Verification of Neural Network Control Systems using Symbolic Zonotopes and Polynotopes

This work has been submitted to the IEEE for possible publication. Copyright may be transferred without notice, after which this version may no longer be accessible.

Carlos Trapiello, Christophe Combastel, and Ali Zolghadri, Senior Member, IEEE

Abstract—Verification and safety assessment of neural network controlled systems (NNCSs) is an emerging challenge. To provide guarantees, verification tools must efficiently capture the interplay between the neural network and the physical system within the control loop. In this paper, a compositional approach focused on inclusion preserving long term symbolic dependency modeling is proposed for the analysis of NNCSs. First of all, the matrix structure of symbolic zonotopes is exploited to efficiently abstract the input/output mapping of the loop elements through (inclusion preserving) affine symbolic expressions, thus maintaining linear dependencies between interacting blocks. Then, two further extensions are studied. Firstly, symbolic polynotopes are used to abstract the loop elements behaviour by means of polynomial symbolic expressions and dependencies. Secondly, an original input partitioning algorithm takes advantage of symbol preservation to assess the sensitivity of the computed approximation to some input directions. The approach is evaluated via different numerical examples and benchmarks. A good trade-off between low conservatism and computational efficiency is obtained.

Index Terms—Reachability, neural networks, verification, symbolic zonotopes, polynotopes, nonlinear dynamics.

I. INTRODUCTION

THE proliferation of data and access to ever-increasing computational power have fueled a renewed interest in deep neural-networks (NNs). These networks have shown a significant ability to address classification/estimation/control tasks that can hardly be formalized and designed from knowledge-based models. However, despite their impressive ability for solving complex problems, it is well known that NNs can be vulnerable to small perturbations or adversarial attacks [1], [2]. This lack of robustness (or fragility) represents a major barrier for their application to safety-critical system where safety assurances are of primary importance. For example, in Guidance, Navigation and Control of flight systems, one must ensure that some output/state trajectories remain inside a flight envelope when some inputs explore a given region. The above issues have fostered a large amount of works that analyze the sensitivity to local disturbances of NNs in isolation (open-loop), as well as the satisfaction of pre-/post- safety conditions [3]. Nevertheless, as reported in [4], reasoning about the safety verification of neural-network control systems (NNCSs), where the NN is used as a feedback controller, still remains a key challenge that requires tractable

The authors are with Univ. Bordeaux, CNRS, Bordeaux INP, IMS, UMR 5218, F-33400 Talence, France; C. Trapiello is also with the Supervision, Safety and Automatic Control Research Center (CS2AC), of the Universitat Politècnica de Catalunya (UPC), 08222 Terrassa, Spain.

methods capable of efficiently integrating the heterogeneous components that make up the control loop.

This paper focuses on the reachability analysis of NNCSs which, in turn, allows for formal reasoning about the satisfaction of safety properties (reachability of a target set, or the avoidance of non-secure sets of states). A key challenge in NNCSs reachability analysis is to successfully retain the system-controller interplay by preserving (at each time instant) the dependencies between relevant variables. This, in fact, discourages a direct application of off-the-shelf verification tools which, although able to compute accurate output bounds for elements in isolation, return coarse approximations when iteratively concatenated for the analysis of closed-loop systems since most of (if not all) the I/O dependencies are quickly broken/lost during the computations [5]–[7]. Furthermore, effective NNCSs verification tools must be able to assess the system state during (relatively) large time intervals. The above issues motivate the development of computationally efficient analysis methods capable of capturing the interaction between the control loop elements while granting a good scalability both in the system dimensions and in the time horizon length.

Another relevant factor that should be taken into account is the size of the initial state set under study. Mainly, the performance of open- and closed-loop verification techniques that are based on (locally) abstracting the system non-linearities, deteriorates considerably for large initial sets. A common approach to address this issue, particularly in NNCSs verification problems where the number of dimensions is relatively small, is to recast the initial reachability problem into simpler subproblems that analyze a subset of the initial conditions [5], [7]–[13]. Nonetheless, the design of efficient and scalable partitioning strategies, specially in closed-loop verification schemes, remains also an open problem.

Related work: Preserving dependencies for NNCS verification has spurred on some recent studies. In [14], the authors abstract the I/O mapping of a ReLU NN controller using a polynomial expression (plus an error interval). The polynomial rule is obtained by regression of I/O samples, whereas a sound error term is derived from solving a mixed-integer program (MIP). In a similar fashion, [15] uses Bernstein polynomials to abstract the NN controller. A theoretical and a sampled-based method is proposed to compute the error term based on the Lipschitz constant of the NN. Although both approaches preserve the system-controller interplay, they are computationally expensive, scaling poorly with the number of NN inputs while requiring to be iteratively repeated for each output. In [9] a NN with differentiable activation functions is transformed into an equivalent hybrid system built upon Taylor models that retain dependencies. However, this approach is not applicable to ReLU functions and the number of states (resp. modes) of the hybrid automaton scales with the number of neurons (resp. layers).

Other approaches preserve system-controller dependencies by formulating the reachable set computation as an optimization problem. The work [16] proposes a semidefinite program for reachability analysis based on the abstraction of the NN non-linearities using quadratic constraints [17]. [11] relies on the tool CROWN [18], and preserves system-controller interaction by solving LP-programs. However, dependencies are broken from one sample to the next. In [10], the closed-loop is firstly abstracted as a conjunction of piecewise-linear functions, and then analyzed using ReLU NNs verification tools like [19], [20].

On the other hand, other works address the reachability problem by chaining different verification tools. In [5], the authors combine a polytopic abstraction of the dynamical system with the tool Sherlock [21], that is used to bound the NN controller outputs. The tool NNV [6], integrates the non-linear dynamics reachability tool CORA [22] with a star sets abstraction of ReLU NN controllers [23]. Besides, [7] combines validated simulation to soundly approximate the dynamical system with common tools for NN output bounding like DeepPoly [24]. In all the above works, dependencies are broken in the switch between the different tools. This latter issue is somehow palliated in [8], where the authors use second order zonotopes (i.e. zonotopes with generators matrix size $n \times 2n$) as an interface between system and NN controller analysis tools. Although capable to retain first order dependencies in the system-to-controller (and controller-tosystem) set transformations, dependencies in the I/O mapping of the NN controller are broken.

Focusing on partitioning strategies, in [25] the gradient of a ReLU NN (open-loop) is used to decide the next input direction to be bisected, whereas in [26] a uniform grid of the initial set is employed. Other works propose a simulation-based splitting strategy. In [12], the bisection is guided by comparing the interval bound of Monte-Carlo samples with a guaranteed Interval Bound Propagation [27] of the initial subsets. Working in a similar fashion, [13] proposes a simulation-guided framework that unifies standard NN output bounding tools. The decision on the bisection order is based on the distance to the simulation samples enclosure. A closed-loop implementation of the latter algorithm is reported in [11].

Contributions: This paper takes a new and original direction based on symbolic zonotopes (s-zonotopes) as a generic tool for the closed-loop verification of discrete-time NN controlled systems. The generators (matrix) representation of s-zonotopes enables to efficiently abstract the input-output mappings of the NN controller and non-linear physical system through (inclusion preserving) affine symbolic expressions. The evolution of the closed-loop system can then be bounded in a propagation-based fashion that benefits from the efficient computation of basic operations granted by s-zonotopes, while preserving system-controller linear dependencies. Besides, the

computational complexity of the verification tool can be fixed by limiting (reducing) the number of independent symbols. Simulations show the good performance/computational efficiency trade-off granted by this approach.

Furthermore, two extensions are proposed. On the one hand, the use of polynomial symbolic expressions to abstract the input-output mapping of the loop elements is explored. In particular, symbolic polynotope (s-polynotope) structures [28] are used to enclose the NN activation functions graph via the non-convex sets that arise from the polynomial map of interval symbols. Polynomial abstractions enable to reduce the conservatism induced by linear relaxations, at the price of increasing the computation needs.

On the other hand, the symbols preservation throughout the control loop is exploited to develop a smart partitioning strategy of the initial conditions set. The proposed algorithm reasons upon the influence of the input symbols in the output set in order to select which dimension to bisect next, and upon the influence of the (independent) error symbols to assess the quality of each over-approximation.

Structure: The paper is organized as follows. Section II is devoted to some useful preliminaries. Section III introduces the problem statement. Then, Section IV analyzes the closed-loop verification using s-zonotopes. In Section V the use of s-polynotopes is investigated, whereas the input partitioning algorithm is detailed in Section VI. Section VII presents simulation results. Finally, some concluding remarks are provided in Section VIII.

Notation: The following notations are used along this work. \mathbb{R}^n , $\mathbb{R}^{m \times n}$ and \mathbb{N} denote the n dimension Euclidean space, the $m \times n$ dimensional Euclidean space and the set of non-negative integers, respectively. The notation v_i stands for the i-th element of vector v and $M_{[i,:]}$ ($M_{[:,j]}$) for the i-th row (j-th column) of matrix M. The 1-norm of the (row) vector v is $\|v\|_1 = |v|\mathbf{1}$, with $|\cdot|$ the elementwise absolute value, and $\mathbf{1}$ a column vector of ones of appropriate size. diag(v) returns a square diagonal matrix with the elements of vector v in the main diagonal, whereas $card(\cdot)$ gives the cardinal.

II. SYMBOLIC DEPENDENCIES IN SET COMPUTATIONS

This section provides preliminary concepts which will be used in the following sections. Throughout this article, s refers to an indexed family of distinct symbolic variables of type unit interval, that is, $\forall i \in \mathbb{N}$, the symbol s_i (uniquely identified by the integer i) refers to a scalar real variable the value of which is only known to belong to the unit interval $\mathcal{D}(s_i) = [-1, +1] \subset \mathbb{R}$. Also, $\mathcal{D}(s) = [-1, +1]^{card(s)}$. In other words, the a priori unknown value ιs_i taken by the symbolic variable s_i satisfies $\iota s_i \in \mathcal{D}(s_i)$. The generic notation ι which reads as "interpretation/valuation of" helps disambiguate between symbols (syntax) and values (semantics) [28]. Note that, in general, several interpretations may coexist. Setvalued interpretations take sets as values. In the following, consistently with the definition domain $\mathcal{D}(s_i)$ related to s_i , the set-valued interpretation of each symbolic variable s_i will be $s_{i|\iota} = [-1, +1]$. In addition, the integer-valued vector I is used to uniquely identify a set of symbols, for example, vector

I = [1, 5, 3] identifies the symbols s_1 , s_5 and s_3 . For brevity of notation, s_I denotes the column vector $[s_i]_{i \in I}$.

Definition 1 (s-zonotope [28]). A symbolic zonotope $\mathcal{X}_{|s}$ is an affine symbolic function that can be written in the form $c+Rs_I$ where vector c and matrix R do not depend on the symbolic variables in s_I . Notation: $\mathcal{X}_{|s} = \langle c, R, I \rangle_{|s} = c + Rs_I$.

Definition 2 (e-zonotope [28]). The e-zonotope $\mathcal{X}_{|\iota}$ related to the s-zonotope $\mathcal{X}_{|s} = \langle c, R, I \rangle_{|s} = c + Rs_I$ is the set-valued interpretation of $\mathcal{X}_{|s}$ as $\mathcal{X}_{|\iota} = \langle c, R, I \rangle_{|\iota} = \{c + R\sigma | \sigma \in \mathcal{D}(s_I)\}.$

A basic example is: given $i \in \mathbb{N}$, $\langle 0,1,i \rangle_{|s} = s_i$ (symbolic expression corresponding to the i-th symbol in s) and $\langle 0,1,i \rangle_{|\iota} = \mathcal{D}(s_i) = [-1,+1]$ (set-valued interpretation of s_i). More generally, s-zonotopes and their interpretation as e-zonotopes make it possible to explicitly perform operations either at symbolic/syntactic level $(\cdot_{|s})$ or at semantic level $(\cdot_{|\iota})$.

Remark 1. In this work, all symbols being of type unit interval, c being a real vector and R a real matrix, $\mathcal{X}_{|\iota}$ is a classical zonotope $\langle c,R\rangle$ with center c and generator matrix R (for extensions to other symbol types, see [28]). Note that $\langle 0,1,i\rangle_{|s}-\langle 0,1,j\rangle_{|s}=s_i-s_j=0$ for i=j, whereas $\langle 0,1,i\rangle_{|\iota}-\langle 0,1,j\rangle_{|\iota}=[-2,+2]$ for all (i,j), that is, even for i=j. Operating at the symbolic/syntactic level thus permits more accurate set evaluations by preserving trace of symbolic dependencies. This is a key point to prevent from pessimistic outer approximations induced by the so-called dependency problem [29] affecting natural interval arithmetic and other classical set-based operations only considering the semantic level.

From a computational point of view, an s-zonotope is defined by storing the triplet (c, R, I). Due to their affine structure, a key aspect is to efficiently trace the identifier $i \in I$ of the symbol that multiplies each column of the matrix R. To that end, Matrices with Labeled Columns (MLCs), constitute a data structure featuring columnwise sparsity: It is defined by the pair (R, I) that allows for efficiently recasting standard operations involving s-zonotopes as set-operations on the identifiers vector (I) and column-wise operations in the projection matrices (R). For how to translate operations such as sum or linear image onto a computational platform using MLCs the interested reader can refer to [30].

Due to their relevance in further developments, the following operations involving s-zonotopes are briefly recalled.

Lemma 1 (common symbols [30]). Any two s-zonotopes $\mathcal{X}_{|s} = \langle c_x, R, I \rangle_{|s}$ and $\mathcal{Y}_{|s} = \langle c_y, G, J \rangle_{|s}$, can be rewritten using a common set of symbols s_K as $\mathcal{X}_{|s} = \langle c_x, \tilde{R}, K \rangle_{|s}$ and $\mathcal{Y}_{|s} = \langle c_y, \tilde{G}, K \rangle_{|s}$, with

$$\tilde{R} = \begin{bmatrix} R_1, & R_2, & 0 \end{bmatrix}, \quad \tilde{G} = \begin{bmatrix} G_1, & 0, & G_2 \end{bmatrix},
K = \begin{bmatrix} I \cap J; & I \setminus J; & J \setminus I \end{bmatrix}.$$
(1)

Matrices (R_1, G_1) in Lemma 1 may be empty matrices if $I \cap J$ is empty (similarly for R_2 and $I \setminus J$ or G_2 and $J \setminus I$).

Definition 3 (basic operations [30]). Given two s-zonotopes $\mathcal{X}_{|s}$ and $\mathcal{Y}_{|s}$ with a common set of symbols s_K as in Lemma 1, then their sum and vertical concatenation are the s-zonotopes

$$\mathcal{X}_{|s} + \mathcal{Y}_{|s} = \langle c_x + c_y, [R_1 + G_1, R_2, G_2], K \rangle_{|s},$$
 (2)

$$[\mathcal{X}_{|s}; \mathcal{Y}_{|s}] = \left\langle \begin{bmatrix} c_x \\ c_y \end{bmatrix}, \begin{bmatrix} R_1 & R_2 & 0 \\ G_1 & 0 & G_2 \end{bmatrix}, K \right\rangle_{|s}.$$
 (3)

Definition 4 (inclusion [28]). The s-zonotope $\mathcal{Y}_{|s}$ is said to include the s-zonotope $\mathcal{X}_{|s}$, if the set-valued interpretation of $\mathcal{Y}_{|s}$ includes the set-valued interpretation of $\mathcal{X}_{|s}$. In other words, the expression $\mathcal{X}_{|s} \subset \mathcal{Y}_{|s}$ interprets as $\mathcal{X}_{|\iota} \subset \mathcal{Y}_{|\iota}$.

Definition 4 paves the way for rewriting rules (at symbolic level) that may be either inclusion preserving or inclusion neutral or none of both at set-based evaluation (semantic) level. A more formal treatment of this topic can be found in the definition 27 (rewriting rules and inclusion) in [28], where a definition of inclusion functions is also given in definition 2.

Definition 5 (reduction [28]). The reduction operator \downarrow_q transforms an s-zonotope $\mathcal{X}_{|s} = \langle c, R, I \rangle_{|s}$ into a new s-zonotope $\tilde{\mathcal{X}}_{|s} = \downarrow_q \mathcal{X}_{|s} = \langle c, G, J \rangle_{|s}$, such that $\tilde{\mathcal{X}}_{|s}$ includes $\mathcal{X}_{|s}$ while depending on at most q symbols, i.e. $card(J) \leq q$.

Reduction is thus an inclusion preserving transform. In Definition 5, $I \cap J \neq \emptyset$ is not mandatory but often useful to prevent from a further propagation of conservative approximations, while controlling the complexity of $\tilde{\mathcal{X}}_{|s}$ through the maximum number q of its symbols/generators. In this context, preserving the more significant symbols/dependencies is often beneficial: as in [30], if p>q a common practice is to replace the p-q+1 less important symbols by a new independent one while guaranteeing the inclusion $\mathcal{X}_{|s}\subseteq \tilde{\mathcal{X}}_{|s}$. Besides, note that new symbols introduced to characterize independent behaviors must be uniquely identified. Wherever needed, the generation of a vector of n new unique symbols identifiers is denoted as !(n). The generation of a pre-specified number of identifiers can be attained by implementing, for example, the Unique Symbols Provider (USP) service introduced in [30].

III. PROBLEM STATEMENT

A. System description

Consider the interconnection of a discrete-time non-linear dynamic model (4) and a neural network. The physical system is modeled as:

$$x(t+1) = f(x(t), u(t), w(t)),$$
 (4)

where $x(t) \in \mathbb{R}^{n_x}$ and $u(t) \in \mathbb{R}^{n_u}$ respectively refer to the state and the control input at time step $t \in \mathbb{N}$. For all $t \geq 0$, vector w(t) accounts for modeling errors and process disturbances and satisfies $w(t) \in \mathcal{W} = [-1, +1]^{n_w}$.

The system (4) is controlled by a state-feedback controller $g(x(t)): \mathbb{R}^{n_x} \mapsto \mathbb{R}^{n_u}$ parameterized by an l-layer feed-forward fully connected neural network. The map $x \mapsto g(x)$ is described by the following recursive equations

$$x^{(0)} = x,$$

$$x^{(k+1)} = \phi^{(k)} (W^{(k)} x^{(k)} + b^{(k)}), \quad k = 0, ..., l-1, \quad (5)$$

$$g(x) = W^{(l)} x^{(l)} + b^{(l)},$$

where $x^{(k)} \in \mathbb{R}^{n_k}$ are the outputs (post-activation) of the k-th layer. The weight matrix $W^{(k)} \in \mathbb{R}^{n_{k+1} \times n_k}$ and bias $b^{(k)} \in \mathbb{R}^{n_{k+1}}$ define the affine mapping $z^{(k)} = W^{(k)}x^{(k)} + b^{(k)}$ for the (k+1)-th layer. Besides, the vector-valued function $\phi^{(k)}: \mathbb{R}^{n_{k+1}} \to \mathbb{R}^{n_{k+1}}$ is applied element-wise to the pre-activation vector $z^{(k)}$, that is, $\phi^{(k)}(z^{(k)}) = [\varphi(z_1^{(k)}), \cdots, \varphi(z_{n_{k+1}}^{(k)})]^T$, where $\varphi: \mathbb{R} \to \mathbb{R}$ is the (scalar) activation function. Common activation choices are: ReLU $\varphi(z) = max(0,z)$; sigmoid $\varphi(z) = \frac{1}{1+e^{-z}}$; and $tanh \varphi(z) = tanh(z)$.

The closed-loop system with dynamics (4) and a previously trained neural-network control policy (5), is governed by

$$x(t+1) = f_q(x(t), w(t)) = f(x(t), g(x(t)), w(t)).$$
 (6)

Accordingly, given an initial set $\mathcal{X}_0 \subset \mathbb{R}^{n_x}$, the forward reachable set of (6) at time step t is denoted as $\mathcal{X}(t)$. For $t \geq 1$, this set is defined as:

$$\mathcal{X}(t) = \left\{ x(t) \mid \exists (x(0), w(0:t-1)) \in \mathcal{X}_0 \times \mathcal{W} \times \dots \times \mathcal{W}, \\ \forall \tau \in [0, t-1], x(\tau+1) = f_g(x(\tau), w(\tau)) \right\}.$$
(7)

B. Finite-time reach-avoid (RA) verification problem

Given a goal set $\mathcal{G} \subset \mathbb{R}^{n_x}$ a sequence of avoid sets $\mathcal{A}(t) \subset \mathbb{R}^{n_x}$ and a finite time horizon $N \in \mathbb{N}^+$, it is desired to test whether

$$\mathcal{X}(N) \subseteq \mathcal{G}$$

 $\mathcal{X}(t) \cap \mathcal{A}(t) = \emptyset, \quad \forall t = 0, ..., N-1$ (8)

holds true for the closed loop system (6). In general, the exact evaluation of (8) for a NNCSS is a computationally intractable problem. Thus, the problem is resorted to iteratively compute a tractable over-approximation of the reachable set $\mathcal{X}(t) \subseteq \bar{\mathcal{X}}(t)$, to test (8) using $\bar{\mathcal{X}}(t)$ instead. Because of the overapproximation, the proposed verification setting only provides one-sided guarantees, that is, if $\bar{\mathcal{X}}(t)$ satisfies (8) then it can be guaranteed that (7) will satisfy the RA property, but no sound conclusion about the safety of (7) can be made if the overapproximation $\bar{\mathcal{X}}(t)$ violates (8). Therefore, the computation of tight over-approximations is of paramount importance, so that a maximum number of truly satisfied specifications can be computationally proven as such.

IV. CLOSED-LOOP VERIFICATION USING S-ZONOTOPES

This section presents the methodology for computing a sound over-approximation of the closed-loop system that preserves system-controller linear dependencies. The computation takes advantage of s-zonotopes described in the previous section. The abstraction of the control loop components using affine symbolic expressions is presented below.

A. Initial set

It is assumed that the initial set can be described by the set-valued interpretation of an s-zonotope $\mathcal{X}_{|s}(0) = \langle c_0, R_0, I_0 \rangle_{|s}$, where $c_0 \in \mathbb{R}^{n_x}$ and $R_0 \in \mathbb{R}^{n_x \times n_0}$ and $I_0 = !(n_0)$ is a set of n_0 unique identifiers for the interval valued symbols s_{I_0} . In other words, it is assumed that $\mathcal{X}_0 = \mathcal{X}_{|\iota}(0)$. Note that, any arbitrary zonotopic set $\{c + R\xi \mid \|\xi\|_{\infty} \leq 1\}$ can be abstracted as an s-zonotope by characterizing the independent behaviour of the generators through new interval type symbols.

B. NN controller affine abstraction

For the sake of simplicity of notations, the temporal notation is dropped here. Given a state bounding s-zonotope $\mathcal{X}_{|s} = \langle c, R, I \rangle_{|s}$ and a NN controller (5), the idea is to abstract the NN behavior through an affine symbolic expression of the form

$$\mathcal{U}_{|s} = \langle C_u, [G, H], [I; J] \rangle_{|s} = c_u + Gs_I + Hs_J, \quad (9)$$

such that, it guarantees the local enclosure of the network outputs, i.e. $g(\mathcal{X}_{|\iota}) \subseteq \mathcal{U}_{|\iota}$. Note that, expression (9) captures the linear dependencies of the state symbols (identified by I), plus the addition of new error symbols (identified by J) that are introduced to guarantee the soundness of the method.

The computation of vector c_u , matrices G, H, and the identifiers vector J is discussed below. The focus is on generating a dependencies-preserving inclusion for an arbitrary layer of NN (5), since a sound enclosure for the whole network follows by induction due its sequential nature. For simplicity, the layer superscript is removed below and the superscript $^+$ is used to denote the next layer.

Affine mapping Given the s-zonotope $\mathcal{X}_{|s} = \langle c, R, I \rangle_{|s}$, the affine mapping $\mathcal{Z}_{|s} = W \mathcal{X}_{|s} + b$ in the layers of (5) yields a (pre-activation) s-zonotope of the form

$$\mathcal{Z}_{|s} = \langle \check{c}, \check{R}, I \rangle_{|s},
\check{c} = Wc + b, \quad \check{R} = WR.$$
(10)

Activation functions Activation functions $\varphi(\cdot)$ in (5) are applied element-wise to the pre-activation vector. Hence, the projection of $\mathcal{Z}_{|s|}$ onto the *i*-th neuron, yields the s-zonotope

$$\mathcal{Z}_{i|s} = \langle \check{c}_i, \check{R}_{[i,:]}, I \rangle_{|s}. \tag{11}$$

Notice that, any point belonging to set-valued interpretation $\mathcal{Z}_{i|\iota}$ of (11) is confined within an interval $[l_i, u_i]$, where, since $\mathcal{Z}_{i|\iota}$ is a one dimensional zonotopic set, it follows that $\mathcal{Z}_{i|\iota} = [l_i, u_i]$ with the lower and upper bounds

$$l_i = \check{c}_i - ||\check{R}_{[i,:]}||_1, \quad u_i = \check{c}_i + ||\check{R}_{[i,:]}||_1.$$
 (12)

Therefore, the soundness of the method can be certified by guaranteeing the inclusion (see Definition 4) of the graph of the activation function in the range $[l_i, u_i]$. To that end, the activation function $\varphi(\cdot)$ is abstracted through an affine symbolic function of the form

$$\mathcal{X}_{i|s}^{+} = \alpha_i \mathcal{Z}_{i|s} + \beta_i + \gamma_i s_j, \tag{13}$$

where s_j represents a new independent symbol (identified through j = !(1)) that must be introduced to guarantee the full coverage of the activation function graph on the considered range, that is, in order to satisfy the condition

$$\begin{bmatrix} \mathcal{Z}_{i|\iota} \\ \varphi(\mathcal{Z}_{i|\iota}) \end{bmatrix} \subseteq \begin{bmatrix} \mathcal{Z}_{i|\iota} \\ \alpha_i \mathcal{Z}_{i|\iota} + \beta_i + \gamma_i \mathcal{D}(s_j) \end{bmatrix}. \tag{14}$$

The *i*-th neuron post-activation s-zonotope $\mathcal{X}_{i|s}^+$ in (13) not only guarantees that its set-valued interpretation encloses the neuron output, but it preserves the linear influence of the symbols s_I in the output set. This later point plays a fundamental role since it allows to retain the interplay between the inputs of the neurons at the same layer. Coherently, the

layer post-activation s-zonotope can be computed by vertically concatenating in a recursive fashion the different $\mathcal{X}_{i|s}^+$ after rewriting them using the same set of symbols

$$\mathcal{X}_{|s}^{+} = [\dots[[\mathcal{X}_{1|s}^{+}; \mathcal{X}_{2|s}^{+}]; \mathcal{X}_{3|s}^{+}] \dots; \mathcal{X}_{n_{k}|s}^{+}]. \tag{15}$$

Proposition 1 (NN s-zonotope). Given the s-zonotope $\mathcal{X}_{|s}^{(0)} = \langle c^{(0)}, R^{(0)}, I^{(0)} \rangle_{|s|}$, and let $\alpha^{(k)}, \beta^{(k)}, \gamma^{(k)} \in \mathbb{R}^{n_k+1}$ be some parameter vectors that guarantee the inclusion of the n_{k+1} activation functions in the k-th layer, then the enclosure of the NN output set $g(\mathcal{X}_{|\iota}^{(0)}) \subseteq \mathcal{U}_{|\iota} = \langle c_u, [G, H], [I; J] \rangle_{\iota}$ is guaranteed for the s-zonotope in (9) with parameters

$$c^{(k+1)} = diag(\alpha^{(k)})(W^{(k)}c^{(k)} + b^{(k)}) + \beta^{(k)}, \tag{16a}$$

$$\tilde{H}^{(k+1)} = \left[diag(\alpha^{(k)}) W^{(k)} \tilde{H}^{(k)}, \quad diag(\gamma^{(k)}) \right], \quad (16b)$$

$$\tilde{G}^{(k+1)} = diag(\alpha^{(k)})W^{(k)}\tilde{G}^{(k)}, \quad k = 1, ..., l-1,$$
 (16c)

$$c_u = W^{(l)}c^{(l)} + b^{(l)}, (16d)$$

$$H = W^{(l)}\tilde{H}^{(l)},\tag{16e}$$

$$K = W^{(l)}\tilde{K}^{(l)},\tag{16f}$$

$$J = [!(n_1); ...; !(n_l)], \tag{16g}$$

where
$$\tilde{G}^{(1)} = diag(\beta^{(0)})W^{(0)}R^{(0)}$$
 and $\tilde{H}^{(1)} = diag(\gamma^{(0)})$.

Proof. Expressions (16a)-(16f) result from the recursive application of Lemma 1 and the vertical concatenation of the post-activation s-zonotopes (13) for the n_{k+1} neurons of the k-th layer. Besides, (16g) reflects the symbols identifier update of the noise terms introduced at the neurons of each laver.

Regarding the output inclusion, starting with an initial set $\mathcal{X}_{|_{L}}^{(0)}$, by induction, given the pre-activation s-zonotope $\mathcal{X}_{ls}^{(k)}$, the operations at the k-th layer are: affine mapping; linear abstraction (inclusion preserving for appropriate triplet $(\alpha_i^{(k)}, \beta_i^{(k)}, \gamma_i^{(k)}))$; and vertical concatenation. Thus, the composition of inclusion functions being an inclusion function, the proof follows.

For each neuron, the triplet of parameters (α, β, γ) must be appropriately designed to satisfy (14), while minimizing the conservatism induced by using an affine relaxation. In this regard, a relevant heuristic consists in minimizing the magnitude of the error symbol introduced to guarantee the activation function graph enclosure, i.e. to minimize $|\gamma|$. Due to the independent behaviour of the error symbol, this can be reformulated as minimizing the area of the enclosing parallelogram [31].

Lemma 2. Given the bounds [l, u] in (12) with l < u, the triplet of parameters $(\alpha^*, \beta^*, \gamma^*)$ that minimizes $|\gamma|$ while guaranteeing the satisfaction of (14) are:

• ReLU function $\varphi(x) = max(0, x)$

$$\alpha^* = \frac{\varphi(u) - \varphi(l)}{u - l}, \quad \beta^* = \gamma^* = \frac{\varphi(l) - \alpha^* \cdot l}{2}. \quad (17)$$

– Sigmoid
$$\varphi(x)=\frac{1}{1+e^{-x}}$$
 with $\varphi'(x)=\varphi(x)(1-\varphi(x))$ – $\tanh \varphi(x)=\tanh(x)$ with $\varphi'(x)=1-\varphi(x)^2$

-
$$tanh \varphi(x) = tanh(x)$$
 with $\varphi'(x) = 1 - \varphi(x)^2$

$$\alpha^* = \min(\varphi'(l), \varphi'(u)),$$

$$\beta^* = \frac{\varphi(u) + \varphi(l) - \alpha^* \cdot (u+l)}{2},$$

$$\gamma^* = \frac{\varphi(u) - \varphi(l) - \alpha^* \cdot (u-l)}{2}.$$
(18)

Remark 2. The proposed NN abstraction method shares a similar structure with the zonotope abstraction based on affine arithmetic presented in [31] for the (open-loop) NN output bounding. However, here, the explicitly computed affine symbolic expression (9) will further play a key role in closedloop verification, and an efficient computation of the projection matrices exploiting the generators (matrix) structure of szonotopes, is also used.

C. Dynamical system affine abstraction

Similar to the NN controller dynamics (5), the function (4) that describes the state evolution at time (t + 1) can be abstracted by means of an (inclusion preserving) affine mapping. The resulting s-zonotope will depend on the symbols that define the state at time t, plus some extra symbols that account for: I) NN controller non-linearities; II) abstract system non-linearities; III) the uncertainty sources.

For the computation of a state bounding s-zonotope, it is assumed that the function $f(\cdot)$ in (4) results from the composition of elementary functions and operators for which an affine symbolic expression (s-zonotope) can be computed. Note that this is not much restrictive since (linear) operations such as linear image, sum or vertical concatenation are closed (i.e. they return s-zonotopes) under affine mappings. Besides, any univariate locally continuous differentiable function can be abstracted through an affine mapping.

Lemma 3. Let $h:[l, u] \to \mathbb{R}$ be a class C^1 function on a given interval $[l, u] \subset \mathbb{R}$. Then, the function $\tilde{h}(x, \epsilon) = \alpha x + \beta + \gamma \epsilon$ satisfies that $\forall x \in [l, u], \exists \epsilon \in [-1, +1], h(x) = \tilde{h}(x, \epsilon)$ for the triplet of parameters:

$$\alpha = \frac{h(u) - h(l)}{u - l}, \quad \beta = \frac{h(\underline{x}) + h(\bar{x}) - \alpha(\underline{x} + \bar{x})}{2},$$
$$\gamma = \frac{h(\bar{x}) - h(\underline{x}) + \alpha(\underline{x} - \bar{x})}{2},$$

where, defining $\xi(x) = h(x) - \alpha x$, then

$$\bar{x} = \underset{x \in \{\delta_1, \dots, \delta_r, u\}}{\arg \max} \xi(x), \quad \underline{x} = \underset{x \in \{\delta_1, \dots, \delta_r, u\}}{\arg \min} \xi(x),$$

with $\delta_1, ..., \delta_n$ the stationary-points of $\xi(\cdot)$ in [l, u].

Lemma 3 provides a method to propagate (inclusion preserving) s-zonotopes through univariate non-linearities. Besides, the approach in Lemma 3 returns an optimal, in the sense of minimizing the magnitude $|\gamma|$ of the error symbol, set of parameters for convex/concave differentiable functions [28]. On the other hand, the interaction between multiple variables can be handled through the sum operation (2) or by overapproximating the product of two s-zonotopes.

Lemma 4. Given two 1-D s-zonotopes $\mathcal{X}_{|s} = \langle c_x, r^T, K \rangle_{|s}$ and $\mathcal{Y}_{|s} = \langle c_y, g^T, K \rangle_{|s}$ with a common set of symbols s_K (with $n = card(s_K)$), then the product $\mathcal{X}_{|s} \times \mathcal{Y}_{|s}$ is included by the s-zonotope $\mathcal{L}_{|s} = \langle c_l, [l^T, m], [K; j] \rangle_{|s}$ with

$$c_{l} = c_{x}c_{y} + \frac{1}{2}\sum_{i=1}^{n} r_{i}g_{i}, \qquad l = c_{x}g + c_{y}r,$$

$$m = \frac{1}{2}\sum_{i=1}^{n} |r_{i}g_{i}| + \sum_{i=1}^{n} \sum_{l>i}^{n} |r_{i}g_{l} + r_{l}g_{i}|,$$
(19)

and j = !(1).

Example 1. Consider the system $x^+ = \sin(x) - u + 0.1w$, with an initial set $\mathcal{X}_0 = [0, 1]$ described by $\mathcal{X}_{|s,0} = 0.5 + 0.5s_1$. This system is controlled by a NN with I layer of 2 neurons that, for \mathcal{X}_0 , is abstracted as $\mathcal{U}_{|s} = 0.1 + 0.2s_1 - 0.1s_2 + 0s_3$. The non-linear function $h(x) = \sin(x)$ is abstracted, for \mathcal{X}_0 , as $\hat{\mathcal{X}}_{|s} = 0.45 + 0.42s_1 + 0.03s_4$. Besides, the independent behaviour of the disturbances is captured by $\mathcal{W}_{|s} = s_5$. Accordingly, a dependency preserving over-approximation of the successor state is given by $\mathcal{X}_{|s}^+ = 0.35 + 0.22s_1 + 0.1s_2 + 0.03s_4 + 0.1s_5$. The a priori knowledge on the number of error symbols introduced at each abstraction (e.g. $\mathcal{U}_{|s}$ introduces up to two error symbols, one per neuron) allows to directly store the generators matrix of each s-zonotope by taking into account the common set of symbols, thus providing an efficient computation of required operations as shown below:

Algorithm 1 Finite-time RA verification

15: isRAok = $(t_{err} == \infty)$

```
Inputs: \mathcal{X}_0, NN param (\boldsymbol{W}, \boldsymbol{b}), \mathcal{G}, \mathcal{A}(i), N, q
 Outputs: isRAok, t_{err}, \mathcal{X}_{|s}(j) j = 0, ..., min(t_{err}, N).
 1: Initialize: Generate \mathcal{X}_{ls}(0); set t_{err} \leftarrow \infty
 2: for i = 0 to N - 1 do
            if \mathcal{X}_{|_L}(i) \cap \mathcal{A}(i) then
 3:
                    t_{err} \leftarrow i
 4:
                    break all
 5:
             else
 6:
 7:
                   \mathcal{U}_{|s}(i) \leftarrow controller(\mathcal{X}_{|s}(i), \boldsymbol{W}, \boldsymbol{b})
                    \vec{\mathcal{X}}_{|s}(i+1) \leftarrow system(f(\cdot), \mathcal{X}_{|s}(i), \mathcal{U}_{|s}(i), \mathcal{W}_{|s}(i))
 8:
                    \mathcal{X}_{|s}(i+1) \leftarrow \downarrow_q \bar{\mathcal{X}}_{|s}(i+1)
 9:
                    if (i == N-1) \wedge (\mathcal{X}_{|s}(N) \not\subseteq \mathcal{G}) then
10:
11:
                           t_{err} \leftarrow i + 1
                    end if
12:
             end if
13:
14: end for
```

D. Closed-loop integration

Algorithm 1 describes the main steps for the closed-loop finite-time reach-avoid verification problem under an s-zonotope formulation. Steps 7 and 8 represent the local abstraction of the NN controller and dynamical system through an affine symbolic expression as described in Section IV-B and Section IV-C, respectively. Note that, in Step 8, the uncertain behaviour of disturbances is adequately modeled by generating a set of independent symbols at each call $\mathcal{W}_{|s}(i) = s_{I_w}$, where $I_w = !(n_w)$. Besides, Step 9 includes the reduction operator introduced in Definition 5. At this step, the less relevant symbols, that is, those that have the least significant impact at the current time instant s-zonotope, are truncated consistently with the tuning of q providing control on the trade-off between computational complexity and accuracy while preserving inclusion.

Remark 3. Symbolic approaches also allow to efficiently handle shorter discretization periods (ΔT) in the (discretetime) system model (4) than the controller update period (Δh), since the dependencies between control inputs repeated at different time steps are preserved. Therefore, the discretization period (ΔT), and thus the discretization error, can be made smaller (up to reduction) at the cost of increasing the number of iterations N for which the system should be evaluated to meet a specified time horizon of $N \cdot \Delta T$. As an example: consider $\Delta T = \Delta h/2$ and the first two iterations of a system with held input over Δh given by x(1) = -x(0) + u(0) and x(2) = -x(1) + u(0), with $x(0) \in (0,1,1)_{\iota}$ and $u(0) \in (0,1,2)_{\iota}$. A classical set-valued evaluation yields $\mathcal{X}_{|\iota}(1) = -\langle 0, 1, 1 \rangle_{\iota} + \langle 0, 1, 2 \rangle_{\iota} = [-2, 2]$ and $\mathcal{X}_{|\iota}(2) = -\mathcal{X}_{|\iota}(1) + \langle 0, 1, 2 \rangle_{\iota} = [-3, 3]$, whereas operating at symbolic level gives $\mathcal{X}_{|s}(1) = \langle 0, [-1, 1], [1, 2] \rangle_{|s|}$ and $\mathcal{X}_{|s}(2) = -\mathcal{X}_{|s}(1) + \langle 0, 1, 2 \rangle_{|s} = \langle 0, 1, 1 \rangle_{|s}$ yielding $\mathcal{X}_{|\iota}(2) = [-1, 1].$

V. POLYNOMIAL SYMBOLIC EXPRESSIONS

The methodology presented in Section IV based on szonotopes can be readily extended to use any other wellformed symbolic expression as long as its set-valued interpretation guarantees the inclusion of the controller-system output sets. In particular, this section investigates the use of symbolic polynotopes (s-polynotopes) as in [28] to compute sound approximations relying on polynomial rather than affine dependencies. To that end, s-polynotopes are briefly recalled in Section V-A. Then, Section V-B investigates the abstraction of the I/O mapping of a NN through an inclusion preserving polynomial symbolic function computed in a propagationbased fashion, that is, in a compositional way possibly benefiting from a systematic use of basic operator overloading for the sake of simple and generic implementations. Besides, Section V-C discusses the main aspects to address a finite-time RA verification problem using a s-polynotope formulation.

A. Symbolic polynotopes

Symbolic polynotopes enable a tractable computational representation of polynomial symbolic functions by encoding all

their relevant information (generators, symbol identifiers and order of the monomials) into matrices.

Definition 6 (s-polynotope [28]). A symbolic polynotope $\mathcal{P}_{|s}$ is a polynomial function that can be written in the form $c + Rs_I^E$ where vector c and matrices R and E do not depend on the symbolic variables in s_I . Notation: $\mathcal{P}_{|s} = \langle c, R, I, E \rangle_{|s} = c + Rs_I^E$.

Definition 6 uses the exponential matrix notation (as in Definitions 23-25 in [28]), where the usually sparse matrix E accounts for exponents of the symbols involved in each monomial. As an example, $s_I = [s_1, s_2]^T$ and $E = [1\,0\,3; 0\,2\,4]$, yields $s_I^E = [s_1, s_2^2, s_1^3 s_2^4]^T$. Similar to s-zonotopes, a distinction is made between an s-polynotope as defined in Definition 6 and its set-valued interpretation defined as the (possibly non-convex) set $\mathcal{P}_{|\iota} = \{c + R\sigma^E \mid \sigma \in \mathcal{D}(s_I)\}$.

Symbolic polynotopes obviously extend s-zonotopes (obtained from $E=\mathcal{I}$, i.e. with identity as exponent matrix) and are closed under the extension of basic operations already defined for s-zonotopes like linear image, sum or concatenation. The reader is referred to [28] for further details on how to define and operate on s-polynotopes.

B. NN controller polynomial abstraction

The abstraction of the I/O map of a NN controller of the form (5) using s-polynotopes is presented below. In particular, given a state bounding s-polynotope $\mathcal{X}_{|s} = \langle c, R, I, E \rangle_{|s}$ (note that any initial s-zonotope in Section IV-A can be directly transformed into an equivalent s-polynotope) the idea is to compute a polynomial symbolic map of the form

$$\mathcal{U}_{|s} = \langle c_u, G, Q, E_u \rangle = c_u + G s_O^{E_u}, \tag{20}$$

such that, the enclosure of the network outputs is guaranteed, i.e. $g(\mathcal{X}_{|\iota}) \subseteq \mathcal{U}_{|\iota}$. The vector of identifiers in (20) has the structure Q = [I, J], thus involving the symbols in the state bounding s-polynotope (identified by I) as well as error symbols (identified by J). Notice that the exponent matrix E_u may also capture cross terms involving symbols with identifiers in both I and J.

Similar to section IV-B, the computation of (20) can be obtained from a forward propagation of $\mathcal{X}_{|s}$ through (in this case) a polynomial relaxation of the activation functions. The pre-activation s-polynotope $\mathcal{Z}_{|s} = W\mathcal{X}_{|s} + b$ and its projection onto the i-th neuron are given by

$$\mathcal{Z}_{|s} = \langle \check{c}, \check{R}, I, E \rangle_{|s}, \qquad \mathcal{Z}_{i|s} = \langle \check{c}_i, \check{R}_{[i,:]}, I, E \rangle_{|s},
\check{c} = Wc + b, \qquad \check{R} = WR.$$
(21)

Bounding the set-valued interpretation of $\mathcal{Z}_{i|s}$ within the interval $[l_i, u_i]$, then the polynomial structure of s-polynotopes enables to obtain a sound over-approximation of the NN output by locally covering the activation function graph through an n-order polynomial expression of the form

$$\mathcal{X}_{i|s}^{+} = \sum_{m=1}^{n} \alpha_{i,m} (\mathcal{Z}_{i|s})^{m} + \beta_{i} + \gamma_{i} s_{j},$$
 (22)

where s_j represents a new independent symbol (identified through j = !(1)) introduced to guarantee the enclosure of

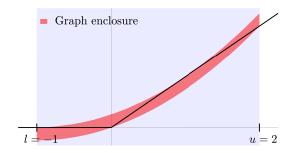


Fig. 1. 2-nd order polynomial enclosure of ReLu function (Example 2)

the activation function graph in the range $[l_i, u_i]$. Therefore, since $\mathcal{X}^+_{i|s}$ results in an s-polynotope arising from the polynomial mapping of s-polynotopes, the layer post-activation s-polynotope $\mathcal{X}^+_{|s|}$ is computed by vertically concatenating the neuron post-activation s-zonotopes.

Polynomial over linear abstractions of the activation function not only allow to reduce the conservatism introduced by the error symbols, but also enable the to compute input-output symbolic relationships that better fit the activation behaviour.

Example 2. Suppose that the projection onto a ReLU neuron is given by the s-polynotope $\mathcal{Z}_{|s}=0.5-0.5s_1+s_1s_2$, whose set-valued interpretation is bounded/included in the interval [l,u]=[-1,2]. Then, the ReLU function can be locally abstracted over this range using an n=2-order polynomial of the form $\mathcal{X}_{|s}^+=\alpha_2(\mathcal{Z}_{|s})^2+\alpha_1\mathcal{Z}_{|s}+\beta+\gamma s_3$, with $(\alpha_2, \alpha_1)=(0.25, 0.5), \ \beta=\gamma=0.125$. This, in turn, generates the post-activation s-polynotope

$$\mathcal{X}_{|s}^{+} = 0.25(0.5 - 0.5s_1 + s_1s_2)^2 + 0.5(0.5 - 0.5s_1 + s_1s_2) + 0.125 + 0.125s_3 = 0.4375 - 0.375s_1 + 0.0625s_1^2 + 0.125s_3 + 0.75s_1s_2 - 0.25s_1^2s_2 + 0.25s_1^2s_2^2.$$

Figure 1 depicts the non-convex local enclosure generated by the set-valued interpretation of $[\mathcal{Z}_{|s}; \mathcal{X}_{|s}^+]$.

Example 2 evidences the complexity/accuracy trade-off inherent to using n-order polynomial abstractions: a high n grants an accurate representation of the activation functions; whereas, on the other hand, it increases the computational complexity due to the increased number of monomials. A reduction strategy is thus used to manage the representation complexity. It can consist in either truncating the maximum degree of the polynomial approximation (20), or limiting the maximum number of monomials involved. To address this latter issue, an approach consists in (independently) assessing the monomials relevance based on the 2-norm of the generators (matrix columns) [32], and use natural interval extension [33] to bound the list of selected monomials through a reduced number of independent symbols.

The ability of the final s-polynotope (20) to generate a sound over-approximation of the network outputs is guaranteed by selecting (for each neuron) a triplet (α, β, γ) , where

 $\alpha = [\alpha_n,...,\alpha_1]^T$ is an n-dimensional vector, that ensures the (local) coverage of the activation function. In the case of the commonly used ReLU activation functions, as shown in Example 2 their convex nature allows describe them more accurately than with the sole affine dependencies by using 2-nd order polynomial expressions. The reduction in the magnitude related to the error symbol is especially significant in those situations where an affine approximation yields a rough description of ReLU function, i.e. for $|l| \approx u$ (if l < 0 < u).

Proposition 2. Given the interval [l, u] with l < 0 < u and a ReLU activation function $\varphi(x) = max(x, 0)$. The set of parameters

$$\begin{array}{ll} \alpha_2=\frac{1}{2u}, \ \alpha_1=1-\alpha_2 u, \ \beta=\gamma=\frac{\alpha_2 u^2}{8} & \mbox{if } |l|\leq u\leq 2|l| \\ \alpha_2=\frac{-1}{2l}, \ \alpha_1=-\alpha_2 l, \ \beta=\gamma=\frac{\alpha_2 l^2}{8} & \mbox{if } u<|l|\leq 2u \\ \mbox{guarantees that } \eta(x,\epsilon)=\alpha_2 x^2+\alpha_1 x+\beta+\gamma \epsilon \mbox{ satisfies that } \\ \forall x\in [l,u], \exists \epsilon[-1,1], \ \varphi(x)=\eta(x,\epsilon) \mbox{ with } |\gamma|\leq \frac{3}{8}|\gamma_{aff}^*|, \\ \mbox{where } \gamma_{aff}^* \mbox{ is the error symbol introduced by the affine } \\ \mbox{abstraction in Lemma 2}. \end{array}$$

Proof. See Appendix B.
$$\Box$$

C. Finite-time RA using s-polynotopes

The main aspects in addressing the RA verification problem using s-polynotopes are discussed below. In general, the same steps presented in Algorithm 1 can be used while adapting the NN controller and dynamical system abstraction to an s-polynotope formulation. In this case, the computation of the s-polynotope $\mathcal{U}_{|s|}$ in Step 7 of Algorithm 1 has already been presented in section V-B. Regarding the abstraction of the nonlinear function $f(\cdot)$ in Step 8, since s-polynotopes constitute an extension of s-zonotopes, $f(\cdot)$ can always be abstracted using (at least) the affine dependency preserving method in Lemma 3. Note that, s-polynotopes also enable the description of (multivariate) polynomial equations without the need to over-approximate them, at least in all intermediate symbolic compositions and up to some tunable computation load.

It must be taken into account that several operations in a s-polynotope formulation of Algorithm 1 such as bounding the projection of an s-polynotope onto a neuron, intersection/inclusion of an s-polynotope with an avoid/reach set or the reduction operator (in Step 9), in turn require the computation of interval bounds from (multivariate) interval polynomial expressions. If computationally affordable, the range bounds computed by a (simple and fast) interval extension may be refined either by iteratively bisecting the variables domain, or resorting to numerically reliable optimization-based methods.

VI. INPUT PARTITIONING STRATEGY

In general, the conservatism induced by abstraction-based verification tools strongly depends on the size of the initial set. It is thus extremely useful to assess the regions of the initial/input space for which meaningful (i.e. not too coarse) over-approximations of the closed-loop system evolution can be obtained. To that end, this section presents an algorithm to split the initial set of a NNCSs verification problem in a smartly guided way. More precisely, the proposed splitting

strategy relies on and benefits from the dependency modeling and tracing used in Section IV. In particular, the algorithm assesses the sensitiveness of the initial/input directions on the satisfaction of a safety property by an s-zonotopic overapproximation through the analysis of the relative influence of the initial symbols. The principle of the algorithm is introduced in Section VI-A. Then, some relevant notions are detailed in section VI-B, whereas the algorithm pseudo-code for a RA problem implementation is reported in Section VI-C. Finally, some further discussion on different settings is presented.

A. Splitting principle

The main idea of the proposed algorithm is to keep a linear increase in the number of subsets by only splitting at each iteration the sole initial/input set symbol that has greater influence on the satisfaction of the safety property $\mathcal S$ to be verified. To that end, notice that given any initial s-zonotope $\mathcal X_{|s|}(0) = \langle c_0, R_0, I \rangle_{|s|}$ as defined in Section IV-A, then the successive computation of forward reachable sets returns overapproximating s-zonotopes structured as

$$\mathcal{X}_{|s}(t) = c_f + R_f s_I + G_f s_J, \tag{23}$$

where the matrix R_f (resp. G_f) reflects the impact of the initial (resp. error) symbols identified by I (resp. J) on the computed over-approximation at time t. Typically, testing $\mathcal{S}(\mathcal{X}_{|s}(t))$ boils down to a metric/size evaluation on the set-valued interpretation of $\mathcal{X}_{|s}(t)$ (e.g. to check threshold trespassing). Hence, due to the linearity of (23), the influence of each input symbol s_i ($i \in I$) can be assessed using a metric that gauges the generator (column of R_f) size that is related to s_i , whereas the accuracy of an s-zonotope approximation to evaluate \mathcal{S} can be determined by measuring the zonotope $\langle 0, G_f \rangle$ spanned by the error symbols.

Therefore, at each iteration of the algorithm, an input szonotope $\mathcal{X}_{|s}(0)$, such that the corresponding output/final szonotope does not satisfy \mathcal{S} , is split into two new input szonotopes that are later evaluated on the satisfaction of \mathcal{S} . The algorithm may run until the satisfaction of the safety property, or until the accuracy of the method (gauged through $\langle 0, G_f \rangle$) is below a certain threshold.

B. Relevant notions

Some relevant notions for the s-zonotope based partitioning algorithm are discussed below.

- 1) Accuracy assessment: considering the evaluation of a safety property for a s-zonotope of the form (23), the accuracy of the over approximation can be assessed by gauging the zonotope $\langle 0, G_f \rangle$ spanned by a (set-valued) interpretation of the error symbols. In particular, further implementations make use of the zonotope F-radius [34], that is,the Frobenius norm of the generators matrix $\|G_f\|_F$ to reasoning upon the quality of the affine approximation.
- 2) Input symbols relative influence: the sensitivity of an input symbol s_i $(i \in I)$ is computed based on the F-radius ratios of the I/O zonotopes spanned by ιs_i . That is, through the ratio $\|R_f^{[i]}\|_2/\|R_0^{[i]}\|_2$ where $R_0^{[i]}$ (and $R_f^{[i]}$) denote the columns

of R_0 (and R_f) that multiply the symbol s_i . This relation is used to quantify how a variation on s_i at the input s-zonotope $\mathcal{X}_{|s}(0)$ affects the output s-zonotope.

- 3) Symbol bisection: bisecting a unit interval symbol s_i $(i \in I)$ is done by rewriting it as $s_i \to 0.5 + 0.5s_j$ and $s_1 \to -0.5 + 0.5s_k$, where j = !(1) and k = !(1), thus generating two new s-zonotopes.
- 4) Polyhedral RA sets: checking the empty intersection and/or the inclusion of a state bounding set with/within a polyhedron in half-space representation of the type $\{h_i^T x \leq r_i, i = 1, ..., m\}$ can be done by evaluating the infimum/supremum of the projections of the bounding set onto the directions $h_i \in \mathbb{R}^{n_x}$ [35]. For a state bounding s-zonotope of the form (23), the supremum of the dot product with h is computed as

$$\sup_{x \in \mathcal{X}_{l_t}(t)} h^T x = h^T c_f + ||h^T R_f||_1 + ||h^T G_f||_1.$$
 (24)

C. Algorithm implementation for finite-time RA

Algorithm 2 reflects the pseudo-code of the proposed partitioning strategy to check the satisfaction of a RA problem over a time horizon N. Algorithm 2 uses square brackets to label the different s-zonotopes that arise after splitting. As an example, $\mathcal{X}_{|s}[0]$ reads as the initial s-zonotope (i.e. $\mathcal{X}_{|s}[0] = \mathcal{X}_{|s}^0$), which is split onto a second $\mathcal{X}_{|s}[1]$ and a third $\mathcal{X}_{|s}[2]$ s-zonotopes (with $\mathcal{X}_{|\iota}[1] \cup \mathcal{X}_{|\iota}[2] = \mathcal{X}_{|\iota}[0]$). Besides, L denotes a set of integer labels/indices (for the above example $L = \{0, 1, 2\}$), and $\mathcal{X}_{|\iota}[L]$ is a shorthand for the set of s-zonotopes $\{\mathcal{X}_{|s}[l] \mid l \in L\}$.

At each iteration, the routine reach runs a slightly modified version of Algorithm 1, that, in this case, returns the last time instant (and the corresponding s-zonotope) for which the RA problem is not satisfied. These times-to-last-error are managed by vector T. The algorithm iteratively selects the label of the initial s-zonotope that yields the largest time-tolast-error (Step 14). The use of this backward management of the information that prioritizes to split until the RA constraints are satisfied at time t, then at time t-1, etc., will be further discussed in the next paragraph. Once the l-th (with $l \in L$) szonotope has been selected, sym-select returns the initial symbol identifier $i \in I$ that has greater relative influence over the violated property. The symbol s_i of the l-th set is split by the routine sym-split that returns two new initial subsets (Step 6). The times-to-last-error for the new s-zonotopes are computed and the set L and vector T updated (Steps 7-11). In particular, Algorithm 2 runs either until the RA problem is satisfied for the whole set L, or until a maximum number n_{max} of splits is reached.

Algorithm 2 manages the information in a backward fashion, that is, it selects a s-zonotope with the higher time-to-last-error. This usually gives better results than working in a forward fashion (that is, selecting the s-zonotope with lower time-to-first-error) since it avoids to get stuck by exhaustively splitting up to the satisfaction of a constraint at time t, which, then, may have a small impact on the constraint satisfaction at t+1. On this subject, the algorithm can be straightforwardly adapted to handle the forward case by directly using Algorithm 1 (instead of reach), using $T(l) \leftarrow N+1$ (instead of

Algorithm 2 Input partitioning for finite-time RA

```
Input: same as Algorithm 1, n_{max}
Output: isRAok, set of s-zonotopes \mathcal{X}_{ls}[L]
 1: Initialize: l = n = 0; L = \{l\}; \mathcal{X}_{|s}[0] = \mathcal{X}_0
 2: (t, \mathcal{X}_{|s}(t)[0]) \leftarrow \operatorname{reach}(\mathcal{X}_{|s}[0], N)
 3: T \leftarrow append(t)
 4: while (\max(T) > 0) \lor (n/2 == n_{max}) do
          i \leftarrow \text{sym-select}(\mathcal{X}_{|s}[l], \mathcal{X}_{|s}(T(l))[l])
          (\mathcal{X}_{|s}[n+1], \mathcal{X}_{|s}[n+2]) \leftarrow \text{sym-split}(\mathcal{X}_{|s}[l], i)
 6:
          for j = 1 to 2 do
 7:
                (t,\mathcal{X}_{|s}(t)[n+j]) \leftarrow \operatorname{reach}(\mathcal{X}_{|s}[n+j], \max(T))
 8:
                L \leftarrow L \cup \{n+j\}
 9:
                T \leftarrow append(t)
10:
          end for
11:
          L \leftarrow L \setminus \{l\}
12:
          T(l) \leftarrow 0
13:
          l \leftarrow \arg\max(T(L))
14:
          n \leftarrow n + 2
15:
16: end while
17: isRAok = (max(T) == 0)
```

 $T(l) \leftarrow 0$) in Step 13 and selecting the minimum (instead of the maximum) of vector T. Besides, the reduction operation used in Algorithm 1 must not truncate the initial symbols even if their relevance decreases with time, so that the input-output mapping of the symbols identified by I is preserved.

D. Other possible settings and applications

Other choices for the proposed input partitioning strategy are as follows:

- The strategy in Algorithm 2 can be adapted to handle open-loop verification problems (e.g. elements like the NN in isolation). In this case, the reach routine will only compute the output s-zonotope for the isolated element for a number of forward steps N=1.
- The maximum number of splits stopping criterion in Algorithm 2 can be modified/complemented with a tolerance on the accuracy assessment (see Section VI-B). In other words, if the accuracy tolerance is fulfilled and a property is still violated, then the algorithm should stop to prevent from further splitting and the safety property is considered as unsatisfied up to the accuracy tolerance.
- Another interesting application is to modify the szonotope split decision rule (Step 14 of Algorithm 2) to focus the split in those regions for which the accuracy of using an affine abstraction is low (i.e. high $||G_f||_F$). This tends to return a set of initial szonotopes such that each locally provides an accurate (affine) abstraction of the system behavior.

VII. SIMULATIONS

A. Benchmark description

The numerical simulations consist in the discrete-time version of some of the verification problems proposed in the ARCH-COMP 2021 [4]. Five dynamical systems are assessed,

namely: single pendulum (S), TORA (T), unicycle car (C), adaptive cruise control (ACC) and double pendulum (D). The above systems have been discretized using the forward Euler method with sampling period ΔT , and they are controlled by a NN controller with control period Δh . The NN controllers are the ones provided in [4] to control the continuous-time version of the models. To address this issue, the dynamical models have been analyzed under sampling times $\Delta T (< \Delta h)$ chosen sufficiently small for the discretization to have negligible impact in the model responses. Under this context, the same safety constraints and initial conditions than the ones proposed in [4] have been re-used to setup the reported simulations. Note that, as discussed in Remark 3, the use of symbolic approaches supports the variation of ΔT (for some Δh) without inducing conservatism due to a loss of dependencies between repeated control inputs. A detailed description of the systems dynamics can be found in [4], whereas the main parameters that define each safety verification problem are shown in Table I.

Problem	Dynamics	NN size	H	Δh	ΔT
S1	Single pendulum	(2,25,25,1)	1	.05	.05
S2	Single pendulum	(2,25,25,1)	1	.05	.001
T1	TORA	(4,100,100,100,1)	20	1	1
T2	TORA	(4,100,100,100,1)	20	1	.01
Т3	TORA	(4,100,100,100,1)	20	1	.001
C1	Unicycle car	(4,500,2)	10	.2	.2
C2	Unicycle car	(4,500,2)	10	.2	.001
ACC1	Cruise control	(5,20,20,20,20,20,1)	5	.1	.1
ACC2	Cruise control	(5,20,20,20,20,20,1)	5	.1	.001
D1	Double pendulum	(4,25,25,2)	1	.05	.05
D2	Double pendulum	(4,25,25,2)	1	.05	.001
D3	Double pendulum	(4,25,25,2)	1	.05	.05

TABLE I BENCHMARKS PARAMETERS

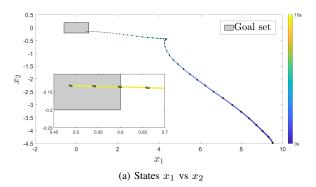
B. Benchmark results using s-zonotopes

All the results reported below were obtained on a standard laptop with Intel Core i7-8550U@1.8GHz \times 4 processor and 16GB RAM running Windows 10. Table II shows the set of initial states, the safety constraints (with their time horizon), as well as the time required by an s-zonotope implementation to verify each problem. The reduction order is q=200 in all the experiments. Some particularities are discussed below:

- Single pendulum (S): in a discrete-time setting, the constraint $x_1 \in [0, 1]$ is guaranteed to be satisfied for problem S1 (with $\Delta T = 0.05$ s) for the time interval $t \in [0.55, 1]$ (that is, for samples $\{11, ..., 20\}$), whereas in S2 (with $\Delta T = 0.001$ s) the constraint satisfaction is guaranteed for the time interval $t \in [0.516, 1]$.
- TORA (T): in T1, the closed-loop system is not stable for the discrete-time model obtained for $\Delta T=1$ s. In this case, an unambiguous constraint violation is achieved at t=3 in 0.036s. On the other hand, the closed-loop model obtained in T2 and T3 is stable, and the s-zonotope method verifies the satisfaction of the safety constraint in both problems without resorting to split the input set.
- Unicycle car (C): the model under study considers the addition of an unknown-but-bounded disturbance $w \in 10^{-4}[-1, +1]$ affecting the fourth state. The safety properties are verified for both C1 and C2. In particular,

- Figure 2 shows the envelope computed for **C2** in the time interval $t \in [0, 10]$ and how the outer-approximation lies within the goal set at t = 10.
- Adaptative cruise control (ACC): both problems ACC1 and ACC2 are verified for the given time horizon.
- Double pendulum (**D**): the set of constraints in problems **D1-2** are violated by the closed-loop system. An unambiguous constraint violation is achieved for **D1** at t = 0.25 and for **D2** at t = 0.278. On the other hand, the problem **D3** cannot be verified from a simple affine abstraction: the accumulated error indeed increases in the reachability analysis of **D3**, not allowing to guarantee the constraints satisfaction or their unambiguous violation, and thus motivating further extensions.

The results presented above show how, despite their low computational complexity, s-zonotopes yield a high performance in NNCSs verification, being able to verify almost all the benchmark problems without splitting the input set. It is also remarkable the scalability of this approach. As an example, for problem **T3** with $\Delta T = 0.001$ s, $\Delta h = 1$ s and time horizon $t \in [0,20]$, the proposed tool only requires of 1.515s to compute and assess $N = 20 \text{s}/\Delta T = 20.000$ forward iterations.



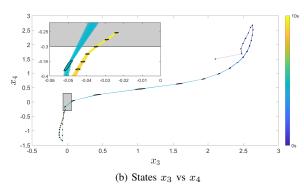


Fig. 2. Problem C2: framed zonotopes represent the computed bounds at each $\Delta h=.2s$; blurred lines represent the bounds update at $\Delta T=.001s$.

C. Use of s-polynotopes

The capability of s-polynotopes to capture the non-convex map of NNs is illustrated below. To that end, the set of randomly generated neural networks used in [17] are analyzed. All the NNs consists of 2 inputs, 2 outputs and they differ on the number of hidden layers and neurons per layer. The

Problem	Time [s]	Initial set	Safety	Sft. Horizon [s]
S1	0.071	$[1, 1.2] \times [0, 0.2]$	$x_1 \in [0, 1]$	$0.5 < t \le 1$
S2	0.111	$[1, 1.2] \times [0, 0.2]$	$x_1 \in [0, 1]$	$0.5 < t \le 1$
T1	0.036	$[0.6, 0.7] \times [-0.7, -0.6] \times [-0.4, -0.3] \times [0.5, 0.6]$	$x \in [-2, 2]^4$	$t \le 20$
T2	0.182	$[0.6, 0.7] \times [-0.7, -0.6] \times [-0.4, -0.3] \times [0.5, 0.6]$	$x \in [-2, 2]^4$	$t \le 20$
Т3	1.515	$[0.6, 0.7] \times [-0.7, -0.6] \times [-0.4, -0.3] \times [0.5, 0.6]$	$x \in [-2, 2]^4$	$t \le 20$
C1	0.37	$[9.5, 9.55] \times [-4.5, -4.45] \times [2.1, 2.11] \times [1.5, 1.51]$	$x \in [6, .6] \times [2, .2] \times [06, .06] \times [3, .3]$	t = 10
C2	21.40	$[9.5, 9.55] \times [-4.5, -4.45] \times [2.1, 2.11] \times [1.5, 1.51]$	$x \in [6, .6] \times [2, .2] \times [06, .06] \times [3, .3]$	t = 10
ACC1	0.095	$[90, 110] \times [32, 32.2] \times 0 \times [10, 11] \times [30, 30.2] \times 0$	$x_1 - x_4 \ge 10 + 1.4x_5$	$t \leq 5$
ACC2	0.439	$[90, 110] \times [32, 32.2] \times 0 \times [10, 11] \times [30, 30.2] \times 0$	$x_1 - x_4 \ge 10 + 1.4x_5$	$t \leq 5$
D1	0.251	$[1, 1.1] \times [1, 1.1] \times [1, 1.2] \times [1, 1.2]$	$x \in [-1, 1.7]^4$	$t \leq 1$
D2	2.113	$[1, 1.1] \times [1, 1.1] \times [1, 1.2] \times [1, 1.2]$	$x \in [-1, 1.7]^4$	$t \leq 1$
D3	Fail	$[1, 1.3] \times [1, 1.3] \times [1, 1.3] \times [1, 1.3]$	$(x_1, x_2) \in [-2, 2]^2, (x_3, x_4) \in [-1.7, 1.7]^2$	$t \leq 1$

TABLE II
BENCHMARK PROBLEMS AND RESULTS FOR AN S-ZONOTOPE IMPLEMENTATION

first four NNs present $l = \{1, 2, 3, 4\}$ hidden layers, each having $n_k = 100$ ReLU neurons per layer. The examined NN input set is $\mathcal{X}_0 = [0.9, 1.1] \times [0.9, 1.1]$. Figure 3 shows the set-valued interpretation of the output bounding s-polynotopes obtained by abstracting the activity functions of active neurons with second order polynomials i.e. with n = 2 in (22). The computation times are $\{0.178, 0.240, 2.021, 3.329\}s$ for the NNs with l = 1 to l = 4 hidden layers, respectively. Similarly, another set of NNs with $l = \{7, 8, 9, 10\}$ hidden layers and $n_k = 10$ ReLU neurons per layer is evaluated for the same input set. Figure 4 represents the interpretation of the resulting s-polynotopes that are computed in $\{0.2389, 0.108, 0.786, 0.155\}$ s, respectively. Those examples (Figure 3 and Figure 4) taken from [17] illustrate the ability of s-polynotopes composition to accurately generate inclusion preserving polynomial I/O mappings of NNs. As a byproduct, an efficient implicit description of possibly non convex output sets is obtained.

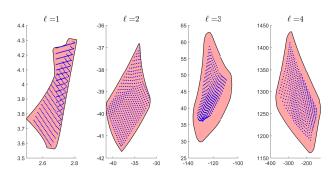


Fig. 3. NNs with 100 neurons per layer and $l = \{1, 2, 3, 4\}$ hidden layers. Set-valued interpretation of the over-approximating s-polynotope (red set); exhaustive evaluation of the NNs (blue dots).

D. Partitioning strategy

Firstly, in order to show the performance of the partitioning algorithm, it will be applied to the open-loop robotic arm example used in [12], [13]. Particularly, the non-linear dynamics of a 2 DOF robot arm are modeled by a (2,5,2) NN with tanh activations. The considered set of joint angles are extended

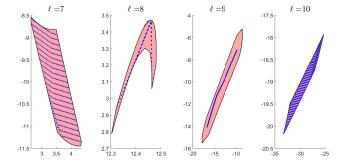


Fig. 4. NNs with 10 neurons per layer and $l = \{7, 8, 9, 10\}$ hidden layers. Set-valued interpretation of the over-approximating s-polynotope (red set); exhaustive evaluation of the NNs (blue dots).

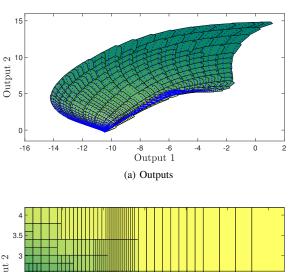
to $(\theta_1,\theta_2)\in [\frac{\pi}{3},\frac{4\pi}{3}]^2$. An implementation of Algorithm 2 adapted to analyze the NN in isolation is executed in order to iteratively minimize the F-radius of the zonotope spanned by the error symbols $(\|G_f\|_F)$ for a fixed number of $n_{max}=400$ splits. The computation time of the algorithm is 0.097s. Figure 5b shows the resulting pattern of 401 input subsets, whereas Figure 5a represents the corresponding s-zonotope interpretations obtained in the output space altogether with an exhaustive evaluation of the NN (blue dots). This latter figure shows how Algorithm 2 achieves an accurate description of the non-convex output set by focusing the splitting effort in those regions of the input space for which an affine abstraction granted by s-zonotopes is not accurate enough.

Furthermore, considering the initial set $(\theta_1, \theta_2) \in [\frac{\pi}{3}, \frac{2\pi}{3}]^2$, Algorithm 2 is set to split up to the satisfaction of the safety constraint $y_1 \leq d$ (where y_1 denotes the first output). Table III reflects the number of splits and the time required by Algorithm 2 to satisfy the above safety constraint for different values of d. Besides, Table III also shows, for a fixed number of splits, the number of existing possible combinations of set selections and symbols bisections, as well as how many among them are able to satisfy the property. As an example, for d=1.2, Algorithm 2 requires 8 splits. For the same

¹The *F*-radius of a zonotope is the Frobenius norm of its generator matrix (see Definition 3 in [34]).

d	Nb of splits (Algo. 2)	Time [s]	4	5	6	7	8	9	10	Nb of splits
			(224)	(1344)	(8448)	$(5.49 \cdot 10^4)$	$(3.66 \cdot 10^5)$	$(2.49 \cdot 10^6)$	$(1.71 \cdot 10^7)$	Possible comb.
1.75	5	.052	.446	.967	*	*	*	*	*	
1.5	6	.057	-	-	.213	*	*	*	*	% of comb.
1.3	7	.072	-	-	.095	.208	*	*	*	satisfying the
1.2	8	.058	-	-	-	.032	.092	*	*	safety prop.
1.1	10	.066	-	-	-	-	-	.002	.008	
		ı	'			TABLE III				

Partitioning Performance: Non existing solutions (-); worse 2 solutions (*).



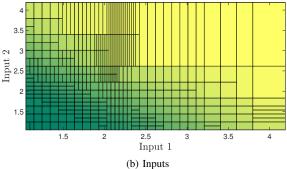


Fig. 5. Robot arm example: input partitioning for $(\theta_1, \theta_2) \in [\frac{\pi}{3}, \frac{4\pi}{3}]^2$; yellow-green colors characterize the corresponding I/O set pairs; exhaustive evaluation of the NN (blue dots).

problem, there are no possible combinations of less than 7 splits for which the property can be proven; there exist 18 out of $5.491 \cdot 10^4$ possibilities that satisfy it with 7 splits (0.032%); and 336 out of $3.66 \cdot 10^5$ possibilities that satisfy it with 8 splits (0.0918%).

Regarding the closed-loop examination of Algorithm 2, this is applied to assess the satisfaction of problem **D3**, which cannot be satisfied by a simple s-zonotope abstraction. To that end, Algorithm 2 is set to split up to the satisfaction of the safety constraints in Table II. The algorithm requires a total of 19 splits (i.e. 20 subsets) computed in 5.12s. Figure 6 shows the time evolution of the interval enclosure of the resulting 20 reachable sets (light blue background), altogether with 50 random simulations of the closed-loop system (blue dots).

VIII. CONCLUSIONS

A compositional approach focused on inclusion preserving long term symbolic dependency modeling is introduced in

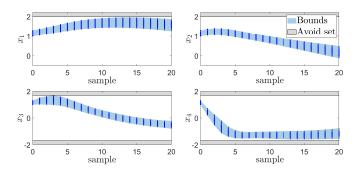


Fig. 6. Input partitioning problem **D3**: interval enclosure of the resulting 20 reachable sets (light blue); random simulations (blue dots).

this work for the analysis of NNCSs, where such long term is to be understood both in time iterations (regarding the controlled system dynamics) and in layer iterations (regarding the sole NNs). This results in a generic method that has been developped in several ways. Firstly, the matrix structure of s-zonotopes enables to compute (fast and simple) affine symbolic mappings to abstract the I/O mapping of the control loop components. Two further extensions are also proposed: the use of s-polynotopes to compute inclusion preserving polynomial mappings capable of accurately describing the non-convex map of NNs, and an input partitioning algorithm that benefits from the ability granted by s-zonotopes to preserve linear dependencies between the loop elements. Simulations show the comparative efficiency of the proposals and support the prevalence of dependency preserving methods for closed-loop analysis over the use of accurate, but dependency breaking, output bounding verification tools. Future works should address the integration with the analysis of continuous-time dynamical systems, as well as the study of optimized affine/polynomial abstractions for achieving better performance in verifying specific safety properties.

ACKNOWLEDGMENTS

The authors would like to thank Prof. Vicenç Puig (UPC, Barcelona, Spain) who initially prompted this collaboration and his support during the research stay of the first author at the University of Bordeaux. This work was in part supported by the Margarita Salas grant from the Spanish Ministry of Universities funded by the European Union NextGenerationEU.

²In the sense that require a higher number of splits to verify the property

APPENDIX A PROOF OF THE LEMMA 3

Define $\alpha = \frac{h(u) - h(l)}{u - l}$ and form $\xi(x) = h(x) - \alpha x$ which is continuous in [l, u], differentiable in (l, u) and satisfies $\xi(l) = \xi(u) = 0$. Then, the maximum $\xi(\bar{x})$ (resp. minimum $\xi(\underline{x})$) of $\xi(x)$ on [l, u] must be for \bar{x} (resp. \underline{x}) in the boundary points $\{u, l\}$ or in its stationary points denoted as $\{\delta_1, ..., \delta_n\}$, that is, the solutions of $\xi'(x) = 0 \to h'(x) = \alpha$.

Given $\xi(\bar{x})$ and $\xi(\underline{x})$, then for all $x \in [l, u]$ $\xi(\underline{x}) \leq \xi(x) \leq \xi(\bar{x}) \Longrightarrow \underline{y}(x) \leq h(x) \leq \bar{y}(x)$, with $\underline{y}(x) = \alpha(x-\underline{x}) + h(\underline{x})$ and $\bar{y}(x) = \alpha(x-\bar{x}) + h(\bar{x})$. Thus, it follows that $\forall x \in [l, u], \exists \epsilon \in [-1, +1]$ such that

$$h(x) = \frac{\bar{y}(x) + \underline{y}(x)}{2} + \frac{\bar{y}(x) - \underline{y}(x)}{2}\epsilon$$

or equivalently $h(x) = \tilde{h}(x, \epsilon) = \alpha x + \beta + \gamma \epsilon$ with

$$\beta = \frac{\bar{y}(x) + \underline{y}(x)}{2} - \alpha x = \frac{h(\underline{x}) + h(\bar{x}) - \alpha(\underline{x} + \bar{x})}{2},$$
$$\gamma = \frac{\bar{y}(x) - \underline{y}(x)}{2} = \frac{h(\bar{x}) - h(\underline{x}) + \alpha(\underline{x} - \bar{x})}{2}.$$

$\begin{array}{c} \text{Appendix B} \\ \text{Proof of the Proposition 2} \end{array}$

Inclusion preservation: the sign criterion $\gamma \geq 0$ is chosen below. Given $\bar{y}(x) = \alpha_2 x^2 + \alpha_1 x + \beta + \gamma$ and $\underline{y}(x) = \alpha_2 x^2 + \alpha_1 x + \beta - \gamma$, then parameters $(\alpha_2, \alpha_1, \beta, \gamma)$ ensure local coverage of $\varphi(x)$ if $y(x) \leq \varphi(x) \leq \bar{y}(x), \forall x \in [l, u]$.

Consider firstly the scenario $|l| \le u \le 2|l|$. In this case, $\alpha_2 = \frac{1}{2u} > 0$ and thus $y(x), \bar{y}(x)$ are strictly convex.

On the one hand, $\beta=\gamma$ and $\alpha_1=1-\alpha_2u$ impose that, $\underline{y}(0)=0=\varphi(0)$ and $\underline{y}(u)=u=\varphi(u)$, whereas $\underline{y}(l)=\frac{1}{2}(\frac{l^2}{u}+l)\leq 0=\varphi(l)$ for $u\geq |l|$ and l<0. Therefore, from $\underline{y}(l)\leq \varphi(l),\,\underline{y}(0)=\varphi(0),\,\underline{y}(u)=\varphi(u)$ and the convexity of $\underline{y}(x)$ wrt x, it follows that $\underline{y}(x)\leq \varphi(x), \forall x\in [l,\,u].$

On the other hand, from $\alpha_1=1-\alpha_2u$ and $\beta=\gamma=\frac{\alpha_2u^2}{8}$, then $\bar{y}(x)$ is tangent to the positive region of $\varphi(x)$ in $\hat{x}=\frac{u}{2}$ (that is, $\bar{y}(\hat{x})=\varphi(\hat{x})=\hat{x}$ and $\bar{y}'(\hat{x})=\varphi'(\hat{x})=1$), and thus, since $\bar{y}(x)$ is convex, it follows that $\bar{y}(x)\geq\varphi(x), \forall x\geq0$. Additionally, for $\alpha_2=\frac{1}{2u}$ the (global) minimum of $\bar{y}(x)$ is $\bar{y}(x^*)=0$ for $x^*=\frac{-u}{2}$ (that is, $\bar{y}'(x^*)=0$), and thus $\bar{y}(x)\geq\varphi(x), \forall x\leq0$.

A similar reasoning can be used to show the inclusion preservation for the scenario $u < |l| \le 2u$.

Conservatism reduction: For the scenario $|l| \leq u \leq 2|l|$, the parameter γ has the value $\gamma = \frac{\alpha_2 u^2}{8} = \frac{u}{16}$. On the other hand, for a ReLU function $\varphi(x) = max(0,x)$ the triplet for an affine abstraction in Lemma 2 yields $\gamma_{aff}^* = \frac{u|l|}{2(u+|l|)}$. Therefore, for $u \leq 2|l|$ the following inequality is obtained

$$\gamma_{aff}^* = \frac{u|l|}{2(u+|l|)} \ge \frac{u|l|}{2(2|l|+|l|)} = \frac{u}{6} > \frac{u}{16} = \gamma$$

and thus $\gamma \leq \frac{3}{8}\gamma_{aff}^* \sim |\gamma| \leq \frac{3}{8}|\gamma_{aff}^*|$ (since $\gamma, \gamma_{aff}^* > 0$). A similar reasoning can be used to prove the case $u < |l| \leq 2u$.

REFERENCES

- C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," arXiv preprint arXiv:1312.6199, 2013.
- [2] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *Artificial intelligence safety and security*. Chapman and Hall/CRC, 2018, pp. 99–112.
- [3] C. Liu, T. Arnon, C. Lazarus, C. Strong, C. Barrett, M. J. Kochenderfer et al., "Algorithms for verifying deep neural networks," Foundations and Trends® in Optimization, vol. 4, no. 3-4, pp. 244–404, 2021.
- [4] T. T. Johnson, D. M. Lopez, L. Benet, M. Forets, S. Guadalupe, C. Schilling, R. Ivanov, T. J. Carpenter, J. Weimer, and I. Lee, "Archcomp21 category report: Artificial intelligence and neural network control systems (ainnes) for continuous and hybrid systems plants." in ARCH@ ADHS, 2021, pp. 90–119.
- [5] G. Yang, G. Qian, P. Lv, and H. Li, "Efficient verification of control systems with neural network controllers," in *Proceedings of the 3rd International Conference on Vision, Image and Signal Processing*, 2019, pp. 1–7.
- [6] H.-D. Tran, X. Yang, D. Manzanas Lopez, P. Musau, L. V. Nguyen, W. Xiang, S. Bak, and T. T. Johnson, "Nnv: the neural network verification tool for deep neural networks and learning-enabled cyber-physical systems," in *International Conference on Computer Aided Verification*. Springer, 2020, pp. 3–17.
- [7] A. Clavière, E. Asselin, C. Garion, and C. Pagetti, "Safety verification of neural network controlled systems," in 2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W). IEEE, 2021, pp. 47–54.
- [8] C. Schilling, M. Forets, and S. Guadalupe, "Verification of neuralnetwork control systems by integrating taylor models and zonotopes," arXiv preprint arXiv:2112.09197, 2021.
- [9] R. Ivanov, T. J. Carpenter, J. Weimer, R. Alur, G. J. Pappas, and I. Lee, "Verifying the safety of autonomous systems with neural network controllers," ACM Transactions on Embedded Computing Systems (TECS), vol. 20, no. 1, pp. 1–26, 2020.
- [10] C. Sidrane, A. Maleki, A. Irfan, and M. J. Kochenderfer, "Overt: An algorithm for safety verification of neural network control policies for nonlinear systems," *Journal of Machine Learning Research*, vol. 23, no. 117, pp. 1–45, 2022.
- [11] M. Everett, G. Habibi, and J. P. How, "Efficient reachability analysis of closed-loop systems with neural network controllers," in 2021 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2021, pp. 4384–4390.
- [12] W. Xiang, H.-D. Tran, X. Yang, and T. T. Johnson, "Reachable set estimation for neural network control systems: A simulation-guided approach," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 5, pp. 1821–1830, 2020.
- [13] M. Everett, G. Habibi, and J. P. How, "Robustness analysis of neural networks via efficient partitioning with applications in control systems," *IEEE Control Systems Letters*, vol. 5, no. 6, pp. 2114–2119, 2020.
- [14] S. Dutta, X. Chen, and S. Sankaranarayanan, "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*, 2019, pp. 157–168.
- [15] C. Huang, J. Fan, W. Li, X. Chen, and Q. Zhu, "Reachnn: Reachability analysis of neural-network controlled systems," ACM Transactions on Embedded Computing Systems (TECS), vol. 18, no. 5s, pp. 1–22, 2019.
- [16] H. Hu, M. Fazlyab, M. Morari, and G. J. Pappas, "Reach-sdp: Reachability analysis of closed-loop systems with neural network controllers via semidefinite programming," in 2020 59th IEEE Conference on Decision and Control (CDC). IEEE, 2020, pp. 5929–5934.
- [17] M. Fazlyab, M. Morari, and G. J. Pappas, "Safety verification and robustness analysis of neural networks via quadratic constraints and semidefinite programming," *IEEE Transactions on Automatic Control*, 2020.
- [18] H. Zhang, T.-W. Weng, P.-Y. Chen, C.-J. Hsieh, and L. Daniel, "Efficient neural network robustness certification with general activation functions," *Advances in neural information processing systems*, vol. 31, 2018
- [19] V. Tjeng, K. Xiao, and R. Tedrake, "Evaluating robustness of neural networks with mixed integer programming," arXiv preprint arXiv:1711.07356, 2017.
- [20] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer, "Reluplex: An efficient smt solver for verifying deep neural networks," in *International conference on computer aided verification*. Springer, 2017, pp. 97–117.

- [21] S. Dutta, S. Jha, S. Sanakaranarayanan, and A. Tiwari, "Output range analysis for deep neural networks," arXiv preprint arXiv:1709.09130, 2017.
- [22] M. Althoff, "An introduction to cora 2015," in Proc. of the workshop on applied verification for continuous and hybrid systems, 2015, pp. 120–151.
- [23] H.-D. Tran, S. Bak, W. Xiang, and T. T. Johnson, "Verification of deep convolutional neural networks using imagestars," in *International* conference on computer aided verification. Springer, 2020, pp. 18–42.
- [24] G. Singh, T. Gehr, M. Püschel, and M. Vechev, "An abstract domain for certifying neural networks," *Proceedings of the ACM on Programming Languages*, vol. 3, no. POPL, pp. 1–30, 2019.
- [25] S. Wang, K. Pei, J. Whitehouse, J. Yang, and S. Jana, "Formal security analysis of neural networks using symbolic intervals," in 27th USENIX Security Symposium (USENIX Security 18), 2018, pp. 1599–1614.
- [26] W. Xiang, H.-D. Tran, and T. T. Johnson, "Output reachable set estimation and verification for multilayer neural networks," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 11, pp. 5777–5783, 2018.
- [27] S. Gowal, K. Dvijotham, R. Stanforth, R. Bunel, C. Qin, J. Uesato, R. Arandjelovic, T. Mann, and P. Kohli, "On the effectiveness of interval bound propagation for training verifiably robust models," arXiv preprint arXiv:1810.12715, 2018.
- [28] C. Combastel, "Functional sets with typed symbols: Mixed zonotopes and polynotopes for hybrid nonlinear reachability and filtering," *Automatica*, vol. 143, 110457, 2022.
- [29] R. E. Moore, R. B. Kearfott, and M. J. Cloud, *Introduction to interval analysis*. SIAM, 2009.
- [30] C. Combastel and A. Zolghadri, "A distributed kalman filter with symbolic zonotopes and unique symbols provider for robust state estimation in cps," *International Journal of Control*, vol. 93, no. 11, pp. 2596–2612, 2020.
- [31] G. Singh, T. Gehr, M. Mirman, M. Püschel, and M. Vechev, "Fast and effective robustness certification," Advances in neural information processing systems, vol. 31, 2018.
- [32] C. Combastel, "A state bounding observer for uncertain non-linear continuous-time systems based on zonotopes," in *Proceedings of the* 44th IEEE Conference on Decision and Control. IEEE, 2005, pp. 7228–7234.
- [33] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter, "Interval analysis," in Applied interval analysis. Springer, 2001, pp. 11–43.
- [34] C. Combastel, "Zonotopes and kalman observers: Gain optimality under distinct uncertainty paradigms and robust convergence," *Automatica*, vol. 55, pp. 265–273, 2015.
- [35] I. Kolmanovsky and E. G. Gilbert, "Theory and computation of disturbance invariant sets for discrete-time linear systems," *Mathematical problems in engineering*, vol. 4, no. 4, pp. 317–367, 1998.



Carlos Trapiello received the M.Sc. degree in Aerospace Engineering from the Universidad Politécnica de Madrid (UPM), Madrid, Spain, in 2015; and the M.Sc. degree in Automatic Control & Robotics and the Ph.D. degree in Control Engineering from the Universitat Politècnica de Catalunya-BarcelonaTech (UPC), Barcelona, Spain, in 2018 and 2021, respectively. From January 2022 to October 2022, he has been a postdoc fellow with the IMS lab, University of Bordeaux, France. He currently holds an industrial position in Toulouse, France.



Christophe Combastel received his MSc degree in electrical engineering (1997) and his PhD in control systems (2000), both from the National Polytechnic Institute of Grenoble (Grenoble-INP), France. From 2001 to 2015, he was associate professor at ENSEA near Paris. Since 2015, he is with the University of Bordeaux and the IMS Lab (CNRS UMR5218). As a member of the ARIA team in the Control System Group of IMS, his research interests include interval, set-membership and stochastic algorithms for integrity control applications (e.g. aerospace) ranging

from on-line diagnosis to verified model-based design, with special emphasis on uncertainty propagation, multi-sensor data fusion, and safety/security of cyber-physical systems.



Ali Zolghadri received his PhD from the University of Bordeaux, France – and has been a Full Professor of Control Systems Engineering there since 2003. His current research interests are centered around autonomy, resilience and safety/(cyber)security of cyber-physical systems. He is member of International Technical Committees "SafeProcess" and "Aerospace" of IFAC – IEEE senior member, and TC member of EuroGNC (Council of European Aerospace societies). He has served as IPC member for various international conferences, and has deliv-

ered a number of plenary lectures and other invited talks at venues worldwide. He is an Associate Editor of the "Journal of the Franklin Institute" (Elsevier, USA) and "Complex Engineering Systems" journal – and Editorial Board member of "Aerospace Science and Engineering", MDPI (Switzerland). He is author / co-author of more than 250 publications in archive journals, refereed conference proceedings and technical book chapters, and co-holder of 15 patents in aerospace. He is the recipient of CNRS Medal of Innovation 2016 which rewards – considering all fields and subfields of research — "outstanding scientific research with innovative applications in the technological and societal fields".