Symbolic Reachability Analysis of High Dimensional Max-Plus Linear Systems

Muhammad Syifa'ul Mufid * **Dieky Adzkiya** **
Alessandro Abate *

* *Department of Computer Science, University of Oxford, UK*
(*e-mail:* {*muhammad.syifaul.mufid,alessandro.abate*}*@cs.ox.ac.uk*)
** *Department of Mathematics, ITS Surabaya, Indonesia*
(*e-mail: dieky@matematika.its.ac.id*)

Abstract: This work discusses the reachability analysis (RA) of Max-Plus Linear (MPL) systems, a class of continuous-space, discrete-event models defined over the max-plus algebra. Given the initial and target sets, we develop algorithms to verify whether there exist trajectories of the MPL system that, starting from the initial set, eventually reach the target set. We show that RA can be solved symbolically by encoding the MPL system, as well as initial and target sets into difference logic, and then checking the satisfaction of the resulting logical formula via an off-the-shelf satisfiability modulo theories (SMT) solver. The performance and scalability of the developed SMT-based algorithms are shown to clearly outperform state-of-the-art RA algorithms for MPL systems, newly allowing to investigate RA of high-dimensional MPL systems: the verification of models with more than 100 continuous variables shows the applicability of these techniques to MPL systems of industrial relevance.

Keywords: max-plus linear systems, reachability analysis, piecewise-affine systems, difference-bound matrices, difference logic, satisfiability modulo theories

1. INTRODUCTION

Max-Plus Linear (MPL) systems are a subclass of discrete-event systems (DES) based on max-plus algebra which uses two binary operations, maximisation and addition. MPL systems are employed to describe synchronization without concurrency, and as such are widely used in transportation (Heidergott et al., 2014) and manufacturing systems (Imaev and Judd, 2008). A fundamental problem for DES is reachability analysis (RA): it investigates whether a certain set of states of the system is attainable from a given set of initial conditions. In the context of MPL systems, RA can be used to determine whether the trajectories of MPL system enter specific conditions that are deemed unsafe: for instance, in a railway network application (Heidergott et al., 2014), whether the delay between two consecutive train departures is ever greater than a given time inteval.

The state-of-the-art approach for RA of MPL systems employs piecewise-affine (PWA) dynamics (Adzkiya et al., 2014b,a, 2015) and generates finite abstractions of MPL systems accordingly (Adzkiya et al., 2013). Forward RA of MPL systems has been discussed in (Adzkiya et al., 2014b). Given an initial set X, it computes the forward image of X w.r.t. the underlying MPL system. Similarly, backward reachability of MPL systems is done by computing the inverse image of target set Y (Adzkiya et al., 2014a), backwards in time. In (Adzkiya et al., 2014b,a), both initial and target sets are assumed to be difference-bound matrices (DBMs) (Dill, 1989) and the MPL dynamics are expressed as PWA models in the event domain.

Whilst the approaches in (Adzkiya et al., 2014b,a, 2015) are scalable much beyond existing results based on simple algebraic operations, it is always desirable to push the envelope and to perform RA for MPL systems with ever larger number of variables (that is, with high continuous dimensions). In (Adzkiya et al., 2014b,a, 2015), PWA systems are characterised by different spatial regions (PWA regions) and corresponding affine dynamics (Sontag, 1981). The translation of MPL systems into PWA dynamics, characterised by spatial regions and corresponding affine dynamics (Sontag, 1981), has an exponential complexity (Adzkiya et al., 2013): the number of PWA regions grows steeply as the dimension of MPL systems and the number of finite entries in the state matrix increase. Furthermore, the forward and backward reach sets are characterised as unions of finitely many DBMs, the number of which grows exponentially w.r.t. the time horizon (Adzkiya et al., 2014a,b).

In order to attain scalability to really large MPL models, this paper proposes a symbolic approach to perform reachability analysis of MPL systems. Instead of computing reach sets explicitly, we use symbolic variables to encode the states of trajectories of MPL systems at each time horizon. Firstly, the MPL system as well as the initial and reach sets are translated into a formula that can be parsed by a satisfiability modulo theory (SMT) solver. An SMT problem deals with the satisfaction of a logical formula w.r.t. a given theory (e.g., linear arithmetics, or bit vectors) (Barrett and Tinelli, 2018). Secondly, the satisfiability of the formula encoding a reachability problem is checked using an SMT solver. If the SMT solver reports

"satisfiable" (resp. "unsatisfiable"), then the target set is reachable (resp. not reachable) from an (resp. any) initial condition within the initial set.

We have implemented the symbolic reachability analysis of MPL systems in C++, using the Z3 SMT solver (De Moura and Bjørner, 2008). According to our numerical benchmark, the symbolic implementation is significantly faster than the state-of-the-art software tool. Furthermore, our implementation can solve the reachability analysis of 100-dimensional MPL systems within reasonable time and memory requirements: these results render RA of MPL systems newly applicable to industrial-sized models.

The paper is structured as follows. Section 2 introduces the basic notions of MPL systems and the brief summary of reachability analysis based on reach sets computation. Section 3 consists of the brief definition of SMT and the main contribution of this paper. The computational benchmarks are provided in Section 4. Finally, we conclude the paper with Section 5.

2. MODEL AND PRELIMINARIES

2.1 Max-Plus Linear Systems

Max-plus algebra is an algebraic structure $(\mathbb{R}_{\max}, \oplus, \otimes)$ where $\mathbb{R}_{\max} := \mathbb{R} \cup \{\varepsilon := -\infty\}$ and

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

for all $a, b \in \mathbb{R}_{\max}$. These operations can be extended to matrices and vectors, as follows:

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

where $A, B \in \mathbb{R}_{\max}^{m\times n}, C \in \mathbb{R}_{\max}^{n\times p}$ and $\alpha \in \mathbb{R}_{\max}$. Given $A \in \mathbb{R}_{\max}^{n\times n}$ and $r \in \mathbb{N}$, $A^{\otimes r}$ denotes $A \otimes \ldots \otimes A$ (r times).

A Max-Plus Linear (MPL) system is defined as

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

where $A \in \mathbb{R}_{\max}^{n\times n}$ is the system matrix and vector $\mathbf{x}(k) = [x_1(k) \ \ldots \ x_n(k)]^\top$ is the state variables (Baccelli et al., 1992). In applications, $\mathbf{x}$ represents the time stamps of the discrete events, while k corresponds to the event counter. Hence, it is more convenient to take $\mathbb{R}^n$ as the state space and A to be a regular matrix, i.e. there exists at least one finite element in each row of A (Heidergott et al., 2014).

Definition 1. (Baccelli et al., 1992). 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 $A(i,j) \neq \varepsilon$. □

The readers are referred to (Baccelli et al., 1992) for more detailed descriptions about $\mathcal{G}(A)$ including the notions of *strongly connected* and *critical circuit*.

Definition 2. (Baccelli et al., 1992). A matrix $A \in \mathbb{R}_{\max}^{n\times n}$ is called *irreducible* if $\mathcal{G}(A)$ is strongly connected. □

Each irreducible matrix $A \in \mathbb{R}_{\max}^{n\times n}$ admits a unique max-plus eigenvalue $\lambda \in \mathbb{R}$, which corresponds to the average weight of critical circuit in $\mathcal{G}(A)$. Furthermore, A satisfies the so-called transient condition:

Proposition 3. (Baccelli et al., 1992). For an irreducible matrix $A \in \mathbb{R}_{\max}^{n\times n}$ and its max-plus eigenvalue $\lambda \in \mathbb{R}$, there exist $k_0, c \in \mathbb{N}$ such that $A^{\otimes(k+c)} = \lambda c \otimes A^{\otimes k}$ for all $k \geq k_0$. The smallest such k_0 and c are called the *transient* and the *cyclicity* of A, respectively. □

2.2 Difference-Bound Matrices

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

Definition 4. (Dill, 1989). A DBM in $\mathbb{R}^n$ is the intersection of sets defined by $x_i - x_j \sim_{i,j} d_{i,j}$, where $\sim_{i,j} \ \in \{>, \geq\}$ and $d_{i,j} \in \mathbb{R}_{\max}$ for $0 \leq i, j \leq n$. The value of special variable x_0 is always equal to 0. □

The variable x_0 is used to represent inequalities with a single variable: $x_i \geq \alpha$ can be written as $x_i - x_0 \geq \alpha$. Unless otherwise stated, in this work we assume that DBM does not contain any inequality with a single variable. Some operations can be applied to DBMs, such as intersection, canonical-form representation, emptiness checking, image and inverse image w.r.t. an affine dynamic (Adzkiya et al., 2013; Mufid et al., 2018).

2.3 Piecewise-Affine Systems

Piecewise-Affine (PWA) systems (Sontag, 1981) are defined by partitioning the input-state space into several domains characterized by polyhedra. Each domain, or PWA region, is associated with an affine function. It is shown in (Heemels et al., 2001) that every MPL system can be transformed into a PWA system. For A in (1), the PWA regions are generated from $\mathbf{g} = (g_1, \ldots, g_n) \in \{1, \ldots, n\}^n$, which satisfies $A(i, g_i) \neq \varepsilon$ for $1 \leq i \leq n$. The region corresponding to $\mathbf{g}$ is

$$\mathsf{R}_{\mathbf{g}} = \bigcap_{i=1}^{n} \bigcap_{j=1}^{n} \{\mathbf{x} \in \mathbb{R}^n | x_{g_i} - x_j \geq A(i,j) - A(i,g_i)\}. \tag{2}$$

Notice that, $\mathsf{R}_{\mathbf{g}}$ is a DBM. The emptiness checking of $\mathsf{R}_{\mathbf{g}}$ can be done using Floyd-Warshall algorithm which has cubic complexity w.r.t. its dimension (Floyd, 1962). The affine dynamics for a non-empty $\mathsf{R}_{\mathbf{g}}$ is

$$x_i(k) = x_{g_i}(k-1) + A(i, g_i), \quad i = 1, \ldots, n. \tag{3}$$

Example 5. Consider a 2×2 MPL system (1) where

$$A = \begin{bmatrix} 2 & 5 \\ 3 & 3 \end{bmatrix}. \tag{4}$$

The resulting PWA regions are $\mathsf{R}_{(1,1)} = \{\mathbf{x} \in \mathbb{R}^2 \mid x_1 - x_2 \geq 3\}$, $\mathsf{R}_{(2,1)} = \{\mathbf{x} \in \mathbb{R}^2 \mid 0 \leq x_1 - x_2 \leq 3\}$, and $\mathsf{R}_{(2,2)} = \{\mathbf{x} \in \mathbb{R}^2 \mid x_1 - x_2 \leq 0\}$. The corresponding affine dynamics is

$$\begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} = \begin{cases} \begin{bmatrix} x_1(k-1) + 2 \\ x_1(k-1) + 3 \end{bmatrix}, & \text{if } \mathbf{x}(k-1) \in \mathsf{R}_{(1,1)}, \\ \begin{bmatrix} x_2(k-1) + 5 \\ x_1(k-1) + 3 \end{bmatrix}, & \text{if } \mathbf{x}(k-1) \in \mathsf{R}_{(2,1)}, \\ \begin{bmatrix} x_2(k-1) + 3 \\ x_2(k-1) + 3 \end{bmatrix}, & \text{if } \mathbf{x}(k-1) \in \mathsf{R}_{(2,2)}. \end{cases}$$

□

3. EXPLICIT REACHABILITY ANALYSIS OF MAX-PLUS LINEAR SYSTEMS

Suppose we have an MPL system (1) and $X, Y \subseteq \mathbb{R}^n$ as the initial and target sets, respectively. The set Y is *reachable at time* k from X if there exist $\mathbf{x}(0) \in X$ such that $\mathbf{x}(k) \in Y$, where $\mathbf{x}(k)$ is computed recursively by (1) from $\mathbf{x}(0)$. The existing approach for solving reachability analysis (RA) of MPL systems is by computing forward and backward reach sets of MPL systems (Adzkiya et al., 2014a,b, 2015).

Given an initial set X, the forward reach set X_k is recursively defined as

$$X_k = \mathtt{Im}(X_{k-1}) = \{A \otimes \mathbf{x} \mid \mathbf{x} \in X_{k-1}\}, \tag{5}$$

where $X_0 = X$. Likewise, from the target set Y, the backward reach set Y_{-k} is defined as

$$Y_{k-1} = \mathtt{Im}^{-1}(Y_k) = \{\mathbf{y} \in \mathbb{R}^n \mid A \otimes \mathbf{y} \in Y_k\}, \tag{6}$$

where $Y_0 = Y$. The initial and target states are assumed to be non-empty DBMs. The forward and backward reach sets can be computed using one-shot procedures as follows:

$$X_k = \{A^{\otimes k} \otimes \mathbf{x} \mid x \in X_0\}, \tag{7}$$

and

$$Y_{-k} = \{\mathbf{y} \in \mathbb{R}^n \mid A^{\otimes k} \otimes \mathbf{y} \in Y_0\}. \tag{8}$$

To compute forward and backward reach sets, one needs to represent an MPL system (1) as a PWA model. The steps to compute X_k are explained in (Adzkiya et al., 2014b) and involve image computation of DBMs w.r.t. the affine dynamics. On the other hand, the inverse image computation of DBMs w.r.t. affine dynamics is used to compute Y_{-k} (Adzkiya et al., 2014a). It has been shown in (Adzkiya et al., 2014a,b) that both forward and backward reach sets are a union of finitely many DBMs. Notice that, $X_k \neq \emptyset$ for $k \geq 0$. However, it is possible that there is an $l > 0$ such that $Y_{-k} = \emptyset$ for all $k \geq l$.

Algorithms 1-4 illustrate ways to perform RA of MPL systems by means of the computation of forward and backward reach sets up to a given bound $N \in \mathbb{N}$. In Algorithms 2 and 4, one needs to generate the PWA system for $A^{\otimes k}$ for each iteration k - notice that this "one-shot" implementation does not simply compute the reach set at the final time N, as it still runs over the entire time horizon; later we will reason about the benefit of such implementation versus the "sequential" Algorithms 1 and 3. For Algorithms 3-4, if the backward reach set $Y_{-k} = \emptyset$ then the algorithms are terminated at the k^{th} iteration with `false` as the output.

If the output of Algorithm 1-4 is `true`, then Y is reachable from X, otherwise Y is not reachable from X, within the given time bound N. Given a negative outcome from above, in general we cannot conclude that Y is not reachable from X within time bounds greater than N. However, for irreducible MPL systems we can prove that there exists a *completeness threshold* (Clarke et al., 2004) $N^* \in \mathbb{N}$ for Algorithms 1-4. This notion is widely used in the model checking literature and applies to RA. Such a scalar is the maximum iteration that is sufficient for the termination of an algorithm: e.g. for Algorithms 1-4, if Y is not reachable from X up to bound N^*, then Y is surely also not reachable from X within any larger bound $N > N^*$. So quite importantly, finding a completeness threshold ensures the completeness of RA procedures.

We show that the completeness threshold is related to transient and cyclicity of irreducible MPL systems. It is important to note that the transient of an irreducible MPL system is not linear w.r.t. its dimension (a small dimensional MPL system may have a relatively large transient).

Proposition 6. If $A \in \mathbb{R}_{\max}^{n \times n}$ is irreducible then the completeness threshold for Algorithms 1-4 is $k_0 + c - 1$, where k_0 and c are the transient and cyclicity of A, respectively.

Proof. It suffices to prove the completeness threshold for Algorithms 1 and 3. Suppose we have forward reach sets $X_0, X_1, \ldots$, where $X_k = \mathtt{Im}(X_{k-1})$. By Proposition 3, for $k \geq k_0$ we have $A^{\otimes(k+c)} = \lambda c \otimes A^{\otimes k}$, which implies $\mathbf{x} \in X_k$ iff $\lambda c \otimes \mathbf{x} \in X_{k+c}$. Recall that the forward reach sets are in general unions of DBMs. Furthermore, DBMs are not affected by shifting operations [1]. Consequently, $X_{k+c} = X_k$ for $k \geq k_0$. From here, we can conclude that we only need to consider a bound before reaching the periodicity, i.e. $k_0 + c - 1$. Now, suppose we have non-empty backward reach sets $Y_0, Y_{-1}, \ldots$, where $Y_{k-1} = \mathtt{Im}^{-1}(Y_k)$. Similarly, by Proposition 3, we have $Y_{-(k+c)} = Y_{-k}$ for $k \geq k_0$ which leads to the same conclusion as previous one. □

By Proposition 6, we can conclude that RA of irreducible MPL system is decidable, provided that the initial and target sets are DBMs.

Alg. 1 RA (forward)

Inputs: $A \in \mathbb{R}_{\max}^{n \times n}$, initial set X, target set Y, $N \in \mathbb{N}$
Output: boolean

```
reach ← false
X_0 ← X
generate PWA system of A
k ← 1
while k ≤ N
  compute X_k by (5)
  if X_k ∩ Y ≠ ∅ then
    reach ← true
    break
  end
  k ← k + 1
end
return reach
```

Alg. 2 RA (one-shot forward)

Inputs: $A \in \mathbb{R}_{\max}^{n \times n}$, initial set X, target set Y, $N \in \mathbb{N}$
Output: boolean

```
reach ← false
X_0 ← X
k ← 1
while k ≤ N
  generate PWA system of A^⊗k
  compute X_k by (7)
  if X_k ∩ Y ≠ ∅ then
    reach ← true
    break
  end
  k ← k + 1
end
return reach
```

Alg. 3 RA (backward)

Inputs: $A \in \mathbb{R}_{\max}^{n \times n}$, initial set X, target set Y, $N \in \mathbb{N}$
Output: boolean

```
reach ← false
Y_0 ← Y
generate PWA system of A
k ← 1
while k ≤ N
  compute Y_-k by (6)
  if Y_-k = ∅ then
    break
  end
  if Y_-k ∩ X ≠ ∅ then
    reach ← true
    break
  end
  k ← k + 1
end
return reach
```

Alg. 4 RA (one-shot backward)

Inputs: $A \in \mathbb{R}_{\max}^{n \times n}$, initial set X, target set Y, $N \in \mathbb{N}$
Output: boolean

```
reach ← false
Y_0 ← Y
k ← 1
while k ≤ N
  generate PWA system of A^⊗k
  compute Y_-k by (8)
  if Y_-k = ∅ then
    break
  end
  if Y_-k ∩ X ≠ ∅ then
    reach ← true
    break
  end
  k ← k + 1
end
return reach
```

[1] Given a DBM D and $\alpha \in \mathbb{R}$, $\alpha \otimes D = \{\alpha \otimes \mathbf{x} \mid \mathbf{x} \in D\} = D$.

We provide an example of reachability analysis of MPL systems via reach sets computation.

Example 7. With the preceding MPL system in Example 5, we define the initial and target sets respectively as $X = \{\mathbf{x} \in \mathbb{R}^2 \mid x_1 - x_2 \geq 3\}$ and $Y = \{\mathbf{x} \in \mathbb{R}^2 \mid x_1 - x_2 \geq 5\}$. One could check that the transient and cyclicity of (4) are $k_0 = c = 2$ and therefore the completeness threshold is $N^* = 3$.

Leaving details aside, the forward reach sets are $X_1 = \{\mathbf{x} \in \mathbb{R}^2 \mid x_1 - x_2 = -1\}, X_2 = \{\mathbf{x} \in \mathbb{R}^2 \mid x_1 - x_2 = 0\}$, and $X_3 = \{\mathbf{x} \in \mathbb{R}^2 \mid x_1 - x_2 = 2\}$. As $X_i \cap Y = \emptyset$ for $i = 1, 2, 3$, we can conclude that Y is not reachable from X. By backward reach set computation, we have $Y_1 = \emptyset$ which leads to the same conclusion. □

There are a few elements contributing to the computational bottleneck (time and memory requirements) of this approach. First of all, the number of regions in the PWA systems depends on the size of state matrix and on the number of finite entries in the matrix. The worst-case complexity of generating the PWA system via (2) is $\mathcal{O}(n^{n+3})$ (Adzkiya et al., 2013). Furthermore, the reachable set and backward reachable set are a union of finitely many DBMs. In the worst case, the number of DBMs grows exponentially with the time horizon.

As shown in (Adzkiya et al., 2014b), the worst-case complexity to generate the sequential (resp. one-shot) reach sets up to bound N is $\mathcal{O}(\sum_{k=0}^{N-1} |X_k| \cdot n^{n+3})$ (resp. $\mathcal{O}((\lfloor \log_2 N \rfloor + |X_0|) \cdot n^{n+3})$) , where $|X_k|$ represents the number of DBMs in X_k. Similarly, the complexity for backward reach sets computations are $\mathcal{O}(\sum_{k=0}^{N-1} |Y_k| \cdot n^{n+3})$ (for sequential) and $\mathcal{O}((\lfloor \log_2 N \rfloor + |Y_0|) \cdot n^{n+3})$ (for one-shot). Surely, the one-shot procedures are more efficient than the sequential ones.

4. SYMBOLIC REACHABILITY ANALYSIS OF MAX-PLUS LINEAR SYSTEMS

4.1 Satisfiability Modulo Theories

Satisfiability Modulo Theories (SMT) deal with the problem of determining the satisfaction of a first-order logical formula w.r.t. some logical theory background, such as Boolean logic (which generalises SAT theory), bit-vectors, real and integer arithmetics, and so on (Barrett and Tinelli, 2018). For instance, the following formula

$$(x \geq 0) \wedge (y < 2) \wedge (x - y < -1)$$

has solutions for $x, y \in \mathbb{R}$ but no solution for $x, y \in \mathbb{Z}$. In general, an SMT formula may contain conjunctions ($\wedge$), disjunctions ($\vee$), and quantifiers ($\exists, \forall$). An SMT solver reports whether the given formula is satisfiable or not satisfiable. For the former case, it usually also provides a *model*, i.e. a satisfying assignment for the formula.

SMT has grown into a very active research subject: it has a standardised library and a collection of benchmarks developed by the SMT community (Barrett et al., 2010), as well as an annual international competition for SMT solvers (Barrett et al., 2005). As a result, there are several powerful SMT solvers, such as MATHSAT5 (Cimatti et al., 2013), Yices 2.2 (Dutertre, 2014), and Z3 (De Moura and Bjørner, 2008). Applications of SMT-solving arise on supervisory control of discrete-event systems (Shoaei et al., 2014), verification of neural networks (Katz et al., 2017), optimization (Li et al., 2014), and beyond.

4.2 SMT-Based Reachability Analysis of MPL systems

This section discusses new procedures to solve RA of MPL systems using SMT-solving. We use quantifier-free *difference logic* as the underlying logical theory for SMT.

Definition 8. (Difference logic, (Cotton et al., 2004)). Let $\mathcal{B} = \{\mathtt{b}_1, \ldots, \mathtt{b}_m\}$ and $\mathcal{V} = \{\mathtt{x}_1, \ldots, \mathtt{x}_n\}$ be sets of Boolean and real-valued variables, respectively. The set of atomic formulae of $DL(\mathcal{B}, \mathcal{V})$ consists of Boolean variables in $\mathcal{B}$ and of inequalities with the form $\mathtt{x}_i - \mathtt{x}_j \sim c$, where $\sim \in \{>, \geq\}$ and $c \in \mathbb{R}$. □

For instance, both $f_1 = (\mathtt{x}_1 - \mathtt{x}_2 \geq 1) \rightarrow (\mathtt{x}_1 - \mathtt{x}_3 > 1)$ and $f_2 = ((\mathtt{x}_1 - \mathtt{x}_2 > 2) \wedge \mathtt{b}_1) \leftrightarrow (\neg(\mathtt{x}_2 - \mathtt{x}_3 \geq 0) \vee \mathtt{b}_2)$ are formulae in difference logic. In this work, we only consider formulae in difference logic where the Boolean variables do not appear, as in f_1. Interestingly, notice that any DBM is a formula in difference logic, where Boolean connectives are exclusively conjunctions ($\wedge$): as such, the non-emptiness of a DBM is equivalent to the satisfiability of its corresponding difference logic formula.

We show that operations in max-plus algebra can be expressed as formulae in difference logic.

Proposition 9. Given real-valued variables $\mathtt{x}_1, \ldots, \mathtt{x}_n$ and real scalars $a_1, \ldots, a_n$, the equation $\mathtt{x}' = \bigoplus_{i=1}^{n} (\mathtt{x}_i \otimes a_i)$ is equivalent to

$$\left(\bigwedge_{i=1}^{n} (\mathtt{x}' - \mathtt{x}_i \geq a_i) \right) \wedge \left(\bigvee_{i=1}^{n} (\mathtt{x}' - \mathtt{x}_i = a_i) \right). \tag{9}$$

Proof. The difference logic formula (9) asserts that: 1) $\forall i \ \mathtt{x}' \geq \mathtt{x}_i + a_i$ and 2) $\exists i \ \mathtt{x}' = \mathtt{x}_i + a_i$. From both conditions, it is straigthforward that $\mathtt{x}'$ can be expressed as $\max\{\mathtt{x}_1 + a_1, \ldots, \mathtt{x}_n + a_n\}$. □

For the rest of the paper, $\mathcal{V}^{(k)} = \{\mathtt{x}_1^{(k)}, \ldots, \mathtt{x}_n^{(k)}\}$ denotes the set of variables encompassing the states of the MPL system in (1) at the k^{th} event. Via Proposition 9, the MPL system in (1) can be expressed as a formula in difference logic as follows:

$$\mathtt{Im}(\mathcal{V}^{(k-1)}, \mathcal{V}^{(k)}) = \bigwedge_{i=1}^{n} (\mathtt{ge}_i \wedge \mathtt{eq}_i), \tag{10}$$

where

$$\mathtt{ge}_i = \bigwedge_{j \in \mathtt{fin}_i} (\mathtt{x}_i^{(k)} - \mathtt{x}_j^{(k-1)} \geq A(i,j)),$$

$$\mathtt{eq}_i = \bigvee_{j \in \mathtt{fin}_i} (\mathtt{x}_i^{(k)} - \mathtt{x}_j^{(k-1)} = A(i,j)),$$

and $\mathtt{fin}_i$ is a set containing the indices of the finite elements of $A(i, \cdot)$.

Consequently, the following SMT formula

$$T = \bigwedge_{k=1}^{N} \mathtt{Im}(\mathcal{V}^{(k-1)}, \mathcal{V}^{(k)}) \tag{11}$$

corresponds to a *symbolic representation* of the states of the trajectory of the MPL system in (1) for $k = 1, \ldots, N$.

It follows that the reachability of the target set Y from the initial set X up to bound N can be equivalently expressed as the satisfiability of the SMT formula

$$X^{(0)} \wedge T \wedge Y^{(N)}, \tag{12}$$

where $X^{(0)}$ (resp. $Y^{(N)}$) is the difference logic representation for X (resp. Y) over $\mathcal{V}^{(0)}$ (resp. $\mathcal{V}^{(N)}$).

Furthermore, the one-shot approach to reachability analysis can be formulated symbolically from (11) as follows: instead of using N conjuncts, (11) can be expressed by

$$T = \mathtt{Im}^N(\mathcal{V}^{(0)}, \mathcal{V}^{(1)}),$$

where $\mathtt{Im}^N$ is generated from (10) for matrix $A^{\otimes N}$. Accordingly, the formula (12) is changed into

$$X^{(0)} \wedge T \wedge Y^{(1)}. \tag{13}$$

Example 10. With the previous example of explicit reachability analysis in Example 7, the corresponding formula (12) for $N = 3$ is

$$(\mathtt{x}_1^{(0)} - \mathtt{x}_2^{(0)} \geq 3) \wedge T_1 \wedge T_2 \wedge T_3 \wedge (\mathtt{x}_1^{(3)} - \mathtt{x}_2^{(3)} \geq 5),$$

where

$$\begin{aligned} T_k = &(\mathtt{x}_1^{(k)} - \mathtt{x}_1^{(k-1)} \geq 2) \wedge (\mathtt{x}_1^{(k)} - \mathtt{x}_2^{(k-1)} \geq 5) \wedge \\ &((\mathtt{x}_1^{(k)} - \mathtt{x}_1^{(k-1)} = 2) \vee (\mathtt{x}_1^{(k)} - \mathtt{x}_2^{(k-1)} = 5)) \wedge \\ &(\mathtt{x}_2^{(k)} - \mathtt{x}_1^{(k-1)} \geq 3) \wedge (\mathtt{x}_2^{(k)} - \mathtt{x}_2^{(k-1)} \geq 3) \wedge \\ &((\mathtt{x}_2^{(k)} - \mathtt{x}_1^{(k-1)} = 3) \vee (\mathtt{x}_2^{(k)} - \mathtt{x}_2^{(k-1)} = 3)). \end{aligned}$$

On the other hand, the one-shot version formula (13) is

$$\begin{aligned} &(\mathtt{x}_1^{(0)} - \mathtt{x}_2^{(0)} \geq 3) \wedge (\mathtt{x}_1^{(1)} - \mathtt{x}_1^{(0)} \geq 11) \wedge (\mathtt{x}_1^{(1)} - \mathtt{x}_2^{(0)} \geq 13) \wedge \\ &((\mathtt{x}_1^{(1)} - \mathtt{x}_1^{(0)} = 11) \vee (\mathtt{x}_1^{(1)} - \mathtt{x}_2^{(0)} = 13)) \wedge (\mathtt{x}_2^{(1)} - \mathtt{x}_1^{(0)} \geq 11) \wedge \\ &(\mathtt{x}_2^{(1)} - \mathtt{x}_2^{(0)} \geq 11) \wedge ((\mathtt{x}_2^{(1)} - \mathtt{x}_1^{(0)} = 11) \vee (\mathtt{x}_2^{(1)} - \mathtt{x}_2^{(0)} = 11)) \wedge \\ &(\mathtt{x}_1^{(1)} - \mathtt{x}_2^{(1)} \geq 5), \end{aligned}$$

where the first (resp. last) conjunct corresponds to the initial (resp. target) set in Example 7. □

Algorithms 5-6 illustrate the SMT-based adaptation of Algorithms 1-2. The function $\mathsf{fresh\_var}(k, n)$ generates a set of n real-valued variables for bound k. F is a *program vector* (not be confused with vectors in linear or max-plus algebra) containing SMT formulae as in (12). The command $\mathsf{push\_back}$ adds a formula into F from the back while $\mathsf{pop\_back}$ removes the last one. For $i \geq 0$, $F[i]$ is the $(i+1)^{\text{th}}$ element of F (from the back).

At the start, both X and Y are expressed as difference logic over $\mathcal{V}^{(0)}$. The function $Y.\mathsf{subs}(\mathcal{V}^{(k-1)}, \mathcal{V}^{(k)})$ substitutes each appearance of $\mathtt{x}_i^{(k-1)}$ in Y with $\mathtt{x}_i^{(k)}$. The non-emptiness checking of a union of DBMs in line 7 of Algorithms 1-2 is now formulated as the satisfiability checking of a difference logic formula (line 12 and 13 of Algorithm 5 and 6, respectively), where $\wedge F$ stands for $\wedge_{0 \leq i < |F|} F[i]$. The check function is implemented by an SMT solver, where $\mathsf{check}(\wedge F) = \mathtt{true}$ means that $\wedge F$ is satisfiable.

In lines 8-11 of Algorithm 5, a difference logic formula for (10) and the target set over $\mathcal{V}^{(k)}$ are added to F for each iteration k. If the condition in line 12 is not fulfilled then the last element of F (i.e., Y) is removed. For Algorithm 6, the number of elements in F is three for each iteration. In line 6, we set a temporary element for $F[1]$, which will be changed at each iteration (line 12).

Alg. 5 SMT-RA (forward)

Inputs: $A \in \mathbb{R}_{\max}^{n \times n}$, initial set X, target set Y, $N \in \mathbb{N}$,
Output: boolean

```
1: reach ← false
2: V^(0) ← fresh_var(0, n)
3: F ← ∅                          ▷ empty vector
4: F.push_back(X)
5: k ← 1
6: while k ≤ N
7:   V^(k) ← fresh_var(k, n)
8:   mpl ← Im(V^(k−1), V^(k))
9:   F.push_back(mpl)
10:  Y.subs(V^(k−1), V^(k))
11:  F.push_back(Y)
12:  if check(∧F) = true then
13:    reach ← true
14:    break
15:  end
16:  F.pop_back()
17:  k ← k + 1
18: end
19: return reach
```

Alg. 6 SMT-RA (one-shot forward)

Inputs: $A \in \mathbb{R}_{\max}^{n \times n}$, initial set X, target set Y, $N \in \mathbb{N}$,
Output: boolean

```
1: reach ← false
2: V^(0) ← fresh_var(0, n)
3: V^(1) ← fresh_var(1, n)
4: F ← ∅                          ▷ empty vector
5: F.push_back(X)
6: F.push_back(true)
7: Y.subs(V^(0), V^(1))
8: F.push_back(Y)
9: k ← 1
10: while k ≤ N
11:   mpl ← Im^k(V^(0), V^(1))
12:   F[1] ← mpl
13:   if check(∧F) = true then
14:     reach ← true
15:     break
16:   end
17:   k ← k + 1
18: end
19: return reach
```

We now describe the approach for SMT-based backward RA. For $k \geq 1$, we use $\mathcal{V}^{(-k)} = \{\mathtt{x}_1^{(-k)}, \ldots, \mathtt{x}_n^{(-k)}\}$ to represent the set of variables encompassing k^{th} backward states obtained from $\mathcal{V}^{(0)}$. The backward version of (12) is

$$Y^{(0)} \wedge \left(\bigwedge_{k=1}^{N} \mathtt{Im}(\mathcal{V}^{(-k)}, \mathcal{V}^{(1-k)}) \right) \wedge X^{(-N)}. \tag{14}$$

Similarly, the one-step version of (14) can be encoded as

$$Y^{(0)} \wedge \mathtt{Im}^N(\mathcal{V}^{(-1)}, \mathcal{V}^{(0)}) \wedge X^{(-1)}. \tag{15}$$

Algorithms 7 and 8 summarise the backward approach to solve RA via SMT-solving. Line 10 of Algorithm 7 and line 11 of Algorithm 8 are equivalent to the emptiness checking in line 7 of Algorithms 3-4.

Alg. 7 SMT-RA (backward)

Inputs: $A \in \mathbb{R}_{\max}^{n \times n}$, initial set X, target set Y, $N \in \mathbb{N}$,
Output: boolean

```
1: reach ← false
2: V^(0) ← fresh_var(0, n)
3: F ← ∅                          ▷ empty vector
4: F.push_back(Y)
5: k ← 1
6: while k ≤ N
7:   V^(−k) ← fresh_var(−k, n)
8:   mpl ← Im(V^(−k), V^(1−k))
9:   F.push_back(mpl)
10:  if check(∧F) = false then
11:    break
12:  end
13:  X.subs(V^(1−k), V^(−k))
14:  F.push_back(X)
15:  if check(∧F) = true then
16:    reach ← true
17:    break
18:  end
19:  F.pop_back()
20:  k ← k + 1
21: end
22: return reach
```

Alg. 8 SMT-RA (one-shot backward)

Inputs: $A \in \mathbb{R}_{\max}^{n \times n}$, initial set X, target set Y, $N \in \mathbb{N}$,
Output: boolean

```
1: reach ← false
2: V^(0) ← fresh_var(0, n)
3: V^(−1) ← fresh_var(−1, n)
4: F ← ∅                          ▷ empty vector
5: F.push_back(Y)
6: F.push_back(true)
7: X.subs(V^(0), V^(−1))
8: k ← 1
9: while k ≤ N
10:   F[1] ← Im^k(V^(−1), V^(0))
11:   if check(∧F) = false then
12:     break
13:   end
14:   F.push_back(X)
15:   if check(∧F) = true then
16:     reach ← true
17:     break
18:   end
19:   F.pop_back()
20:   k ← k + 1
21: end
22: return reach
```

As we mentioned before, the SMT-based RA of MPL systems is done symbolically in a sense that the SMT

formula (12) (resp. (14)) consists of variables from $\mathcal{V}^{(0)} \cup \ldots \cup \mathcal{V}^{(N)}$ (resp. $\mathcal{V}^{(0)} \cup \ldots \cup \mathcal{V}^{(-N)}$). Therefore, if the dimension of matrix in (1) is n then there are $(N+1) \times n$ variables. The number of variables is reduced to $2n$ for the one-shot versions in (13) and (15).

The performance of the symbolic Algorithms 5-8 depends on the number of "constraints" (inequalities and equalities) in (10). If the matrix in A in (1) has m finite elements in each row, then there are $2mn$ constraints.

5. COMPUTATIONAL BENCHMARKS

We compare the performance of the SMT-based RA of MPL systems presented in this paper with the existing approach in (Adzkiya et al., 2014a,b, 2015). The experiments for both procedures are implemented in C++. For the SMT solver, we use Z3 (De Moura and Bjørner, 2008). 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 $m \leq n$. For each dimension n (i.e., number of continuous variables), we generate 20 irreducible matrices $A \in \mathbb{R}_{\max}^{n \times n}$ with m finite elements in each row, where their values are taken to be between 1 and 20. The locations of the finite elements are chosen randomly. The initial and target sets for each experiment are $X = \{\mathbf{x} \in \mathbb{R}^n \mid x_1 \geq \ldots \geq x_5\}$ and $Y = \{\mathbf{x} \in \mathbb{R}^n \mid x_1 \leq \ldots \leq x_5\}$, respectively.

Table 1 (columns 2-5) shows the average running time of the reachability analysis via Algorithms 1 and 3 and of symbolic reachability analysis (SMT-RA) via Algorithms 5 and 7. The 6$^{\text{th}}$ column reports the number of experiments (out of 20) with a `true` outcome, whilst the last one represents average and maximum completeness threshold, as obtained from the 20 experiments.

Table 1. Computational benchmark for sequential reachability analysis of MPL systems

(n, m)	RA		SMT-RA		#`true`	N^*
	Alg. 1	Alg. 3	Alg. 5	Alg. 7		
$(5, 3)$	$0.03s$	$0.03s$	$0.02s$	$0.01s$	7	$\{12.25, 30\}$
$(6, 3)$	$0.31s$	$0.05s$	$0.08s$	$0.01s$	4	$\{11.20, 39\}$
$(7, 3)$	$5.26s$	$0.47s$	$0.09s$	$0.01s$	7	$\{10.45, 30\}$
$(8, 3)$	$23.89s$	$3.94s$	$0.09s$	$0.01s$	10	$\{14.65, 49\}$
$(8, 4)$	$42.14s$	$11.02s$	$0.16s$	$0.01s$	10	$\{12.85, 21\}$
$(8, 5)$	$57.84s$	$21.71s$	$0.09s$	$0.01s$	11	$\{11.50, 33\}$
$(8, 6)$	$46.42s$	$40.39s$	$0.18s$	$0.01s$	15	$\{11.95, 20\}$
$(8, 7)$	$51.28s$	$28.34s$	$0.08s$	$0.01s$	10	$\{10.55, 22\}$
$(8, 8)$	$68.51s$	$40.50s$	$0.15s$	$0.02s$	13	$\{9.65, 30\}$
$(9, 9)$	$2650.51s$	$701.29s$	$0.88s$	$0.01s$	9	$\{13.00, 25\}$

As one can see in Table 1, the SMT-based algorithms are significantly faster than those that explicitly compute reach sets. With regards to the comparison between the forward and backward approaches (for both RA and SMT-RA), the latter seems to be faster. This is likely due to the break condition in line 7 of Algorithm 3 and line 10 of Algorithm 7, which cause the backward algorithms to terminate earlier than the specified step bound N whenever the RA problem has empty solution. It should also be noted that the completeness threshold also affects the overall running time.

Table 2 reports the average running times obtained using one-shot approaches over the same tests of Table 1 (the last two columns are indeed equal). The one-shot strategy improves the running time over its sequential counterpart, particularly in the case of the forward sequential RA algorithms. Within the one-shot procedures, again the SMT-based algorithms outperform those sequentially computing the reach sets.

Table 2. Computational benchmark for one-shot reachability analysis of MPL systems

(n, m)	RA		SMT-RA		#`true`	N^*
	Alg. 2	Alg. 4	Alg. 6	Alg. 8		
$(5, 3)$	$0.03s$	$0.02s$	$0.01s$	$0.01s$	7	$\{12.25, 30\}$
$(6, 3)$	$0.22s$	$0.19s$	$0.02s$	$0.01s$	4	$\{11.20, 39\}$
$(7, 3)$	$1.36s$	$0.91s$	$0.02s$	$0.01s$	7	$\{10.45, 30\}$
$(8, 3)$	$9.06s$	$6.56s$	$0.02s$	$0.01s$	10	$\{14.65, 49\}$
$(8, 4)$	$13.32s$	$9.02s$	$0.02s$	$0.01s$	10	$\{12.85, 21\}$
$(8, 5)$	$20.58s$	$14.62s$	$0.02s$	$0.01s$	11	$\{11.5, 33\}$
$(8, 6)$	$27.69s$	$24.64s$	$0.02s$	$0.01s$	15	$\{11.95, 20\}$
$(8, 7)$	$32.55s$	$29.40s$	$0.02s$	$0.01s$	10	$\{10.55, 22\}$
$(8, 8)$	$42.60s$	$37.69s$	$0.02s$	$0.01s$	13	$\{9.65, 30\}$
$(9, 9)$	$843.13s$	$693.99s$	$0.03s$	$0.01s$	9	$\{13, 25\}$

The impressive (and almost constant) outcomes of the SMT-RA (Algorithms 6,8) in Table 2 suggest to push their scalability to the limit. Hence, we provide a computational benchmark for high-dimensional MPL systems in Table 3. We focus the benchmark exclusively on one-shot algorithms, as we have seen that sequential algorithms are slower. To balance success and failures of RA, we re-define the (dimension of) initial and target sets to be function of the model dimension, as follows: $X = \{\mathbf{x} \in \mathbb{R}^n \mid x_1 \geq \ldots \geq x_p\}, Y = \{\mathbf{x} \in \mathbb{R}^n \mid x_1 \leq \ldots \leq x_p\}$ where $p = \frac{n}{2}$.

Table 3. Computational benchmark for SMT-based reachability analysis of high-dimensional MPL systems

(n, m)	SMT-RA		#`true`	N^*
	Alg. 6	Alg. 8		
$(20, 10)$	$0.23s$	$0.05s$	8	$\{19.15, 44\}$
$(30, 15)$	$0.87s$	$0.14s$	5	$\{20.70, 30\}$
$(40, 20)$	$3.18s$	$0.30s$	2	$\{23.35, 47\}$
$(50, 25)$	$5.67s$	$0.55s$	1	$\{22.10, 50\}$
$(60, 30)$	$8.86s$	$1.76s$	3	$\{19.65, 34\}$
$(70, 35)$	$16.59s$	$3.25s$	1	$\{18.35, 37\}$
$(80, 40)$	$29.20s$	$6.62s$	0	$\{16.15, 25\}$
$(90, 45)$	$31.93s$	$12.29s$	2	$\{13.65, 21\}$
$(100, 50)$	$46.01s$	$21.34s$	5	$\{12.05, 14\}$
$(110, 55)$	$70.15s$	$43.57s$	4	$\{11.10, 12\}$
$(120, 60)$	$102.40s$	$70.99s$	2	$\{11.13, 13\}$
$(140, 70)$	$154.72s$	$92.28s$	4	$\{9.6, 11\}$
$(160, 80)$	$220.23s$	$222.71s$	6	$\{8.3, 10\}$
$(180, 90)$	$380.96s$	$539.16s$	11	$\{8, 9\}$
$(200, 100)$	$682.10s$	$1592.28s$	12	$\{7.35, 12\}$

Similar to the results in Table 2, Table 3 shows that the performance of Algorithm 8 (backward RA) is better than that of Algorithm 6 (forward RA) up to dimension of 140. Instead, for larger dimensions the forward RA algorithm outperforms the backward one. There are two possible reasons for this outcome. First, the larger proportion of `true` experiments: we argue that if the RA problem yields `true`, then Algorithm 6 (which performs SMT-checking once) is likely faster than Algorithm 8 (which uses SMT-checking twice for each iteration). Second, the smaller values of completeness thresholds also contribute to the relative speedup of the forward algorithm.

Recall that we expect one-shot algorithms to be faster: as an example, for $(n, m) = (50, 25)$ the average running time for Algorithm 5 and 7 would be 706.73 second and 1.33 second, respectively. Indeed, for the SMT-RA procedures, the one-shot algorithms handle less complex difference logic formulae than the sequential ones: notice that in line 9 of Algorithms 5 and 7, at any iteration k a new formula encompassing the k^{th} image of the MPL system is added and sent to the SMT solver; instead, in Algorithms 6 and 8, at every iteration the SMT formula is replaced by a new one, and this is likely to result in simpler formulae.

6. CONCLUSIONS AND FUTURE WORK

This paper has introduced a symbolic approach to solve reachability problems over (high-dimensional) MPL systems. We encode the problems as a formulae in difference logic and verify their satisfaction using an SMT solver. The procedure has been tested on computational benchmarks, which have shown a significant improvement over alternative, state-of-the-art techniques.

We are interested to extend the symbolic reachability analysis procedure to *uncertain* MPL systems and to use SMT-based approaches for the analysis of general properties of MPL systems, as in Abate et al. (2020).

ACKNOWLEDGEMENTS

Support by the Indonesia Endowment Fund for Education (LPDP) and by the Alan Turing Institute, London, UK.

REFERENCES

Abate, A., Cimatti, A., Micheli, A., and Mufid, M. (2020). Computation of transient in max-plus linear systems via SMT-solving. In N. Jansen and N. Bertrand (eds.), *Int. Conf. Formal Modeling and Analysis of Timed Systems* (FORMATS'20), 161–177. Springer, LNCS 12288.

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

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

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

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

Baccelli, F., Cohen, G., Olsder, G.J., and Quadrat, J.P. (1992). *Synchronization and Linearity: An Algebra for Discrete Event Systems*. John Wiley & Sons Ltd.

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

Barrett, C., Stump, A., and Tinelli, C. (2010). The satisfiability modulo theories library. URL `http://smtlib.cs.uiowa.edu`.

Barrett, C. and Tinelli, C. (2018). Satisfiability modulo theories. In *Handbook of Model Checking*, 305–343. Springer.

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

Clarke, E., Kroening, D., Ouaknine, J., and Strichman, O. (2004). Completeness and complexity of bounded model checking. In *International Workshop on Verification, Model Checking, and Abstract Interpretation*, 85–96. Springer.

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

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

Dill, D.L. (1989). Timing assumptions and verification of finite-state concurrent systems. In J. Sifakis (ed.), *Intl. Conf. on Computer Aided Verification* (CAV'89), volume 407 of *Lecture Notes in Computer Science*, 197–212. Springer, Hiedelberg.

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

Floyd, R.W. (1962). Algorithm 97: shortest path. *Communications of the ACM*, 5(6), 345.

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

Heidergott, B., Olsder, G.J., and Van der Woude, J. (2014). *Max Plus at Work−Modeling and Analysis of Synchronized Systems: A Course on Max-Plus Algebra and Its Applications*. Princeton University Press.

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

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

Li, Y., Albarghouthi, A., Kincaid, Z., Gurfinkel, A., and Chechik, M. (2014). Symbolic optimization with smt solvers. In *ACM SIGPLAN Notices*, volume 49, 607–618. ACM.

Mufid, M., Adzkiya, D., and Abate, A. (2018). Tropical abstractions of max-plus linear systems. In D. Jansen and P. Prabhakar (eds.), *Int. Conf. Formal Modeling and Analysis of Timed Systems* (FORMATS'18), 271–287. Springer, LNCS 11022.

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

Sontag, E. (1981). Nonlinear regulation: The piecewise linear approach. *IEEE Transactions on Automatic Control*, 26(2), 346–358.