Delft University of Technology

Delft Center for Systems and Control

Technical report 14-010

Forward reachability computation for autonomous max-plus-linear systems*

D. Adzkiya, B. De Schutter, and A. Abate

If you want to cite this report, please use the following reference instead:
D. Adzkiya, B. De Schutter, and A. Abate, "Forward reachability computation for autonomous max-plus-linear systems," *Proceedings of the 20th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2014)*, Grenoble, France, pp. 248–262, Apr. 2014.

Delft Center for Systems and Control
Delft University of Technology
Mekelweg 2, 2628 CD Delft
The Netherlands
phone: +31-15-278.51.19 (secretary)
fax: +31-15-278.66.79
URL: `http://www.dcsc.tudelft.nl`

*This report can also be downloaded via `http://pub.deschutter.info/abs/14_010.html`

Forward Reachability Computation for Autonomous Max-Plus-Linear Systems *

Dieky Adzkiya[1], Bart De Schutter[1], and Alessandro Abate[2,1]

[1] Delft Center for Systems and Control
TU Delft – Delft University of Technology, The Netherlands
{d.adzkiya,b.deschutter,a.abate}@tudelft.nl
[2] Department of Computer Science
University of Oxford, United Kingdom
alessandro.abate@cs.ox.ac.uk

Abstract. This work discusses the computation of forward reachability for autonomous (that is, deterministic) Max-Plus-Linear (MPL) systems, a class of continuous-space discrete-event models that are relevant for applications dealing with synchronization and scheduling. Given an MPL model and a set of initial states, we characterize and compute its "reach tube," namely the sequential collection of the sets of reachable states (these sets are regarded step-wise as "reach sets"). We show that the exact computation of the reach sets can be quickly and compactly performed by manipulations of difference-bound matrices, and derive explicit worst-case bounds for the complexity of these operations. The concepts and techniques are implemented within the toolbox VeriSiMPL, and are practically elucidated by a running example. We further display the computational performance of the approach by two concluding numerical benchmarks: the technique comfortably handles reachability computations over twenty-dimensional MPL models (i.e., models with twenty continuous variables), and it clearly outperforms an alternative state-of-the-art approach in the literature.

1 Introduction

Reachability analysis is a fundamental problem in the areas of formal methods and of systems theory. It is concerned with assessing whether a certain set of states of a system is attainable from a given set of initial conditions. The problem is particularly interesting and compelling over models with continuous components – either in time or in the (state) space. For the first class of models, reachability has been widely investigated over discrete-space systems, such as timed automata [1, 2], or (timed continuous) Petri nets [3], or hybrid automata [4]. On the other hand, much research has been done to enhance and scale the

* This work has been supported by the European Commission STREP project MoVeS 257005, by the European Commission Marie Curie grant MANTRAS 249295, by the European Commission IAPP project AMBI 324432, by the European Commission NoE Hycon2 257462, and by the NWO VENI grant 016.103.020.

reachability analysis of continuous-space models. Among the many approaches for deterministic dynamical systems, we report here the use of face lifting [5], the computation of flow-pipes via polyhedral approximations [6, 7], the formulation as solution of Hamilton-Jacobi equations [8] (related to the study of forward and backward reachability [9]), the use of ellipsoidal techniques [10, 11], differential inclusions [12], support functions [13], and Taylor models [14].

Max-Plus-Linear (MPL) models are discrete-event systems [15] with continuous variables that express the timing of the underlying sequential events. Autonomous MPL models are characterized by deterministic dynamics. MPL models are employed to describe the timing synchronization between interleaved processes, and as such are widely employed in the analysis and scheduling of infrastructure networks, such as communication and railway systems [16] and production and manufacturing lines [17]. They are related to a subclass of timed Petri nets, namely timed-event graphs [15], however not directly to time Petri nets [18] nor to timed automata [1]. MPL models are classically analyzed over properties such as transient and periodic regimes [15]. They can be simulated (though not verified) via the max-plus toolbox Scilab [19].

Reachability analysis of MPL systems from a *single* initial condition has been investigated in [20, 21] by the computation of the reachability matrix (as for discrete-time linear dynamical systems). It has been shown in [22, Sect. 4.13] that the reachability problem for autonomous MPL systems with a single initial condition is decidable – this result however does not hold for a general, uncountable set of initial conditions. Under the limiting assumption that the set of initial conditions is expressed as a max-plus polyhedron [23, 24], forward reachability analysis can be performed over the max-plus algebra. In conclusion, to the best of our knowledge, there exists no computational toolbox for general reachability analysis of MPL models, nor it is possible to leverage software for related timed-event graphs or timed Petri nets. As an alternative, reachability computation for MPL models can be studied using the Multi-Parametric Toolbox (MPT) [25] (cf. Section 4).

In this work, we extend the state-of-the-art results for forward reachability analysis of MPL models by considering an arbitrary (possibly uncountable) set of initial conditions, and present a new computational approach to forward reachability analysis of MPL models. We first alternatively characterize MPL dynamics by Piece-wise Affine (PWA) models, and show that the dynamics can be fully represented by Difference-Bound Matrices (DBM) [26, Sect. 4.1], which are structures that are quite simple to manipulate. We further claim that DBM are closed over PWA dynamics, which leads to being able to map DBM-sets through MPL models. We then characterize and compute, given a set of initial states, its "reach tube," namely the sequential collection of the sets of reachable states (aggregated step-wise as "reach sets"). With an emphasis on computational and implementation aspects, we provide a quantification of the worst-case complexity of the algorithms discussed throughout the work. Notice that although DBM are a structure that has been used in reachability analysis of timed automata, this

does not imply that we can employ related techniques for reachability analysis of MPL systems, since the two modeling frameworks are not equivalent.

While this new approach reduces reachability analysis of MPL models to a computationally feasible task, the foundations of this contribution go beyond mere manipulations of DBM: the technique is inspired by the recent work in [27], which has developed an approach to the analysis of MPL models based on finite-state abstractions. In particular, the procedure for forward reachability computation on MPL models discussed in this work is implemented in the VeriSiMPL ("very simple") software toolbox [28], which is freely available. While the general goals of VeriSiMPL go beyond the topics of this work and are thus left to the interested reader, in this article we describe the details of the implementation of the suite for reachability analysis within this toolbox over a running example. With an additional numerical case study, we display the scalability of the tool as a function of model dimension (the number of its continuous variables): let us emphasize that related approaches for reachability analysis of discrete-time dynamical systems based on finite abstractions do not reasonably scale beyond models with a few variables [29], whereas our procedure comfortably handles models with about twenty continuous variables. In this numerical benchmark we have purposely generated the underlying dynamics randomly: this allows deriving empirical outcomes that are general and not biased towards possible structural features of a particular model. Finally, we successfully benchmark the computation of forward reachability sets against an alternative approach based on the well-developed MPT software tool [25].

2 Models and Preliminaries

2.1 Max-Plus-Linear Systems

Define $\mathbb{R}_\varepsilon$, ε and e respectively as $\mathbb{R}\cup\{\varepsilon\}$, $-\infty$ and 0. For $\alpha,\beta\in\mathbb{R}_\varepsilon$, introduce the two operations $\alpha\oplus\beta=\max\{\alpha,\beta\}$ and $\alpha\otimes\beta=\alpha+\beta$, where the element ε is considered to be absorbing w.r.t. $\otimes$ [15, Definition 3.4]. Given $\beta\in\mathbb{R}$, the max-algebraic power of $\alpha\in\mathbb{R}$ is denoted by $\alpha^{\otimes\beta}$ and corresponds to $\alpha\beta$ in the conventional algebra. The rules for the order of evaluation of the max-algebraic operators correspond to those of conventional algebra: max-algebraic power has the highest priority, and max-algebraic multiplication has a higher priority than max-algebraic addition [15, Sect. 3.1].

The basic max-algebraic operations are extended to matrices as follows. If $A,B\in\mathbb{R}_\varepsilon^{m\times n}$; $C\in\mathbb{R}_\varepsilon^{m\times p}$; $D\in\mathbb{R}_\varepsilon^{p\times n}$; and $\alpha\in\mathbb{R}_\varepsilon$, then $[\alpha\otimes A](i,j)=\alpha\otimes A(i,j)$; $[A\oplus B](i,j)=A(i,j)\oplus B(i,j)$; $[C\otimes D](i,j)=\bigoplus_{k=1}^{p}C(i,k)\otimes D(k,j)$; for $i=1,\dots,m$ and $j=1,\dots,n$. Notice the analogy between $\oplus$, $\otimes$ and $+$, $\times$ for matrix and vector operations in conventional algebra. Given $m\in\mathbb{N}$, the m-th max-algebraic power of $A\in\mathbb{R}_\varepsilon^{n\times n}$ is denoted by $A^{\otimes m}$ and corresponds to $A\otimes\cdots\otimes A$ (m times). Notice that $A^{\otimes 0}$ is an n-dimensional max-plus identity matrix, i.e. the diagonal and nondiagonal elements are e and ε, respectively. In this paper, the following notation is adopted for reasons of convenience. A vector with each component that is equal to 0 (or $-\infty$) is also denoted by e (resp., ε).

Furthermore, for practical reasons, the state space is taken to be $\mathbb{R}^n$, which also implies that the state matrix has to be row-finite (cf. Definition 1).

An autonomous (that is, deterministic) MPL model [15, Remark 2.75] is defined as:

$$x(k) = A \otimes x(k-1) \ , \tag{1}$$

where $A \in \mathbb{R}_\varepsilon^{n\times n}$, $x(k-1) = [x_1(k-1) \dots x_n(k-1)]^T \in \mathbb{R}^n$ for $k \in \mathbb{N}$. The independent variable k denotes an increasing discrete-event counter, whereas the state variable x defines the (continuous) timing of the discrete events. Autonomous MPL models are characterized by deterministic dynamics. Related to the state matrix A is the notion of regular (or row-finite) matrix and that of irreducibility.

Definition 1 (Regular (Row-Finite) Matrix, [16, Sect. 1.2]) *A max-plus matrix $A \in \mathbb{R}_\varepsilon^{n\times n}$ is called regular (or row-finite) if A contains at least one element different from ε in each row.*

A matrix $A \in \mathbb{R}_\varepsilon^{n\times n}$ is irreducible if the nondiagonal elements of $\bigoplus_{k=1}^{n-1} A^{\otimes k}$ are finite (not equal to ε). If A is irreducible, there exists a unique max-plus eigenvalue $\lambda \in \mathbb{R}$ [15, Th. 3.23] and the corresponding eigenspace $E(A) = \{x \in \mathbb{R}^n : A \otimes x = \lambda \otimes x\}$ [15, Sect. 3.7.2].

Example: Consider the following autonomous MPL model from [16, Sect. 0.1], representing the scheduling of train departures from two connected stations $i = 1, 2$ ($x_i(k)$ denotes the time of the k-th departure for station i):

$$\begin{aligned} x(k) &= \begin{bmatrix} 2 & 5 \\ 3 & 3 \end{bmatrix} \otimes x(k-1) \ , \ \text{or equivalently} \ , \\ \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} &= \begin{bmatrix} \max\{2 + x_1(k-1), 5 + x_2(k-1)\} \\ \max\{3 + x_1(k-1), 3 + x_2(k-1)\} \end{bmatrix} \ . \end{aligned} \tag{2}$$

Matrix A is a row-finite matrix and irreducible since $A(1,2) \neq \varepsilon \neq A(2,1)$. □

Proposition 1 ([16, Th. 3.9]) *Let $A \in \mathbb{R}_\varepsilon^{n\times n}$ be an irreducible matrix with max-plus eigenvalue $\lambda \in \mathbb{R}$. There exist $k_0, c \in \mathbb{N}$ such that $A^{\otimes(k+c)} = \lambda^{\otimes c} \otimes A^{\otimes k}$, for all $k \geq k_0$. The smallest k_0 and c verifying the property are defined as the length of the transient part and the cyclicity, respectively.*

Proposition 1 allows to establish the existence of a periodic behavior. Given an initial condition $x(0) \in \mathbb{R}^n$, there exists a finite $k_0(x(0))$, such that $x(k+c) = \lambda^{\otimes c} \otimes x(k)$, for all $k \geq k_0(x(0))$. Notice that we can seek the length of the transient part $k_0(x(0))$ specifically for the initial condition $x(0)$, which is in general less conservative than the global $k_0 = k_0(A)$, as in Proposition 1. Upper bounds for the length of the transient part k_0 and for its computation have been discussed in [30].

Example: In the example (2), from Proposition 1 we obtain a max-plus eigenvalue $\lambda = 4$, cyclicity $c = 2$, and a (global) length of the transient part

$k_0 = 2$. The length of the transient part specifically for $x(0) = [3, 0]^T$ can be computed observing the trajectory

$$\begin{bmatrix}3\\0\end{bmatrix}, \begin{bmatrix}5\\6\end{bmatrix}, \begin{bmatrix}11\\9\end{bmatrix}, \begin{bmatrix}14\\14\end{bmatrix}, \begin{bmatrix}19\\17\end{bmatrix}, \begin{bmatrix}22\\22\end{bmatrix}, \begin{bmatrix}27\\25\end{bmatrix}, \begin{bmatrix}30\\30\end{bmatrix}, \begin{bmatrix}35\\33\end{bmatrix}, \begin{bmatrix}38\\38\end{bmatrix}, \dots$$

The periodic behavior occurs (as expected) after 2 event steps, i.e. $k_0([3, 0]^T) = 2$, and shows a period equal to 2, namely $x(4) = 4^{\otimes 2} \otimes x(2) = 8 + x(2)$, and similarly $x(5) = 4^{\otimes 2} \otimes x(3)$. Furthermore $x(k + 2) = 4^{\otimes 2} \otimes x(k)$ for $k \geq 2$. □

2.2 Piece-wise Affine Systems

This section discusses Piece-wise Affine (PWA) systems [31] generated by an autonomous MPL model. In the following section, PWA systems will play an important role in forward reachability analysis. PWA systems are characterized by a cover of the state space, and by affine (linear plus constant) dynamics that are active within each set of the cover.

Every autonomous MPL model characterized by a row-finite matrix $A \in \mathbb{R}_{\varepsilon}^{n\times n}$ can be expressed as a PWA system in the event domain. The affine dynamics are characterized, along with its corresponding region, by the coefficients $g = (g_1, \dots, g_n) \in \{1, \dots, n\}^n$ or, more precisely, as:

$$R_g = \bigcap_{i=1}^{n} \bigcap_{j=1}^{n} \{x \in \mathbb{R}^n : A(i, g_i) + x_{g_i} \geq A(i, j) + x_j\} \ ; \tag{3}$$

$$x_i(k) = x_{g_i}(k - 1) + A(i, g_i) \ , \qquad 1 \leq i \leq n \ . \tag{4}$$

Implementation: VeriSiMPL employs a backtracking algorithm to generate the PWA system. Recall that we are looking for all coefficients $g = (g_1, \dots, g_n)$ such that R_g is not empty. In the backtracking approach, the partial coefficients are $(g_1, \dots, g_k)$ for $k = 1, \dots, n$ and the corresponding region is

$$R_{(g_1,\dots,g_k)} = \bigcap_{i=1}^{k} \bigcap_{j=1}^{n} \{x \in \mathbb{R}^n : A(i, g_i) + x_{g_i} \geq A(i, j) + x_j\} \ .$$

Notice that if the region associated with a partial coefficient $(g_1, \dots, g_k)$ is empty, then the regions associated with the coefficients $(g_1, \dots, g_n)$ are also empty, for all $g_{k+1}, \dots, g_n$. The set of all coefficients can be represented as a potential search tree. For a 2-dimensional MPL model, the potential search tree is given in Fig. 1. The backtracking algorithm traverses the tree recursively, starting from the root, in a depth-first order. At each node, the algorithm checks whether the corresponding region is empty: if the region is empty, the whole sub-tree rooted at the node is skipped (pruned).

The function `maxpl2pwa` is used to construct a PWA system from an autonomous MPL model. The autonomous MPL model is characterized by a row-finite state matrix (`Ampl`), whereas the PWA system is characterized by a collection of regions (`D`) and a set of affine dynamics (`A`,`B`). The affine dynamics that

are active in the j-th region are characterized by the j-th column of both `A` and `B`. Each column of `A` and the corresponding column of `B` contain the coefficients $[g_1, \ldots, g_n]^T$ and the constants $[A(1, g_1), \ldots, A(n, g_n)]^T$, respectively. The data structure of `D` will be discussed in Section 2.3.

Considering the autonomous MPL example in (2), the following script generates the PWA system:

```
>> Ampl = [2 5;3 3], [A,B,D] = maxpl2pwa(Ampl)
```

It will become clear in Section 2.3 that the nonempty regions of the PWA system produced by the script are: $R_{(1,1)} = \{x \in \mathbb{R}^2 : x_1 - x_2 \geq 3\}$, $R_{(2,1)} = \{x \in \mathbb{R}^2 : e \leq x_1 - x_2 \leq 3\}$, and $R_{(2,2)} = \{x \in \mathbb{R}^2 : x_1 - x_2 \leq e\}$. The affine dynamics corresponding to a region R_g are characterized by g, e.g. those for region $R_{(2,1)}$ are given by $x_1(k) = x_2(k-1) + 5$, $x_2(k) = x_1(k-1) + 3$. □

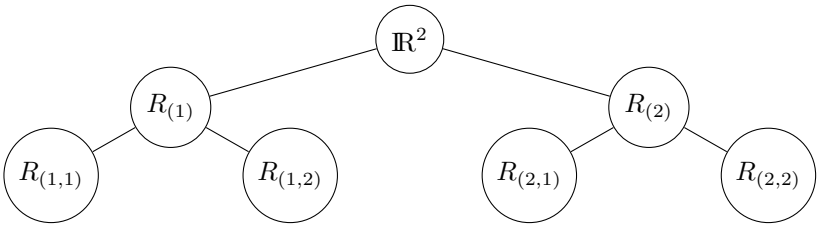

Fig. 1. Potential search tree for a 2-dimensional MPL model.

2.3 Difference-Bound Matrices

This section introduces the definition of a DBM [26, Sect. 4.1] and of its canonical-form representation. DBM provide a simple and computationally advantageous representation of the MPL dynamics, and will be further used in the next section to represent the initial conditions and reach sets.

Definition 2 (Difference-Bound Matrix) *A DBM is the intersection of finitely many sets defined by* $x_j - x_i \bowtie_{i,j} \alpha_{i,j}$, *where* $\bowtie_{i,j} \in \{<, \leq\}$, $\alpha_{i,j} \in \mathbb{R} \cup \{+\infty\}$, *for* $0 \leq i \neq j \leq n$ *and the value of* x_0 *is always equal to* 0.

The special variable x_0 is used to represent bounds on a single variable: $x_i \leq \alpha$ can be written as $x_i - x_0 \leq \alpha$. A "stripe" is defined as a DBM that does not contain x_0. Definition 2 can be likewise given over the input and the corresponding augmented space.

Implementation: VeriSiMPL represents a DBM in $\mathbb{R}^n$ as a 1×2 cell: the first element is an $(n+1)$-dimensional real-valued matrix representing the upper bound α, and the second element is an $(n+1)$-dimensional Boolean matrix representing the value of $\bowtie$. More precisely, the $(i+1, j+1)$-th element represents the upper bound and the strictness of the sign of $x_j - x_i$, for $i = 0, \ldots, n$ and $j = 0, \ldots, n$ (cf. Definition 2). Furthermore, a collection of DBM is also

represented as a 1×2 cell, where the corresponding matrices are stacked along the third dimension. □

Each DBM admits an equivalent and unique canonical-form representation, which is a DBM with the tightest possible bounds [26, Sect. 4.1]. The Floyd-Warshall algorithm can be used to obtain the canonical-form representation of a DBM, with a complexity that is cubic w.r.t. its dimension. One advantage of the canonical-form representation is that it is easy to compute orthogonal projections w.r.t. a subset of its variables, which is simply performed by deleting rows and columns corresponding to the complementary variables [26, Sect. 4.1].

Implementation: The Floyd-Warshall algorithm has been implemented in the function `floyd_warshall`. Given a collection of DBM, this function generates its canonical-form representation. The following MATLAB script computes the canonical-form representation of `D` $= \{x \in \mathbb{R}^4 : x_1 - x_4 \leq -3, x_2 - x_1 \leq -3, x_2 - x_4 \leq -3, x_3 - x_1 \leq 2\}$:

```
>> D = cell(1,2), ind = sub2ind([5,5],[4,1,4,1]+1,[1,2,2,3]+1)
>> D{1} = Inf(5), D{1}(1:6:25) = 0, D{1}(ind) = [-3,-3,-3,2]
>> D{2} = false(5), D{2}(1:6:25) = true, D{2}(ind) = true
>> Dcf = floyd_warshall(D)
```

Let us discuss the steps in the construction of the DBM `D`. We first initialize `D` with $\mathbb{R}^4$ as `D = cell(1,2)`, `D{1} = Inf(5)`, `D{1}(1:6:25) = 0`, `D{2} = false(5)`, `D{2}(1:6:25) = true`. The variable `ind` contains the location, in linear index format, of each inequality in the matrix. We define the upper bounds and the strictness in `D{1}(ind) = [-3,-3,-3,2]` and `D{2}(ind) = true`, respectively. The output is `Dcf` $= \{x \in \mathbb{R}^4 : x_1 - x_4 \leq -3, x_2 - x_1 \leq -3, x_2 - x_4 \leq -6, x_3 - x_1 \leq 2, x_3 - x_4 \leq -1\}$. Notice that the bounds of $x_2 - x_4$ and $x_3 - x_4$ are tighter. Moreover, the orthogonal projection of `D` (or `Dcf`) w.r.t. $\{x_1, x_2\}$ is $\{x \in \mathbb{R}^2 : x_2 - x_1 \leq -3\}$. □

The following result plays an important role in the computation of reachability for MPL models.

Proposition 2 ([27, Th. 1]) *The image of a DBM with respect to affine dynamics (in particular the PWA expression* (4) *generated by an MPL model) is a DBM.*

Implementation: The procedure to compute the image of a DBM in $\mathbb{R}^n$ w.r.t. the affine dynamics (4) involves: 1) computing the cross product of the DBM and $\mathbb{R}^n$; then 2) determining the DBM generated by the expression of the affine dynamics (each equation can be expressed as the difference between variables at event k and $k-1$); 3) intersecting the DBM obtained in steps 1 and 2; 4) generating the canonical-form representation; finally 5) projecting the DBM over the variables at event k, i.e. $\{x_1(k), \ldots, x_n(k)\}$. The worst-case complexity critically depends on computing the canonical-form representation (in the fourth step) and is $\mathcal{O}(n^3)$.

The procedure has been implemented in `dbm_image`. It computes the image of a collection of DBM w.r.t. the corresponding affine dynamics. The following

example computes the image of $\mathtt{D} = \{x \in \mathbb{R}^2 : e \leq x_1 - x_2 \leq 3\}$ w.r.t. $x_1(k) = x_2(k-1) + 5$, $x_2(k) = x_1(k-1) + 3$:

```
>> D = cell(1,2), ind = sub2ind([3,3],[2,1]+1,[1,2]+1)
>> D{1} = Inf(3), D{1}(1:4:9) = 0, D{1}(ind) = [3,0]
>> D{2} = false(3), D{2}(1:4:9) = true, D{2}(ind) = true
>> A = [2;1], B =[5;3], Dim = dbm_image(A,B,D)
```

The image is $\mathtt{Dim} = \{x \in \mathbb{R}^2 : -1 \leq x_1 - x_2 \leq 2\}$, which is a DBM. □

The result in Proposition 2 allows computing the image of a DBM in $\mathbb{R}^n$ w.r.t. the MPL model characterized by a row-finite matrix $A \in \mathbb{R}_\varepsilon^{n \times n}$. In order to do so, we leverage the corresponding PWA system dynamics and separate the procedure in the following steps: 1) intersecting the DBM with each region of the PWA system; then 2) computing the image of nonempty intersections according to the corresponding affine dynamics (cf. Theorem 2). The worst-case complexity depends on the last step and is $\mathcal{O}(|\mathsf{R}(A)| \cdot n^3)$, where $|\mathsf{R}(A)|$ is the number of regions in the PWA system generated by matrix A.

Proposition 2 can be extended as follows.

Corollary 1 *The image of a union of finitely many DBM w.r.t. the PWA system generated by an MPL model is a union of finitely many DBM.*

3 Forward Reachability Analysis

The goal of forward reachability analysis is to quantify the set of possible states that can be attained under the model dynamics, from a set of initial conditions. Two main notions can be defined.

Definition 3 (Reach Set) *Given an MPL model and a nonempty set of initial positions $X_0 \subseteq \mathbb{R}^n$, the reach set X_N at the event step $N > 0$ is the set of all states $\{x(N) : x(0) \in X_0\}$ obtained via the MPL dynamics.*

Definition 4 (Reach Tube) *Given an MPL model and a nonempty set of initial positions $X_0 \subseteq \mathbb{R}^n$, the reach tube is defined by the set-valued function $k \mapsto X_k$ for any given $k > 0$ where X_k is defined.*

Unless otherwise stated, in this work we focus on *finite-horizon* reachability: in other words, we compute the reach set for a finite index N (cf. Definition 3) and the reach tube for $k = 1, \ldots, N$, where $N < \infty$ (cf. Definition 4). While the reach set can be obtained as a by-product of the (sequential) computations used to obtain the reach tube, we will argue that it can be as well calculated by a tailored procedure (one-shot).

In the computation of the quantities defined above, the set of initial conditions $X_0 \subseteq \mathbb{R}^n$ will be assumed to be a union of finitely many DBM. In the more general case of arbitrary sets, these will be over- or under-approximated by DBM. As it will become clear later, this will in general shape the reach set X_k at event step $k > 0$ as a union of finitely many DBM. For later use, we assume that X_k is a union of $|X_k|$ DBM and in particular that the set of initial conditions X_0 is a union of $|X_0|$ DBM.

3.1 Sequential Computation of the Reach Tube

This approach uses the one-step dynamics for autonomous MPL systems iteratively. In each step, we leverage the DBM representation and the PWA dynamics to compute the reach set.

Given a set of initial conditions X_0, the reach set X_k is recursively defined as the image of X_{k-1} w.r.t. the MPL dynamics as

$$X_k = \mathcal{I}(X_{k-1}) = \{A \otimes x : x \in X_{k-1}\} \ .$$

In the dynamical systems and automata literature the mapping $\mathcal{I}$ is also known as *Post* [32, Definition 2.3]. Under the assumption that X_0 is a union of finitely many DBM, by Corollary 1 it can be shown by induction that the reach set X_k is also a union of finitely many DBM, for each $k \in \mathbb{N}$.

Implementation: Given a state matrix A and a set of initial conditions X_0, the general procedure for obtaining the reach tube works as follows: first, we construct the PWA system generated by A; then, for each $k = 1, \ldots, N$, the reach set X_k is obtained by computing $\mathcal{I}(X_{k-1})$.

The worst-case complexity of the procedure (excluding that related to the generation of PWA system) can be assessed as follows. The complexity of computing $\mathcal{I}(X_{k-1})$ is $\mathcal{O}(|X_{k-1}| \cdot |\mathsf{R}(A)| \cdot n^3)$, for $k = 1, \ldots, N$. This results in an overall complexity of $\mathcal{O}(|\mathsf{R}(A)| \cdot n^3 \sum_{k=0}^{N-1} |X_k|)$. Notice that quantifying explicitly the cardinality $|X_k|$ of the DBM union at each step k is not possible in general (cf. Benchmark in Section 4).

The procedure has been implemented in `maxpl_reachtube_for`. The inputs are the PWA system (`A`, `B`, `D`), the initial states (`D0`), and the event horizon (`N`). The set of initial states `D0` is a collection of finitely many DBM and the event horizon `N` is a natural number. The output is a $1 \times (\mathtt{N}+1)$ cell. For each $1 \leq i \leq \mathtt{N}+1$, the i-th element contains the reach set X_{i-1}, which is a collection of finitely many DBM (cf. Section 2.3).

Let us consider the unit square as the set of initial conditions $X_0 = \{x \in \mathbb{R}^2 : 0 \leq x_1 \leq 1, 0 \leq x_2 \leq 1\}$. The following MATLAB script computes the reach tube for two steps:

```
>> Ampl = [2 5;3 3], [A,B,D] = maxpl2pwa(Ampl), N = 2
>> D0 = cell(1,2), ind = sub2ind([3,3],[1,2,0,0]+1,[0,0,1,2]+1)
>> D0{1} = Inf(3), D0{1}(1:4:9) = 0, D0{1}(ind) = [0,0,1,1]
>> D0{2} = false(3), D0{2}(1:4:9) = true, D0{2}(ind) = true
>> D0N = maxpl_reachtube_for(A,B,D,D0,N)
```

The reach sets are DBM given by $X_1 = \{x \in \mathbb{R}^2 : 1 \leq x_1 - x_2 \leq 2, 5 \leq x_1 \leq 6, 3 \leq x_2 \leq 4\}$, $X_2 = \{x \in \mathbb{R}^2 : 0 \leq x_1 - x_2 \leq 1, 8 \leq x_1 \leq 9, 8 \leq x_2 \leq 9\}$, and are shown in Fig. 2 (left). □

Recall that, given a set of initial conditions X_0 and a finite event horizon $N \in \mathbb{N}$, in order to compute X_N, we have to calculate $X_1, \ldots, X_{N-1}$. If the autonomous MPL system is irreducible, we can exploit the periodic behavior (cf. Proposition 1) to simplify the computation.

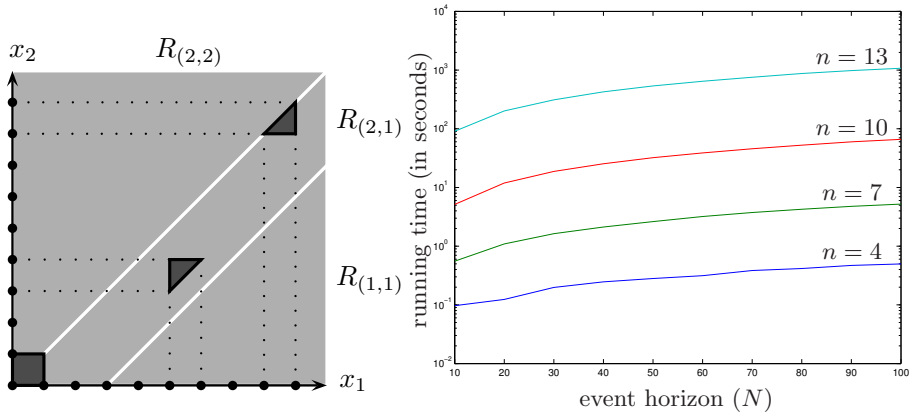

Fig. 2. (Left plot) Reach tube for autonomous MPL model over 2 event steps. (Right plot) Time needed to generate reach tube of autonomous models for different models size and event horizons, cf. Section 4.

Proposition 3 *Let $A \in \mathbb{R}_\varepsilon^{n\times n}$ be an irreducible matrix with max-plus eigenvalue $\lambda \in \mathbb{R}$ and cyclicity $c \in \mathbb{N}$. There exists a $k_0(X_0) = \max_{x\in X_0} k_0(x)$, such that $X_{k+c} = \lambda^{\otimes c} \otimes X_k$, for all $k \geq k_0(X_0)$.*

Proof. Recall that for each $x(0) \in \mathbb{R}^n$, there exists a $k_0(x(0))$ such that $x(k + c) = \lambda^{\otimes c} \otimes x(k)$, for all $k \geq k_0(x(0))$. Since $k_0(X_0) = \max_{x\in X_0} k_0(x)$, for each $x(0) \in X_0$, we have $x(k+c) = \lambda^{\otimes c}\otimes x(k)$, for $k \geq k_0(X_0)$. Recall from Definition 3 that $X_k = \{x(k) : x(0) \in X_0\}$, for all $k \in \mathbb{N}$. □

Thus if the autonomous MPL system is irreducible, we only need to compute $X_1, \ldots, X_{k_0(X_0)\wedge N}$ in order to calculate X_N, for any $N \in \mathbb{N}$, where $k_0(X_0)\wedge N = \min\{k_0(X_0), N\}$.

If the initial condition X_0 is a stripe, the infinite-horizon reach tube can be computed in a finite time, as stated in the following theorem.

Theorem 1 *Let $A \in \mathbb{R}_\varepsilon^{n\times n}$ be an irreducible matrix with cyclicity $c \in \mathbb{N}$. If X_0 is a union of finitely many stripes, then $\bigcup_{i=0}^{k_0(X_0)+c-1} X_i = \bigcup_{i=0}^{k} X_i$, for all $k \geq k_0(X_0) + c - 1$.*

Proof. First we will show that X_k is a union of finitely many stripes for all $k \in \mathbb{N}$. By using the procedure to compute the image of a DBM w.r.t. an affine dynamics, it can be shown that the image of a stripe w.r.t. affine dynamics (generated by an MPL model) is a stripe. Following the arguments after Theorem 2, it can be shown that the image of a union of finitely many stripes w.r.t. the PWA system generated by an MPL model is a union of finitely many stripes.

Since a stripe is a collection of equivalence classes [16, Sect. 1.4], then $X_0 \otimes \alpha = X_0$, for each $\alpha \in \mathbb{R}$. From Proposition 3 and the previous observations, $X_{k+c} = X_k$ for all $k \geq k_0(X_0)$. □

Example: The set of initial conditions can also be described as a stripe, for example $X_0 = \{x \in \mathbb{R}^2 : -1 \leq x_1 - x_2 \leq 1\}$. The reach sets are stripes given

by $X_1 = \{x \in \mathbb{R}^2 : 1 \leq x_1 - x_2 \leq 2\}$ and $X_2 = \{x \in \mathbb{R}^2 : 0 \leq x_1 - x_2 \leq 1\}$. Additionally, we obtain $X_1 = X_{2k-1}$ and $X_2 = X_{2k}$, for all $k \in \mathbb{N}$. It follows that the infinite-horizon reach tube is $\bigcup_{k=0}^{+\infty} X_k = \bigcup_{k=0}^{2} X_k = \{x \in \mathbb{R}^2 : -1 \leq x_1 - x_2 \leq 2\}$. □

3.2 One-Shot Computation of the Reach Set

In this section we discuss a procedure for computing the reach set for a specific event step N using a tailored (one-shot) procedure. Given a set of initial conditions X_0, we compute the reach set at event step N by using

$$X_N = (\mathcal{I} \circ \cdots \circ \mathcal{I})(A) = \mathcal{I}^N(A) = \{A^{\otimes N} \otimes x : x \in X_0\} \ .$$

Using Corollary 1, it can be seen that the reach set X_N is a union of finitely many DBM.

Implementation: Given a state matrix A, a set of initial conditions X_0 and a finite index N, the general procedure for obtaining X_N is: 1) computing $A^{\otimes N}$; then 2) constructing the PWA system generated by it; finally 3) computing the image of X_0 w.r.t. the obtained PWA system.

Let us quantify the total complexity of the first and third steps in the procedure. The complexity of computing N-th max-algebraic power of an $n \times n$ matrix (cf. Section 2.1) is $\mathcal{O}(\lceil \log_2(N) \rceil \cdot n^3)$. Excluding the generation of the PWA system – step 2), see above – the overall complexity of the procedure is $\mathcal{O}(\lceil \log_2(N) \rceil \cdot n^3 + |X_0| \cdot |\mathsf{R}(A^{\otimes N})| \cdot n^3)$.

The procedure has been implemented in `maxpl_reachset_for`. The inputs are the state matrix (`Ampl`), the initial states (`D0`), and the event horizon (`N`). The set of initial states `D0` is a collection of finitely many DBM (cf. Section 2.3) and the event horizon `N` is a natural number. The output is a 1×2 cell: the first element is the set of initial states and the second one is the reach set at event step `N`. Recall that both the initial states and the reach set are a collection of finitely many DBM.

Let us consider the unit square as the set of initial conditions $X_0 = \{x \in \mathbb{R}^2 : 0 \leq x_1 \leq 1, 0 \leq x_2 \leq 1\}$. The following MATLAB script computes the reach set for two steps:

```
>> Ampl = [2 5;3 3], N = 2
>> D0 = cell(1,2), ind = sub2ind([3,3],[1,2,0,0]+1,[0,0,1,2]+1)
>> D0{1} = Inf(3), D0{1}(1:4:9) = 0, D0{1}(ind) = [0,0,1,1]
>> D0{2} = false(3), D0{2}(1:4:9) = true, D0{2}(ind) = true
>> D0N = maxpl_reachset_for(Ampl,D0,N)
```

As expected, the reach set is a DBM given by $X_2 = \{x \in \mathbb{R}^2 : 0 \leq x_1 - x_2 \leq 1, 8 \leq x_1 \leq 9, 8 \leq x_2 \leq 9\}$. □

Intuitively, the sequential approach involves step-wise computations, and yields correspondingly more information than the one-shot procedure as an output. The complexities of both the sequential and one-shot computations depend

on the number of PWA regions corresponding to, respectively, the models related to matrix A and $A^{\otimes N}$. Thus, in order to compare the performance of both methods, we need to assess the cardinality of the PWA regions generated by $A^{\otimes k}$, for different values of k: from our experiments, it seems that the cardinality of PWA regions grows if k increases, hence the one-shot approach may not always result in drastic computational advantages. More work is needed to conclusively assess this feature.

4 Numerical Benchmark

4.1 Implementation and Setup

The technique for forward reachability computation on MPL models discussed in this work is implemented in the VeriSiMPL ("very simple") version 1.3, which is freely available at [28]. VeriSiMPL is a software tool originally developed to obtain finite abstractions of Max-Plus-Linear (MPL) models, which enables their verification against temporal specifications via a model checker. The algorithms have been implemented in MATLAB 7.13 (R2011b) and the experiments have been run on a 12-core Intel Xeon 3.47 GHz PC with 24 GB of memory.

In order to test the practical efficiency of the proposed algorithms, we compute the runtime needed to determine the reach tube of an autonomous MPL system, for event horizon $N = 10$ and an increasing dimension n of the MPL model. We also keep track of the number of regions of the PWA system generated from the MPL model. For any given n, we generate matrices A with 2 finite elements (in a max-plus sense) that are randomly placed in each row. The finite elements are randomly generated integers between 1 and 100. The set of initial conditions is selected as the unit hypercube, i.e. $\{x \in \mathbb{R}^n : 0 \leq x_1 \leq 1, \ldots, 0 \leq x_n \leq 1\}$.

Over 10 independent experiments, Table 1 reports the average time needed to generate the PWA system and to compute the reach tube, as well as the corresponding number of regions. As confirmed by Table 1, the time needed to compute the reach tube is monotonically increasing w.r.t. the dimension of the MPL model (as we commented previously this is not the case for the cardinality of reach sets, which hinges on the structure of the MPL models). For a fixed model size and dynamics, the growth of the computational time for forward reachability is linear (in the plot, logarithmic over logarithmic time scale) with the event horizon as shown in Fig. 2 (right). We have also performed reachability computations for the case of the set of initial conditions described as a stripe, which has yielded results that are analogue to those in Table 1.

4.2 Comparison with Alternative Reachability Computations

To the best of the authors knowledge, there exist no approaches for general forward reachability computation over MPL models. Forward reachability can be alternatively assessed only leveraging the PWA characterization of the model dynamics (cf. Section 2). Forward reachability analysis of PWA models can be best computed by the Multi-Parametric Toolbox (MPT, version 2.0) [25]. However, the toolbox has some implementation requirements: the state space matrix A has

Table 1. Numerical benchmark, autonomous MPL model: computation of the reach tube (average over 10 experiments)

size of MPL model	time for generation of PWA system	number of regions of PWA system	time for generation of reach tube	number of regions of X_{10}
3	0.09 [sec]	5.80	0.09 [sec]	4.20
5	0.14 [sec]	22.90	0.20 [sec]	6.10
7	0.52 [sec]	89.60	0.72 [sec]	13.40
9	2.24 [sec]	340.80	2.25 [sec]	4.10
11	10.42 [sec]	1.44×10^3	15.49 [sec]	3.20
13	46.70 [sec]	5.06×10^3	5.27 [min]	16.90
15	3.48 [min]	2.01×10^4	25.76 [min]	10.10
17	15.45 [min]	9.07×10^4	3.17 [hr]	68.70
19	67.07 [min]	3.48×10^5	7.13 [hr]	5.00

to be invertible – this is in general not the case for MPL models; the reach sets X_k have to be bounded – in our case the reach sets can be unbounded, particularly when expressed as stripes; further, MPT deals only with full-dimensional polytopes – whereas the reach sets of interest may not necessarily be so; finally, MPT handles convex regions and over-approximates the reach sets X_k when necessary – our approach computes instead the reach sets exactly.

For the sake of comparison, we have constructed randomized examples (with invertible dynamics) and run both procedures in parallel, with focus on computation time rather than the actual obtained reach tubes. Randomly generating the underlying dynamics allows deriving general results that are not biased towards possible structural features of the model. MPT can handle in a reasonable time frame models with dimension up to 10: in this instance (as well as lower-dimensional ones) we have obtained that our approach performs better (cf. Table 2). Notice that this is despite MPT being implemented in the *C* language, whereas VeriSiMPL runs in MATLAB: this leaves quite some margin of computational improvement to our techniques.

Table 2. Time for generation of the reach tube of 10-dimensional autonomous MPL model for different event horizons (average over 10 experiments)

event horizon	20	40	60	80	100
VeriSiMPL	11.02 [sec]	17.94 [sec]	37.40 [sec]	51.21 [sec]	64.59 [sec]
MPT	47.61 [min]	1.19 [hr]	2.32 [hr]	3.03 [hr]	3.73 [hr]

5 Conclusions and Future Work

This work has discussed the computation of forward reachability analysis of Max-Plus-Linear models by fast manipulations of DBM through PWA dynamics.

Computationally, we are interested in further optimizing the software for reachability computations, by leveraging symbolic techniques based on the use

of decision diagrams and by developing an implementation in the *C* language. We are presently exploring a comparison of the proposed approach with Flow* [14].

We plan to investigate *backward* reachability, as well as reachability of *non-autonomous* models, which embed non-determinism in the form of a control input, by tailoring or extending the techniques discussed in this work.

References

1. Alur, R., Dill, D.: A theory of timed automata. Theoretical Computer Science **126**(2) (1994) 183–235
2. Behrmann, G., David, A., Larsen, K.: A tutorial on UPPAAL. In Bernardo, M., Corradini, F., eds.: Formal Methods for the Design of Real-Time Systems (SFM-RT'04). Volume 3185 of Lecture Notes in Computer Science. Springer, Heidelberg (September 2004) 200–236
3. Kloetzer, M., Mahulea, C., Belta, C., Silva, M.: An automated framework for formal verification of timed continuous Petri nets. IEEE Trans. Ind. Informat. **6**(3) (2010) 460–471
4. Henzinger, T., Rusu, V.: Reachability verification for hybrid automata. In Henzinger, T., Sastry, S., eds.: Hybrid Systems: Computation and Control (HSCC'98). Volume 1386 of Lecture Notes in Computer Science. Springer, Heidelberg (1998) 190–204
5. Dang, T., Maler, O.: Reachability analysis via face lifting. In Henzinger, T., Sastry, S., eds.: Hybrid Systems: Computation and Control (HSCC'98). Volume 1386 of Lecture Notes in Computer Science. Springer, Heidelberg (1998) 96–109
6. Chutinan, A., Krogh, B.: Computational techniques for hybrid system verification. IEEE Trans. Autom. Control **48**(1) (January 2003) 64–75
7. CheckMate [Online]
8. Mitchell, I., Bayen, A., Tomlin, C.: A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games. IEEE Trans. Autom. Control **50**(7) (July 2005) 947–957
9. Mitchell, I.: Comparing forward and backward reachability as tools for safety analysis. In Bemporad, A., Bicchi, A., Buttazzo, G., eds.: Hybrid Systems: Computation and Control (HSCC'07). Volume 4416 of Lecture Notes in Computer Science., Springer, Heidelberg (2007) 428–443
10. Kurzhanskiy, A., Varaiya, P.: Ellipsoidal techniques for reachability analysis of discrete-time linear systems. IEEE Trans. Autom. Control **52**(1) (January 2007) 26–38
11. Kurzhanskiy, A., Varaiya, P.: Ellipsoidal toolbox. Technical report, EECS Department, University of California, Berkeley (May 2006)
12. Asarin, E., Schneider, G., Yovine, S.: Algorithmic analysis of polygonal hybrid systems, part I: Reachability. Theoretical Computer Science **379**(12) (2007) 231–265
13. Le Guernic, C., Girard, A.: Reachability analysis of hybrid systems using support functions. In Bouajjani, A., Maler, O., eds.: Computer Aided Verification (CAV'09). Volume 5643 of Lecture Notes in Computer Science. Springer, Heidelberg (2009) 540–554
14. Chen, X., Ábrahám, E., Sankaranarayanan, S.: Flow*: An analyzer for non-linear hybrid systems. In Sharygina, N., Veith, H., eds.: Computer Aided Verification (CAV'13). Volume 8044 of Lecture Notes in Computer Science. Springer, Heidelberg (2013) 258–263

15. Baccelli, F., Cohen, G., Olsder, G., Quadrat, J.P.: Synchronization and Linearity, An Algebra for Discrete Event Systems. John Wiley and Sons (1992)
16. Heidergott, B., Olsder, G., van der Woude, J.: Max Plus at Work–Modeling and Analysis of Synchronized Systems: A Course on Max-Plus Algebra and Its Applications. Princeton University Press (2006)
17. Roset, B., Nijmeijer, H., van Eekelen, J., Lefeber, E., Rooda, J.: Event driven manufacturing systems as time domain control systems. In: Proc. 44th IEEE Conf. Decision and Control and European Control Conf. (CDC-ECC'05). (December 2005) 446–451
18. Merlin, P., Farber, D.J.: Recoverability of communication protocols–implications of a theoretical study. IEEE Trans. Commun. **24**(9) (1976) 1036–1043
19. Plus, M.: Max-plus toolbox of Scilab [Online] (1998) Available at `www.cmap.polytechnique.fr/~gaubert/MaxplusToolbox.html`.
20. Gazarik, M., Kamen, E.: Reachability and observability of linear systems over max-plus. Kybernetika **35**(1) (1999) 2–12
21. Gaubert, S., Katz, R.: Reachability and invariance problems in max-plus algebra. In Benvenuti, L., De Santis, A., Farina, L., eds.: Positive Systems. Volume 294 of Lecture Notes in Control and Information Science. Springer, Heidelberg (April 2003) 15–22
22. Gaubert, S., Katz, R.: Reachability problems for products of matrices in semirings. International Journal of Algebra and Computation **16**(3) (2006) 603–627
23. Gaubert, S., Katz, R.: The Minkowski theorem for max-plus convex sets. Linear Algebra and its Applications **421**(2-3) (2007) 356–369
24. Zimmermann, K.: A general separation theorem in extremal algebras. Ekonom.-Mat. Obzor **13**(2) (1977) 179–201
25. Kvasnica, M., Grieder, P., Baotić, M.: Multi-parametric toolbox (MPT) (2004)
26. Dill, D.: Timing assumptions and verification of finite-state concurrent systems. In Sifakis, J., ed.: Automatic Verification Methods for Finite State Systems. Volume 407 of Lecture Notes in Computer Science. Springer, Heidelberg (1990) 197–212
27. Adzkiya, D., De Schutter, B., Abate, A.: Finite abstractions of max-plus-linear systems. IEEE Trans. Autom. Control **58**(12) (December 2013) 3039–3053
28. Adzkiya, D., Abate, A.: VeriSiMPL: Verification via biSimulations of MPL models. In Joshi, K., Siegle, M., Stoelinga, M., D'Argenio, P., eds.: Proc. 10th Int. Conf. Quantitative Evaluation of Systems (QEST'13). Volume 8054 of Lecture Notes in Computer Science., Springer, Heidelberg (September 2013) 253–256
29. Yordanov, B., Belta, C.: Formal analysis of discrete-time piecewise affine systems. IEEE Trans. Autom. Control **55**(12) (December 2010) 2834–2840
30. Charron-Bost, B., Függer, M., Nowak, T.: Transience bounds for distributed algorithms. In Braberman, V., Fribourg, L., eds.: Formal Modeling and Analysis of Timed Systems (FORMATS'13). Volume 8053 of Lecture Notes in Computer Science. Springer, Heidelberg (2013) 77–90
31. Sontag, E.D.: Nonlinear regulation: The piecewise-linear approach. IEEE Trans. Autom. Control **26**(2) (April 1981) 346–358
32. Baier, C., Katoen, J.P.: Principles of Model Checking. The MIT Press (2008)