# SMT-based Reachability Analysis of High Dimensional Interval Max-Plus Linear Systems

Muhammad Syifa'ul Mufid, Dieky Adzkiya, and Alessandro Abate

***Abstract*—This paper discusses the reachability analysis (RA) of Interval Max-Plus Linear (IMPL) systems, a subclass of continuous-space, discrete-event systems defined over the max-plus algebra. Unlike standard Max-Plus Linear (MPL) systems, where the transition matrix is fixed at each discrete step, IMPL systems allow for uncertainty on state matrices. Given an initial and a target set, we develop algorithms to verify the existence of IMPL system trajectories that, starting from the initial set, eventually reach the target set. We show that RA can be solved by encoding the IMPL system, as well as initial and target sets, into linear real arithmetic expressions, and then checking the satisfaction of a resulting logical formula via a satisfiability modulo theory (SMT) solver. The performance and scalability of the developed SMT-based algorithms are shown to drastically outperform state-of-the-art RA algorithms applied to IMPL systems, which promises to usher their use in practical, industrial-sized IMPL models.**

***Index Terms*—max-plus linear systems, reachability analysis, piecewise-affine systems, difference-bound matrices, linear real arithmetic, satisfiability modulo theory**

## I. Introduction

Max-Plus Linear (MPL) systems are a subclass of discrete-event systems (DES) with a continuous state space defined over the so-called max-plus algebra. They are commonly used to describe synchronisation without concurrency, under the assumption that the timing of the "next" discrete event depends linearly (within the max-plus algebra) on the current times. MPL systems are widely applied for the analysis of models where the timing of discrete events is of interest, such as in transportation networks [22] and in manufacturing systems [23]. Another application of MPL systems deals with biological systems [12], [16].

Recently, the notion of *uncertainty* has also been considered for the analysis and control of MPL systems [35], where delays between successive events are characterised by random quantities. There are several ways to address the uncertainty in MPL systems. For instance, in Stochastic MPL (SMPL) systems [33], the random variables of the matrices can be defined on a common probability space. Whilst in Interval MPL (IMPL) systems [14], the entries of matrices belong to a fixed interval. In practical applications, IMPL systems are more realistic than the simple MPL ones: for instance, in a model for manufacturing production lines, the processing time of a machine depends on the machine's and the raw materials' conditions. IMPL systems have been studied for control and analysis of uncertain models, and used in fault-tolerant control for automated vehicles [34] and in robust control and disturbance rejection [30].

Reachability analysis (RA) is the problem to assess whether a given target set of a dynamical system is reachable from a set of initial conditions. As for dynamical models, also for MPL systems the RA can be performed by computing the forward reach sets from the initial set [4], or alternatively computing the backward reach sets from the target [3]. Both methods employ the translation of an MPL system into an equivalent Piecewise-Affine (PWA) system [5], which is characterised by spatial partitions (PWA regions) and corresponding affine dynamics. They also use Difference-Bound Matrices (DBM) to express initial and target sets. It has been shown in [14] that the aforementioned procedures from [3], [4], [5] can be applied for IMPL systems.

Whilst the approaches in [14] can be applied to systems with multiple initial conditions (rather than starting from a single vector), they only work for small-dimensional IMPL systems. Furthermore, there are a few elements contributing to the computational bottleneck (time and memory requirements) of the approach. First, the translation of IMPL systems into PWA systems has an exponential complexity [2], [14]. Second, the forward and backward reach sets are characterised as unions of finitely many DBMs, which grow exponentially with the time horizon [3], [4], [14].

This paper proposes a new approach to perform the RA of IMPL systems based on SMT solving. Instead of computing reach sets explicitly, as in [14], we employ symbolic variables to encode the states of trajectories of IMPL systems at each time instant. Namely, the trajectories of the IMPL system, together with initial and target sets, are translated into a formula that can be parsed by a Satisfiability Modulo Theory (SMT) solver. An SMT problem refers to the satisfaction of a logic formula w.r.t. a given theory, such as linear arithmetics or bit vectors [11], with possible quantified ($\exists, \forall$) variables. The satisfiability of the formula encoding a reachability problem is checked using an SMT solver. If the SMT solver reports "satisfiable" (resp. "unsatisfiable"), this entails that the target set is reachable (resp. not reachable) from an (resp. any) initial condition within the initial set. In the proposed approach, the initial and target sets are defined as the set of states expressed as linear real arithmetic expressions, which are more general than DBM. In addition to the novel approach to the RA of IMPL systems, the paper also develops a procedure to solve

M. S. Mufid and A. Abate are with the Department of Computer Science, University of Oxford, Oxford OX1 2JD, United Kingdom. (e-mail: {muhammad.syifaul.mufid,alessandro.abate}@cs.ox.ac.uk)

D. Adzkiya is with the Department of Mathematics, Institut Teknologi Sepuluh Nopember, Surabaya 60111, Indonesia. (e-mail: dieky@matematika.its.ac.id)

M. S. Mufid is supported by Indonesia Endowment Fund for Education (LPDP)

*quantified reachability analysis* (cf. Section 5.3) by allowing quantifiers over initial conditions and state matrices.

We have implemented the SMT-based RA of IMPL systems in C++, using Z3 [18] as the SMT solver, and Armadillo [29] for matrix operations in max-plus algebra. According to the computational benchmark, the proposed implementation is significantly faster than the existing procedure in [14]. Furthermore, our implementation can be pushed to perform RA of high-dimensional IMPL systems within reasonable time.

The remaining parts of this paper are structured as follows. Section 2 introduces the brief description of MPL and IMPL systems followed by the definition of DBM and PWA systems. The novel method to compute the image and inverse image of DBM w.r.t. an IMPL system is described in Section 3. The resulting procedure is inspired by the computation of the image and inverse image of DBM w.r.t. an MPL system in [27]. Section 4 summarises the reachability analysis of IMPL systems based on reach sets computation. In Section 5, we present the basic definition of SMT and the main contributions of this paper. The computational benchmarks are presented in Section 6, and we conclude with Section 7.

## II. Models and Preliminaries

### A. Max-Plus Algebra

In max-plus algebra, $\mathbb{R}, \mathbb{R}_{\max}, \varepsilon$ are defined respectively as the set of real numbers, $\mathbb{R} \cup \{\varepsilon\}$, and $-\infty$. The set $\mathbb{R}_{\max}$ is equipped with two binary operations, $\oplus$ and $\otimes$, where

$$a \oplus b := \max\{a, b\} \quad \text{and} \quad a \otimes b := a + b, \tag{1}$$

for all $a, b \in \mathbb{R}_{\max}$. The algebraic structure $(\mathbb{R}_{\max}, \oplus, \otimes)$ is a semi-ring with $\varepsilon$ and 0 as the null and unit elements, respectively [7], [22].

By $\mathbb{R}_{\max}^{m\times n}$, we denote the set of $(m \times n)$-dimensional max-plus algebraic matrices whose elements are in $\mathbb{R}_{\max}$. The notation $A(i, j)$ represents the entry of matrix $A$ at $i^{\text{th}}$ row and $j^{\text{th}}$ column. Furthermore, $A(i, \cdot)$ and $A(\cdot, j)$ denote the vectors corresponding to the $i^{\text{th}}$ row and $j^{\text{th}}$ column of $A$, respectively. The operations in (1) can be extended to matrices, as follows. For $A, B \in \mathbb{R}_{\max}^{m\times n}, C \in \mathbb{R}_{\max}^{n\times p}$ and $\alpha \in \mathbb{R}_{\max}$,

$$[A \oplus B](i, j) = A(i, j) \oplus B(i, j),$$
$$[A \otimes C](i, j) = \bigoplus_{k=1}^{n} A(i, k) \otimes C(k, j),$$
$$[\alpha \otimes A](i, j) = \alpha \otimes A(i, j) = \alpha + A(i, j),$$

for all $i, j$ ranging within the corresponding dimensions. The linear order $\leq$ can be applied to max-plus algebra as follows:

$$a \leq b \text{ if and only if } a \oplus b = b,$$
$$A \leq B \text{ if and only if } A \oplus B = B,$$

for all $a, b \in \mathbb{R}_{\max}$ and $A, B \in \mathbb{R}_{\max}^{m\times n}$.

Given a natural number $k$, the $k$-th max-plus algebraic power of $A \in \mathbb{R}_{\max}^{n\times n}$ is denoted by $A^{\otimes k}$ and corresponds to $A \otimes \ldots \otimes A$ ($k$ times). For $k = 0$, $A^{\otimes 0}$ is an $n$-dimensional identity matrix $I_n$ whose diagonal and non-diagonal elements are 0 and $\varepsilon$, respectively. Similarly, for $a, b \in \mathbb{R}$, the max-plus algebraic power of $a$ w.r.t. $b$ is denoted by $a^{\otimes b}$ and equals to $ab$ in conventional algebra.

A matrix $A \in \mathbb{R}_{\max}^{n\times n}$ is called *regular* if there is at least one finite element in each row [22]. Furthermore, a tuple $\mathbf{g} = (g_1, \ldots, g_n) \in \{1, \ldots, n\}^n$ is said to be a *finite coefficient* of $A$ if $A(i, g_i) \neq \varepsilon$ for $1 \leq i \leq n$. The *region matrix* of a regular matrix $A$ w.r.t. a finite coefficient $\mathbf{g}$ is defined as

$$A_{\mathbf{g}}(i, j) = \begin{cases} A(i, j), & \text{if } g_i = j, \\ \varepsilon, & \text{otherwise.} \end{cases} \tag{2}$$

The *conjugate* of $A$ is $A^{\mathsf{c}}$, where

$$A^{\mathsf{c}}(i, j) = \begin{cases} -A(j, i), & \text{if } A(j, i) \neq \varepsilon, \\ \varepsilon, & \text{otherwise.} \end{cases} \tag{3}$$

### B. Max-Plus and Interval Max-Plus Linear Systems

An autonomous Max-Plus Linear (MPL) system is defined as

$$\mathbf{x}(k) = A \otimes \mathbf{x}(k-1), \quad k = 1, 2, \ldots \tag{4}$$

where $A \in \mathbb{R}_{\max}^{n\times n}$ is the system matrix and vector $\mathbf{x}(k) = [x_1(k) \ \ldots \ x_n(k)]^\top$ denotes the state variables [7]. The variable $\mathbf{x}$ represents the time instances of discrete events, while $k$ corresponds to the counter of discrete events. Hence, it makes practical sense to take $\mathbb{R}^n$ as the state space and $A$ to be a regular matrix.

**Definition 1.** *[7]. The precedence graph of $A \in \mathbb{R}_{\max}^{n\times n}$, denoted by $\mathcal{G}(A)$, is a weighted directed graph with nodes $1, \ldots, n$ and an edge from $j$ to $i$ with weight $A(i, j)$ for each $A(i, j) \neq \varepsilon$.* □

**Definition 2.** *[7]. A matrix $A \in \mathbb{R}_{\max}^{n\times n}$ is called irreducible if $\mathcal{G}(A)$ is strongly connected.* □

Recall that a directed graph is strongly connected if for two different nodes $i, j$ there exists a path from $i$ to $j$ [7]. The weight of a path $p = i_1 i_2 \ldots i_k$ is equal to the sum of edge-weight in $p$. A circuit, a path that begins and ends at the same node, is called *critical* if it has maximum average weight, which is the weight divided by the length of path [7].

An interval over max-plus algebra is defined as

$$\mathbf{a} = [\underline{a}, \overline{a}] = \{a \in \mathbb{R}_{\max} \mid \underline{a} \leq a \leq \overline{a}\}, \tag{5}$$

where $\underline{a} \leq \overline{a}$. If $\underline{a}$ and $\overline{a}$ are both finite, (5) coincides with a closed interval over $\mathbb{R}$. On the other hand, if $\underline{a} = \varepsilon$ and $\overline{a} > \varepsilon$, (5) is a right-bounded interval. Notice that $[\varepsilon \ \varepsilon]$ is also an interval over the max-plus algebra: for the sake of simplicity, we denote it by $\varepsilon$ instead. The extensions of max-plus algebraic operations to intervals are defined as follows:

$$[\underline{a}, \overline{a}] \oplus [\underline{b}, \overline{b}] = [\underline{a} \oplus \underline{b}, \overline{a} \oplus \overline{b}],$$
$$[\underline{a}, \overline{a}] \otimes [\underline{b}, \overline{b}] = [\underline{a} \otimes \underline{b}, \overline{a} \otimes \overline{b}].$$

By $(\mathbb{IR}_{\max}, \oplus, \otimes)$ we denote the interval max-plus algebraic structure, where $\mathbb{IR}_{\max}$ is the set of intervals introduced in (5). Similarly, $\mathbb{IR}_{\max}^{m\times n}$ denotes the set of $m \times n$ interval matrices over $\mathbb{IR}_{\max}$. One could check that the algebraic structure $(\mathbb{IR}_{\max}, \oplus, \otimes)$ is a semi-ring with $[\varepsilon, \varepsilon]$ and $[0, 0]$ as the null and unit element, respectively.

For each interval matrix $\mathbf{A} = \{\mathbf{a}_{ij}\} \in \mathbb{IR}_{\max}^{m\times n}$, we define $\underline{A}, \overline{A} \in \mathbb{R}_{\max}^{m\times n}$ respectively as the lower and upper matrix of

$\mathbf{A}$ i.e., $\underline{A}(i,j) = \underline{a}_{ij}$ and $\overline{A}(i,j) = \overline{a}_{ij}$ for $1 \le i \le m, 1 \le j \le n$. Hence, $\mathbf{A}$ can be writen as $[\underline{A}, \overline{A}]$ and $A \in \mathbf{A}$ iff $\underline{A} \le A \le \overline{A}$. For $\mathbf{A}, \mathbf{B} \in \mathbb{IR}_{\max}^{m \times n}, \mathbf{C} \in \mathbb{IR}_{\max}^{n \times p}$ and $\alpha \in \mathbb{IR}_{\max}$, the operations for interval matrices are defined as follows:

$$\mathbf{A} \oplus \mathbf{B} = [\underline{A} \oplus \underline{B}, \overline{A} \oplus \overline{B}],$$
$$\mathbf{A} \otimes \mathbf{C} = [\underline{A} \otimes \underline{C}, \overline{A} \otimes \overline{C}],$$
$$\alpha \otimes \mathbf{A} = [\underline{\alpha} \otimes \underline{A}, \overline{\alpha} \otimes \overline{A}].$$

It is straightforward to see that if $A_1, A_2 \in \mathbf{A}$, then so is $A_1 \oplus A_2$. The $k^{\text{th}}$ power of $\mathbf{A} \in \mathbb{IR}_{\max}^{n \times n}$ is given by $\mathbf{A}^{\otimes k} = [\underline{A}^{\otimes k}, \overline{A}^{\otimes k}]$. We call an interval matrix $\mathbf{A} = [\underline{A}, \overline{A}]$ to be irreducible if both $\underline{A}$ and $\overline{A}$ are irreducible. In fact, it is straightforward to see that if the lower matrix is irreducible than so is the upper one.

Interval MPL (IMPL) systems are the extension of MPL systems, in the sense that the state matrix is event varying. More precisely, in IMPL systems, for each $k$, each entry of state matrix $A$ takes values in a given interval. The dynamical equation of an autonomous IMPL system with an interval matrix $\mathbf{A} = [\underline{A}\ \overline{A}]$ is

$$\mathbf{x}(k) = A_{k-1} \otimes \mathbf{x}(k-1), \quad k = 1, 2, \ldots \tag{6}$$

where $\mathbf{x}(k) \in \mathbb{R}^n$, $A_k \in \mathbf{A}$ for each $k \ge 0$. Notice that, in (6), $\mathbf{x}(k) \in [\underline{A} \otimes \mathbf{x}(k-1), \overline{A} \otimes \mathbf{x}(k-1)]$ which can be expressed as

$$\bigoplus_{j=1}^{n}(\underline{A}(i,j) + x_j(k-1)) \le x_i(k) \le \bigoplus_{j=1}^{n}(\overline{A}(i,j) + x_j(k-1)), \tag{7}$$

for each $i \in \{1, \ldots, n\}$.

**Remark 1.** *In this work, for each $i, j \in \{1, \ldots, n\}$, it is always the case that either both $\underline{A}(i,j) > \varepsilon$ and $\overline{A}(i,j) > \varepsilon$, or $\underline{A}(i,j) = \overline{A}(i,j) = \varepsilon$.* □

**Example 1.** *Consider a two-dimensional IMPL system (6), where*

$$\underline{A} = \begin{bmatrix} 1 & 3 \\ 2 & 3 \end{bmatrix}, \qquad \overline{A} = \begin{bmatrix} 2 & 5 \\ 3 & 5 \end{bmatrix}. \tag{8}$$

*The IMPL system (8) represents the simple railway network shown in Fig. 1 which consists of two stations and four trains. The intervals represent the time instance required by the trains to pass through the corresponding tracks. For example, the shortest and longest time to travel from station $S_1$ to station $S_2$ is 2 and 3 time units, respectively.* □

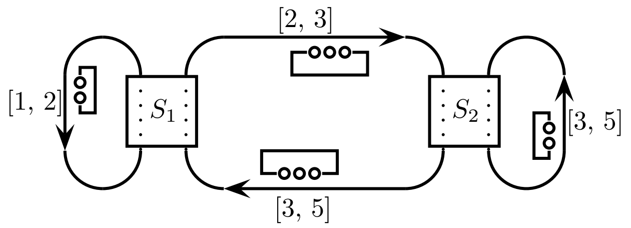

Fig. 1. A simple railway network represented by the IMPL system in (8).

## C. Difference-Bound Matrices

Difference-Bound Matrices (DBMs) are defined as the intersection of sets characterised by the difference of two variables.

**Definition 3.** *[19]. A DBM in $\mathbb{R}^n$ is the intersection of sets defined by $x_i - x_j \sim_{i,j} d_{i,j}$, where $\sim_{i,j} \in \{>, \ge\}$ and $d_{i,j} \in \mathbb{R}_{\max}$ for $0 \le i, j \le n$. The value of the special variable $x_0$ is always equal to 0.* □

The variable $x_0$ is used to represent inequalities involving a single variable: $x_i \ge \alpha$ can be written as $x_i - x_0 \ge \alpha$. A DBM in $\mathbb{R}$ can be expressed as a pair of matrices $(D, S)$: $D(i,j)$ corresponds to the bound variable $d_{i,j}$, while $S(i,j) = 1$ if $\sim_{i,j} = \ge$, and 0 otherwise. Notice that, by Definition 3, $D$ is an $(n+1)$-dimensional max-plus algebraic matrix, and $S$ is an $(n+1)$-dimensional binary matrix.

Some operations can be applied to DBMs, such as intersection, canonical-form representation (the expression of a DBM with the tightest possible bounds [19, Sec. 4.1]), emptiness checking, image and inverse image w.r.t. affine dynamics [2], [27].

**Remark 2.** *In the rest of this paper, we may use the bound matrix only whenever recalling a DBM, especially when all inequalities in the DBM are non-strict ones. For example a DBM $D = \{\boldsymbol{x} \in \mathbb{R}^2 \mid 1 \le x_1 - x_2 \le 2\}$ can be expressed as*

$$D = \begin{bmatrix} 0 & \varepsilon & \varepsilon \\ \varepsilon & 0 & 1 \\ \varepsilon & -2 & 0 \end{bmatrix}.$$

□

## D. Piecewise-Affine Systems

Piecewise-Affine (PWA) systems [32] are defined by partitioning the state space into several domains characterised by polyhedra. Each domain, or PWA region, is associated with an affine function expressing the local transition function. It is shown in [21] that every MPL system can be transformed into a PWA system. For the model in (4), the PWA regions are generated from $\mathbf{g} = (g_1, \ldots, g_n) \in \{1, \ldots, n\}^n$, where $A(i, g_i) \ne \varepsilon$ for $1 \le i \le n$. The region corresponding to $\mathbf{g}$ is

$$\mathsf{R}_{\mathbf{g}} = \bigcap_{i=1}^{n} \bigcap_{j=1}^{n} \{\mathbf{x} \in \mathbb{R}^n | x_{g_i} - x_j \ge A(i,j) - A(i, g_i)\}. \tag{9}$$

Notice that $\mathsf{R}_{\mathbf{g}}$ is a DBM. The affine dynamics associated to a non-empty $\mathsf{R}_{\mathbf{g}}$ are

$$x_i(k) = x_{g_i}(k-1) + A(i, g_i), \quad i = 1, \ldots, n. \tag{10}$$

As shown in [27], the PWA region (9) can be also generated by matrix operations as follows

$$\mathsf{R}_{\mathbf{g}} = ([A_{\text{ext}}]_{\mathbf{g}}^{\mathsf{c}} \otimes A_{\text{ext}}) \oplus I_{n+1}, \tag{11}$$

where $A_{\text{ext}}$ is the extension of $A$ by adding the 0th row and column with $A(0,0) = 0, A(0,i) = A(i,0) = \varepsilon$ for $1 \le i \le n$ and $\mathbf{g}$ is also extended into $(g_0, g_1, \ldots, g_n)$ with $g_0 = 0$. Moreover, we may add square brackets to represent the region matrix (2) and the conjugate (3) of $A_{\text{ext}}$, i.e. $[A_{\text{ext}}]_{\mathbf{g}}^{\mathsf{c}}$.

**Remark 3.** *For the sake of simplicity, instead of using $\mathbf{x}(k-1)$ and $\mathbf{x}(k)$, we may write $\mathbf{x} = [x_1, \ldots, x_n]^\top$ and $\mathbf{x}' = [x'_1, \ldots, x'_n]^\top$ to represent the "current" and the "next" variables.* □

**Remark 4.** *Following the introduction of* (11)*, in the rest of this paper, all matrices (after extension) and tuples are indexed starting from zero. In accordance to* (12) *and* (13)*, it is always the case that $x_0 = x'_0 = 0$.* □

As shown in [14], the translation into PWA system from an IMPL system can be done by generating the PWA regions from its upper matrix $\overline{A}$, i.e.,

$$\overline{\mathsf{R}}_{\mathbf{g}} = \bigcap_{i=0}^{n} \bigcap_{j=0}^{n} \left\{ \mathbf{x} \in \mathbb{R}^n | x_{g_i} - x_j \geq \overline{A}_{\text{ext}}(i,j) - \overline{A}_{\text{ext}}(i,g_i) \right\}. \tag{12}$$

If (12) is not empty then the corresponding affine dynamics are

$$\left.\begin{array}{l} x'_i \geq \max\{x_0 + \underline{A}_{\text{ext}}(i,0), \ldots, x_n + \underline{A}_{\text{ext}}(i,n)\} \\ x'_i \leq x_{g_i} + \overline{A}_{\text{ext}}(i,g_i) \end{array}\right\}, \tag{13}$$

for $i = 0, \ldots, n$. Let us remark that the IMPL dynamics in (13) are evidently richer, more complex than the MPL ones in (10). Furthermore, whilst they are semantically equivalent to those in (7), they are syntactically simpler, as they do not use the maximum operation on the right-hand side of (7).

**Example 2.** *Let us generate the PWA regions for the IMPL system* (8)*. As mentioned above, the regions can be generated from the upper matrix $\overline{A}$: notice that there are four possible finite coefficients **g** of $\overline{A}_{\text{ext}}$, namely $(0,1,1), (0,1,2), (0,2,1)$, and $(0,2,2)$. By* (11)*, we have*

$$\overline{\mathsf{R}}_{(0,1,1)} = \begin{bmatrix} 0 & \varepsilon & \varepsilon \\ \varepsilon & -2 & -3 \\ \varepsilon & \varepsilon & \varepsilon \end{bmatrix} \otimes \begin{bmatrix} 0 & \varepsilon & \varepsilon \\ \varepsilon & 2 & 5 \\ \varepsilon & 3 & 5 \end{bmatrix} \oplus I_3 = \begin{bmatrix} 0 & \varepsilon & \varepsilon \\ \varepsilon & 0 & 3 \\ \varepsilon & \varepsilon & 0 \end{bmatrix},$$

$$\overline{\mathsf{R}}_{(0,1,2)} = \begin{bmatrix} 0 & \varepsilon & \varepsilon \\ \varepsilon & -2 & \varepsilon \\ \varepsilon & \varepsilon & -5 \end{bmatrix} \otimes \begin{bmatrix} 0 & \varepsilon & \varepsilon \\ \varepsilon & 2 & 5 \\ \varepsilon & 3 & 5 \end{bmatrix} \oplus I_3 = \begin{bmatrix} 0 & \varepsilon & \varepsilon \\ \varepsilon & 0 & 3 \\ \varepsilon & -2 & 0 \end{bmatrix},$$

$$\overline{\mathsf{R}}_{(0,2,1)} = \begin{bmatrix} 0 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & -3 \\ \varepsilon & -5 & \varepsilon \end{bmatrix} \otimes \begin{bmatrix} 0 & \varepsilon & \varepsilon \\ \varepsilon & 2 & 5 \\ \varepsilon & 3 & 5 \end{bmatrix} \oplus I_3 = \begin{bmatrix} 0 & \varepsilon & \varepsilon \\ \varepsilon & 0 & 2 \\ \varepsilon & -3 & 0 \end{bmatrix},$$

$$\overline{\mathsf{R}}_{(0,2,2)} = \begin{bmatrix} 0 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & -5 & -5 \end{bmatrix} \otimes \begin{bmatrix} 0 & \varepsilon & \varepsilon \\ \varepsilon & 2 & 5 \\ \varepsilon & 3 & 5 \end{bmatrix} \oplus I_3 = \begin{bmatrix} 0 & \varepsilon & \varepsilon \\ \varepsilon & 0 & \varepsilon \\ \varepsilon & -2 & 0 \end{bmatrix},$$

*which respectively corresponds to DBMs*

$$\begin{aligned} \overline{\mathsf{R}}_{(0,1,1)} &= \{\mathbf{x} \in \mathbb{R}^2 \mid x_1 - x_2 \geq 3\}, \\ \overline{\mathsf{R}}_{(0,1,2)} &= \{\mathbf{x} \in \mathbb{R}^2 \mid 3 \leq x_1 - x_2 \leq 2\}, \\ \overline{\mathsf{R}}_{(0,2,1)} &= \{\mathbf{x} \in \mathbb{R}^2 \mid 2 \leq x_1 - x_2 \leq 3\}, \\ \overline{\mathsf{R}}_{(0,2,2)} &= \{\mathbf{x} \in \mathbb{R}^2 \mid x_1 - x_2 \leq 2\}. \end{aligned}$$

*It is straightforward to see that $\overline{\mathsf{R}}_{(0,1,2)}$ is empty.* □

## III. Computation of Image and Inverse Image of Sets over Interval Max-Plus Linear Systems

This section describes a procedure to compute the image and inverse image of a set $X$ w.r.t. the IMPL system (7),

$$\mathsf{Img}(X) = \{A \otimes \mathbf{x} \mid \mathbf{x} \in X, A \in \mathbf{A}\}, \tag{14}$$

$$\mathsf{Inv}(X) = \{\mathbf{x} \in \mathbb{R}^n \mid \exists A \in \mathbf{A} \text{ s.t. } A \otimes \mathbf{x} \in X\}. \tag{15}$$

We first show the steps to compute the image and inverse image of a single DBM w.r.t. IMPL dynamics (13). Notice that (13) can be rewritten as a DBM in $\mathbb{R}^{2n}$

$$\begin{aligned} &\bigcap_{i=0}^{n} \{[\mathbf{x}^\top, (\mathbf{x}')^\top]^\top \mid x_{g_i} - x'_i \geq -\overline{A}_{\text{ext}}(i,g_i)\} \cap \\ &\bigcap_{i=0}^{n} \bigcap_{j=0}^{n} \{[\mathbf{x}^\top, (\mathbf{x}')^\top]^\top \mid x'_i - x_j \geq \underline{A}_{\text{ext}}(i,j)\}. \end{aligned} \tag{16}$$

As summarised in [13, Algorithm 5.1], the general procedure to compute the image of a DBM $D$ w.r.t. (13) involves: 1) computing the intersection of $D \times \mathbb{R}^n$ and the DBM associated with the dynamics (16), i.e.

$$\begin{aligned} &\left\{[\mathbf{x}^\top, (\mathbf{x}')^\top]^\top \in \mathbb{R}^{2n} \mid \mathbf{x} \in D\right\} \cap \\ &\bigcap_{i=1}^{n} \{[\mathbf{x}^\top, (\mathbf{x}')^\top]^\top \in \mathbb{R}^{2n} \mid x_{g_i} - x'_i \geq -\overline{A}_{\text{ext}}(i,g_i)\} \cap \\ &\bigcap_{i=1}^{n} \bigcap_{j=1}^{n} \{[\mathbf{x}^\top, (\mathbf{x}')^\top]^\top \in \mathbb{R}^{2n} \mid x'_i - x_j \geq \underline{A}_{\text{ext}}(i,j)\}; \end{aligned}$$

then 2) calculating the canonical-form representation (cf. Section II-C) of the obtained intersection; and finally 3) projecting the canonical-form representation over $x'_1, \ldots, x'_n$ by removing the complementary variables, i.e. $x_1, x_2, \ldots, x_n$. The complexity of the procedure depends critically on the second step, which runs in cubic time w.r.t. the number of variables $2n$.

Likewise, the procedure [13, Algorithm 5.2] to compute the inverse image of a DBM $D'$ w.r.t. (13) is: 1) intersecting $\mathbb{R}^n \times D'$ and (16), i.e.

$$\begin{aligned} &\left\{[\mathbf{x}^\top, (\mathbf{x}')^\top]^\top \in \mathbb{R}^{2n} \mid \mathbf{x}' \in D'\right\} \cap \\ &\bigcap_{i=1}^{n} \{[\mathbf{x}^\top, (\mathbf{x}')^\top]^\top \in \mathbb{R}^{2n} \mid x_{g_i} - x'_i \geq -\overline{A}_{\text{ext}}(i,g_i)\} \cap \\ &\bigcap_{i=1}^{n} \bigcap_{j=1}^{n} \{[\mathbf{x}^\top, (\mathbf{x}')^\top]^\top \in \mathbb{R}^{2n} \mid x'_i - x_j \geq \underline{A}_{\text{ext}}(i,j)\}; \end{aligned}$$

2) calculating the canonical-form representation of the obtained intersection; and 3) projecting the canonical-form representation over $x_1, \ldots, x_n$ by removing the complementary variables, i.e., $x'_1, \ldots, x'_n$. The complexity of this procedure is again cubic in $2n$.

**Remark 5.** *In view of the fact that the state-space of IMPL systems can be partitioned by PWA regions* (12) *and can be expressed as DBMs, in this work we assume that the set $X$ in* (14) *and* (15) *can be expressed as a union of finitely many DBMs. We emphasise that recent work [3], [4], [14] has also followed this assumption.* □

Next, we show that the computation of image and inverse image of a DBM w.r.t. affine dynamics (13) can be done using matrix operations in max-plus algebra which, unlike the above procedures, do not involve the canonical-form computation of DBMs. These operations later underpin the new Algorithms 1-2.

**Proposition 1.** *The image of DBM $D$ w.r.t. the dynamics in* (13) *is a $D' = \bigcap_{i=0}^{n}\bigcap_{j=0}^{n}\{\mathbf{x}' \in \mathbb{R}^n \mid x'_i - x'_j \sim_{ij} d'_{ij}\}$, where $d'_{ij} = \bigoplus_{k=0}^{n}\{\underline{A}_{\text{ext}}(i,k) + D(k,g_j) - \overline{A}_{\text{ext}}(j,g_j)\}$ and $D(k,g_j)$ represents the bound of $x_j - x_{g_j}$ in DBM $D$. The operator $\sim_{ij}$ depends on the argmax of $d'_{ij}$ (see Algorithm 1). Alternatively, considering the bound matrix only, the image computation can be expressed as*

$$D' = (\underline{A}_{\text{ext}} \otimes D \otimes [\overline{A}_{\text{ext}}]^{\mathsf{c}}_{\mathbf{g}}) \oplus I_{n+1},$$

*where $[\overline{A}_{\text{ext}}]_{\mathbf{g}}$ is the region matrix of $\overline{A}_{\text{ext}}$ w.r.t $\mathbf{g}$ and $[\overline{A}_{\text{ext}}]^{\mathsf{c}}_{\mathbf{g}}$ is the conjugate of $[\overline{A}_{\text{ext}}]_{\mathbf{g}}$.*

*Proof.* The DBM $D$ is defined over variabes $x_0, \ldots, x_n$ while the resulting image will be represented as a DBM $D'$ over variables $x'_0, \ldots, x'_n$. Let us denote $\overline{a}_{ij} = \overline{A}_{\text{ext}}(i,j)$ and $\underline{a}_{ij} = \underline{A}_{\text{ext}}(i,j)$ for $i,j \in \{0,\ldots,n\}$. By (13), for each $i,j \in \{0,\ldots,n\}$, we have

$$x'_i - x'_j \geq \max\{x_0 + \underline{a}_{i0}, \ldots, x_n + \underline{a}_{in}\} - (x_{g_j} + \overline{a}_{jg_j}). \quad (17)$$

Because $x_i - x_j$ is (lower) bounded by $D(i,j)$ in DBM $D$, (17) can be rewritten as

$$x'_i - x'_j \geq \max\{D(0,g_j) + \underline{a}_{i0}, \ldots, D(n,g_j) + \underline{a}_{in}\} - \overline{a}_{jg_j},$$

which can be expressed as

$$x'_i - x'_j \geq \bigoplus_{k=0}^{n} \left(\underline{A}(i,k) \otimes D(k,g_j) - \overline{a}_{jg_j}\right). \quad (18)$$

By setting $d'_{ij}$ as the right-hand side of (18), the resulting image can be expressed as $D' = \bigcap_{i=0}^{n}\bigcap_{j=0}^{n}\{\mathbf{x}' \in \mathbb{R}^n \mid x'_i - x'_j \sim_{ij} d'_{ij}\}$, where $\sim_{ij}$ depends of the argmax of $d'_{ij}$.

We recall that $[\overline{A}_{\text{ext}}]_{\mathbf{g}}$ is a region matrix with only one finite element in each row. Furthermore, we have $[\overline{A}_{\text{ext}}]^{\mathsf{c}}_{\mathbf{g}}(g_j,j) = -[\overline{A}_{\text{ext}}]_{\mathbf{g}}(j,g_j) = -\overline{a}_{jg_j}$. Thus, the expression (18) is equivalent to

$$x'_i - x'_j \geq [\underline{A}_{\text{ext}} \otimes D \otimes [\overline{A}_{\text{ext}}]^{\mathsf{c}}_{\mathbf{g}}](i,j).$$

Considering the bound matrix only, one can write $D'(i,j) = [\underline{A}_{\text{ext}} \otimes D \otimes [\overline{A}_{\text{ext}}]^{\mathsf{c}}_{\mathbf{g}}](i,j)$. It is possible that, for $i = j$, $[\underline{A}_{\text{ext}} \otimes D \otimes [\overline{A}_{\text{ext}}]^{\mathsf{c}}_{\mathbf{g}}](i,j) < 0$. Hence, the final expression is $D' = (\underline{A}_{\text{ext}} \otimes D \otimes [\overline{A}_{\text{ext}}]^{\mathsf{c}}_{\mathbf{g}}) \oplus I_{n+1}$. □

Bolstered by Proposition 1, Algorithm 1 shows a procedure to compute the image of a DBM $(D,S)$ w.r.t. the affine dynamics in (13). It is important to note that Algorithm 1 is applied to the "sub-system" of the IMPL system that is characterised by the PWA region in (12) and its corresponding dynamics in (13). Therefore, it assumes that there is exactly one region $\overline{\mathsf{R}}_{\mathbf{g}}$ such that $(D,S) \subseteq \overline{\mathsf{R}}_{\mathbf{g}}$.

**Algorithm 1** Image computation of a DBM $(D,S)$ w.r.t. affine dynamics in (13) for IMPL system

```
Inputs: $\mathbf{A} = [\underline{A}, \overline{A}]$, where $\underline{A}, \overline{A} \in \mathbb{R}^{n\times n}_{\max}$ with $\underline{A} \leq \overline{A}$,
        $D$, the bound matrix of the DBM,
        $S$, the sign matrix of the DBM,
        $\mathbf{g} = (g_0, \ldots, g_n)$, the finite coefficient w.r.t. (13)
Output: a DBM $(D', S')$
 1: $n \leftarrow$ ROW($\overline{A}$)
 2: $\underline{A}_{\text{ext}} \leftarrow$ EXTEND($\underline{A}$)
 3: $\overline{A}_{\text{ext}} \leftarrow$ EXTEND($\overline{A}$)
 4: $D' \leftarrow I_{n+1}$
 5: $S' \leftarrow$ EYE($n+1$)
 6: for $i \in \{0,\ldots,n\}$ do
 7:     for $j \in \{0,\ldots,n\}$ do
 8:         $v \leftarrow \underline{A}_{\text{ext}}(i,\cdot) + D(\cdot,g_j)^\top - \overline{A}_{\text{ext}}(j,g_j)$
 9:         $val \leftarrow \max(v)$
10:         $idx \leftarrow \text{argmax}(v)$
11:         if $val > D'(i,j)$ then        ▷ $D'(i,j)$ is either 0 or $\varepsilon$
12:             $D'(i,j) \leftarrow val$
13:             $S'(i,j) \leftarrow \min_{k\in idx} S(k,g_j)$
14: return $(D', S')$
```

At the start of Algorithm 1, we extend the upper and lower matrix by by adding the 0th row and column as $\underline{A}_{\text{ext}}(0,\cdot) = \overline{A}_{\text{ext}}(0,\cdot) = \underline{A}_{\text{ext}}(\cdot,0)^\top = \overline{A}_{\text{ext}}(\cdot,0)^\top = [0\ \varepsilon\ \ldots\ \varepsilon]$. Then, we initialise $D'$ as a max-plus identity matrix and $S'$ as a Boolean (identity) matrix. Notice that the initial DBM $(D',S')$ represents $\mathbb{R}^n$. In line 8, vector $v$ corresponds to the right side of (18). Hence, for each $i,j \in \{0,\ldots,n\}$, $D'(i,j)$ is updated by $\max(v)$ if $D'(i,j) < \max(v)$. In line 10, $idx$ is a set of index(es) corresponding to the maximum elements in $v$. Notice that the sign in (18) depends on $S(k,g_j)$ for $k \in idx$. Also, $>$ (denoted by 0 in $S$) is stricter than $\geq$ (denoted by 1). Thus, we update $S'(i,j)$ by $\min_{k\in idx} S(k,g_j)$. The worst-case complexity of Algorithm 1 is in $\mathcal{O}(n^3)$: the for loops in lines 6-7 involve $(n+1)^2$ iterations, and the steps in lines 9-10 run in linear time.

We recall that the the translation of an IMPL system into a PWA one allows for the simplification of (7) into (13). Thus, in order to compute the image of a DBM $D$ in $\mathbb{R}^n$ w.r.t. the IMPL system dynamics (7), it is necessary to intersect $D$ with all the non-empty PWA regions in (12). In general, the image computation involves the following steps: 1) intersecting the DBM with each region of the corresponding PWA system; 2) computing the image of non-empty intersections w.r.t. its corresponding affine dynamics (13) using Algorithm 1; 3) collecting the resulting images. In general, this procedure results in a union of finitely many DBMs.

**Example 3.** *Let us compute the image of $X_0 = \{\mathbf{x} \in \mathbb{R}^2 \mid 0 \leq x_1 \leq 2, 0 \leq x_2 \leq 2\}$ w.r.t. the IMPL system* (8). *The intersection of $X_0$ and the PWA regions are $D_1 = X_0 \cap \overline{\mathsf{R}}_{(0,1,1)} = \emptyset, D_2 = X_0 \cap \overline{\mathsf{R}}_{(0,2,1)} = \{\mathbf{x} \in \mathbb{R}^2 \mid x_1 = 2, x_2 = 0\}$, and $D_3 = X_0 \cap \overline{\mathsf{R}}_{(0,2,2)} = \{\mathbf{x} \in \mathbb{R}^2 \mid 0 \leq x_1 \leq 2, 0 \leq x_2 \leq 2, -2 \leq x_1 - x_2 \leq 2\}$. The image of $D_2$ is*

$$D'_2 = \left(\begin{bmatrix}0 & \varepsilon & \varepsilon\\ \varepsilon & 1 & 3\\ \varepsilon & 2 & 3\end{bmatrix} \otimes \begin{bmatrix}0 & -2 & 0\\ 2 & 0 & 2\\ 0 & -2 & 0\end{bmatrix} \otimes \begin{bmatrix}0 & \varepsilon & \varepsilon\\ \varepsilon & \varepsilon & -3\\ \varepsilon & -5 & \varepsilon\end{bmatrix}\right) \oplus I_3$$
$$= \begin{bmatrix}0 & -7 & -5\\ 3 & -2 & -2\\ 4 & -1 & -1\end{bmatrix} \oplus I_3 = \begin{bmatrix}0 & -7 & -5\\ 3 & 0 & -2\\ 3 & -1 & 0\end{bmatrix},$$

*which can be expressed as $D'_2 = \{\mathbf{x}' \in \mathbb{R}^2 \mid 3 \le x'_1 \le 7, 4 \le x'_2 \le 5, -2 \le x'_1 - x'_2 \le 1\}$. The image of $D_3$ is*

$$D'_3 = \left(\begin{bmatrix}0 & \varepsilon & \varepsilon\\ \varepsilon & 1 & 3\\ \varepsilon & 2 & 3\end{bmatrix} \otimes \begin{bmatrix}0 & -2 & -2\\ 0 & 0 & -2\\ 0 & -2 & 0\end{bmatrix} \otimes \begin{bmatrix}0 & \varepsilon & \varepsilon\\ \varepsilon & \varepsilon & \varepsilon\\ \varepsilon & -5 & -5\end{bmatrix}\right) \oplus I_3$$
$$= \begin{bmatrix}0 & -7 & -7\\ 3 & -2 & -2\\ 3 & -2 & -2\end{bmatrix} \oplus I_3 = \begin{bmatrix}0 & -7 & -7\\ 3 & 0 & -2\\ 3 & -2 & 0\end{bmatrix},$$

*which can be expressed as $D'_3 = \{\mathbf{x}' \in \mathbb{R}^2 \mid 3 \le x'_1 \le 7, 3 \le x'_2 \le 7, -2 \le x'_1 - x'_2 \le 2\}$. It is a coincidence that, $D'_2 \subset D'_3$. Hence, the image of $X_0$ w.r.t. IMPL system* (8) *is* $\mathsf{Img}(X_0) = D'_3$*, as depicted in Figure 2.*

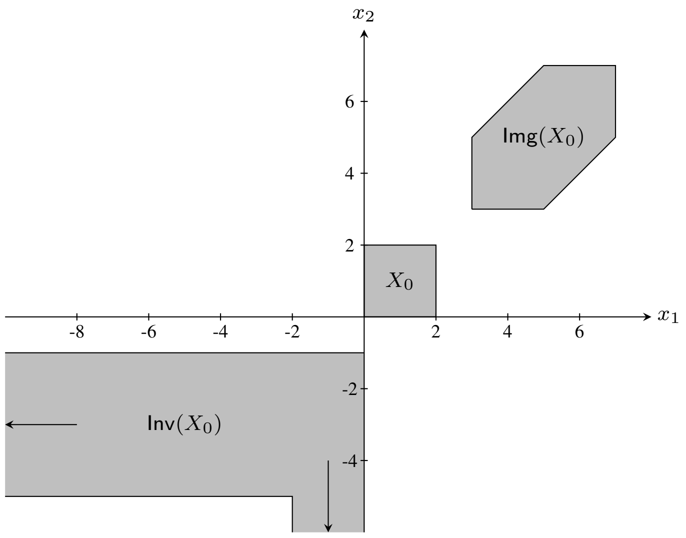

Fig. 2. The image and inverse image of $X_0$ w.r.t. the IMPL system in (8). The arrows in $\mathsf{Inv}(X_0)$ indicate there is no finite lower bound for $x_1$ and $x_2$.

**Proposition 2.** *The inverse image of DBM $D'$ w.r.t. the dynamics* (13) *is $D = \bigcap_{i=0}^n \bigcap_{j=0}^n \{\mathbf{x} \in \mathbb{R}^n \mid x_i - x_j \sim_{ij} d_{ij}\}$, where $d_{ij} = \bigoplus_{j=0}^n(-\overline{A}_{ext}(j, g_j) + \bigoplus_{i=0}^n(D'(j,i) + \underline{A}_{ext}(i,k)))$ and $D'(i,j)$ represents the bound of $x'_i - x'_j$ in DBM $D'$. The operator $\sim_{ij}$ depends on the argmax of $d_{ij}$ (see Algorithm 2). Alternatively, considering the bound matrix only, the inverse image computation can be expressed as*

$$D = ([\overline{A}_{\text{ext}}]^{\mathsf{c}}_{\mathbf{g}} \otimes D' \otimes \underline{A}_{\text{ext}}) \oplus I_{n+1}.$$

*Proof.* The DBM $D'$ is defined over variabes $x'_0, \dots, x'_n$ while the resulting inverse image will be represented as a DBM $D$ over the variables $x_0, \dots, x_n$. Notice that (17) is equivalent to

$$\bigwedge_{k=0}^{n} (x_{g_j} - x_k \ge x'_j - x'_i + \underline{A}_{\text{ext}}(i,k) - \overline{A}_{\text{ext}}(j, g_j)).$$

Thus, for each fixed $j, k \in \{0, \dots, n\}$, we have

$$x_{g_j} - x_k \ge -\overline{A}_{\text{ext}}(j, g_j) + \bigoplus_{i=0}^{n}(D'(j,i) + \underline{A}_{\text{ext}}(i,k)).$$

In general, the above expression can be rewritten as

$$x_{g_j} - x_k \ge \bigoplus_{j=0}^{n}(-\overline{A}_{\text{ext}}(j, g_j) + \bigoplus_{i=0}^{n}(D'(j,i) + \underline{A}_{\text{ext}}(i,k))), \tag{19}$$

because it is possible that there exists such that $l \ne j$ and $g_l = g_j$. By setting $d_{ij}$ as the right-hand side of (19), the resulting inverse image can be expressed as $D = \bigcap_{i=0}^n \bigcap_{j=0}^n \{\mathbf{x} \in \mathbb{R}^n \mid x_i - x_j \sim_{ij} d_{ij}\}$ where $\sim_{ij}$ depends on the argmax of $d_{ij}$.

Considering the bound matrix only, (19) can be expressed as

$$D(g_j, k) = [[\overline{A}_{\text{ext}}]^{\mathsf{c}}_{\mathbf{g}} \otimes D' \otimes \underline{A}_{\text{ext}}](g_j, k).$$

Again, it is possible that $[[\overline{A}_{\text{ext}}]^{\mathsf{c}}_{\mathbf{g}} \otimes D' \otimes \underline{A}_{\text{ext}}](g_j, k) < 0$ for $j, k \in \{0, \dots, n\}$ such that $g_j = k$. Thus, the final expression is $D = ([\overline{A}_{\text{ext}}]^{\mathsf{c}}_{\mathbf{g}} \otimes D' \otimes \underline{A}_{\text{ext}}) \oplus I_{n+1}$. ☐

Algorithm 2 illustrates the steps to compute the inverse image of DBM $(D', S')$ w.r.t. affine dynamics (13). The steps at lines 1-5 are similar to those in Algorithm 1. In line 8, vector $v$ corresponds to the right side of (19). For each $j, k \in \{0, \dots, n\}$, we need to update $D(g_j, k)$ and $S(g_j, k)$. The variables $val$ in line 9 and $sign$ in line 11, respectively, represent the new bound and operator of $x_{g_j} - x_k$: that is, $x_{g_j} - x_k \ge val$ if $sign = 1$ and $x_{g_j} - x_k > val$ if $sign = 0$. If the new bound is larger than the old bound, then the new bound and operator replaces the old ones. In case the new bound is equal to the old bound, we only need to update the operator. Similar to Algorithm 1, the complexity of Algorithm 2 is $\mathcal{O}(n^3)$.

**Algorithm 2** Inverse image computation of a DBM $(D', S')$ w.r.t. affine dynamics in (13) for IMPL system

```
Inputs: $\mathbf{A} = [\underline{A}, \overline{A}]$, where $\underline{A}, \overline{A} \in \mathbb{R}^{n\times n}_{\max}$ with $\underline{A} \le \overline{A}$,
        $D'$, the bound matrix of the DBM,
        $S'$, the sign matrix of the DBM,
        $\mathbf{g} = (g_0, \dots, g_n)$, the finite coefficient w.r.t. (13)
Output: a DBM $(D, S)$
 1: $n \leftarrow \textsc{Row}(\overline{A})$
 2: $\underline{A}_{\text{ext}} \leftarrow \textsc{Extend}(\underline{A})$
 3: $\overline{A}_{\text{ext}} \leftarrow \textsc{Extend}(\overline{A})$
 4: $D \leftarrow I_{n+1}$
 5: $S \leftarrow \textsc{Eye}(n+1)$
 6: for $j \in \{0, \dots, n\}$ do
 7:     for $k \in \{0, \dots, n\}$ do
 8:         $v \leftarrow \underline{A}_{\text{ext}}(\cdot, k)^\top + D'(j, \cdot) - \overline{A}_{\text{ext}}(j, g_j)$
 9:         $val \leftarrow \max(v)$
10:         $idx \leftarrow \text{argmax}(v)$
11:         $sign \leftarrow \min_{i \in idx} S'(j,i)\}$
12:         if $val > D(g_j, k)$ then
13:             $D(g_j, k) \leftarrow val$
14:             $S(g_j, k) \leftarrow sign$
15:         else if $val = D(g_j, k)$ then
16:             $S(g_j, k) \leftarrow \min\{S(g_j, k), sign\}$
17: return $(D, S)$
```

Similarly, the inverse image of a DBM in $\mathbb{R}^n$ w.r.t. IMPL system dynamics (7) characterised by $[\underline{A}, \overline{A}]$ can be computed by: 1) computing the inverse image of the DBM w.r.t each affine dynamics (13) of the PWA system; 2) intersecting the resulting inverse image with the corresponding PWA region; and 3) collecting the non-empty intersections. Again, this procedure results in a union of finitely many DBMs. However, it is possible that the resulting inverse image is an empty set.

**Example 4.** *Let us determine the inverse image of $X_0 = \{\mathbf{x} \in \mathbb{R}^2 \mid 0 \leq x_1 \leq 2, 0 \leq x_2 \leq 2\}$ w.r.t. IMPL system* (8). *The inverse image of $X_0$ w.r.t. the affine dynamics for $\overline{\mathsf{R}}_{(0,1,1)}, \overline{\mathsf{R}}_{(0,2,1)}$ and $\overline{\mathsf{R}}_{(0,2,2)}$ is respectively $E_1, E_2$ and $E_3$, where*

$$\begin{aligned}
E_1 &= \left(\begin{bmatrix} 0 & \varepsilon & \varepsilon \\ \varepsilon & -2 & -3 \\ \varepsilon & \varepsilon & \varepsilon \end{bmatrix} \otimes \begin{bmatrix} 0 & -2 & -2 \\ 0 & 0 & \varepsilon \\ 0 & \varepsilon & 0 \end{bmatrix} \otimes \begin{bmatrix} 0 & \varepsilon & \varepsilon \\ \varepsilon & 1 & 3 \\ \varepsilon & 2 & 3 \end{bmatrix}\right) \oplus I_3 \\
&= \begin{bmatrix} 0 & 0 & 1 \\ -2 & -1 & 1 \\ \varepsilon & \varepsilon & \varepsilon \end{bmatrix} \oplus I_3 = \begin{bmatrix} 0 & 0 & 1 \\ -2 & -1 & 1 \\ \varepsilon & \varepsilon & 0 \end{bmatrix} \\
&= \{\mathbf{x} \in \mathbb{R}^2 \mid -2 \leq x_1 \leq 0, x_2 \leq -1, x_1 - x_2 \geq 1\}, \\
E_2 &= \left(\begin{bmatrix} 0 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & -3 \\ \varepsilon & -5 & \varepsilon \end{bmatrix} \otimes \begin{bmatrix} 0 & -2 & -2 \\ 0 & 0 & \varepsilon \\ 0 & \varepsilon & 0 \end{bmatrix} \otimes \begin{bmatrix} 0 & \varepsilon & \varepsilon \\ \varepsilon & 1 & 3 \\ \varepsilon & 2 & 3 \end{bmatrix}\right) \oplus I_3 \\
&= \begin{bmatrix} 0 & 0 & 1 \\ -3 & -1 & 0 \\ -5 & -4 & -2 \end{bmatrix} \oplus I_3 = \begin{bmatrix} 0 & 0 & 1 \\ -3 & 0 & 0 \\ -5 & -4 & 0 \end{bmatrix} \\
&= \{\mathbf{x} \in \mathbb{R}^2 \mid -3 \leq x_1 \leq 0, -5 \leq x_2 \leq -1, 0 \leq x_1 - x_2 \leq 4\}, \\
E_3 &= \left(\begin{bmatrix} 0 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & -5 & -5 \end{bmatrix} \otimes \begin{bmatrix} 0 & -2 & -2 \\ 0 & 0 & \varepsilon \\ 0 & \varepsilon & 0 \end{bmatrix} \otimes \begin{bmatrix} 0 & \varepsilon & \varepsilon \\ \varepsilon & 1 & 3 \\ \varepsilon & 2 & 3 \end{bmatrix}\right) \oplus I_3 \\
&= \begin{bmatrix} 0 & 0 & 1 \\ \varepsilon & \varepsilon & \varepsilon \\ -5 & -3 & -2 \end{bmatrix} \oplus I_3 = \begin{bmatrix} 0 & 0 & 1 \\ \varepsilon & 0 & \varepsilon \\ -5 & -3 & 0 \end{bmatrix} \\
&= \{\mathbf{x} \in \mathbb{R}^2 \mid x_1 \leq 0, -5 \leq x_2 \leq -1, x_1 - x_2 \leq 3\}.
\end{aligned}$$

*The intersection of the resulting inverse image w.r.t. the corresponding PWA region is $E_1 \cap \overline{\mathsf{R}}_{(0,1,1)} = \{\mathbf{x} \in \mathbb{R}^2 \mid -2 \leq x_1 \leq 0, x_2 \leq -3, x_1 - x_2 \geq 3\}$, $E_2 \cap \overline{\mathsf{R}}_{(0,2,1)} = \{\mathbf{x} \in \mathbb{R}^2 \mid -3 \leq x_1 \leq 0, -5 \leq x_2 \leq -2, 2 \leq x_1 - x_2 \leq 3\}$, and $E_3 \cap \overline{\mathsf{R}}_{(0,2,2)} = \{\mathbf{x} \in \mathbb{R}^2 \mid x_1 \leq 0, -5 \leq x_2 \leq -1, x_1 - x_2 \leq 2\}$. Hence the inverse image of $X_0$ is $\mathsf{Inv}(X_0) = (E_1 \cap \overline{\mathsf{R}}_{(0,1,1)}) \cup (E_2 \cap \overline{\mathsf{R}}_{(0,2,1)}) \cup (E_3 \cap \overline{\mathsf{R}}_{(0,2,2)})$, as illustrated in Figure 2.* □

**Remark 6.** *It is important to note that, one can find $\mathbf{x} \in \mathsf{Inv}(X)$ such that $A \otimes \mathbf{x} \notin X$ for some $A \in [\underline{A}, \overline{A}]$. This is due to the existential quantifier in the inverse image computation* (15). *For instance, from the preceding example, we have $\mathbf{x} = [0\ -1]^\top \in \mathsf{Inv}(X_0)$, $\underline{A} \otimes \mathbf{x} = [2\ 2]^\top \in X_0$ but $\overline{A} \otimes \mathbf{x} = [4\ 4] \notin X_0$.* □

## IV. Explicit Reachability Analysis of Interval Max-Plus Linear Systems

This section discusses the reachability analysis problem for IMPL systems and an existing procedure to study it.

**Problem 1.** *Suppose we have the IMPL system in* (6), *$X_0, Y_0 \subseteq \mathbb{R}^n$ respectively as the initial and target sets and a positive integer $N$; the bounded reachability analysis refers to the problem of determining whether the set $Y$ is reachable in at most $N$-steps from $X_0$: there exist $\mathbf{x}(0) \in X_0$ and $1 \leq k \leq N$ such that $\mathbf{x}(k) \in Y_0$, where $\mathbf{x}(k)$ is computed recursively by* (6) *from $\mathbf{x}(0)$. It is assumed that $X_0 \cap Y_0 = \emptyset$.* □

The procedure to solve Problem 1 has been discussed in [14] by explicity computing the reach sets (cf. Definitions 4-5). The presented results are the extension of work for MPL systems in [3], [4].

**Definition 4** (Forward reach set [14])**.** *Given an IMPL system* (6) *and a non-empty set of initial conditions $X_0 \subseteq \mathbb{R}^n$, the forward reach set $X_N$ at the event step $N > 0$ is the set of all states $\{\mathbf{x}(N) \mid \mathbf{x}(0) \in X_0\}$ that can be reached by the dynamics in* (7). □

**Definition 5** (Backward reach set [14])**.** *Given an IMPL system* (6) *and a non-empty target set $Y_0 \subseteq \mathbb{R}^n$, the backward reach set $Y_{-N}$ is the set of all states $\{\mathbf{y}(-N)\}$ that may lead to $Y_0$ in $N$ steps using the IMPL dynamics* (7). □

It is assumed that both $X_0$ and $Y_0$ can be expressed as the union of finitely many DBMs. For an autonomous IMPL system (6), given a non-empty set of initial conditions $X_0$, the forward reach set $X_k$ at time horizon $k$ can be computed recursively as the image of $X_{k-1}$

$$X_k = \mathsf{Img}(X_{k-1}) = \{A \otimes \mathbf{x} \mid A \in \mathbf{A}, \mathbf{x} \in X_{k-1}\}. \quad (20)$$

Likewise, the backward reach set $Y_{-k}$ from a non-empty target set $Y_0$ is computed recursively as the inverse image of $Y_{-k+1}$ w.r.t. IMPL dynamics (7)

$$\begin{aligned} Y_{-k} &= \mathsf{Inv}(Y_{-k+1}) \\ &= \{\mathbf{x} \in \mathbb{R}^n \mid \exists A \in \mathbf{A} \text{ s.t. } A \otimes \mathbf{x} \in Y_{-k+1}\}. \end{aligned} \quad (21)$$

The computation of (20) (resp. (21)) is performed by calculating the image (resp. inverse image) of each DBM in $X_{k-1}$ (resp. $Y_{-k+1}$), as described in Section III. While it is evident that $X_k \neq \emptyset$ for $k \geq 0$, it is possible that there exists an $l > 0$ such that $Y_{-k} = \emptyset$ for $k \geq l$.

**Remark 7.** *Inspired by a similar procedure in [3], [4] for MPL systems, a "one-shot" procedure to compute* (20) *and* (21) *is introduced in [13] as follows, where $\mathbf{A}^{\otimes k} = [\underline{A}^{\otimes k}, \overline{A}^{\otimes k}]$:*

$$X_k = \{A \otimes \mathbf{x} \mid A \in \mathbf{A}^{\otimes k}, \mathbf{x} \in X_0\}, \quad (22)$$

$$Y_{-k} = \{\mathbf{x} \in \mathbb{R}^n \mid \exists A \in \mathbf{A}^{\otimes k} \text{ s.t. } A \otimes \mathbf{x} \in Y_0\}. \quad (23)$$

*In this work, we argue that the resulting reach sets in* (22)-(23) *are in general not equal to their sequential counterparts* (20)-(21). *It is true that* (20) $\subseteq$ (22) *and* (21) $\subseteq$ (23). *However, the inclusion relations* (22) $\subseteq$ (20) *and* (23) $\subseteq$ (21) *in general do not hold. For instance, using an IMPL system* (8), *we have*

$$\underline{A}^{\otimes 2} = \begin{bmatrix} 5 & 6 \\ 5 & 6 \end{bmatrix}, \ \overline{A}^{\otimes 2} = \begin{bmatrix} 8 & 10 \\ 8 & 10 \end{bmatrix}, \ A = \begin{bmatrix} 5 & 6 \\ 8 & 10 \end{bmatrix} \in \mathbf{A}^{\otimes 2},$$

*and $A \otimes [0\ 0]^\top = [6\ 10]^\top$. One can check that it is impossible to find $A_1, A_2 \in [\underline{A}, \overline{A}]$ and $\mathbf{x} \in \mathbb{R}^2$ such that $A_1 \otimes A_2 \otimes \mathbf{x} = [6\ 10]^\top$. In general, given $A \in [\underline{A}^{\otimes k}, \overline{A}^{\otimes k}]$, it is not always the case that there exist $A_1, \ldots, A_k \in [\underline{A}, \overline{A}]$ such that $A_1 \otimes \ldots \otimes A_k = A$.* □

Algorithm 3 summarises the procedure to solve the reachability analysis of IMPL systems via forward reach sets computation. In line 2, we translate the underlying IMPL system into an equivalent PWA system w.r.t. its upper matrix $\overline{A}$. Starting from bound $k = 1$, we compute the reach set $X_k$ and then intersect the resulting set with the target set $Y_0$. If the intersection is not empty (line 6), then the procedure is terminated. In this case, one can conclude that $Y_0$ is reachable from $X_0$ at bound $k$. On the other hand, if the intersection is empty then the bound is increased by one. This process is repeated until either the condition in line 6 is fulfilled or $k$ exceeds the maximum bound $N$.

**Algorithm 3** Explicit forward RA of IMPL systems

```
Inputs: $\mathbf{A} = [\underline{A}, \overline{A}]$ where $\underline{A}, \overline{A} \in \mathbb{R}_{\max}^{n \times n}$ with $\underline{A} \leq \overline{A}$,
        initial set $X_0 \subseteq \mathbb{R}^n$,
        target set $Y_0 \subseteq \mathbb{R}^n$,
        $N \in \mathbb{N}$
Output: boolean
 1: $reach \leftarrow$ false
 2: generate PWA system w.r.t. $\overline{A}$
 3: $k \leftarrow 1$
 4: while $k \leq N$ do
 5:     $X_k \leftarrow \mathsf{Img}(X_{k-1})$
 6:     if $X_k \cap Y_0 \neq \emptyset$ then
 7:         $reach \leftarrow$ true
 8:         break
 9:     $k \leftarrow k + 1$
10: return $reach$
```

Algorithm 4 summarises the steps to solve reachability analysis of IMPL systems via backward reach sets computation. Again, we first generate the PWA system from the upper matrix $\bar{A}$. Starting from $k = 1$, the backward reach set $Y_{-k}$ is computed. If $Y_{-k} = \emptyset$ then the algorithm is terminated with `false` as output. In case $Y_{-k} \neq \emptyset$, one needs to check the emptiness of $Y_{-k} \cap X_0$. If it is not an empty set, then one can conclude that $Y_0$ is reachable from $X_0$ at step $k$. Otherwise, the bound $k$ is increased by one. This process is repeated until either one of the conditions in lines 6 and 8 is fulfilled, or $k$ exceeds the maximum bound $N$.

**Algorithm 4** Explicit backward RA of IMPL systems

```
Inputs: $\mathbf{A} = [\underline{A}, \overline{A}]$ where $\underline{A}, \overline{A} \in \mathbb{R}_{\max}^{n \times n}$ with $\underline{A} \leq \overline{A}$,
        initial set $X_0 \subseteq \mathbb{R}^n$,
        target set $Y_0 \subseteq \mathbb{R}^n$,
        $N \in \mathbb{N}$
Output: boolean
 1: $reach \leftarrow$ false
 2: generate PWA system w.r.t. $\overline{A}$
 3: $k \leftarrow 1$
 4: while $k \leq N$ do
 5:     $Y_{-k} \leftarrow \mathsf{Inv}(Y_{-k+1})$
 6:     if $Y_{-k} = \emptyset$ then
 7:         break
 8:     if $Y_{-k} \cap X_0 \neq \emptyset$ then
 9:         $reach \leftarrow$ true
10:         break
11:     $k \leftarrow k + 1$
12: return $reach$
```

**Remark 8.** *Both Algorithms 3 and 4 are similar to the existing procedure for RA of IMPL systems in [13]. The only difference is that we compute the images from the initial set $X_0$ and the inverse images from the target set $Y_0$ using Algorithm 1 and 2, respectively.* □

It is straightforward to conclude that both Algorithms 3 and 4 are sound and complete: they are able to provide the correct answer (`true` or `false`) for arbitrary inputs. Thus, the (bounded) reachability analysis for IMPL systems in Problem 1 is indeed decidable. However, to determine whether there exists an integer $N > 0$ such that the target set $Y_0$ is reachable from $X_0$ at $N$-steps is undecidable; this is due to the fact that if Algorithms 3 and 4 yield `false`, in general we cannot conclude that that $Y_0$ is not reachable from $X_0$ within time horizons greater than $N$.

**Example 5.** *With the preceding IMPL system in Example 1, we define the initial and target sets respectively as $X_0 = \{\mathbf{x} \in \mathbb{R}^2 \mid x_1 - x_2 \leq 3\}$ and $Y_0 = \{\mathbf{x} \in \mathbb{R}^2 \mid x_1 - x_2 \geq 5\}$.*

*The forward reach sets are $X_k = \{\mathbf{x} \in \mathbb{R}^2 \mid -2 \leq x_1 - x_2 \leq 2\}$ for all $k \geq 1$. Hence, we can conclude that $Y_0$ is not reachable from $X_0$. By backward reach set computation, we have $Y_1 = \emptyset$, leading to the same conclusion.* □

The main drawback of the new approach is its scalability. First of all, the number of regions in the PWA system depends on the size of state matrix $A$ and on the number of finite entries in $A$. The worst-case complexity for generating the PWA system via (9) is $\mathcal{O}(n^{n+3})$ [2], where $n$ is the dimension of the state matrix. Furthermore, the forward and backward reachable sets are a union of finitely many DBMs. In the worst case, the number of DBMs grows exponentially w.r.t. the time horizon. As shown in [14], the worst-case complexity to generate the forward reach sets up to bound $N$ is $\mathcal{O}(\sum_{k=0}^{N-1} |X_k| \cdot n^{n+3})$, where $|X_k|$ represents the number of DBMs used to express $X_k$. Similarly, the complexity for backward reach sets computations is $\mathcal{O}(\sum_{k=0}^{N-1} |Y_k| \cdot n^{n+3})$. In order to mitigate the exposed scalability limitations, we provide next a fresh new look at reachability analysis for IMPL, based on SMT solving.

## V. SMT-based Reachability Analysis of Interval Max-Plus Linear Systems

This section discusses a novel approach to solve Problem 1 via SMT solving. We first mention basic notions of Satisfiability Modulo Theory (SMT) and then describe the new, SMT-based procedure for reachability analysis of IMPL systems. The resulting procedure is inspired by similar work applied to MPL systems in [28]. At the end of this section, we show that the SMT-based procedure can be used to solve generalisations of Problem 1 obtained by allowing logical quantifiers over the initial set as well as the state matrices in (6).

### A. Satisfiability Modulo Theory

Satisfiability Modulo Theory (SMT) concerns the problem of determining the satisfaction of a first-order logical formula w.r.t. a background logical theory, such as Boolean logic (which generalises SAT problems), bit-vectors, real and integer arithmetics, etc. [9], [11], [25]. For instance, the following formula

$$(x \geq 0) \land (y < 2) \land (x - y < -1) \qquad (24)$$

admits solutions for $x, y \in \mathbb{R}$, but has no solution for $x, y \in \mathbb{Z}$.

In general, an SMT formula may contain conjunctions ($\land$), disjunctions ($\lor$), and quantifiers ($\exists, \forall$). An SMT solver is a software tool that reports whether the given formula is satisfiable or not. In the former case, it can provide one of the satisfying assignments for the formula. The quantifier $\exists$ (resp. $\forall$) is used to express that a formula holds for at least one assignment (respectively, all possible assignments) over the quantified variables. For instance, the formula $\exists x, y \in \mathbb{R}.\ F$ (where $F$ is as in (24)) holds, while $\forall x, y \in \mathbb{R}.\ F$ does not. To negate a quantified formulae, one can "switch" the quantifiers (keeping the same order of quantifiers) and then negate the quantifier-free sub-formula (De Morgan's law).

One of the widely used theories in SMT is Linear Real Arithmetic (LRA) [25, Chapter 5]. Given real variables $\mathtt{x}_1, \ldots, \mathtt{x}_n$, a formula in LRA is an arbitrary Boolean combination, or quantification, of atoms in the form $\sum_{i=1}^{n} a_i \mathtt{x}_i \bowtie c$, where $\bowtie \in \{>, <, \geq, \leq, \neq, =\}$ and $a_1, \ldots, a_n, c$ are rational constants. Real Difference Logic (RDL) [17] is a sub-logic of LRA in which all atoms are restricted to the form $x_i - x_j \bowtie c$. The quantifier-free fragment of LRA and RDL is denoted as QF-LRA and QF-RDL, respectively. Both theories are decidable [11, Section 26].

Notice that each DBM can be expressed as a QF-LRA formula, where Boolean connectives are exclusively conjunctions ($\wedge$). Moreover, if a DBM does not contain any inequality with single variable, then it can be translated into a formula in QF-RDL [28]. The non-emptiness of a DBM is equivalent to the satisfiability of its corresponding QF-LRA formula.

SMT has grown into a very active research subject: it has standardised libraries and a collection of benchmarks developed by the SMT community [10], as well as an annual international competition for SMT solvers [8]. As a result, there are several powerful SMT solvers, such as MATHSAT5 [15], Yices 2.2 [20], and Z3 [18]. Applications of SMT-solving arise on supervisory control of discrete-event systems [31], verification of neural networks [24], optimization [26], and sound synthesis of Lyapunov functions [6] and of control architectures [1].

### *B. SMT-Based Reachability Analysis of Interval Max-Plus Linear Systems*

This subsection presents the novel procedure to solve RA of IMPL systems via SMT-solving. The main idea underpinning the new method is to transform the underlying IMPL system, as well as its initial and target sets, into a formula in QF-LRA, and then passing the formula into an SMT solver. A similar procedure has been successfully applied to solve RA of simpler MPL systems in [28].

As shown in [28], the MPL system in (4) with state matrix $A \in \mathbb{R}_{\max}^{n\times n}$ can be expressed as a formula in QF-RDL, as follows:

$$\bigwedge_{i=1}^{n} \left( \left( \bigwedge_{j \in \mathtt{fin}_i} \mathtt{x}_i^{(k)} - \mathtt{x}_j^{(k-1)} \geq a_{ij} \right) \wedge \left( \bigvee_{j \in \mathtt{fin}_i} \mathtt{x}_i^{(k)} - \mathtt{x}_j^{(k-1)} = a_{ij} \right) \right), \tag{25}$$

where $\mathtt{x}_1^{(k)}, \ldots, \mathtt{x}_n^{(k)}$ represents the symbolic variables at a given step $k$, and $\mathtt{fin}_i$ is a set containing the indices of the finite elements of $A(i, \cdot)$.

For IMPL systems, we know that the state matrix is not fixed at each step $k$ and bounded by the lower matrix $\underline{A}$ and upper matrix $\overline{A}$. Thus, formula (25) is modified into a QF-LRA formula

$$\bigwedge_{i=1}^{n} \left( \left( \bigwedge_{j \in \mathtt{fin}_i} \mathtt{x}_i^{(k)} - \mathtt{x}_j^{(k-1)} \geq \mathtt{a}_{ij}^{(k-1)} \right) \wedge \left( \bigvee_{j \in \mathtt{fin}_i} \mathtt{x}_i^{(k)} - \mathtt{x}_j^{(k-1)} = \mathtt{a}_{ij}^{(k-1)} \right) \wedge \left( \bigwedge_{j \in \mathtt{fin}_i} (\mathtt{a}_{ij}^{(k-1)} \geq \underline{a}_{ij}) \wedge (\mathtt{a}_{ij}^{(k-1)} \leq \overline{a}_{ij}) \right) \right), \tag{26}$$

where $\mathtt{a}_{ij}^{(k)}$ represents a symbolic variable for the element of the state matrix at the $k$-th step from (6), at row $i$ and column $j$, and $\mathtt{fin}_i$ is a set containing the indices of the finite elements of $\overline{A}(i, \cdot)$.

**Remark 9.** *A similar translation to a QF-LRA formula can be obtained for a time-varying IMPL system of the form* $\boldsymbol{x}(k) = B_k \otimes \boldsymbol{x}(k) \oplus A_{k-1} \otimes \boldsymbol{x}(k-1)$ *for* $B_k \in [\underline{B}, \overline{B}]$ *and* $A_k \in [\underline{A}, \overline{A}]$*, which has seen practical use in the literature. The main idea is that any maximisation operation* $x_i' = \max\{x_1 + a_i, \ldots, x_n + a_n\}$ *can be expressed into a QF-RDL (a sublogic of QF-LRA) formula as in* (25)*. Furthermore, one can also translate a minimisation operation* $x_i' = \min\{x_1 + a_i, \ldots, x_n + a_n\}$ *to QF-RDL formula due to the duality relation* $\min\{x_1 + a_i, \ldots, x_n + a_n\} = -\max\{-x_1 - a_i, \ldots, -x_n - a_n\}$. $\square$

For simplicity, we introduce the sets of symbolic variables $\mathcal{A}^{(k-1)} = \{\mathtt{a}_{ij}^{(k-1)} \mid 1 \leq i \leq n, j \in \mathtt{fin}_i\}$ and $\mathcal{V}^{(k)} = \{\mathtt{x}_1^{(k)}, \ldots, \mathtt{x}_n^{(k)}\}$. Also, the first two conjuncts of (26) will be denoted by $\mathtt{Im}(\mathcal{V}^{(k-1)}, \mathcal{V}^{(k)})$, whilst the last conjunct (a constraint) is expressed by $\mathtt{M}(\mathcal{A}^{(k-1)})$.

Consequently, the following QF-LRA formula

$$\bigwedge_{k=1}^{N} \left( \mathtt{Im}(\mathcal{V}^{(k-1)}, \mathcal{V}^{(k)}) \wedge \mathtt{M}(\mathcal{A}^{(k-1)}) \right)$$

is a *symbolic representation* of the states of the trajectory of the IMPL system in (6) for $k = 1, \ldots, N$. Furthermore, it follows that the reachability of the target set $Y_0$ from the initial set $X_0$ up to bound $N$ can be equivalently expressed as the satisfiability of the formula

$$X^{(0)} \wedge \left( \bigwedge_{k=1}^{N} \mathtt{Im}(\mathcal{V}^{(k-1)}, \mathcal{V}^{(k)}) \wedge \mathtt{M}(\mathcal{A}^{(k-1)}) \right) \wedge \bigvee_{k=1}^{N} Y^{(k)}, \tag{27}$$

where $X^{(0)}$ (resp. $Y^{(k)}$) is the QF-LRA representation of set $X_0$ (resp. $Y_0$) over $\mathcal{V}^{(0)}$ (resp. $\mathcal{V}^{(k)}$).

Algorithm 5 illustrates the SMT-based version of Algorithm 3. The command $\mathsf{symb\_mat}(\overline{A}, k-1)$ generates the $k$-th symbolic matrix according to the finite elements of $\overline{A}$ while the function $\mathsf{symb\_var}(k, n)$ generates a set of $n$ real-valued variables for time horizon $k$. In line 4, $F$ is a last-in/first-out (LIFO) *program stack* containing SMT formulae, as in (27). The command push adds a new formula into $F$, while pop removes the last one.

At the start of the procedure, both $X_0$ and $Y_0$ are expressed as QF-LRA formulae over $\mathcal{V}^{(0)}$. The function $Y.\mathsf{subs}(\mathcal{V}^{(k-1)}, \mathcal{V}^{(k)})$ substitutes each appearance of $\mathrm{x}_i^{(k-1)}$ in $Y$ with $\mathrm{x}_i^{(k)}$. The non-emptiness checking of a union of DBMs in line 6 of Algorithm 3 is now formulated as the satisfiability checking of a QF-LRA formula in line 14 of Algorithm 5, where $\mathsf{mk\_and}(F)$ stands for $\wedge_{f\in F} f$. The check function is implemented by an SMT solver, where $\mathsf{check}(\mathsf{mk\_and}(F)) = \mathsf{SAT}$ means that $\mathsf{mk\_and}(F)$ is satisfiable. In lines 10-13 of Algorithm 5, a QF-LRA formula for (26) and the target set over $\mathcal{V}^{(k)}$ are added to $F$ at each iteration $k$. If the condition in line 14 is not fulfilled, then the last element of $F$ (i.e., $Y^{(k)}$) is removed.

**Algorithm 5** SMT-based Forward RA of IMPL systems

```
Inputs: $\mathbf{A} = [\underline{A}, \overline{A}]$ where $\underline{A}, \overline{A} \in \mathbb{R}_{\max}^{n\times n}$ with $\underline{A} \leq \overline{A}$,
        initial set $X_0 \subseteq \mathbb{R}^n$ expressed as a QF-LRA formula,
        target set $Y_0 \subseteq \mathbb{R}^n$ expressed as a QF-LRA formula,
        $N \in \mathbb{N}$
Output: boolean
 1: $reach \leftarrow \texttt{false}$
 2: $n \leftarrow \mathrm{Row}(\underline{A})$                    ▷ the number of rows of $\underline{A}$
 3: $\mathcal{V}^{(0)} \leftarrow \mathsf{symb\_var}(0, n)$
 4: $F \leftarrow \emptyset$                                    ▷ empty stack
 5: $F.\mathsf{push}(X_0)$                           ▷ $X_0$ is defined over $\mathcal{V}^{(0)}$
 6: $k \leftarrow 1$
 7: while $k \leq N$ do
 8:     $\mathcal{A}^{(k-1)} \leftarrow \mathsf{symb\_mat}(\overline{A}, k-1)$
 9:     $\mathcal{V}^{(k)} \leftarrow \mathsf{symb\_var}(k, n)$
10:     $F.\mathsf{push}(\mathtt{Im}(\mathcal{V}^{(k-1)}, \mathcal{V}^{(k)}))$
11:     $F.\mathsf{push}(\mathtt{M}(\mathcal{A}^{(k-1)}))$
12:     $Y_0.\mathsf{subs}(\mathcal{V}^{(k-1)}, \mathcal{V}^{(k)})$        ▷ $Y_0$ is now defined over $\mathcal{V}^{(k)}$
13:     $F.\mathsf{push}(Y_0)$
14:     if $\mathsf{check}(\mathsf{mk\_and}(F)) = \mathsf{SAT}$ then
15:         $reach \leftarrow \texttt{true}$
16:         break
17:     $F.\mathsf{pop}()$
18:     $k \leftarrow k+1$
19: return $reach$
```

We now will describe the approach for SMT-based backward RA of IMPL systems. For $k \geq 1$, we introduce $\mathcal{V}^{(-k)} = \{\mathrm{x}_1^{(-k)}, \ldots, \mathrm{x}_n^{(-k)}\}$ to represent the set of variables encompassing the $k$-step backward states obtained from $\mathcal{V}^{(0)}$, and $\mathcal{A}^{(1-k)}$ to express the corresponding $k$-step backward state matrix. The backward version of (27) is

$$Y^{(0)} \wedge \left( \bigwedge_{k=1}^{N} \mathtt{Im}(\mathcal{V}^{(-k)}, \mathcal{V}^{(1-k)}) \wedge \mathtt{M}(\mathcal{A}^{(1-k)}) \right) \wedge \bigvee_{k=1}^{N} X^{(-k)}. \tag{28}$$

Algorithm 6 describes the SMT-based procedure to solve Problem 1 using the backward approach. The satisfiability checking in line 12 of Algorithm 6 is equivalent to the non-emptiness checking in line 6 of Algorithm 4. If the SMT solver reports "unsatisfiable," then the procedure terminates and outputs false. Otherwise, the initial set over $\mathcal{V}^{(-k)}$ is added to the formula before checking its satisfiability again in line 16. If it is satisfiable then the algorithm is terminated with true as the output, otherwise the last element of $F$ (i.e., $X^{(-k)}$) is removed.

**Algorithm 6** SMT-based Backward RA of IMPL systems

```
Inputs: $\mathbf{A} = [\underline{A}, \overline{A}]$ where $\underline{A}, \overline{A} \in \mathbb{R}_{\max}^{n\times n}$ with $\underline{A} \leq \overline{A}$,
        initial set $X_0 \subseteq \mathbb{R}^n$ expressed as a QF-LRA formula,
        target set $Y_0 \subseteq \mathbb{R}^n$ expressed as a QF-LRA formula,
        $N \in \mathbb{N}$
Output: boolean
 1: $reach \leftarrow \texttt{false}$
 2: $n \leftarrow \mathrm{Row}(\underline{A})$                    ▷ the number of rows of $\underline{A}$
 3: $\mathcal{V}^{(0)} \leftarrow \mathsf{symb\_var}(0, n)$
 4: $F \leftarrow \emptyset$                                    ▷ empty stack
 5: $F.\mathsf{push}(Y_0)$                           ▷ $Y_0$ is defined over $\mathcal{V}^{(0)}$
 6: $k \leftarrow 1$
 7: while $k \leq N$ do
 8:     $\mathcal{A}^{(1-k)} \leftarrow \mathsf{symb\_mat}(\underline{A}, 1-k)$
 9:     $\mathcal{V}^{(-k)} \leftarrow \mathsf{symb\_var}(-k, n)$
10:     $F.\mathsf{push}(\mathtt{Im}(\mathcal{V}^{(-k)}, \mathcal{V}^{(1-k)}))$
11:     $F.\mathsf{push}(\mathtt{M}(\mathcal{A}^{(1-k)}))$
12:     if $\mathsf{check}(\mathsf{mk\_and}(F)) = \mathsf{UNSAT}$ then
13:         break
14:     $X_0.\mathsf{subs}(\mathcal{V}^{(1-k)}, \mathcal{V}^{(-k)})$        ▷ $X_0$ is now defined over $\mathcal{V}^{(-k)}$
15:     $F.\mathsf{push}(X_0)$
16:     if $\mathsf{check}(\mathsf{mk\_and}(F)) = \mathsf{SAT}$ then
17:         $reach \leftarrow \texttt{true}$
18:         break
19:     $F.\mathsf{pop}()$
20:     $k \leftarrow k+1$
21: return $reach$
```

In comparison with Algorithms 3-4, Algorithms 5-6 do not need to generate the PWA system from the underlying IMPL system and to compute explicitly the reach sets. The performance of the SMT-based Algorithms 5-6 critically hinges on the number of constraints (inequalities and equalities) in (26): if both $\underline{A}, \overline{A}$ have $m$ finite elements in each row, then there are $4mn$ constraints in (26). Therefore, excluding the constraints from initial and target sets, there are $4mnN$ constraints in (27) and (28). Indeed, the more constraints in (27) and (28), the slower the running time for Algorithms 5 and 6. In the next Section we shall display the drastic increase in scalability of Algorithms 5-6 over Algorithms 3-4.

With regards to the initial and target sets, the SMT-based algorithms are more general than the explicit ones, which compute reach sets. As we mentioned before, Algorithms 3-4 use DBMs to express the initial and target sets. We have argued that each DBM can be translated into a QF-LRA formula, and Algorithms 5-6 can more generally accept any initial and target sets, as long as they are expressible as formulae in QF-LRA.

### C. Quantified Reachability Analysis of Interval Max-Plus Linear Systems

This subsection discusses extensions of Problem 1 by allowing quantifiers, $\exists$ or $\forall$, over either initial sets or state matrices in (6), thus resulting in four possible new problems. Consequently, reachability analysis is studied by modifying the forward SMT-based RA of IMPL systems (Algorithm 5). These extensions allow to perform *safety analysis* of IMPL systems with respect to uncertainty on the state matrices: for instance, the dynamics of an IMPL system up to bound $N$ never reach the undesired conditions regardless the sequence of state matrices $A_0, \ldots, A_{N-1}$. Let us remark that these extensions (especially those with universal quantifiers) cannot be solved by standard approaches, namely by computing the reach sets: the termination condition in line 6 of Algorithm 3 does not necessarily imply $\mathbf{x}(k) \in Y_0$ for all $\mathbf{x}(0) \in X_0$ nor for all $A_0, \ldots, A_{N-1} \in [\underline{A}, \overline{A}]$. Therefore, the new problem can be uniquely addressed by the proposed SMT-based approaches.

**Problem 2** (Quantified Reachability Analysis)**.** *Suppose we have an IMPL system* (6)*, $X_0, Y_0 \subseteq \mathbb{R}^n$ respectively as the initial and target sets and a positive integer $N$, the bounded quantified reachability analysis refers to one of the following:*

1. $\exists \mathbf{x}(0) \in X_0.\ \exists A_0, \ldots, A_{N-1} \in [\underline{A}, \overline{A}].\ \exists 1 \leq k \leq N.\ \mathbf{x}(k) = A_{k-1} \otimes \ldots \otimes A_0 \otimes \mathbf{x}(0) \in Y_0,$
2. $\forall \mathbf{x}(0) \in X_0.\ \exists A_0, \ldots, A_{N-1} \in [\underline{A}, \overline{A}].\ \exists 1 \leq k \leq N.\ \mathbf{x}(k) = A_{k-1} \otimes \ldots \otimes A_0 \otimes \mathbf{x}(0) \in Y_0,$
3. $\exists \mathbf{x}(0) \in X_0.\ \forall A_0, \ldots, A_{N-1} \in [\underline{A}, \overline{A}].\ \exists 1 \leq k \leq N.\ \mathbf{x}(k) = A_{k-1} \otimes \ldots \otimes A_0 \otimes \mathbf{x}(0) \in Y_0,$
4. $\forall \mathbf{x}(0) \in X_0.\ \forall A_0, \ldots, A_{N-1} \in [\underline{A}, \overline{A}].\ \exists 1 \leq k \leq N.\ \mathbf{x}(k) = A_{k-1} \otimes \ldots \otimes A_0 \otimes \mathbf{x}(0) \in Y_0.$ □

Essentially, the above problems differ on how we quantify the initial vector $\mathbf{x}(0) \in X_0$ and the $N$ state matrices $A_0, \ldots, A_{N-1}$ of (6). Notice that Problem 1 is equivalent to Problem 2.1 and can be considered as the weakest formulation. On the other contrary, Problem 2.4 assesses whether for all $\mathbf{x}(0) \in X_0$ and for all $A_0, \ldots, A_{N-1} \in [\underline{A}, \overline{A}]$ there exists $1 \leq k \leq N$ such that if we compute $\mathbf{x}(1), \ldots, \mathbf{x}(N)$ according to (6) then at least one of these vectors belongs to $Y_0$. It is evident that Problem 2.4 is the strictest one: if the answer for Problem 2.4 is `true`, then so are the others. The implication relation within Problems 2.1-2.4 is illustrated in Figure 3.

$$\begin{tikzcd}
& \text{Problem 2.4} \arrow[dl] \arrow[dr] \arrow[dd] & \\
\text{Problem 2.3} \arrow[dr] & & \text{Problem 2.2} \arrow[dl] \\
& \text{Problem 2.1} &
\end{tikzcd}$$

Fig. 3. Logical relations amongst Problem 2. An arrow corresponds to a logical implication.

**Remark 10.** *There are two additional possible orders of quantifiers for Problem 2, namely*

$$\exists A_0, \ldots, A_{N-1} \in [\underline{A}, \overline{A}].\ \forall \mathbf{x}(0) \in X_0.\ \exists 1 \leq k \leq N.$$
$$\mathbf{x}(k) = A_{k-1} \otimes \ldots \otimes A_0 \otimes \mathbf{x}(0) \in Y_0,$$

*and*

$$\forall A_0, \ldots, A_{N-1} \in [\underline{A}, \overline{A}].\ \exists \mathbf{x}(0) \in X_0.\ \exists 1 \leq k \leq N.$$
$$\mathbf{x}(k) = A_{k-1} \otimes \ldots \otimes A_0 \otimes \mathbf{x}(0) \in Y_0.$$

*This work does not discuss these problems, because it does not make practical sense to choose state matrices $A_0, \ldots, A_{N-1}$ prior to the initial vectors $\mathbf{x}(0)$.*

*The backward formulation of Problems 2.1-2.4*

$$Q_1(\mathbf{y} \in Y_0).\ Q_2(A_0, \ldots, A_{N-1} \in [\underline{A}, \overline{A}]).\ \exists \mathbf{x}(0) \in X_0.$$
$$\exists 1 \leq k \leq N.\ \mathbf{y} = A_{k-1} \otimes \ldots A_0 \otimes \mathbf{x}(0),$$

*where the quantifiers $Q_1, Q_2 \in \{\exists, \forall\}$, is also not discussed in this work, because it is really unusual to quantify the target set for a reachability problem.* □

**Example 6.** *With regards to the IMPL system in Example 1 and the initial and target sets in Example 5, we already know that the result for Problem 2.1 is* `false`*. Therefore, the output for other problems is also* `false`*.*

*Now, consider two new initial and target sets, namely $X_0 = \{\mathbf{x} \in \mathbb{R}^2 \mid x_1 - x_2 \geq 3\}, Y_0 = \{\mathbf{x} \mid \mathbb{R}^2 \mid x_1 - x_2 = -2\}$ and $N = 3$. We will show that the answer for Problems 2.1 and 2.2 is* `true` *while for the remaining ones the output is* `false`*. By taking*

$$A_0 = A_1 = A_2 = \begin{bmatrix} 1 & 3 \\ 3 & 5 \end{bmatrix},$$

*it is evident that for all $\mathbf{x}(0) \in X_0$ we have $A_0 \otimes \mathbf{x}(0) \in Y_0$ (notice that both columns of $A_0$ belong to $Y_0$). Thus, the output for Problem 2.2 is* `true`*, and then so is Problem 2.1.*

*On the other hand, by taking $A_0 = A_1 = A_2 = \underline{A}$, one could show that for all $\mathbf{x}(0) = [x_1\ x_2] \in X_0$, the trajectories of* (8) *up to $k = 3$ are*

$$\mathbf{x}(1) = \begin{bmatrix} x_1 + 1 \\ x_1 + 2 \end{bmatrix}, \mathbf{x}(2) = \begin{bmatrix} x_1 + 5 \\ x_1 + 5 \end{bmatrix}, \mathbf{x}(3) = \begin{bmatrix} x_1 + 8 \\ x_1 + 8 \end{bmatrix}.$$

*It is clear that $\mathbf{x}(k) \notin Y$ for $k = 1, 2, 3$ (even for $k > 3$). Thus, the negation of Problem 2.3 holds. As a result, the output for Problems 2.3 and 2.4 is* `false`*.* □

We argue that, in general, Problems 2.2-2.4 cannot be solved explicitly by computing forward reach sets. Indeed, if Problem 2.1 (which can be solved by Algorithm 3) does not hold, then so do not the other problems. If instead the output is `true`, then the condition on line 6 of Algorithm 3 is fulfilled. However, regardless the corresponding state matrices $A_0, \ldots, A_{k-1}$, we cannot conclude whether it is satisfied by all vectors $\mathbf{x}(0) \in X_0$. In general, given two different vectors $\mathbf{x}(0), \mathbf{y}(0) \in X_0$ which eventually reach a target set $Y_0$, it is possible that they "enter" $Y_0$ with: i) different time horizons, $\mathbf{x}(k_1) \in Y_0$ and $\mathbf{y}(k_2) \in Y_0$ with $k_1 \neq k_2$, or ii) different state matrices.

Before describing the procedure to solve Problems 2.1-2.4, we will explain how to generate a quantified formula from quantifier-free ones. Given quantifier-free formulae $F_1$ and $F_2$ over variables $\mathtt{x}_1, \ldots, \mathtt{x}_n$, to express that all satisfying assigments of $F_1$ also satisfy $F_2$ we could write $\forall \mathtt{x}_1, \ldots, \mathtt{x}_n\ (F_1 \rightarrow F_2)$. On the other hand, to say that there exists one assignment for $F_1$ that also satisfies $F_2$, we can write $\exists \mathtt{x}_1, \ldots, \mathtt{x}_n\ (F_1 \wedge F_2)$.

Now we show how to express Problems 2.1-2.4 as LRA formulae. Recall that the quantifier-free expression for Problems 2.1-2.4 can be formulated as (27). The assertion "$\exists k, 1 \leq k \leq$

$N$, such that $\mathbf{x}(k) \in Y$", present in Problems 2.1-2.4, can be expressed as a sub-formula of (26), as

$$F \equiv \left( \bigwedge_{k=1}^{N} \mathtt{Im}(\mathcal{V}^{(k-1)}, \mathcal{V}^{(k)}) \right) \wedge \left( \bigvee_{k=1}^{N} Y^{(k)} \right). \tag{29}$$

Building on this, we can add quantifiers for the respective symbolic variables $\mathcal{V}^{(0)}, \ldots, \mathcal{V}^{(N)}$ and symbolic matrices $\mathcal{A}^{(0)}, \ldots, \mathcal{A}^{(N)}$.

The quantifier for $\mathcal{V}^{(0)}$ is restricted by the formula $X^{(0)}$, while the quantifiers for state matrices $\mathcal{A}^{(0)}, \ldots, \mathcal{A}^{(N)}$ are restricted by the formula $\mathtt{M}(\mathcal{A}^{(0)}) \wedge \ldots \wedge \mathtt{M}(\mathcal{A}^{(N-1)})$. For $k \geq 1$, the set of variables $\mathcal{V}^{(k)}$ is always preceded by quantifier $\exists$ because the vector $\mathbf{x}(k)$ is uniquely determined by the chosen initial vector $\mathbf{x}(0)$ and state matrices $A_0, \ldots, A_{k-1}$. Furthermore, the location for their quantifiers will be on the last ones. Thus, Problems 2.1 and 2.2 can be respectively formulated as

$$\exists \mathcal{V}^{(0)} \exists \mathcal{A}^{[N]} \exists \mathcal{V}^{[N]} \left( X^{(0)} \wedge \bigwedge_{k=0}^{N-1} \mathtt{M}(A^{(k)}) \wedge F \right), \tag{30a}$$

$$\forall \mathcal{V}^{(0)} \exists \mathcal{A}^{[N]} \exists \mathcal{V}^{[N]} \left( X^{(0)} \rightarrow \left( \bigwedge_{k=0}^{N-1} \mathtt{M}(A^{(k)}) \wedge F \right) \right), \tag{30b}$$

where $\exists \mathcal{A}^{[N]}$ and $\exists \mathcal{V}^{[N]}$ are respectively the abbreviation of $\exists \mathcal{A}^{(0)} \ldots \exists \mathcal{A}^{(N-1)}$ and $\exists \mathcal{V}^{(1)} \ldots \exists \mathcal{V}^{(N)}$.

For the remaining two Problems 2.3-2.4 where the state matrices are universally quantified, we express their negation instead. This will positively affect the performance of the procedure, as it is known that the fewer are the universal quantifiers in a quantified formula, the faster it is for an SMT-solver to check its satisfiability. For instance, the negation for Problem 2.3 is $\forall \mathbf{x}(0) \in X_0.\ \exists A_0, \ldots, A_{N-1} \in [\underline{A}, \overline{A}].\ \forall 1 \leq k \leq N.\ \mathbf{x}(k) = A_{k-1} \otimes \ldots \otimes A_0 \otimes \mathbf{x}(0) \notin Y$. The quantifier-free expression of this statement can be expressed as

$$G \equiv \left( \bigwedge_{k=1}^{N} \mathtt{Im}(\mathcal{V}^{(k-1)}, \mathcal{V}^{(k)}) \right) \wedge \left( \bigwedge_{k=1}^{N} \neg Y^{(k)} \right). \tag{31}$$

As a result, the negation of Problems 2.3 and 2.4 can be respectively formulated as

$$\forall \mathcal{V}^{(0)} \exists \mathcal{A}^{[N]} \exists \mathcal{V}^{[N]} \left( X^{(0)} \rightarrow \left( \bigwedge_{k=0}^{N-1} \mathtt{M}(A^{(k)}) \wedge G \right) \right), \tag{32a}$$

$$\exists \mathcal{V}^{(0)} \exists \mathcal{A}^{[N]} \exists \mathcal{V}^{[N]} \left( X^{(0)} \wedge \bigwedge_{k=0}^{N-1} \mathtt{M}(A^{(k)}) \wedge G \right). \tag{32b}$$

Notice that, similar to (30b), there is only one universal quantifier in (32a). Similarly, the formula (32b) does not contain any universal quantifier: just like in (30a).

Algorithm 7 summarises the steps to solve Problems 2.1-2.4. At the start, we generate empty stacks that will contain the formula encoding the trajectories ($Trj$) and the interval matrices ($I$) of the IMPL system and of the target set ($Trg$). Other stacks, $Var$ and $Mat$, correspond to the symbolic variables and matrices. In lines 5-11, we collect the symbolic variables and matrices in (26) and introduce the LRA formulae for (26). Then, in lines 13-16, a quantifier-free formula w.r.t. (29) or (31) is generated: the command $\mathsf{mk\_or}(Trg)$ stands for $\vee_{f \in Trg} f$. The quantifier formula corresponding to one of four formulae in (30a)-(30b) or (32a)-(32b) is then generated in lines 17-20. Finally, we check the satisfaction of the resulting formula using an SMT solver. The result is expressed as a boolean variable $res$, which initialised as $\mathtt{false}$. If the SMT solver reports "satisfiable" then $res$ is changed to $\mathtt{true}$.

The output of the quantified reachability analysis depends on the type of the problem (represented by an integer $type \in \{1, 2, 3, 4\}$) and boolean variable $res$. If $1 \leq type \leq 2$, this corresponds to one of Problems 2.1 and 2.2, then the value of $res$ is the output (line 25). Conversely, recalling that for Problems 2.3-2.4, the procedure in Algorithm 7 checks the satisfaction of their negation, if $3 \leq type \leq 4$, then the negation of $res$ becomes the output instead (line 27).

**Algorithm 7** Quantified RA of IMPL systems

```
Inputs: $\mathbf{A} = [\underline{A}, \overline{A}]$ where $\underline{A}, \overline{A} \in \mathbb{R}_{\max}^{n \times n}$ with $\underline{A} \leq \overline{A}$,
        initial set $X_0 \subseteq \mathbb{R}^n$ expressed as a QF-LRA formula,
        target set $Y_0 \subseteq \mathbb{R}^n$ expressed as a QF-LRA formula,
        $N \in \mathbb{N}$,
        $type \in \{1 \ldots, 4\}$
Output: boolean
 1: $n \leftarrow$ ROW($\underline{A}$)                          ▷ the number of rows of $\underline{A}$
 2: $Traj, Trg, I, Var, Mat \leftarrow \emptyset$             ▷ empty stacks
 3: $\mathcal{V}^{(0)} \leftarrow \mathsf{symb\_var}(0, n)$
 4: for $1 \leq k \leq N$ do
 5:     $Mat.\mathsf{push}(\mathcal{A}^{(k-1)})$                ▷ symbolic matrices
 6:     $\mathcal{V}^{(k)} \leftarrow \mathsf{symb\_var}(k, n)$
 7:     $Var.\mathsf{push}(\mathcal{V}^{(k)})$                  ▷ symbolic variables
 8:     $I.\mathsf{push}(\mathtt{M}(\mathcal{A}^{(k-1)}))$
 9:     $Traj.\mathsf{push}(\mathtt{Im}(\mathcal{V}^{(k-1)}, \mathcal{V}^{(k)}))$
10:     $Y_0.\mathsf{subs}(\mathcal{V}^{(k-1)}, \mathcal{V}^{(k)})$
11:     $Trg.\mathsf{push}(Y_0)$
12: $F \leftarrow \mathsf{mk\_and}(Traj)$
13: if $(type = 1 \vee type = 2)$ then
14:     $F \leftarrow F \wedge \mathsf{mk\_or}(Trg)$            ▷ formula in (29)
15: else if $(type = 3 \vee type = 4)$ then
16:     $F \leftarrow F \wedge \neg\mathsf{mk\_or}(Trg)$        ▷ formula in (31)
17: if $(type = 1 \vee type = 4)$ then
18:     $F \leftarrow \exists \mathcal{V}^{(0)}.\ \exists Mat.\ \exists Var.\ (X_0 \wedge \mathsf{mk\_and}(I) \wedge F)$
19: else if $(type = 2 \vee type = 3)$ then
20:     $F \leftarrow \forall \mathcal{V}^{(0)}.\ \exists Mat.\ \exists Var.\ (X_0 \rightarrow (\mathsf{mk\_and}(I) \wedge F))$
21: $res \leftarrow \mathtt{false}$
22: if check($F$) = SAT then
23:     $res \leftarrow \mathtt{true}$
24: if $(type = 1 \vee type = 2)$ then
25:     return $res$
26: else if $(type = 3 \vee type = 4)$ then
27:     return $\neg res$
```

The performance of Algorithm 7 crucially depends on the SMT-checking in line 22. As mentioned before, more universal quantifiers in a (quantified) formula result in a slower SMT solver to check its satisfiability. Thus, the running time of solving Problem 2.1 and 2.4 is potentially faster than those of Problems 2.2-2.3: this argument is confirmed in the experiments of the next section (cf. Table V).

## VI. Computational Benchmarks

We compare the performance of the new, SMT-based reachability analysis of IMPL systems presented in this paper with the explicit approaches in [14]. The experiments for both procedures are implemented in C++. For the SMT solver, we use Z3 [18]. We use Armadillo [29] to enable matrix operations in max-plus algebra. The computational benchmark has been implemented on an Intel® Xeon® CPU E5-1660 v3, 16 cores, 3.0GHz each, and 16GB of RAM.

We work with pairs $(n, m)$, where $m < n$. For each dimension $n$ (i.e., number of continuous variables), we generate 20 matrices $\underline{A}, \overline{A} \in \mathbb{R}_{\max}^{n\times n}$ (of dimension $n$), with $m$ finite elements in each row, where their values are taken to be between 1 and 20. We recall that the upper and lower matrices $\underline{A}, \overline{A}$ are generated under the assumptions in Remark 1. The locations of the finite elements in each row of are chosen randomly. It is important to note that, across different experiments (Tables I-V), we have re-used the generated matrices for the same pairs of $(n, m)$.

In the first benchmark, we present the comparison of the procedure to compute image and inverse image of DBMs w.r.t. an IMPL system from [13] and the new methods in Algorithms 1-2. For each experiment, we generate the PWA system from the given IMPL system, then compute the image and inverse images of each PWA region w.r.t. its corresponding affine dynamics. Table I summarises the outcomes: the second column shows the average number of PWA regions, while the last four columns depict the average running time of the procedures.

TABLE I
COMPUTATIONAL BENCHMARK FOR IMAGE AND INVERSE IMAGE COMPUTATION OF DBMS W.R.T. IMPL SYSTEMS.

| $(n,m)$ | avg. num. of regions | image comp. | | inverse image comp. | |
|---|---|---|---|---|---|
| | | [13, Alg. 5.1] | Alg. 1 | [13, Alg. 5.2] | Alg. 2 |
| $(4,2)$ | 11.30 | 1.63ms | 0.28ms | 1.74ms | 0.31ms |
| $(4,3)$ | 25.20 | 3.59ms | 0.54ms | 3.76ms | 0.74ms |
| $(5,2)$ | 23.75 | 5.18ms | 0.74ms | 5.58ms | 0.89ms |
| $(5,3)$ | 74.35 | 14.38ms | 2.27ms | 14.39ms | 2.50ms |
| $(6,2)$ | 45.00 | 15.23ms | 1.46ms | 13.16ms | 1.78ms |
| $(6,3)$ | 194.15 | 51.47ms | 8.36ms | 51.83ms | 9.64ms |
| $(7,2)$ | 89.40 | 36.75ms | 4.81ms | 35.11ms | 4.81ms |
| $(7,3)$ | 505.35 | 98.94ms | 14.92ms | 99.36ms | 16.49ms |
| $(8,2)$ | 179.80 | 88.15ms | 9.25ms | 90.81ms | 12.55ms |
| $(8,3)$ | 1465.40 | 265.20ms | 35.68ms | 271.74ms | 41.34ms |

It is clear that the new procedures to compute the image and inverse image of DBMs w.r.t. an IMPL system via Algorithms 1-2, outperform the existing ones in [13]. Although all algorithms have cubic complexity, our proposed algorithms are indeed more efficient as they only involve $n + 1$ variables $x_0, x_1, \ldots, x_n$. In comparison, the existing procedures in [13, Alg. 5.1] and [13, Alg. 5.2] employ additional variables $x'_1, x'_2, \ldots, x'_n$.

Next, we compare the performance of the explicit reachability analysis of IMPL systems via Algorithms 3-4 to that of SMT-based RA by Algorithms 5-6. We select $X_0 = \{\mathbf{x} \in \mathbb{R}^n \mid 0 \leq x_1 \leq \ldots \leq x_p \leq 1\}$ and $Y = \{\mathbf{x} \in \mathbb{R}^n \mid x_1 > \ldots > x_p, x_1 - x_p > 20\}$ with $p = \lceil \frac{n}{2} \rceil$ as the set of initial and target sets, respectively. The experiments have been implemented to perform the reachability analysis over $N = 10$ steps. We set 30 minutes as a *timeout* condition: if a procedure runs more than a half hour then it will be terminated.

Table II (resp. Table III) shows the average and maximum running time from 20 experiments of the reachability analysis of IMPL systems via Algorithms 3 and 4 (resp. Algorithms 5 and 6). The last column of Tables II and III report the number of experiments, out of 20, with a `true` outcome. The notation "timeout $(r)$" corresponds to "there are $r$ experiments whose running time exceed 30 minutes".

TABLE II
COMPUTATIONAL BENCHMARK FOR EXPLICIT REACHABILITY ANALYSIS OF IMPL SYSTEMS.

| $(n,m)$ | runing times | | #`true` |
|---|---|---|---|
| | Algorithm 3 | Algorithm 4 | |
| $(4,2)$ | $\{0.32, 2.09\}$s | $\{0.02, 0.09\}$s | 11 |
| $(4,3)$ | $\{1.33, 16.17\}$s | $\{0.03, 0.12\}$s | 10 |
| $(5,2)$ | $\{0.38, 3.16\}$s | $\{0.04, 0.16\}$s | 14 |
| $(5,3)$ | $\{36.54, 522.13\}$s | $\{0.19, 0.47\}$s | 10 |
| $(6,2)$ | $\{62.11, 1238.01\}$s | $\{0.10, 0.15\}$s | 18 |
| $(6,3)$ | timeout $(1)$ | $\{1.11, 2.56\}$s | 15 |
| $(7,2)$ | $\{69.49, 1091.54\}$s | $\{17.89, 226.45\}$s | 16 |
| $(7,3)$ | timeout $(4)$ | $\{10.55, 87.96\}$s | 14 |
| $(8,2)$ | timeout $(2)$ | timeout $(2)$ | - |
| $(8,3)$ | timeout $(1)$ | timeout $(1)$ | - |

TABLE III
COMPUTATIONAL BENCHMARK FOR SMT-BASED REACHABILITY ANALYSIS OF IMPL SYSTEM.

| $(n,m)$ | running times | | #`true` |
|---|---|---|---|
| | Algorithm 5 | Algorithm 6 | |
| $(4,2)$ | $\{0.02, 0.05\}$s | $\{0.01, 0.03\}$s | 11 |
| $(4,3)$ | $\{0.02, 0.05\}$s | $\{0.01, 0.08\}$s | 10 |
| $(5,2)$ | $\{0.01, 0.03\}$s | $\{0.01, 0.02\}$s | 14 |
| $(5,3)$ | $\{0.03, 0.06\}$s | $\{0.01, 0.09\}$s | 10 |
| $(6,2)$ | $\{0.01, 0.07\}$s | $\{0.01, 0.02\}$s | 18 |
| $(6,3)$ | $\{0.02, 0.06\}$s | $\{0.01, 0.02\}$s | 15 |
| $(7,2)$ | $\{0.01, 0.06\}$s | $\{0.01, 0.13\}$s | 16 |
| $(7,3)$ | $\{0.02, 0.07\}$s | $\{0.01, 0.02\}$s | 14 |
| $(8,2)$ | $\{0.01, 0.06\}$s | $\{0.01, 0.11\}$s | 17 |
| $(8,3)$ | $\{0.01, 0.08\}$s | $\{0.01, 0.02\}$s | 19 |
| $(9,2)$ | $\{0.01, 0.06\}$s | $\{0.02, 0.03\}$s | 17 |
| $(9,3)$ | $\{0.02, 0.18\}$s | $\{0.02, 0.33\}$s | 18 |
| $(10,2)$ | $\{0.01, 0.06\}$s | $\{0.01, 0.02\}$s | 17 |
| $(10,3)$ | $\{0.02, 0.21\}$s | $\{0.02, 0.34\}$s | 18 |

As one can see comparing Tables II and III, the SMT-based algorithms are significantly faster than those that explicitly compute the reach sets. With regards to the comparison between the forward and backward approaches for the explicit RA of IMPL systems (Algorithms 3 and 4), the latter seems to be faster. This is likely due to the break condition in line 8 of Algorithm 4, which causes the backward procedure to terminate earlier than the specified step bound $N$. On the other hand, due to the "almost constant" running times in Table III, we cannot conclude that Algorithm 6 is faster than Algorithm 5: this suggest to challenge both approaches over IMPL systems with drastically larger dimensions, which is pursued next.

Table IV presents a benchmark for SMT-based reachability analysis for high-dimensional IMPL systems. It is now clear that that the backward approach is faster then the forward one. As the dimension increases, we see an increasing gap on the average and maximum running time. Additionally, based on our experiments, Algorithm 6 has an advantage when the experiment yields `false`, which is owed to the break condition in line 13 of Algorithm 6.

TABLE IV
COMPUTATIONAL BENCHMARK FOR SMT-BASED REACHABILITY ANALYSIS OF HIGH-DIMENSIONAL IMPL SYSTEMS.

| $(n, m)$ | running times | | #`true` |
|---|---|---|---|
| | Algomrithm 5 | Algorith 6 | |
| $(20, 10)$ | $\{0.25, 1.59\}$s | $\{0.05, 0.18\}$s | 17 |
| $(30, 15)$ | $\{1.07, 12.01\}$s | $\{0.41, 2.54\}$s | 18 |
| $(40, 20)$ | $\{6.34, 51.68\}$s | $\{0.81, 2.68\}$s | 14 |
| $(50, 25)$ | $\{6.84, 79.30\}$s | $\{1.14, 7.08\}$s | 18 |
| $(60, 30)$ | $\{33.27, 112.95\}$s | $\{1.65, 4.78\}$s | 13 |
| $(70, 35)$ | $\{12.02, 213.59\}$s | $\{2.15, 3.78\}$s | 19 |
| $(80, 40)$ | timeout $(1)$ | $\{5.24, 19.81\}$s | 14 |
| $(90, 45)$ | - | $\{5.84, 9.69\}$s | 14 |
| $(100, 50)$ | - | $\{9.78, 21.76\}$s | 14 |
| $(110, 55)$ | - | $\{12.27, 22.78\}$s | 18 |
| $(120, 60)$ | - | $\{28.09, 104.85\}$s | 16 |
| $(130, 65)$ | - | $\{55.88, 441.55\}$s | 15 |
| $(140, 70)$ | - | $\{81.46, 333.28\}$s | 18 |
| $(150, 75)$ | - | $\{123.04, 427.03\}$s | 17 |
| $(160, 80)$ | - | $\{180.46, 1090.61\}$s | 18 |

Table IV also suggests that the SMT-based RA of IMPL system (Algorithms 5-6) is much more scalable than explicit RA (Algorithms 3-4). For $(n, m) = (70, 35)$, the average running time for Algorithms 5 and 6 is just over 12 and 2 second, respectively. It seems that the forward approach cannot be pushed to really high dimensions. Because we get a timeout experiment for Algorithm 5 with $(n, m) = (80, 40)$, we continue the study with Algorithm 6 only. The average running time of Algorithm 6 is about 3 minutes for $(n, m) = (160, 80)$: this suggests that the new approach can be practically applicable to models of industrial scale.

For the last benchmark, we show the average running time of Algorithm 7. It is evident that the presence of universal quantifiers heavily burdens the performance of Algorithm 7. Thanks to formula (32b), checking the satisfaction of Problem 2.1 is as fast as that of Problem 2.4. Notice that the number of experiments with `true` output for Problem 2.1 (sixth column of Table V) is the same with that of Problem 1 (last column of Table III). On the other hand, for Problem 2.4, there is no single experiment with `true` output, because of its more restrictive requirements.

TABLE V
COMPUTATIONAL BENCHMARK FOR QUANTIFIED REACHABILITY ANALYSIS OF IMPL SYSTEMS.

| $(n, m)$ | running times | | | | #`true` | | | |
|---|---|---|---|---|---|---|---|---|
| | P1 | P2 | P3 | P4 | P1 | P2 | P3 | P4 |
| $(4, 2)$ | 0.02s | 0.07s | 0.09s | 0.02s | 11 | 6 | 7 | 0 |
| $(4, 3)$ | 0.03s | 0.06s | 0.11s | 0.03s | 10 | 3 | 8 | 0 |
| $(5, 2)$ | 0.03s | 0.18s | 0.19s | 0.03s | 14 | 4 | 1 | 0 |
| $(5, 3)$ | 0.04s | 0.14s | 0.30s | 0.04s | 10 | 7 | 3 | 0 |
| $(6, 2)$ | 0.04s | 0.93s | 0.69s | 0.03s | 18 | 15 | 9 | 0 |
| $(6, 3)$ | 0.06s | 0.69s | 1.25s | 0.05s | 15 | 6 | 2 | 0 |
| $(7, 2)$ | 0.05s | 0.96s | 1.11s | 0.05s | 16 | 10 | 1 | 0 |
| $(7, 3)$ | 0.08s | 1.24s | 2.91s | 0.07s | 14 | 8 | 1 | 0 |
| $(8, 2)$ | 0.06s | 11.36s | 10.72s | 0.05s | 17 | 11 | 5 | 0 |
| $(8, 3)$ | 0.11s | 18.66s | 19.52s | 0.09s | 19 | 11 | 6 | 0 |
| $(9, 2)$ | 0.08s | 18.17s | 33.58s | 0.07s | 17 | 12 | 1 | 0 |
| $(9, 3)$ | 0.14s | 66.74s | 81.31s | 0.10s | 18 | 14 | 0 | 0 |
| $(10, 2)$ | 0.10s | timeout $(2)$ | timeout $(2)$ | 0.09s | 17 | - | - | 0 |
| $(10, 3)$ | 0.16s | 739.59s | timeout $(1)$ | 0.13s | 18 | 15 | - | 0 |

## VII. CONCLUSIONS

This paper has introduced a new, SMT-based approach to solve reachability problems over IMPL systems based on SMT solving. The procedure has been tested on computational benchmarks, which have shown a significant improvement over existing explicit reachability techniques. Furthermore, the procedure is scalable as it allows to perform reachability analysis of very high-dimensional IMPL systems.

## REFERENCES

[1] A. Abate, C. David, P. Kesseli, D. Kroening, and E. Polgreen. Counterexample guided inductive synthesis modulo theories. In *Proceedings of CAV, LNCS 10981*, pages 270–288, 2018.

[2] D. Adzkiya, B. De Schutter, and A. Abate. Finite abstractions of max-plus-linear systems. *IEEE Transactions on Automatic Control*, 58(12):3039–3053, 2013.

[3] D. Adzkiya, B. De Schutter, and A. Abate. Backward reachability of autonomous max-plus-linear systems. *WODES, IFAC Proceedings Volumes*, 47(2):117–122, 2014.

[4] D. Adzkiya, B. De Schutter, and A. Abate. Forward reachability computation for autonomous max-plus-linear systems. In E. Abraham and K. Havelund, editors, *Intl. Conf. on Tools and Algorithms for the Construction and Analysis of Systems* (TACAS'14), volume 8413 of *LNCS*, pages 248–262. Springer, 2014.

[5] D. Adzkiya, B. D. Schutter, and A. Abate. Computational techniques for reachability analysis of max-plus-linear systems. *Automatica*, 53(3):293–302, 2015.

[6] D. Ahmed, A. Peruffo, and A. Abate. Automated and sound synthesis of Lyapunov Functions with SMT solvers. In *Proceedings of TACAS, LNCS 12078*, pages 97–114, 2020.

[7] F. Baccelli, G. Cohen, G. J. Olsder, and J.-P. Quadrat. *Synchronization and linearity: an algebra for discrete event systems*. John Wiley & Sons Ltd, 1992.

[8] C. Barrett, L. De Moura, and A. Stump. SMT-COMP: Satisfiability modulo theories competition. In K. Etessami and S. K. Rajamani, editors, *Intl. Conf. on Computer Aided Verification* (CAV'05), volume 3576 of *LNCS*, pages 20–23. Springer, 2005.

[9] C. Barrett, R. Sebastiani, S. A. Seshia, and C. Tinelli. Satisfiability modulo theories. In A. Biere, H. van Maaren, and T. Walsh, editors, *Handbook of Satisfiability*, volume 4, chapter 8. IOS Press, 2009.

[10] C. Barrett, A. Stump, and C. Tinelli. The satisfiability modulo theories library, 2010. http://smtlib.cs.uiowa.edu.

[11] C. Barrett and C. Tinelli. Satisfiability modulo theories. In *Handbook of Model Checking*, pages 305–343. Springer, 2018.

[12] C. A. Brackley, D. S. Broomhead, M. C. Romano, and M. Thiel. A max-plus model of ribosome dynamics during mRNA translation. *Journal of Theoretical Biology*, 303:128–140, 2012.

[13] R. M. F. Cândido. *Reachability Analysis of Uncertain Max Plus Linear Systems*. PhD thesis, Universitè d'Angers, 2017.

[14] R. M. F. Cândido, L. Hardouin, M. Lhommeau, and R. S. Mendes. Conditional reachability of uncertain max plus linear systems. *Automatica*, 94:426–435, 2018.

[15] A. Cimatti, A. Griggio, B. J. Schaafsma, and R. Sebastiani. The MATHSAT5 SMT solver. In N. Piterman and S. A. Smolka, editors, *Intl. Conf. on Tools and Algorithms for the Construction and Analysis of Systems* (TACAS'13), volume 7795 of *LNCS*, pages 93–107. Springer, 2013.

[16] J.-P. Comet. Application of max-plus algebra to biological sequence comparisons. *Theoretical computer science*, 293(1):189–217, 2003.

[17] S. Cotton, E. Asarin, O. Maler, and P. Niebert. Some progress in satisfiability checking for difference logic. In *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, pages 263–276. Springer, 2004.

[18] L. De Moura and N. Bjørner. Z3: An efficient SMT solver. In C. R. Ramakrishnan and J. Rehof, editors, *Intl. Conf. on Tools and Algorithms for the Construction and Analysis of Systems* (TACAS'08), volume 4963 of *LNCS*, pages 337–340. Springer, 2008.

[19] D. L. Dill. Timing assumptions and verification of finite-state concurrent systems. In J. Sifakis, editor, *Proc. International Conference on Computer Aided Verification*, volume 407 of *Lecture Notes in Computer Science*, pages 197–212, Hiedelberg, 1989. Springer.

[20] B. Dutertre. Yices 2.2. In *Intl. Conf. on Computer Aided Verification* (CAV'14), volume 8559 of *LNCS*, pages 737–744, 2014.

[21] W. Heemels, B. De Schutter, and A. Bemporad. Equivalence of hybrid dynamical models. *Automatica*, 37(7):1085–1091, July 2001.

[22] B. Heidergott, G. J. Olsder, and J. Van der Woude. *Max Plus at work: modeling and analysis of synchronized systems: a course on Max-Plus algebra and its applications*. Princeton University Press, 2014.

[23] A. Imaev and R. P. Judd. Hierarchial modeling of manufacturing systems using max-plus algebra. In *Proc. American Control Conference, 2008*, pages 471–476, June 2008.

[24] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer. Reluplex: An efficient SMT solver for verifying deep neural networks. In R. Majumdar and V. Kunčak, editors, *Intl. Conf. on Computer Aided Verification* (CAV'17), volume 10426 of *LNCS*, pages 97–117. Springer, 2017.

[25] D. Kroening and O. Strichman. *Decision procedures*. Springer, 2016.

[26] Y. Li, A. Albarghouthi, Z. Kincaid, A. Gurfinkel, and M. Chechik. Symbolic optimization with SMT solvers. In *ACM SIGPLAN Notices*, volume 49, pages 607–618. ACM, 2014.

[27] M. Mufid, D. Adzkiya, and A. Abate. Tropical abstractions of max-plus linear systems. In D. Jansen and P. Prabhakar, editors, *Int. Conf. Formal Modeling and Analysis of Timed Systems* (FORMATS'18), pages 271–287. Springer, 2018.

[28] M. S. Mufid, D. Adzkiya, and A. Abate. Symbolic reachability analysis of high dimensional max-plus linear systems. *IFAC-PapersOnLine*, 53(4):459–465, 2020. 15th IFAC Workshop on Discrete Event Systems WODES 2020 — Rio de Janeiro, Brazil, 11-13 November 2020.

[29] C. Sanderson and R. Curtin. Armadillo: a template-based C++ library for linear algebra. *Journal of Open Source Software*, 1(2):26, 2016.

[30] Y. Shang, L. Hardouin, M. Lhommeau, and C. A. Maia. Robust controllers in disturbance decoupling of uncertain max-plus linear systems: an application to a high throughput screening system for drug discovery. In *International Workshop on Discrete Event Systems* (WODES'16), pages 404–409. IEEE, 2016.

[31] M. R. Shoaei, L. Kovács, and B. Lennartson. Supervisory control of discrete-event systems via IC3. In *Haifa Verification Conference*, pages 252–266. Springer, 2014.

[32] E. Sontag. Nonlinear regulation: The piecewise linear approach. *IEEE Transactions on automatic control*, 26(2):346–358, 1981.

[33] S. E. Z. Soudjani, D. Adzkiya, and A. Abate. Formal verification of stochastic max-plus-linear systems. *IEEE Transactions on Automatic Control*, 61(10):2861 – 2876, 2016.

[34] M. Witczak, P. Majdzik, R. Stetter, and G. Bocewicz. Interval max-plus fault-tolerant control under resource conflicts and redundancies: application to the seat assembly. *International Journal of Control*, pages 1–13, 2019.

[35] J. Xu, T. van den Boom, and B. De Schutter. Model predictive control for stochastic max-plus linear systems with chance constraints. *IEEE Transactions on Automatic Control*, 64(1):337–342, 2018.

**Muhammad Syifa'ul Mufid** received the B.Sc. degree in September 2012 and the M.Sc. degree in October 2013, both in Mathematics from the Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia. He is currently a DPhil student in Department of Computer Science, University of Oxford, United Kingdom.

His research interests are in the analysis and verification of max-plus-linear systems and in their applications.

**Dieky Adzkiya** is an Assistant Professor in the Department of Mathematics and a member of Mechatronics and Industrial Automation Research Center, both at Institut Teknologi Sepuluh Nopember, Indonesia. He received the B.Sc. degree in September 2005 and the M.Sc. degree in October 2008, both in Mathematics from the Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia. He received the Ph.D. degree in Systems and Control in October 2014 and after that he continued as a postdoctoral researcher until June 2015, both at the Delft Center for Systems and Control, Delft University of Technology, Delft, The Netherlands. His research interests are in the analysis and verification of max-plus-linear systems and in their applications.

**Alessandro Abate** (S'02-M'08-SM'19) is Professor of Verification and Control in the Department of Computer Science at the University of Oxford, and a fellow of the Alan Turing Institute for Data Sciences in London. He received a Laurea in Electrical Engineering in October 2002 from the University of Padova (IT), an MS in May 2004 and a PhD in December 2007, both in Electrical Engineering and Computer Sciences, at UC Berkeley (USA). He has been an International Fellow in the CS Lab at SRI International in Menlo Park (USA), and a PostDoctoral Researcher at Stanford University (USA), in the Department of Aeronautics and Astronautics. He has also been an Assistant Professor at the Delft Centre for Systems and Control, TU Delft (NL).