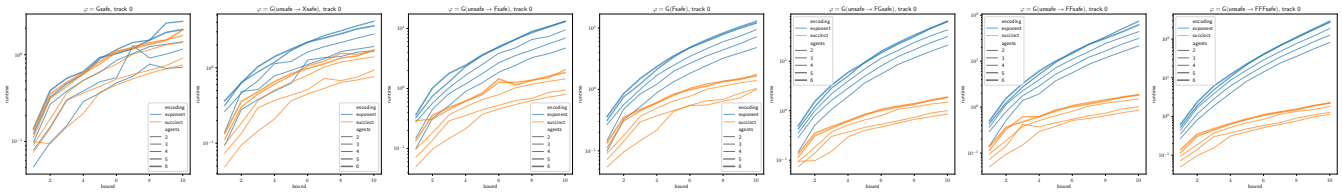

Figure 3: Runtimes for our encoding (succinct) and SOTA (exponent) on ACC across formulae φ^k_{1-7} for $k \in [1, 10]$, branching factor 1.

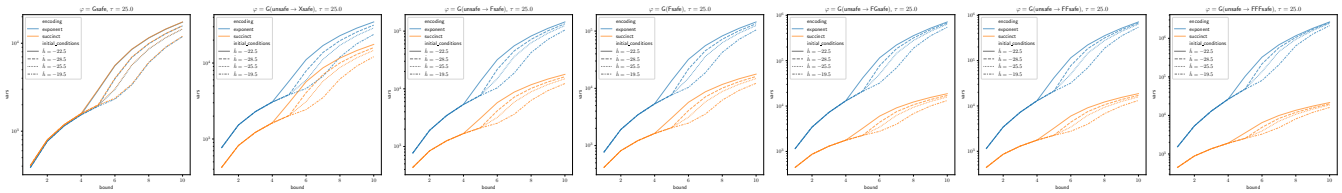

Figure 4: Number of variables for VCAS-1 across formulae φ^k_{1-7} with $k \in [1, 10]$ and branching factor 2.

nesting of temporal operators. Our results reveal a clear separation in runtimes between the two encodings. We observe that for both encodings verification times increase both with the temporal depth and number of agents. However, the slope is significantly less pronounced in our case. Finally, for specification where nesting does not feature prominently, the two encodings exhibit similar performance, with no clear winner emerging. These results are aligned with our observations on VCAS and highlight the complementary benefits of our encoding compared to VENMAS.

4.3 Encoding Size Evaluation

Beyond execution time, we empirically evaluated the compactness of our MILP formulation. We collected statistics on the number of variables and constraints generated by our linear encoding compared to the rewriting-based approach across all benchmarks. The data confirms that our method is consistently more succinct, producing significantly smaller programs as the complexity of the LTL specification increases. As theoretically predicted, both encodings result in the same number of variables and constraints for simple bounded safety properties without nested operators. For all other specifications, our approach avoids the exponential growth inherent in rewriting. This trend is exemplified in Figure 4, which compares the number of variables for the VCAS-1 benchmark (branching factor 2) for the two encodings across all specifications. Detailed comparative plots for the full suite of experiments are available in the appendix.

5 Conclusion

This paper presented a novel MILP encoding for the verification of neuro-symbolic MAS against LTL specifications. Unlike existing approaches that are limited to bounded LTL fragments, our encoding handles general LTL specifications and is linear in both temporal depth and formula size. Experimental results across various benchmarks demonstrate the benefits of our approach, complementing existing methods for neuro-symbolic MAS verification.

The empirical results obtained highlight a complementary relationship between our encoding and VENMAS. In par-

ticular, the efficiency of our encoding is more pronounced when verifying formulae containing nested temporal operators, where VENMAS often encounters memory or timeout issues. These results suggest that our method provides a valuable alternative for verifying systems against intricate temporal requirements that push existing state-of-the-art methods to their computational limits.

References

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

Michael E. Akintunde, Andreea Kevorchian, Alessio Lomuscio, and Edoardo Pirovano. Verification of rnn-based neural agent-environment systems. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI 2019)*, pages 6006–6013. AAAI Press, 2019.

Michael Akintunde, Elena Botoeva, Panagiotis Kouvaros, and Alessio Lomuscio. Verifying strategic abilities of neural-symbolic multi-agent systems. In *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning (KR 2020)*, pages 22–32. IJCAI Press, 2020.

Michael E. Akintunde, Elena Botoeva, Panagiotis Kouvaros, and Alessio Lomuscio. Formal verification of neural agents in non-deterministic environments. *Auton. Agents Multi Agent Syst.*, 36(1):6, 2022.

M. E. Akintunde, E. Botoeva, P. Kouvaros, and A. Lomuscio. VENMAS: Verification of neural-symbolic multi-agent systems, https://github.com/NeuralMAS/venmas/, 2024.

Armin Biere, Alessandro Cimatti, Edmund M. Clarke, and Yunshan Zhu. Symbolic model checking without bdds. In *Proceedings of the 5th International Conference on Tools and Algorithms for Construction and Analysis of Systems (TACAS 1999)*, volume 1579 of *LNCS*, pages 193–207. Springer, 1999.

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

Sergiy Bogomolov, Marcelo Forets, Goran Frehse, Kostiantyn Potomkin, and Christian Schilling. Juliareach: a toolbox for set-based reachability. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control (HSCC 2019)*, pages 39–44. ACM, 2019.

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

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

Timon Gehr, Matthew Mirman, Dana Drachsler-Cohen, Petar Tsankov, Swarat Chaudhuri, and Martin T. Vechev. AI2: safety and robustness certification of neural networks with abstract interpretation. In *Proceedings of the 39th IEEE Symposium on Security and Privacy (SP 2018)*, pages 3–18. IEEE Computer Society, 2018.

Ian J. Goodfellow, Yoshua Bengio, and Aaron C. Courville. *Deep Learning*. Adaptive computation and machine learning. MIT Press, 2016.

Patrick Henriksen and Alessio R. Lomuscio. Efficient neural network verification via adaptive refinement and adversarial search. In *Proceedings of the 24th European Conference on Artificial Intelligence (ECAI 2020)*, volume 325 of *Frontiers in Artificial Intelligence and Applications*, pages 2513–2520. IOS Press, 2020.

Chao Huang, Jiameng Fan, Xin Chen, Wenchao Li, and Qi Zhu. POLAR: A polynomial arithmetic framework for verifying neural-network controlled systems. In *Proceedings of the 20th International Symposium on Automated Technology for Verification and Analysis (ATVA 2022)*, volume 13505 of *LNCS*, pages 414–430. Springer, 2022.

R. Ivanov, J. Weimer, R. Alur, G. J. Pappas, and I. Lee. 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 (HSCC 19)*, pages 169–178. ACM, 2019.

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

Guy Katz, Clark W. Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer. Reluplex: An efficient SMT solver for verifying deep neural networks. In *Proceedings of the 29th International Conference on Computer Aided Verification (CAV 2017)*, volume 10426 of *LNCS*, pages 97–117. Springer, 2017.

Diego Manzanas Lopez, Sung Woo Choi, Hoang-Dung Tran, and Taylor T. Johnson. NNV 2.0: The neural network verification tool. In *Proceedings of the 35th International Conference on Computer Aided Verification (CAV 2023)*, volume 13965 of *LNCS*, pages 397–412. Springer, 2023.

A. Pnueli. The temporal logic of programs. In *Proceedings of the 18th International Symposium Foundations of Computer Science (FOCS77)*, pages 46–57, 1977.

Yuki Sugiyama, Minoru Fukui, Macoto Kikuchi, Katsuya Hasebe, Akihiro Nakayama, Katsuhiro Nishinari, Shin-ichi Tadaki, and Satoshi Yukawa. Traffic jams without bottlenecks—experimental evidence for the physical mechanism of the formation of a jam. *New Journal of Physics*, 10(3):033001, 2008.

Hoang-Dung Tran, Stanley Bak, Weiming Xiang, and Taylor T. Johnson. Verification of deep convolutional neural networks using imagestars. In *Proceedings of the 32nd International Conference on Computer Aided Verification (CAV 2020)*, volume 12224 of *LNCS*, pages 18–42. Springer, 2020.

Huan Zhang, Shiqi Wang, Kaidi Xu, Yihan Wang, Suman Jana, Cho-Jui Hsieh, and J. Zico Kolter. A branch and bound framework for stronger adversarial attacks of relu networks. In *Proceedings of the 39th International Conference on Machine Learning (ICML 2022)*, volume 162 of *Proceedings of Machine Learning Research*, pages 26591–26604. PMLR, 2022.