Delft University of Technology

Delft Center for Systems and Control

Technical report 13-018

Finite abstractions of 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, "Finite abstractions of max-plus-linear systems," *IEEE Transactions on Automatic Control*, vol. 58, no. 12, pp. 3039–3053, Dec. 2013.

Delft Center for Systems and Control
Delft University of Technology
Mekelweg 2, 2628 CD Delft
The Netherlands
phone: +31-15-278.24.73 (secretary)
URL: `https://www.dcsc.tudelft.nl`

*This report can also be downloaded via `https://pub.deschutter.info/abs/13_018.html`

Finite Abstractions of Max-Plus-Linear Systems

Dieky Adzkiya, *Student Member, IEEE,* Bart De Schutter, *Senior Member, IEEE,*
and Alessandro Abate, *Member, IEEE*

***Abstract*—This work puts forward a novel technique to generate finite abstractions of autonomous and nonautonomous Max-Plus-Linear (MPL) models, a class of discrete-event systems used to characterize the dynamics of the timing related to successive events that synchronize autonomously. Nonautonomous versions of MPL models embed within their dynamics nondeterminism, namely a signal choice that is usually regarded as an exogenous control or schedule. In this paper, abstractions of MPL models are characterized as finite-state Labeled Transition Systems (LTS). LTS are obtained first by partitioning the state space (and, for the nonautonomous model, by covering the input space) of the MPL model and by associating states of the LTS to the introduced partitions, then by defining relations among the states of the LTS based on dynamical transitions between the corresponding partitions of the MPL state space, and finally by labeling the LTS edges according to the one-step timing properties of the events of the original MPL model. In order to establish formal equivalences, the finite abstractions are proven to either simulate or to bisimulate the original MPL model. This approach enables the study of general properties of the original MPL model by verifying (via model checking) equivalent logical specifications over the finite LTS abstraction. The computational aspects related to the abstraction procedure are thoroughly discussed and its performance is tested on a numerical benchmark.**

***Index Terms*—Max-plus algebra, discrete-event systems, piecewise affine models, labeled transition systems, model abstractions, difference-bound matrices, bisimulations, model checking.**

I. Introduction

MAX-PLUS-LINEAR (MPL) systems are a class of discrete-event systems [2], [3] with a continuous state space characterizing the timing of the underlying sequential discrete events. MPL models are predisposed to describe the timing synchronization between interleaved processes, under the assumption that timing events are linearly dependent (within the max-plus algebra) on previous event occurrences and (for nonautonomous models) on exogenous schedules (cf. Section II). MPL models are widely employed in the analysis and scheduling of infrastructure networks, such as communication and railway systems [4], production and manufacturing lines [5], [6], or biological systems [7]. They cannot model concurrency and are related to a subclass of Timed Petri Nets, namely Timed-Event Graphs [8].

The authors are with the Delft Center for Systems and Control, TU Delft - Delft University of Technology, The Netherlands. A. Abate is also with the Department of Computer Science, University of Oxford, United Kingdom. Emails: {d.adzkiya,b.deschutter,a.abate}@tudelft.nl

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. A subset of the presented material has appeared in [1].

Classical dynamical analysis of MPL models is grounded on their algebraic [9] or geometric features [10]. It allows investigating model properties such as its transient behavior, its periodic regimes, or its ultimate dynamical behavior [11]. This work explores a new, alternative approach to analysis that is based on finite-state abstractions [12] of autonomous and nonautonomous MPL models. Furthermore, by employing a novel representation of the quantities into play (regions over the state and the control spaces, as well as model dynamics), this work seeks to synthesize techniques that are computationally agile.

With regards to the abstraction procedure (cf. Section III), we put forward a new technique that generates a finite-state Transition System (TS) in a finite number of steps. The states of the TS are obtained by finite partitioning of the state space of the MPL model, whereas relations between pairs of TS states are established by checking whether a trajectory of the original MPL model is allowed to transition between the corresponding partitioning regions. The finite-state TS, however, may contain transitions that are unfeasible for the MPL model and as such may overapproximate the dynamics of the MPL model [13]. Computationally, this characterization is performed by forward-reachability analysis over a Piece-Wise Affine (PWA) representation of the MPL dynamics. PWA models [14] are characterized by vector fields that are affine (linear, plus an offset), within convex polytopes over the state and input spaces. In the nonautonomous (control-dependent) case, the technique embeds the one-step dynamics within an "augmented space" [4], namely the cross product of state and input spaces. The transitions between pairs of TS states thus depend on a choice of the input. In order to establish formal relationships between the concrete model and its abstraction, we argue that in general the LTS abstraction simulates the original MPL model [12], and furthermore we provide sufficient conditions to establish a bisimulation relation between abstract and concrete models [15]. We design a finite-time procedure with formally quantified complexity to construct it, if the sufficient conditions are fulfilled. The overall approach furthermore leverages a novel representation of the spatial regions and of the dynamics based on Difference-Bound Matrices (DBM) [16]. This representation allows for compact and computationally fast operations on regions of the state space, and thus for fast computation of the quantities of interest.

The approach to attain abstractions developed in this work is inspired by those developed for other models in [12], [17], [18], and can be interpreted in the context of literature focused on the construction of finite-state (quotient) models of given

systems. Notice that we leverage a PWA representation of the given MPL dynamics [19] – a particular case of the PWA model used in [18] – to build the finite-state LTS abstraction. However, techniques for abstractions of PWA systems developed in the literature [18] do not appear to be directly usable in the context of the models derived from MPL systems, since spatial boundaries can non-trivially affect the semantics of the trajectories [18, Remark 1] (see Section III). Likewise, related verification approaches developed for Timed Petri Nets (such as that for safety analysis based on existential zones [20]) do not appear to being exportable to MPL models.

Next, the TS abstraction is decorated with labels (cf. Section IV), which results in models, known as Labeled Transition Systems (LTS), that are particularly useful in computational formal verification techniques. The labels of the LTS model are defined in two possible ways: 1) they either characterize the difference between the timing of the same event for any two variables of the original MPL model, or 2) they represent the time difference between consecutive events of the MPL model. For a nonautonomous MPL model, it is furthermore possible to associate each transition with a label describing: 1) the subset of the outgoing partitioning region that is actively enabled by the transition under some input signals; or 2) the subset of the control inputs that actively leads states from the outgoing partitioning region to the destination region. This plurality of semantical definitions allows for flexibility in using the LTS abstraction for analysis of the original MPL model.

The computational aspects related to the abstraction procedure have been under particular scrutiny, and have brought to 1) the selection of DBM as a framework for the representation and manipulation of regions over the state and control spaces; and 2) the use of PWA representations of the MPL dynamics [19], which nicely couples with quantities expressed as DBM. The computational costs of the abstraction procedure are discussed in detail and its overall performance is benchmarked over a case study in Section V.

By expressing general dynamical properties as specifications in a modal logic such as Linear Temporal Logic (LTL) [15], we argue that the LTS abstraction allows for the formal verification of classes of properties by using model checking techniques. In particular, we focus on properties related to the cyclicity, the length of the transient part, and the ultimate behavior of the autonomous MPL model (cf. Section VI-A). More generally, LTL specifications over LTS can be efficiently model checked by a number of existing software tools – in this work we use the SPIN model checker [21] for our purposes. Moreover, LTS models in general are prone to be further simulated or bisimulated [15], which may lead to additional computational savings. Section VI elaborates a few examples to display the overall approach.

The abstraction technique developed in this work has been implemented as a MATLAB software tool, "Verification via biSimulations of MPL models" (VeriSiMPL, as in "very simple") [22], which is freely available for download at `http://sourceforge.net/projects/verisimpl/`

II. Models and Preliminaries

This section introduces the definition of an MPL model [8], recalls a few of its basic properties, and presents a representation of the MPL model by a PWA system. Finally, the notion of DBM [16], along with its properties, is formally described.

A. 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

$$\alpha\oplus\beta=\max\{\alpha,\beta\}\qquad\text{and}\qquad\alpha\otimes\beta=\alpha+\beta,$$

where the element ε is considered to be absorbing w.r.t. $\otimes$ [8, 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 [8, Sec. 3.1]. The basic max-algebraic operations are extended to matrices as follows. If $A,B\in\mathbb{R}_\varepsilon^{m\times n}$ and $C\in\mathbb{R}_\varepsilon^{n\times p}$,

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

for all i,j. Notice the analogy between $\oplus$, $\otimes$ and $+$, $\times$ for matrix and vector operations in conventional algebra. Given an $m\in\mathbb{N}$, the 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 equals to 0 (or $-\infty$) is also denoted by e (resp., ε). Furthermore, the state space is taken to be $\mathbb{R}^n$, which also implies that the state matrix A has to be row-finite (cf. Definition 2).

An autonomous MPL model [8, 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.

Related to matrix A is the notion of precedence (or communication) graph and of regular (or row-finite) matrix.

Definition 1 (Precedence graph, [8, Definition 2.8]): The precedence graph of $A\in\mathbb{R}_\varepsilon^{n\times n}$, denoted by $\mathcal{G}(A)$, is a weighted directed graph with vertices $1,\dots,n$ and an arc (j,i) with weight $A(i,j)$ for each $A(i,j)\neq\varepsilon$. □

Definition 2 (Regular (row-finite) matrix, [4, Sec. 1.2]): A 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. □

Example: Consider the two-dimensional MPL model of a simple railway network between two cities [4, Sec. 0.1]:

$$x(k)=\begin{bmatrix}2&5\\3&3\end{bmatrix}\otimes x(k-1).\tag{2}$$

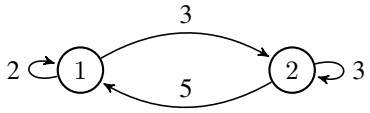


Fig. 1. Precedence graph of matrix A for the MPL model in (2)

The precedence graph of A is shown in Fig. 1 and A is a row-finite matrix. □

The notion of irreducible matrix, to be used shortly, can be given via that of precedence graph.

Definition 3 (Irreducible matrix, [8, Th. 2.14]): A matrix $A \in \mathbb{R}_{\varepsilon}^{n\times n}$ is called irreducible if its precedence graph $\mathcal{G}(A)$ is strongly connected. □

Recall that a directed graph is strongly connected if for any pair of different vertices i, j of the graph, there exists a path from i to j [8, p. 37]. From a max-algebraic perspective, 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), since this condition means that for two arbitrary vertices i and j of $\mathcal{G}(A)$ with $i \neq j$ there exists at least one path (of length 1, 2, … or $n-1$) from j to i.

Example: For the preceding example in (2), since $A(1,2) \neq \varepsilon \neq A(2,1)$, the matrix A is irreducible. Equivalently, notice that the precedence graph in Fig. 1 is strongly connected. □

If A is irreducible, there exists a unique max-plus eigenvalue $\lambda \in \mathbb{R}$. From a graph-theoretical point of view, the max-plus eigenvalue is defined as the maximum cycle mean of the associated precedence graph [8, Th. 3.23]. Algorithms have been developed to compute this quantity [23], [24, Sec. 4].

Proposition 1 (Length of the transient part, [4, 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 $k_0 \in \mathbb{N} \cup \{0\}$ and $c \in \mathbb{N}$ 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 length of the transient part $k_0(x(0))$, such that $x(k+c) = \lambda^{\otimes c} \otimes x(k)$, for all $k \geq k_0(x(0))$. Notice that we can seek a specific length of the transient part $k_0(x(0))$, in general less conservative than the global $k_0 = k_0(A)$, as in Proposition 1. Proposition 1 further implies there exists a max-plus base vector with a length of the transient part that equals k_0. Upper bounds for the length of the transient part k_0 and for its computation have been discussed in [25] and [26, Th. 10 and Th. 13].

Example: In the previous numerical example, characterized by the irreducible matrix A in (2), we have a max-plus eigenvalue $\lambda = 4$, a cyclicity $c = 2$, and a global length of the transient part $k_0 = 2$. □

Observe that we can associate with the max-plus eigenvalue of an MPL model characterized by an irreducible system matrix A an eigenspace, which: 1) is formally defined as $E(A) = \{x \in \mathbb{R}^n : A \otimes x = \lambda \otimes x\}$; 2) coincides with the periodic behavior associated with $c = 1$; and 3) is a linear combination (in a max-plus sense) of some vectors in $\mathbb{R}^n$ [8, Th. 3.100-3.101]. Furthermore, the complete periodic behaviors are encompassed by the eigenspace of $A^{\otimes c}$, where c is the cyclicity of A, which is formulated as $E(A^{\otimes c}) = \{x \in \mathbb{R}^n : A^{\otimes c} \otimes x = \lambda^{\otimes c} \otimes x\}$, and contains the eigenspace of A, i.e. $E(A) \subseteq E(A^{\otimes c})$.

A nonautonomous MPL model [8, Corollary 2.82] is defined by embedding an external input u in the dynamics of (1) as:

$$x(k) = A \otimes x(k-1) \oplus B \otimes u(k), \tag{3}$$

where $A \in \mathbb{R}_{\varepsilon}^{n\times n}$, $B \in \mathbb{R}_{\varepsilon}^{n\times m}$, $x(k-1) \in \mathbb{R}^n$, $u(k) \in \mathbb{R}^m$, for $k \in \mathbb{N}$. As suggested in [8, Sec. 2.5.4], the nonautonomous MPL model (3) can be transformed into an augmented MPL model

$$x(k) = \bar{A} \otimes \bar{x}(k-1), \tag{4}$$

where $\bar{A} = [A\ B]$, $\bar{x}(k-1) = [x(k-1)^T\ u(k)^T]^T$.

Example: A timetable can be incorporated in (2) as the input [4, p. 137]. We obtain a nonautonomous MPL model

$$x(k) = \begin{bmatrix} 2 & 5 \\ 3 & 3 \end{bmatrix} \otimes x(k-1) \oplus \begin{bmatrix} e & \varepsilon \\ \varepsilon & e \end{bmatrix} \otimes u(k). \tag{5}$$

The augmented MPL model is simply

$$x(k) = \begin{bmatrix} 2 & 5 & e & \varepsilon \\ 3 & 3 & \varepsilon & e \end{bmatrix} \otimes \bar{x}(k-1), \tag{6}$$

where $x(k) \in \mathbb{R}^2$ and $\bar{x}(k-1) \in \mathbb{R}^4$, for $k \in \mathbb{N}$. □

B. Piecewise-Affine Systems

This section discusses PWA systems generated by an autonomous and by a nonautonomous MPL model. In the abstraction procedure, the PWA system will be used to determine transitions (cf. Section III) and to compute labels (cf. Section IV) of the LTS.

Every autonomous MPL model as in (1) can be expressed as a PWA system in the event domain [19, Sec. 3]. A PWA system is described by a set of affine dynamics defined over a corresponding region in the state space. Both are characterized by the coefficients $g = (g_1, \ldots, g_n) \in \{1, \ldots, n\}^n$ or, more precisely, as:

$$R_g = \bigcap_{i,j=1}^{n} \{x \in \mathbb{R}^n : A(i, g_i) + x_{g_i} \geq A(i,j) + x_j\}; \tag{7}$$

$$x_i(k) = x_{g_i}(k-1) + A(i, g_i), \qquad 1 \leq i \leq n. \tag{8}$$

The set g of coefficients is found based on the region and the affine dynamics. A point $x \in \mathbb{R}^n$ is in R_g if $\max_j A(i,j) + x_j = A(i, g_i) + x_{g_i}$, for all i. Thus, as shown in (8), set g comprises the indices of the state variable x at event $k-1$.

Similarly, each nonautonomous MPL model (3) can be expressed as a PWA system in the event domain via its representation as an augmented MPL model. Each region with its affine dynamics is characterized by $\bar{g} = (\bar{g}_1, \ldots, \bar{g}_n) \in \{1, \ldots, n+m\}^n$ or, more precisely, as:

$$\bar{R}_{\bar{g}} = \bigcap_{i=1}^{n} \bigcap_{j=1}^{n+m} \{\bar{x} \in \mathbb{R}^{n+m} : \bar{A}(i, \bar{g}_i) + \bar{x}_{\bar{g}_i} \geq \bar{A}(i,j) + \bar{x}_j\};$$

$$x_i(k) = \bar{x}_{\bar{g}_i}(k-1) + \bar{A}(i, \bar{g}_i), \qquad 1 \leq i \leq n. \tag{9}$$

input: $A \in \mathbb{R}_{\varepsilon}^{n\times p}$, a row-finite max-plus matrix
output: $\mathsf{R}, \mathsf{A}, \mathsf{B}$, a PWA system over $\mathbb{R}^p$
1: **function** [R,A,B] = MPL2PWA(A)
2: $\mathsf{R} \leftarrow \emptyset$, $\mathsf{A} \leftarrow \emptyset$, $\mathsf{B} \leftarrow \emptyset$ ▷ $\emptyset$ is an empty collection
3: **for all** $(g_1, \ldots, g_n) \in \{1, \ldots, p\}^n$ **do**
4: $R_g \leftarrow \mathbb{R}^p$, $A_g \leftarrow \text{zeros}(n, p)$, $B_g \leftarrow \text{zeros}(n, 1)$
5: **for all** $1 \le i \le n$ **do**
6: **for all** $1 \le j \le p$ **do** ▷ define regions (7)
7: $R_g \leftarrow R_g \cap \{A(i, g_i) + x_{g_i} \ge A(i, j) + x_j\}$
8: **end for**
9: $A_g(i, g_i) \leftarrow 1$, $B_g(i) \leftarrow A(i, g_i)$ ▷ eqn (8)
10: **end for**
11: **if** R_g is not empty **then** ▷ check emptiness
12: $\mathsf{R} \leftarrow \mathsf{R} \cup \{R_g\}$, $\mathsf{A} \leftarrow \mathsf{A} \cup \{A_g\}$, $\mathsf{B} \leftarrow \mathsf{B} \cup \{B_g\}$
13: **end if**
14: **end for**
15: **end function**

Fig. 2. Algorithm generating a PWA system from a row-finite MPL matrix

Given a row-finite max-plus matrix A, the algorithm in Fig. 2[1] describes a general procedure to construct a PWA system corresponding to the autonomous MPL model. Similarly, if we run the algorithm with the augmented matrix $\bar{A}$, we obtain a PWA system related to the nonautonomous MPL model. Notice that the affine dynamics associated with a dynamical system generated by Algorithm 2 are a special case of the general PWA dynamics as defined in [14, Sec. 1].

It can be shown that if the intersection of two regions generated by Algorithm 2 is not empty, then its dimension is strictly less than p (here p is a parameter that can be set to be equal to n in the autonomous case and to $n + m$ in the nonautonomous case). Indeed, let $g = (g_1, \ldots, g_n), g' = (g'_1, \ldots, g'_n) \in \{1, \ldots, p\}^n$, where $R_g \cap R_{g'}$ is not empty and $g \neq g'$ – without loss of generality, let us assume that $g_1 \neq g'_1$. From step 7, $R_g \subseteq \{x \in \mathbb{R}^p : x_{g'_1} - x_{g_1} \le A(1, g_1) - A(1, g'_1)\}$ and $R_{g'} \subseteq \{x \in \mathbb{R}^p : x_{g_1} - x_{g'_1} \le A(1, g'_1) - A(1, g_1)\}$. Thus, $R_g \cap R_{g'} \subseteq \{x \in \mathbb{R}^p : x_{g'_1} - x_{g_1} = A(1, g_1) - A(1, g'_1)\}$, which corresponds to a lower-dimensional subspace. Algorithm 2 is a finite-time procedure, since the maximum number of iterations (steps 3, 5 and 6) is finite and the computations over convex polytopes involved in steps 7 and 11 are also finite. Its complexity is formally quantified in the next section.

Example: Considering the autonomous MPL model defined in (2), the nonempty regions of the PWA system are: $R_{(1,1)} = \{x \in \mathbb{R}^2 : x_1 - x_2 \ge 3\}$, $R_{(2,1)} = \{x \in \mathbb{R}^2 : e \le x_1 - x_2 \le 3\}$ and $R_{(2,2)} = \{x \in \mathbb{R}^2 : x_1 - x_2 \le e\}$, as depicted in Fig. 3. Region $R_{(1,2)}$ does not appear since it corresponds to an empty set. Notice that the overlap between regions is over sets of zero measure (lines). As explained above, the affine dynamics corresponding to a region are characterized by set g: for example the affine dynamics of $R_{(2,1)}$ are given by $x_1(k) = x_2(k-1)+5$, $x_2(k) = x_1(k-1)+3$. Similarly, for the nonautonomous MPL model in (5), the nonempty regions of the corresponding PWA system are: $\bar{R}_{(1,1)} = \{\bar{x} \in \mathbb{R}^4 : x_1 - x_2 \ge 3,\ x_1 - u_1 \ge -2,\ x_1 - u_2 \ge -3\}$; $\bar{R}_{(1,4)} = \{\bar{x} \in \mathbb{R}^4 : x_1 - x_2 \ge 3,\ x_1 - u_1 \ge -2,\ x_1 - u_2 \le -3,\ x_2 - u_2 \le -3\}$; $\bar{R}_{(2,1)} = \{\bar{x} \in \mathbb{R}^4 : e \le x_1 - x_2 \le 3,\ x_1 - u_2 \ge -3,\ x_2 - u_1 \ge -5\}$; $\bar{R}_{(2,2)} = \{\bar{x} \in \mathbb{R}^4 : x_1 - x_2 \le e,\ x_2 - u_1 \ge -5,\ x_2 - u_2 \ge -3\}$; $\bar{R}_{(2,4)} = \{\bar{x} \in \mathbb{R}^4 : x_1 - x_2 \le 3,\ x_1 - u_2 \le -3,\ x_2 - u_1 \ge -5,\ x_2 - u_2 \le -3\}$; $\bar{R}_{(3,1)} = \{\bar{x} \in \mathbb{R}^4 : x_1 - x_2 \ge e,\ x_1 - u_1 \le -2,\ x_1 - u_2 \ge -3,\ x_2 - u_1 \le -5\}$; $\bar{R}_{(3,2)} = \{\bar{x} \in \mathbb{R}^4 : x_1 - x_2 \le e,\ x_1 - u_1 \le -2,\ x_2 - u_1 \le -5,\ x_2 - u_2 \ge -3\}$; $\bar{R}_{(3,4)} = \{\bar{x} \in \mathbb{R}^4 : x_1 - u_1 \le -2,\ x_1 - u_2 \le -3,\ x_2 - u_1 \le -5,\ x_2 - u_2 \le -3\}$. □

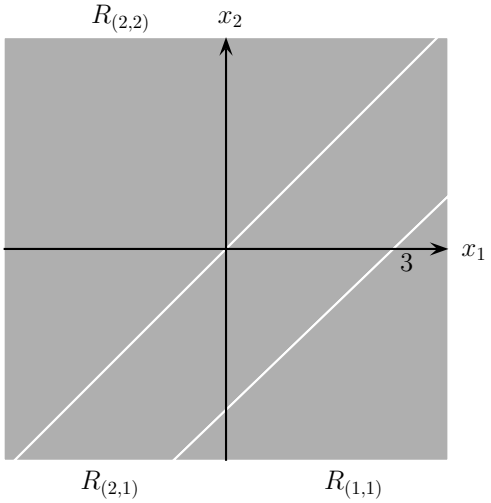


Fig. 3. Regions associated with the PWA system generated by the autonomous MPL model in (2)

C. Difference-Bound Matrices

This section introduces the definition of a DBM [16, Sec. 4.1] and of its canonical-form representation. DBM will be used in Section III-B, III-C2, and IV-B to represent partitioning regions, the set of inputs, and labels, respectively.

Definition 4 (Difference-bound matrix): A DBM in $\mathbb{R}^n$ is the intersection of finitely many sets defined by $x_i - x_j \bowtie_{i,j} \alpha_{i,j}$, where $\bowtie_{i,j} \in \{<, \le\}$, $\alpha_{i,j} \in \mathbb{R} \cup \{+\infty\}$, for $1 \le i \ne j \le n$. □

Notice that the DBM in Definition 4 is a special case of that in [16, Sec. 4.1]. Definition 4 can be likewise given over the input and augmented spaces. Each DBM admits an equivalent and unique canonical-form representation, which is a DBM with the tightest possible bounds [16, Sec. 4.1]. As intuitive, a DBM in $\mathbb{R}^n$ can be represented by an n-dimensional matrix D (namely, a "potential graph" [27, Sec. 3]). Each element $D(i, j)$ represents a right-bounded interval for the set of possible values of $x_i - x_j$. The interval is characterized by its upper bound $\alpha_{i,j}$ and by the operator $\bowtie_{i,j}$. Since computing the canonical-form representation of a DBM is equivalent to the all-pairs shortest path problem over the corresponding potential graph [16, Sec. 4.1], the Floyd-Warshall algorithm [28] can be used over the graph with a complexity that is cubic w.r.t. its dimension.

Example: Consider the PWA system generated by the nonautonomous MPL model (5). A few regions are not in

[1]For simplicity, when referring to an algorithm we use the term "Algorithm 2" rather than "the Algorithm in Fig. 2".

the canonical-form representation, and can then be expressed as follows: $\mathrm{cf}(\bar{R}_{(1,4)}) = \{\bar{x} \in \mathbb{R}^4 : x_1 - x_2 \geq 3,\ x_1 - u_1 \geq -2,\ x_1 - u_2 \leq -3,\ x_2 - u_2 \leq -6,\ u_1 - u_2 \leq -1\}, \mathrm{cf}(\bar{R}_{(2,1)}) = \{\bar{x} \in \mathbb{R}^4 : e \leq x_1 - x_2 \leq 3,\ x_1 - u_1 \geq -5,\ x_1 - u_2 \geq -3,\ x_2 - u_1 \geq -5,\ x_2 - u_2 \geq -6\}, \mathrm{cf}(\bar{R}_{(2,4)}) = \{\bar{x} \in \mathbb{R}^4 : x_1 - x_2 \leq 3,\ x_1 - u_2 \leq -3,\ x_2 - u_1 \geq -5,\ x_2 - u_2 \leq -3,\ u_1 - u_2 \leq 2\}, \mathrm{cf}(\bar{R}_{(3,1)}) = \{\bar{x} \in \mathbb{R}^4 : x_1 - x_2 \geq e,\ x_1 - u_1 \leq -2,\ x_1 - u_2 \geq -3,\ x_2 - u_1 \leq -5,\ u_1 - u_2 \geq -1\}, \mathrm{cf}(\bar{R}_{(3,2)}) = \{\bar{x} \in \mathbb{R}^4 : x_1 - x_2 \leq e,\ x_1 - u_1 \leq -5,\ x_2 - u_1 \leq -5,\ x_2 - u_2 \geq -3,\ u_1 - u_2 \geq 2\}$, where cf is a generic operator yielding the canonical form [16, Sec. 4.1]. Other regions appear already in canonical form, for instance $\bar{R}_{(1,1)} = \mathrm{cf}(\bar{R}_{(1,1)})$. □

One advantage of the canonical-form representation is that it is straightforward to compute orthogonal projections w.r.t. a subset of its variables. This is simply performed by deleting rows and columns corresponding to the complementary variables [16, Sec. 4.1]. The orthogonal projection of a DBM in canonical-form is again in canonical form [16, Observation 1].

Definition 5 (Orthogonal projection): The orthogonal projection w.r.t. state variables (input variables) of a region in the augmented space is defined as $\Pi_X : \mathbb{R}^{n+m} \to \mathbb{R}^n$ ($\Pi_U : \mathbb{R}^{n+m} \to \mathbb{R}^m$), where $\Pi_X : [x_1 \dots u_m]^T \mapsto [x_1 \dots x_n]^T$ ($\Pi_U : [x_1 \dots u_m]^T \mapsto [u_1 \dots u_m]^T$). □

Another advantage of the canonical-form representation is that it is fast to check its emptiness. By using the potential graph representation, the unfeasible sets of constraints are only those which form a circuit with a strictly negative weight in the graph. As a consequence, in order to test whether a DBM is empty or not, we simply have to check for the existence of such a circuit: this can be achieved by the Bellman-Ford algorithm [29, Sec. 5], which is cubic w.r.t. its dimension. Whenever a DBM is in canonical form, testing for strictly negative cycles can be reduced to check whether there is an i such that $\bowtie_{i,i}$ is $<$ or $\alpha_{i,i} < e$. Thus, its complexity is linear w.r.t. its dimension.

Each region and the corresponding affine dynamics of the PWA system generated by Algorithm 2 (for both autonomous and nonautonomous MPL models) can be characterized by a DBM. More precisely, from step 7 of Algorithm 2, each region of the PWA system generated by a row-finite max-plus matrix is a DBM in $\mathbb{R}^p$. From step 9 of Algorithm 2, the affine dynamics can be expressed, by virtue of simple algebraic manipulations (each equation can be expressed as the difference between variables at time k and $k-1$), via a DBM in $\mathbb{R}^{n+p}$.

Looking back at Algorithm 2, its worst-case complexity can be precisely formulated as follows. Notice that the maximum number of iterations in steps 3, 5, and 6 is p^n, n, and p, respectively. Furthermore, the complexity in steps 7 and 11 is constant and amounts to $\mathcal{O}(p^3)$, respectively. Thus, the worst-case complexity of Algorithm 2 is $\mathcal{O}(p^n(np + p^3))$.

Theorem 1: The image and the inverse image of a DBM w.r.t. affine dynamics is a DBM.

Proof: The general procedure to compute the image and the inverse image of a DBM w.r.t. affine dynamics involves: 1) computing the intersection of the DBM with a DBM generated by the expression of affine dynamics; then 2) calculating the canonical form of the intersection; finally 3) projecting the canonical-form representation over a subset of its variables. The claim follows by noticing that the intersection of two DBM is a DBM, that the canonical form of a DBM is a DBM and that the orthogonal projection of a DBM is again a DBM. □

It is possible to quantify the worst-case complexity related to the computation of the image and of the inverse image of a DBM w.r.t. affine dynamics. In both cases the overall worst-case complexity critically depends on computing the canonical-form representation (the second step) and is cubic w.r.t. the sum of its dimension and of the dimension of the (inverse) image.

As it will become clear in Section III-C, we need to relate the notion of DBM and that of projective space [4, Sec. 1.4].

Lemma 1: Let D be a DBM, then $\beta \otimes x \in D$ for each $\beta \in \mathbb{R}$ and $x \in D$.

Proof: Since $[\beta \otimes x]_i - [\beta \otimes x]_j = (\beta + x_i) - (\beta + x_j) = x_i - x_j$ for each i, j, then $\beta \otimes x \in D$ (cf. Definition 4). □

In order to apply formal verification tools on LTS models (cf. Section VI), we may need to check inclusion properties over DBM. These are characterized by a partial order on DBM, denoted by $D \subseteq D'$ iff $D(i,j) \subseteq D'(i,j)$, for all i and j [16, Sec. 4.1]: in other words, $D \subseteq D'$ whenever the interval associated with $D(i,j)$ is a subset of the interval associated with $D'(i,j)$. Furthermore, the smallest DBM containing a union of q DBM can be computed using the following procedure, which has complexity $\mathcal{O}(n^2 q)$.

Theorem 2: The smallest DBM containing the union of finitely many DBM $\bigcup_{k=1}^{q} D_k$ is given by $D(i,j) = \bigcup_{k=1}^{q} D_k(i,j)$, for $1 \leq i \neq j \leq n$.

Proof: Let D' be a DBM containing $\bigcup_{k=1}^{q} D_k$, then $D_k \subseteq D'$ for each $1 \leq k \leq q$. By using the DBM inclusion property, we know that $D_k(i,j) \subseteq D'(i,j)$ for each $1 \leq i \neq j \leq n$, $1 \leq k \leq q$. It follows that $\bigcup_{k=1}^{q} D_k(i,j) \subseteq D'(i,j)$ or equivalently, $D(i,j) \subseteq D'(i,j)$. Thus, from the DBM inclusion property, $D \subseteq D'$. □

III. MPL Abstractions as Transition Systems

This section starts by introducing TS models and the notions of simulation and bisimulation, then proceeds describing the abstraction procedure – constructing a finite-state or a quotient TS.

A. Transition Systems

Definition 6 (Transition system): A TS is a tuple (S, δ, I) that consists of

- a (possibly uncountable) set S of states;
- a transition relation $\delta \subseteq S \times S$;
- a set $I \subseteq S$ of initial states. □

If the cardinality of S, denoted by $|S|$, is infinite, it is called an infinite-state TS. On the other hand, if $|S| < \infty$, it is called a finite-state TS. If for each state there is at most one outgoing transition, then the TS is called deterministic. Otherwise, the TS is said to be nondeterministic. Initial states are useful for verification purposes (cf. Section VI). As it will become clear in Section III-C, we do not require a single initial state as

in [15, Definition 2.5]. Unless otherwise stated, we will assume that $I = S$. We provide an extension of the notion of TS to a labeled version (LTS) in Section IV.

Inspired by the work on bisimilar dynamical models in [12, Definition 4], let us introduce a TS (S, δ, I) related to an autonomous MPL model. The TS is characterized by a set of states $S = \mathbb{R}^n$. There exists a transition relation δ from x to x', denoted by $(x, x') \in \delta$, if and only if $x' = A \otimes x$. Since for each $x \in \mathbb{R}^n$, $A \otimes x$ is uniquely defined, the TS is deterministic. With focus on a TS generated by a nonautonomous MPL model, there exists a transition from x to x' if and only if there exists an input u such that $x' = A \otimes x \oplus B \otimes u$ – in general this renders the TS nondeterministic.

Given a TS (S, δ, I) and an equivalence relation $\sim$, the definition of quotient TS $(S/_\sim, \delta_\sim, I_\sim)$ is straightforward [15, Definition 7.51]. The set of equivalence classes is denoted by $S/_\sim$. The transition relation $\delta_\sim$ will be induced from δ, as discussed in Section III-C. The initial states $I_\sim$ are defined as $\{s_\sim \in S/_\sim : I \cap s_\sim \neq \emptyset\}$.

Definition 7 (Simulation): The quotient TS $(S/_\sim, \delta_\sim, I_\sim)$ simulates the original TS (S, δ, I) if for each $(x, x') \in \delta$, there exists a pair $(s_\sim, s'_\sim) \in \delta_\sim$ such that $x \in s_\sim$ and $x' \in s'_\sim$. □

Definition 8 (Bisimulation): The quotient TS $(S/_\sim, \delta_\sim, I_\sim)$ bisimulates the original TS (S, δ, I) if it simulates it and if, for each pair $(s_\sim, s'_\sim) \in \delta_\sim$, the following holds: for each $x \in s_\sim$, there exists an $x' \in s'_\sim$ such that $(x, x') \in \delta$. □

We seek to construct specifically a *finite-state* TS, derived from either the autonomous or the nonautonomous MPL model, which simulates and, under certain conditions, also bisimulates the TS (S, δ, I) (and thus also the original MPL model). This objective is pursued next.

B. States: Partitioning Procedure

We discuss different approaches to construct $S/_\sim$, where its elements are DBM in $\mathbb{R}^n$, then prove their relationship: the first technique focuses on autonomous MPL models, while the second works with nonautonomous MPL models.

1) Autonomous Case: We determine a partitioning of the state space based on the value of $A(i,j) + x_j$, similar to Section II-B. Given an autonomous MPL model characterized by a row-finite max-plus matrix $A \in \mathbb{R}_\varepsilon^{n \times n}$ and a generic $x \in \mathbb{R}^n$, for notational purposes we define $W_x(i,j) = A(i,j) + x_j - [A \otimes x]_i$. Notice that each element of W_x is nonpositive, depends (given a matrix A) only on x, and that there exists a nonempty set of null (e) elements in each of its rows. Each region generated by this approach is characterized by a parameter set $f = (f_1, \ldots, f_n) \in (\mathcal{P}(\{1, \ldots, n\}) \setminus \{\emptyset\})^n$, where $f_i = \{j : W_x(i,j) = e\} = \{j : [A \otimes x]_i = A(i,j) + x_j\}$ for $1 \leq i \leq n$, and where $\mathcal{P}$ denotes the power-set operator. More precisely, the region characterized by f, denoted by R_f, is defined as the set of points $x \in \mathbb{R}^n$ verifying the condition for matrix W_x, i.e. $R_f = \{x \in \mathbb{R}^n$: for each $1 \leq i \leq n, W_x(i,j) = e$ iff $j \in f_i\}$.

In order to design a procedure for the proposed approach, we need to characterize each point $x \in R_f$ based on the value of $A(i,j) + x_j$. For each $1 \leq i \leq n$, $j \in f_i$ and $1 \leq j' \leq n$, the following property holds: if $j' \in f_i$, then $A(i,j) + x_j = A(i,j') + x_{j'}$; if $j' \notin f_i$, then $A(i,j) + x_j > A(i,j') + x_{j'}$. Thus a constructive definition of $R_f \subseteq \mathbb{R}^n$ is as follows:

$$\bigcap_{i=1}^{n} \bigcap_{j \in f_i} \bigcap_{j'=1}^{n} \begin{cases} \{A(i,j) + x_j = A(i,j') + x_{j'}\}, & \text{if } j' \in f_i, \\ \{A(i,j) + x_j > A(i,j') + x_{j'}\}, & \text{if } j' \notin f_i. \end{cases}$$

input: $A \in \mathbb{R}_\varepsilon^{n \times n}$, a row-finite max-plus matrix
output: $S/_\sim$, a partition of $\mathbb{R}^n$

1: $S/_\sim \leftarrow \emptyset$ ▷ initialize $S/_\sim$ with an empty collection
2: **for all** $(f_1, \ldots, f_n) \in (\mathcal{P}(\{1, \ldots, n\}) \setminus \{\emptyset\})^n$ **do**
3: $R_f \leftarrow \mathbb{R}^n$ ▷ notice that $\mathbb{R}^n$ is a DBM
4: **for all** $1 \leq i \leq n$ **do** ▷ constructive definition of R_f
5: **for all** $j \in f_i$, $1 \leq j' \leq n$ **do**
6: **if** $j' \in f_i$ **then**
7: $R_f \leftarrow R_f \cap \{A(i,j) + x_j = A(i,j') + x_{j'}\}$
8: **else if** $j' \notin f_i$ **then**
9: $R_f \leftarrow R_f \cap \{A(i,j) + x_j > A(i,j') + x_{j'}\}$
10: **end if**
11: **end for**
12: **end for**
13: **if** R_f is not empty **then** ▷ check emptiness
14: $S/_\sim \leftarrow S/_\sim \cup \{R_f\}$ ▷ add the region to $S/_\sim$
15: **end if**
16: **end for**

Fig. 4. Generation of the partition of the state space of an autonomous MPL model

Algorithm 4 details the procedure to compute the collection of regions by using the proposed approach. The following property characterizes its output:

Theorem 3: Algorithm 4 generates a partition of $\mathbb{R}^n$.

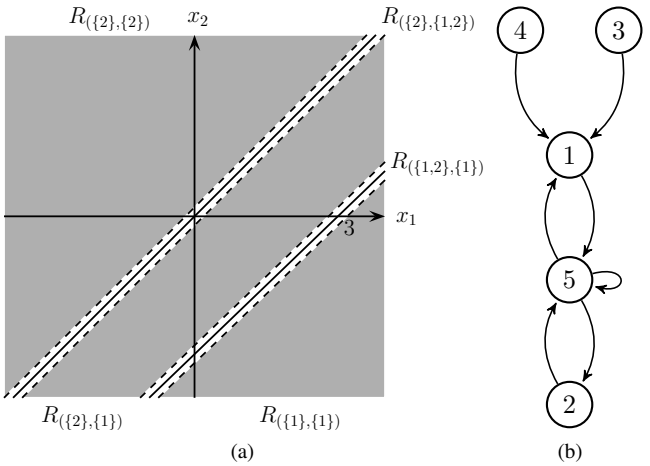
Proof: First, we show that the collection of the regions is a cover of $\mathbb{R}^n$. Let $x \in \mathbb{R}^n$ be arbitrary. Based on x, we can compute W_x and f, which means that there is an R_f : $x \in R_f$. Next, we show (by contradiction) that the regions are pairwise disjoint. Notice that each point x corresponds to a unique characterization f, since each element of W_x depends only on x and the characterization f depends only on W_x. Let us then suppose that $x \in \mathbb{R}^n$ is an arbitrary element in the intersection of two regions: then x would correspond to two different region characterizations, which is a contradiction. □

Algorithm 4 allows to derive a general representation of each region, which can be expressed as a set of linear inequalities of the following form:

$$\alpha_{i,j} \bowtie_{i,j} x_i - x_j \bowtie_{i,j} \beta_{i,j}, \tag{10}$$

where $\alpha_{i,j}, \beta_{i,j} \in \mathbb{R} \cup \{-\infty, +\infty\}$; $\bowtie_{i,j}$ is $<$ if $\alpha_{i,j} \neq \beta_{i,j}$, whereas $\bowtie_{i,j}$ is $\leq$ if $\alpha_{i,j} = \beta_{i,j}$, for $1 \leq i < j \leq n$. Notice that each region generated by the approach is a DBM in $\mathbb{R}^n$ (cf. Section II-C), which can be expressed in its canonical form.

Let us quantify the worst-case complexity of Algorithm 4. Notice that the maximum number of iterations in steps 2, 4, 5 and 13 is $(2^n - 1)^n$, n, n^2 and n^3, respectively. Thus, the worst-case complexity is exponential, or more precisely $\mathcal{O}(n^3(2^n - 1)^n)$. However, this worst case is rarely incurred in



(a) (b)

Fig. 5. (a) Partitioning regions obtained using Algorithm 4. (b) TS transitions, where states 1, ..., 5 represent respectively $R_{(\{2\},\{2\})}$, $R_{(\{2\},\{1,2\})}$, $R_{(\{1\},\{1\})}$, $R_{(\{1,2\},\{1\})}$, $R_{(\{2\},\{1\})}$. The (nondeterministic) TS and its partitioning regions generated as an abstraction of the autonomous MPL model in (2)

practice. In order to improve the performance of the approach, we have applied standard pruning tricks [30, Sec. 3], which practically results in efficient outcomes, as will be shown with the benchmark in Section V.

Example: Consider the autonomous MPL model in (2). The regions generated by the scheme in Algorithm 4 are $R_{(\{1\},\{1\})} = \{x \in \mathbb{R}^2 : x_1 - x_2 > 3\}$, $R_{(\{1,2\},\{1\})} = \{x \in \mathbb{R}^2 : x_1 - x_2 = 3\}$, $R_{(\{2\},\{1\})} = \{x \in \mathbb{R}^2 : e < x_1 - x_2 < 3\}$, $R_{(\{2\},\{1,2\})} = \{x \in \mathbb{R}^2 : x_1 - x_2 = e\}$, $R_{(\{2\},\{2\})} = \{x \in \mathbb{R}^2 : x_1 - x_2 < e\}$. The regions are shown in Fig. 5a. □

Looking back at Section II-B, each region can be related to the affine dynamics in (8) of a PWA system generated by the autonomous MPL model. Such dynamics can also be characterized by the matrix W_x introduced in this section: more precisely, for each $x \in \mathbb{R}^n$, the dynamics in (8) are characterized by g if and only if $W_x(i, g_i) = e$, for $1 \le i \le n$. Furthermore, the region characterized by g is defined as $R_g = \{x \in \mathbb{R}^n : \text{for each } 1 \le i \le n, W_x(i, j) = e \text{ if } j = g_i\}$. The above relationship can be formalized as follows.

Proposition 2: For each $g \in \{1, \dots, n\}^n$, $R_g = \bigcup_{f \in F(g)} R_f$, where $F(g) = \{f : \text{for each } 1 \le i \le n, g_i \in f_i\}$. Furthermore, for each $f \in (\mathcal{P}(\{1, \dots, n\}) \setminus \{\emptyset\})^n$, $R_f = (\cap_{g \in G(f)} R_g) \setminus (\cup_{g' \in G'(f)} R_{g'})$, where $G(f) = \{g : \text{for each } 1 \le i \le n, g_i \in f_i\}$ and $G'(f) = \{g' : \text{there exists an } 1 \le i \le n \text{ such that } g'_i \notin f_i\}$.

Proof: For the first statement, notice that $R_g = \{x \in \mathbb{R}^n : \text{for each } 1 \le i \le n, W_x(i, g_i) = e\} = \bigcup_{f \in F(g)} \{x \in \mathbb{R}^n : \text{for each } 1 \le i \le n, W_x(i, j) = e,\ \forall j \in f_i\} = \bigcup_{f \in F(g)} R_f$.

For the second statement, it can be shown that the condition for matrix W_x associated with $(\cap_{g \in G(f)} R_g) \setminus (\cup_{g' \in G'(f)} R_{g'})$ and R_f are the same. Notice that $\cap_{g \in G(f)} R_g = \{x \in \mathbb{R}^n : \text{for each } 1 \le i \le n, W_x(i, j) = e \text{ if } j \in f_i\}$. The set difference guarantees the "only if" part. □

It follows that each region R_f generated by the scheme in Algorithm 4 is a subset of a unique collection of regions in the PWA system generated by the autonomous MPL model (recall that the collection of regions in the PWA system is a cover of $\mathbb{R}^n$).

input: $A \in \mathbb{R}_{\varepsilon}^{n \times n}$, a row-finite max-plus matrix
output: $S/_\sim$, a partition of $\mathbb{R}^n$

```
S/∼ ← MPL2PWA(A)                          ▷ Algorithm 2
for all R_g, R_g' ∈ S/∼ do
    if R_g > R_g' then                    ▷ Definition 9
        R_g' ← R_g' \ R_g                 ▷ Proposition 3
    else if R_g' > R_g then
        R_g ← R_g \ R_g'
    end if
end for
remove the empty regions from S/∼
```

Fig. 6. Generation of state-space partitioning via refinement of regions generated by the state space matrix

A partitioning of the state space can also be obtained by refining the regions of the PWA system generated by the state matrix. The refinement procedure obtaining a partition is not unique: with focus on memory usage, we propose one that leads in general to a partition that is smaller (in cardinality) than the one generated by Algorithm 4. Let us start with the following concept.

Definition 9 (Adjacent regions): Let R_g and $R_{g'}$ be regions generated by an n-dimensional state space matrix. We say that they are adjacent ($R_g > R_{g'}$) if there exists a single $1 \le i \le n$ such that $g_i > g'_i$ and $g_j = g'_j$ for each $j \ne i$. □

Given a collection of regions generated by the state space matrix using Algorithm 2, the refinement procedure (cf. Algorithm 6) works as follows. For each pair of adjacent regions, their intersection is combined to the region with higher index. It can be shown that this refinement procedure generates a partition of $\mathbb{R}^n$. From Proposition 2, each point $x \in \mathbb{R}^n$ is an element of some regions $\{R_g : g \in G(f)\}$. After running the procedure, there is a single region $R_{g'} \in \{R_g : g \in G(f)\}$ such that $x \in R_{g'}$, where $g'_i = \bigoplus_{g \in G(f)} g_i$, for each $1 \le i \le n$.

Proving that the refinement procedure does not increase the number of regions equates to showing that the set difference of two adjacent regions is a DBM.

Proposition 3: If $R_g > R_{g'}$, then $R_{g'} \setminus R_g = R_{g'} \cap \{x \in \mathbb{R}^n : A(i, g'_i) + x_{g'_i} > A(i, g_i) + x_{g_i}\}$, which is a DBM.

Proof: The ingredients are (7) and two properties of set-theoretic operations: $(A \cap B) \setminus (A \cap C) = A \cap (B \setminus C)$ and $A \setminus B = A \setminus (A \cap B)$, where A, B, C are sets.

We define $R = \bigcap_{i=2}^n \bigcap_{j=1}^n \{x \in \mathbb{R}^n : A(i, j) + x_j \le A(i, g_i) + x_{g_i}\}$ and assume $g_1 > g'_1$, then $R_{g'} = R \cap \bigcap_{j=1}^n \{x \in \mathbb{R}^n : A(1, j) + x_j \le A(1, g'_1) + x_{g'_1}\}$. From the first property, $R_{g'} \setminus R_g = R \cap [\bigcap_{j=1}^n \{x \in \mathbb{R}^n : A(1, j) + x_j \le A(1, g'_1) + x_{g'_1}\} \setminus \bigcap_{j=1}^n \{x \in \mathbb{R}^n : A(1, j) + x_j \le A(1, g_1) + x_{g_1}\}]$.

Let us compute the intersection of two terms in the square bracket. Considering $j = g_1$ and $j = g'_1$ in the first and second terms respectively leads to $\{x \in \mathbb{R}^n : A(1, g_1) + x_{g_1} = A(1, g'_1) + x_{g'_1}\}$. Thus the intersection is $\bigcap_{j=1}^n \{x \in \mathbb{R}^n : A(1, j) + x_j \le A(1, g'_1) + x_{g'_1}\} \cap \{x \in \mathbb{R}^n : A(1, g_1) + x_{g_1} = A(1, g'_1) + x_{g'_1}\}$. Finally both properties are used to obtain the explicit form of $R_{g'} \setminus R_g$. □

Let us quantify the worst-case complexity of Algorithm 6.

Step 1 is performed in $\mathcal{O}(n^{n+3})$ (cf. Section II-C). Since the maximum number of regions generated by MPL2PWA(A) is n^n, the maximum number of iterations in step 2 is n^{2n}. Checking whether two regions are adjacent has a linear complexity, i.e. $\mathcal{O}(n)$. From discussions in the preceding paragraph, the set difference between two adjacent regions can be done in a constant time. Further, since checking emptiness of a region can be done in $\mathcal{O}(n^3)$, the complexity of step 9 is $\mathcal{O}(n^{n+3})$. It follows that the overall worst-case complexity is $\mathcal{O}(n^{2n+1})$.

Example: The partitioning regions generated by the proposed refinement procedure are $\{x \in \mathbb{R}^2 : x_1 - x_2 > 3\}$, $\{x \in \mathbb{R}^2 : e < x_1 - x_2 \leq 3\}$, and $\{x \in \mathbb{R}^2 : x_1 - x_2 \leq e\}$. □

2) Nonautonomous Case: Given a nonautonomous MPL model as in (3), a state-space partitioning can be simply obtained by considering its autonomous version and by applying the procedures in Section III-B1. However, as it will become clear in Section III-C2, a covering of the augmented space in (4) is needed to compute the transitions associated with the partitioning regions. A covering can be obtained from the regions of the PWA system associated with the (augmented) nonautonomous MPL model in (4), and further refined to obtain the state-space partitioning.

C. Transitions: One-Step Reachability

In this section, we investigate a technique to determine the transition relations between two states of the finite-state TS, that is between two partitioning regions (cf. Section III-B) of the concrete MPL model.

1) Autonomous Case: At any event step k, there is a transition from $s_\sim$ to $s'_\sim$ if and only if there exists an $x(k-1) \in s_\sim$ such that $x(k) \in s'_\sim$, where $s_\sim, s'_\sim \in S/_\sim$. Such a transition can be determined by a forward- or backward-reachability approach. According to the former, we calculate $s'_\sim \cap \{x(k) : x(k-1) \in s_\sim\}$, whereas if we use the backward approach we compute $s_\sim \cap \{x(k-1) : x(k) \in s'_\sim\}$. The non-emptiness of the resulting set characterizes the presence of a transition from $s_\sim$ to $s'_\sim$.

In this work we focus on the forward-reachability approach, since it is computationally more attractive than the backward one. More precisely (cf. Theorem 1), since both approaches leverage the affine dynamics associated with the outgoing partitioning region, the number of image computations in the forward-reachability approach is linear w.r.t. the number of partitioning regions, whereas the number of inverse-image computations in the backward-reachability approach is quadratic w.r.t. the number of partitioning regions.

With focus on the forward-reachability approach, given a partitioning region $s_\sim$ we employ its dynamics to compute its image as: $\mathcal{I}(s_\sim) = \{A \otimes x : x \in s_\sim\}$. In the dynamical systems and automata literature, the mapping $\mathcal{I}$ is also known as *Post* [15, Definition 2.3]. Furthermore, since each partitioning region has associated affine dynamics, we exploit their DBM representation as in Theorem 1 to compute the image. The complete approach to determine the transitions is shown in Algorithm 7, which incurs a worst-case complexity (checking emptiness over the cycles in step 2) of $\mathcal{O}(n^3|S/_\sim|^2)$.

Example: Consider the regions generated by the scheme in Algorithm 4. The image of $R_{(\{2\},\{1\})}$ is $\mathcal{I}(R_{(\{2\},\{1\})}) = \{x \in \mathbb{R}^2 : -1 < x_1 - x_2 < 2\}$. Thus, there are three outgoing transitions from $R_{(\{2\},\{1\})}$ with destinations respectively $R_{(\{2\},\{1\})}$, $R_{(\{2\},\{2\})}$, $R_{(\{2\},\{1,2\})}$. The complete transition relation is shown in Fig. 5b. □

input: $S/_\sim$, a partition of $\mathbb{R}^n$
output: $\delta_\sim \subseteq S/_\sim \times S/_\sim$, a transition relation
1: $\delta_\sim \leftarrow \emptyset$
2: **for all** $s_\sim, s'_\sim \in S/_\sim$ **do**
3: **if** $\mathcal{I}(s_\sim) \cap s'_\sim$ is not empty **then** ▷ Theorem 1
4: $\delta_\sim \leftarrow \delta_\sim \cup \{(s_\sim, s'_\sim)\}$
5: **end if**
6: **end for**

Fig. 7. Computation of the transitions for an autonomous model via forward-reachability analysis

Notice that the obtained TS can be nondeterministic: by construction of its transitions, the TS obtained by Algorithm 7 simulates the autonomous MPL model. In general the opposite relation (leading to a bisimulation) does not hold. In fact, whenever the TS obtained by Algorithm 7 is nondeterministic, there exists a partitioning region $s_\sim$ that contains more than one outgoing transition. If we assume that this happens with reference to a point $x \in s_\sim$, for each $x \in \mathbb{R}^n$ the value of $A \otimes x$ is instead unique (A denoting again the MPL system matrix). The previous argument leads to the following result.

Theorem 4: The TS obtained by Algorithm 7 is deterministic if and only if it bisimulates the autonomous MPL model.

Proof: The necessity is a direct consequence of the preceding discussion. On the other hand, if a TS bisimulates the autonomous MPL model it is deterministic, since the infinite-state TS generated by an autonomous MPL model (cf. Section III-A) is also deterministic. □

Having obtained a nondeterministic TS by abstraction, it makes sense to attempt deriving a deterministic TS by successive refinement: within a refinement step, each nondeterministic partitioning region is split and its (incoming and outgoing) transitions are updated. Whenever a deterministic TS is obtained, we can establish the preceding property.

Example: From Fig. 5b, $R_{(\{2\},\{1\})}$ has three outgoing transitions. After computing the inverse image of each destination w.r.t. the dynamics on $R_{(\{2\},\{1\})}$, we obtain: $R^{(\{2\},\{1\})}_{(\{2\},\{1\})} = \{x \in \mathbb{R}^2 : e < x_1 - x_2 < 2\}$; $R^{(\{2\},\{2\})}_{(\{2\},\{1\})} = \{x \in \mathbb{R}^2 : 2 < x_1 - x_2 < 3\}$; $R^{(\{2\},\{1,2\})}_{(\{2\},\{1\})} = \{x \in \mathbb{R}^2 : x_1 - x_2 = 2\}$. In order to simplify the notation, we define $s^1_\sim = R_{(\{2\},\{2\})}$, $s^2_\sim = R_{(\{2\},\{1,2\})}$, $s^3_\sim = R_{(\{1\},\{1\})}$, $s^4_\sim = R_{(\{1,2\},\{1\})}$, $s^5_\sim = R^{(\{2\},\{2\})}_{(\{2\},\{1\})}$, $s^6_\sim = R^{(\{2\},\{1\})}_{(\{2\},\{1\})}$, $s^7_\sim = R^{(\{2\},\{1,2\})}_{(\{2\},\{1\})}$. The obtained deterministic TS and the graphical representation of its regions are shown in Fig. 8. □

Unfortunately, such a procedure in general does not necessarily terminate, especially in the presence of a cycle in the TS containing the nondeterministic partitioning regions. An upper bound on the number of (partitioning) regions can be used as a stopping criterion. In the remainder of this subsection, sufficient conditions for the existence of a bisimilar TS will be discussed.

Given an irreducible MPL model (cf. Definition 3) cha-

racterized by system matrix A, notice that the union of partitioning regions associated with the states of the obtained TS equals to $\mathbb{R}^n$ (it is a "global" TS). A TS over $E(A^{\otimes c})$ is defined similarly, that is as the union of partitioning regions that corresponds to $E(A^{\otimes c})$.

Theorem 5: Given an irreducible MPL model characterized by matrix A with cyclicity c and the obtained TS, if the TS over $E(A^{\otimes c})$ is deterministic, then there exists a (globally) deterministic TS.

Proof: Given a TS over $E(A^{\otimes c})$, we describe a procedure to construct a global TS. Notice that $E(A^{\otimes c})$ can be expressed as $\{x(0) \in \mathbb{R}^n : k_0(x(0)) = 0\}$. The procedure to build the global TS works as follows: 1) initialize k with value 1; 2) compute $\{x(0) \in \mathbb{R}^n : k_0(x(0)) \leq k\}$ by using backward-reachability analysis; 3) determine $\{x(0) \in \mathbb{R}^n : k_0(x(0)) = k\}$ by using set-difference computations; finally 4) if $\{x(0) \in \mathbb{R}^n : k_0(x(0)) = k\}$ is not empty, then define the transition relations originating from it, set $k \leftarrow k+1$ and go to step 2); else stop.

The TS obtained by the procedure is deterministic, since each new region is the inverse image of a region, i.e. there is a single outgoing transition for each new region. Since the MPL model is irreducible and the corresponding infinite-state TS (cf. Section III-A) is deterministic, the obtained TS covers $\mathbb{R}^n$ and its regions are pairwise disjoint. Notice that the preceding steps constitute a finite-time procedure, since $\max_{x(0)\in\mathbb{R}^n} k_0(x(0))$ is finite, however its time complexity is difficult to quantify since the length of the transient part can be arbitrarily large, even for matrices of small size [4, p. 56]. □

Remark: Given an irreducible MPL model and the obtained TS, the procedure in the proof of Theorem 5 can be used to construct the mapping $x(0) \mapsto k_0(x(0))$, for $x(0) \in \mathbb{R}^n$. As such, it computes the maximum length of the transient part over the whole state space, i.e. $\max_{x(0)\in\mathbb{R}^n} k_0(x(0))$, which generalizes and improves the results in [25], [26, Th. 10 and Th. 13]. To the best of our knowledge, this problem cannot be solved in the literature by more efficient means. □

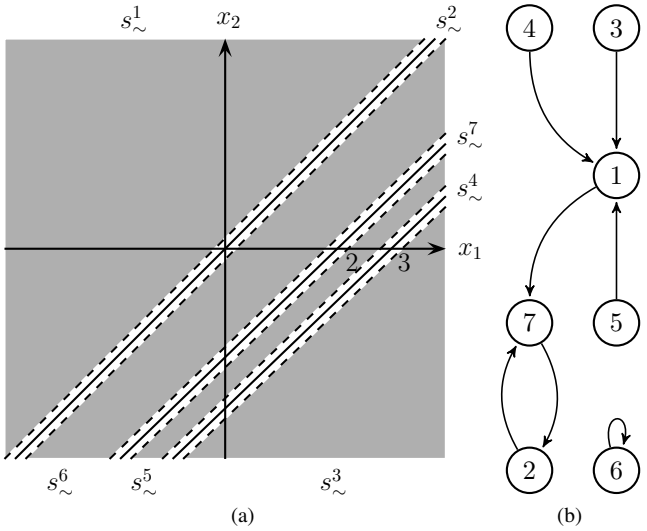
Theorem 5 implies the existence of (globally) deterministic TS for any 2-dimensional irreducible MPL system, since the MPL system characterized by $A^{\otimes c}$ is irreducible and the eigenspace of each 2-dimensional irreducible MPL system is a DBM. For a higher dimensional irreducible MPL system, the existence of (globally) deterministic TS depends on the cyclicity of A, as the following result claims.

Theorem 6: Given an irreducible MPL model with system matrix A, if the cyclicity of A is equal to 1, then the model admits a deterministic TS abstraction.

Proof: If the cyclicity of A is 1, then $E(A) = E(A^{\otimes c})$ and via [8, Th. 3.100-3.101] we conclude that $E(A)$ is a linear combination (in a max-plus sense) of a set of finitely many vectors. Equivalently, $E(A)$ is the image of a linear map (in a max-plus sense) governed by a max-plus matrix made up of those vectors. Exploiting the PWA representation of the max-plus matrix, $E(A)$ equals to the union of the image of each region w.r.t. its affine dynamics. From Theorem 1, $E(A)$ is a union of finitely many DBM that are not necessarily pairwise disjoint: in order to obtain a partition of $E(A)$ we can leverage a generic refinement procedure. Each obtained partitioning region has a self loop, since it is a subset of $E(A)$. It follows that the TS over $E(A)$ is deterministic, which leads to the conclusion by application of Theorem 5. □

TABLE I
CONDITIONS ON THE MPL MODEL FOR SIMILARITY OR BISIMILARITY OF THE TS ABSTRACTION

model dimension	A **irreducible**		A **reducible**
	$c=1$	$c>1$	
$n=2$	bisimulation		
$n>2$		simulation	



(a) (b)

Fig. 8. (a) Graphical representation of the partitioning regions for the deterministic TS. (b) Deterministic TS abstraction, where state i represents partitioning region $s^i_\sim$, for $i = 1, \ldots, 7$. Deterministic TS with partitioning regions after refinement, as an abstraction of the autonomous MPL model in (2)

Remark: In the proof of Theorem 6 we have observed that each linear combination (in a max-plus sense) of some vectors in $\mathbb{R}^n$ is a union of finitely many DBM. As a special case, the eigenspace $E(A)$ and the whole periodic behavior $E(A^{\otimes c})$ (cf. Section II-A) can be represented as unions of finitely many DBM. Since in particular $E(A^{\otimes c})$ can be expressed as a union of q pairwise disjoint DBM $\bigcup_{k=1}^{q} D_k$, the partitioning regions associated with the states of the TS over $E(A^{\otimes c})$ can be computed by $\{s_\sim \cap D_k : s_\sim \in S/_\sim, 1 \leq k \leq q\}$ and its transition relations can be determined by Algorithm 7. □

Table I summarizes a simple procedure that can be carried out on a given MPL model in order to check if it is possible to obtain an abstraction that is bisimilar to it. Given a state space matrix A, we check its irreducibility. If it is reducible, the abstraction will simulate the MPL model. Otherwise, if A is a 2×2 matrix, the abstraction will bisimulate the concrete model. Else, we check the cyclicity of A: if the cyclicity is 1, then the abstraction will bisimulate the model, otherwise it will simulate it.

2) Nonautonomous Case: Next, we investigate a technique to determine the transition relations $\bar{\delta}_\sim$ between two states of the finite-state TS generated from a nonautonomous MPL

model. At any given event step k, given a pair of partitioning regions $s_\sim$ and $s'_\sim$ and a set of inputs $U \subseteq \mathbb{R}^m$, there exists a transition from $s_\sim$ to $s'_\sim$ if and only if there exists an $x(k-1) \in s_\sim$ and a $u(k) \in U$, such that $x(k) \in s'_\sim$. Such a transition can be determined either by forward- or by backward-reachability computation, that is calculating either $s'_\sim \cap \{x(k) : x(k-1) \in s_\sim,\ u(k) \in U\}$, or $s_\sim \cap \{x(k-1) : x(k) \in s'_\sim,\ u(k) \in U\}$, and by checking the non-emptiness of the resulting set.

We assume that the set of inputs $U \subseteq \mathbb{R}^m$ is represented via a DBM (cf. Section II-C). Practically, this enables expressing upper or lower bounds on the separation between input events (schedules) (cf. Definition 4). If on the other hand there are no constraints on input events, we define $U = \mathbb{R}^m$, which is also a DBM.

Notice that the forward- and backward-reachability approaches yield the same outcome, since both are equivalent in checking the non-emptiness of the set $\{x \in s_\sim : \exists u \in U \text{ s.t. } A \otimes x \oplus B \otimes u \in s'_\sim\}$. As in the autonomous case, we focus on the forward-reachability approach, since it is computationally more attractive than the backward one. Given a partitioning region $s_\sim \in S/_\sim$, we employ the PWA representation over the augmented space to compute its image: $\bar{\mathcal{I}}(s_\sim \times U) = \{\bar{A} \otimes \bar{x} : \bar{x} \in s_\sim \times U\}$, where $s_\sim \times U$ denotes the cross product of the sets $s_\sim$ and U. Since both $s_\sim$ and U are DBM, $s_\sim \times U$ is also a DBM in the augmented space. In order to exploit the DBM structure for both $s_\sim$ and U, the general procedure for obtaining $\bar{\mathcal{I}}(s_\sim \times U)$ is as follows: 1) calculating $s_\sim \times U$; then 2) intersecting $s_\sim \times U$ with each region of the PWA system generated by the nonautonomous MPL model; finally 3) computing the image of nonempty intersections (cf. Section II-C). Notice that in general $\bar{\mathcal{I}}(s_\sim \times U)$ is a union of finitely many DBM that are not necessarily pairwise disjoint, since $s_\sim \times U$ may intersect with more than one region and regions of the PWA system generated by a nonautonomous MPL model are not pairwise disjoint. The complexity of image computations critically depends on the last step and is $\mathcal{O}((n+m)^3|\bar{\mathsf{R}}|)$. Algorithm 9 details the complete approach. Let us quantify the worst-case complexity of Algorithm 9. Its global worst-case complexity depends on the number of iterations related to cycles in step 2, which is $|S/_\sim|$. This, compounded with the complexity of DBM intersection, DBM image computation, and with the check of emptiness for a DBM, amounts to a total complexity of $\mathcal{O}((n+m)^3|S/_\sim|^2|\bar{\mathsf{R}}|)$.

Given a row-finite max-plus matrix A, a max-plus matrix B, and a set of inputs U that is a DBM, then Lemma 1 implies that for each state $x(k-1) \in \mathbb{R}^n$ there exists a sufficiently small input $u(k) \in U$ such that states $x(k)$ obtained from (1) and (3) coincide. This observation leads to the following result.

Proposition 4: Each transition generated by the autonomous MPL model is included in the transition relation generated by the nonautonomous MPL model, i.e. $\delta_\sim \subseteq \bar{\delta}_\sim$. □

Proposition 5: For any $s'_\sim \in S/_\sim$, if $s'_\sim \cap (B \otimes U)$ is not empty, then $(s_\sim, s'_\sim) \in \bar{\delta}_\sim$, for each $s_\sim \in S/_\sim$.

Proof: Given a row-finite max-plus matrix A, a max-plus matrix B, a set of inputs U that is a DBM, and a partitioning region $s'_\sim \in S/_\sim$ such that $s'_\sim \cap (B \otimes U)$ is not empty, since each partitioning region $s_\sim$ is a DBM, then Lemma 1 implies that for each state $x(k) \in s'_\sim \cap (B \otimes U)$ there exists a state $x(k-1) \in s_\sim$ such that $x(k) = A \otimes x(k-1) \oplus x(k)$. □

input: $\bar{A} \in \mathbb{R}_\varepsilon^{n \times (n+m)}$, a row-finite augmented matrix;
$S/_\sim$, a partition of $\mathbb{R}^n$;
$\bar{\mathsf{R}} \leftarrow$ MPL2PWA$(\bar{A})$, a cover of $\mathbb{R}^{n+m}$ where $|\bar{\mathsf{R}}| < \infty$;
U, the input set
output: $\bar{\delta}_\sim \subseteq S/_\sim \times S/_\sim$, a transition relation

1: $\bar{\delta}_\sim \leftarrow \emptyset$ ▷ $\bar{\mathcal{I}}(s_\sim \times U)$ is a collection of DBM
2: **for all** $s_\sim, s'_\sim \in S/_\sim$ **do** ▷ $s''_\sim$ is a DBM
3: **if** $\exists s''_\sim \in \bar{\mathcal{I}}(s_\sim \times U)$ s.t. $s''_\sim \cap s'_\sim$ is not empty **then**
4: $\bar{\delta}_\sim \leftarrow \bar{\delta}_\sim \cup \{(s_\sim, s'_\sim)\}$
5: **end if**
6: **end for**

Fig. 9. Computation of the transitions for nonautonomous MPL model via forward-reachability analysis

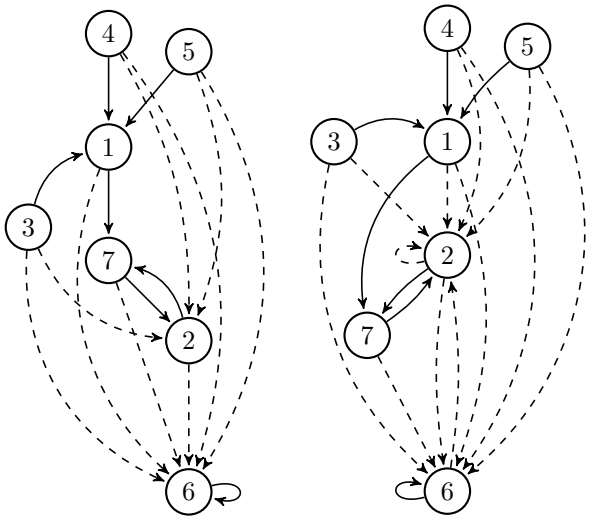
The outcome of Algorithm 9 is in general a nondeterministic TS. Its relationship with the nonautonomous MPL model is clear: the TS obtained by Algorithm 9 simulates the nonautonomous MPL model, whereas in general the opposite direction does not hold. In order to provide sufficient conditions to obtain a bisimulation relation, we employ a backward-reachability analysis over the augmented MPL model: $\bar{\mathcal{I}}^{-1}(s'_\sim) = \{\bar{x} \in \mathbb{R}^n \times U : \bar{A} \otimes \bar{x} \in s'_\sim\}$.

The general procedure for obtaining $\bar{\mathcal{I}}^{-1}(s'_\sim)$ is as follows: 1) computing the inverse image of $s'_\sim$ w.r.t. each affine dynamics (9) of the PWA system generated by the nonautonomous MPL model; then 2) intersecting each inverse image with the corresponding region in the augmented space and $\mathbb{R}^n \times U$; finally 3) obtaining $\bar{\mathcal{I}}^{-1}(s'_\sim)$ as the union of the nonempty intersections. The worst-case complexity of inverse-image computations is $\mathcal{O}((n+m)^3|\bar{\mathsf{R}}|)$, where $\bar{\mathsf{R}} = \text{MPL2PWA}(\bar{A})$ (cf. Algorithm 2). Notice that $\Pi_X(\bar{\mathcal{I}}^{-1}(s'_\sim))$ represents the set of states that transition into $s'_\sim$ or, more formally $\{x \in \mathbb{R}^n : \exists u \in U \text{ s.t. } A \otimes x \oplus B \otimes u \in s'_\sim\}$. Furthermore, there exists a transition from $s_\sim$ to $s'_\sim$ if and only if $s_\sim \cap \Pi_X(\bar{\mathcal{I}}^{-1}(s'_\sim))$ is not empty. This leads to the following result.

Proposition 6: The TS obtained by Algorithm 9 bisimulates the nonautonomous MPL model if for each pair $(s_\sim, s'_\sim) \in \bar{\delta}_\sim$, the following holds: for each $x \in s_\sim$, there exists an input $u \in U$ such that $A \otimes x \oplus B \otimes u \in s'_\sim$. Equivalently, for each $(s_\sim, s'_\sim) \in \bar{\delta}_\sim$, $s_\sim \subseteq \Pi_X(\bar{\mathcal{I}}^{-1}(s'_\sim))$. □

The procedure to check whether a finite-state TS bisimulates a nonautonomous MPL model practically works as follows. If $s_\sim \setminus \Pi_X(\bar{\mathcal{I}}^{-1}(s'_\sim))$ is empty for each $(s_\sim, s'_\sim) \in \bar{\delta}_\sim$, then the finite-state TS bisimulates the nonautonomous MPL model. Otherwise it simulates it.

Example: Let us consider an input set U that coincides with region $s^6_\sim$, i.e. $U = \{u \in \mathbb{R}^2 : e < u_1 - u_2 < 2\}$. Let us explicitly check the existence of a transition from $s^3_\sim$ to $s^2_\sim$. First, we determine the regions that intersect with $s^3_\sim \times U = \{\bar{x} \in \mathbb{R}^4 : x_1 - x_2 > 3,\ e < u_1 - u_2 < 2\}$: after performing an emptiness check of the intersections, we are left with the regions $\bar{R}_{(1,1)}$, $\bar{R}_{(3,1)}$, and $\bar{R}_{(3,4)}$. Thereafter, we determine whether the image of $\bar{R}_{(1,1)} \cap (s^3_\sim \times U)$,



(a) Input set $U = s^6_\sim$ (b) Input set $U = s^2_\sim \cup s^6_\sim$

Fig. 10. TS abstraction for the MPL model in (5), where i represents $s^i_\sim$, for $i = 1, \ldots, 7$. Solid arrows refer to the autonomous version in (2) of the MPL model, whereas dashed ones are related to the nonautonomous model

$\bar{R}_{(3,1)} \cap (s^3_\sim \times U)$, and $\bar{R}_{(3,4)} \cap (s^3_\sim \times U)$ intersect with $s^2_\sim$: after computing the sets, we conclude that only the image of $\bar{R}_{(3,1)} \cap (s^3_\sim \times U)$ intersects with $s^2_\sim$. Thus, there is a transition from $s^3_\sim$ to $s^2_\sim$. The overall TS for this example is shown in Fig. 10a. Alternatively, if the set of inputs U is defined as $s^6_\sim \cup s^2_\sim$, then we obtain the TS depicted in Fig. 10b. In Fig. 10, the solid transitions are generated by autonomous versions of the given MPL model, whereas the dashed ones are related to the nonautonomous models. It can be checked that both TS bisimulate the nonautonomous MPL model in (5). □

In the remainder of this article, unless otherwise stated, we assume that the obtained TS simulates the original MPL model.

IV. MPL Abstractions as Labeled Transition Systems

Definition 10 (Labeled transition system): An LTS (S, δ^L, I, L) is a direct extension of the TS (S, δ, I) in Definition 6 and additionally consists of

- a (possibly infinite) set L of transition labels;
- a labeled transition relation $\delta^L \subseteq S \times L \times S$. □

The definition above can be alternatively given over state labels [15, Definition 2.1]. With the goal of working with finite-state LTS as extensions of the TS abstractions obtained in the previous section, we consider either sort of labels, referring to states or to transitions, respectively. State labels are defined as the explicit representation of the regions associated with states in $S/\sim$, whereas transition labels depend on the partitioning region (the source of the transition), its affine dynamics, and (possibly) the set of inputs. More precisely,

1) state labels characterize the difference between the timing of an event for any two variables of the original model, and are defined as all the possible values of $x_i(k) - x_j(k)$, where $1 \leq i < j \leq n$. Given a partitioning region, we can easily compute the labels using its explicit representation: the label of each state in the finite-state LTS is defined as the system of linear inequalities characterizing the partitioning region. Since each partitioning region is represented by a DBM in its canonical-form representation (cf. Section III-A), the bounds on state labels are the tightest possible and the representation is unique [16, Th. 2].
2) transition labels represent the time difference between consecutive events of the MPL model, and are defined as all the possible values of the difference $x_i(k) - x_i(k-1)$, for $1 \leq i \leq n$. Since the possible interval characterizing the difference $x_i(k) - x_i(k-1)$ may depend on the value for other variables, in general labels associated with each transition in the finite-state LTS are overapproximations of the actual differences (see example in the next subsection).

Notice the structural difference between the labels defined in this work (quantities associated with states or transitions) and either the canonical state labels or the actions as discussed in [15, Definition 2.1]. Since labels over the states can be directly derived from the characterization of the partitioning regions, we next focus on the characterization of the labels over the transitions.

A. Autonomous Case

Computing $l_\sim \in L_\sim$ – the label of a specific transition – involves substituting the affine dynamics (8) into $x_i(k) - x_i(k-1)$ for $1 \leq i \leq n$, and applying the explicit representation (10) if needed. A label is an n-dimensional vector of real-valued intervals, which is overapproximated by a box – a special case of DBM – in $\mathbb{R}^{2n}$. Using a box is advantageous, since: 1) the possible values of $x_i(k) - x_i(k-1)$ are now independent for each $1 \leq i \leq n$; and 2) they are computationally easier to work with. The box lies in $\mathbb{R}^{2n}$ since the total number of variables is $2n$ (n variables for both event steps k and $k-1$). Let us denote the procedure to compute the label of a transition as Labels$(s_\sim)$, where $s_\sim$ is the outgoing partitioning region and its complexity is $\mathcal{O}(n)$. Thus, the complexity of computing the set of transition labels $L_\sim$ hinges on the number of transition relations in the LTS and it is $\mathcal{O}(|S/\sim|^2 n)$ in the worst case. Furthermore, in the nondeterministic case, evaluating the transition labels requires proper subdivision of partitioning regions using backward-reachability analysis: in this case, the worst-case complexity of the complete labeling procedure increases to $\mathcal{O}(|S/\sim|^2 n^3)$.

Example: Considering the TS of Fig. 10a, let us determine the label of the transition from $s^5_\sim$ to $s^1_\sim$. Substituting the dynamical system of $s^5_\sim$, i.e. $x_1(k) = x_2(k-1) + 5$, $x_2(k) = x_1(k-1) + 3$, into $x(k) - x(k-1)$ we obtain $x_1(k) - x_1(k-1) = 5 - (x_1(k-1) - x_2(k-1))$ and $x_2(k) - x_2(k-1) = 3 + x_1(k-1) - x_2(k-1)$. Applying the definition of $s^5_\sim$, the exact transition label is $\{[5 - \alpha \; 3 + \alpha]^T : 2 < \alpha < 3\}$, which can be overapproximated by the smallest box containing it, i.e. $\{[5 - \alpha \; 3 + \alpha']^T : 2 < \alpha < 3, 2 < \alpha' < 3\}$. The complete labels of this example is shown as intervals with bold typeset over the LTS with continuous edges in Fig. 11. □

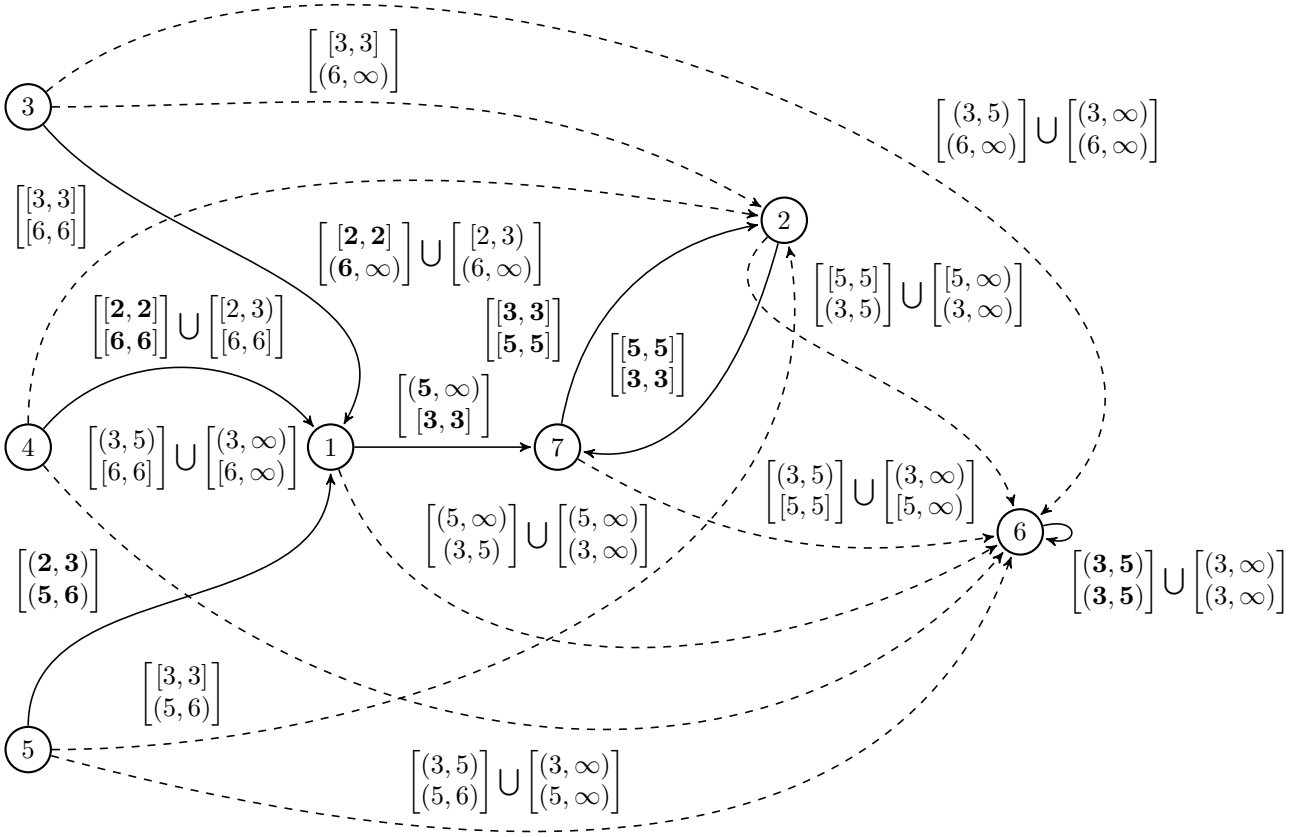


Fig. 11. LTS endowed with transition labels. For the autonomous MPL model in (2) of Fig. 8b the LTS is represented with continuous edges and the labels with bold typeset. For the nonautonomous MPL model in (6) of Fig. 10a the LTS is represented with dashed edges and the labels with normal typeset. State i represents $s^i_\sim$, for $i = 1, \dots, 7$

B. Nonautonomous Case

Computing the label $\bar{l}_\sim \in \bar{L}_\sim$ of a transition from $s_\sim$ to $s'_\sim$ consists of two steps: 1) determining $\{\bar{x} \in s_\sim \times U : \bar{A} \otimes \bar{x} \in s'_\sim\}$ by using backward-reachability analysis, i.e. $\bar{\mathcal{I}}^{-1}(s'_\sim) \cap (s_\sim \times U)$, which is in general a union of finitely many DBM; then, similar to the autonomous case, 2) collecting information from expressions such as (9) and (10), for each DBM. Thus, a transition label for a nonautonomous MPL model is a union of finitely many transition labels (see example below), as defined in the preceding section for the autonomous case.

Example: We are going to determine the labels of the transition from $s^3_\sim$ to $s^2_\sim$. First we calculate $(s^3_\sim \times U) \cap \bar{\mathcal{I}}^{-1}(s^2_\sim)$: after iterating through all states, we obtain a region described by $\{\bar{x} \in \mathbb{R}^4 : x_1 - x_2 > 3,\ x_1 - u_1 = -3,\ x_1 - u_2 = -2,\ x_2 - u_1 < -6,\ x_2 - u_2 < -5,\ u_1 - u_2 = 1\}$. Since the region is a subset of $\bar{R}_{(3,1)}$, the transition label is obtained by applying the procedure described in the preceding section to the region and the affine dynamics of $\bar{R}_{(3,1)}$: $x_1(k) - x_1(k-1) = u_1(k) - x_1(k-1) = 3$ and $x_2(k) - x_2(k-1) = x_1(k-1) + 3 - x_2(k-1) > 6$. Figure 11 depicts the full LTS endowed with transition labels. □

Labels in $\bar{L}_\sim$ allow to be further particularized as follows. Each transition can be associated with a subset of the partitioning region of its source, made up of the points that are affected by the transition w.r.t. some input signals, i.e. $\{x \in s_\sim : \exists u \in U \text{ s.t. } A \otimes x \oplus B \otimes u \in s'_\sim\}$. This can be computed by projecting the inverse image of $s'_\sim$ over the state variables: $t_X(s_\sim, s'_\sim) = \Pi_X(\bar{\mathcal{I}}^{-1}(s'_\sim) \cap (s_\sim \times U))$. Similarly, the subset of control inputs that steer some states in the source region to the target region is denoted by $\{u \in U : \exists x \in s_\sim \text{ s.t. } A \otimes x \oplus B \otimes u \in s'_\sim\}$ and can be computed by projecting w.r.t. input variables the inverse image of $s'_\sim$: $t_U(s_\sim, s'_\sim) = \Pi_U(\bar{\mathcal{I}}^{-1}(s'_\sim) \cap (s_\sim \times U))$. Since, as discussed above, backward-reachability analysis yields unions of finitely many DBM, also $t_X(s_\sim, s'_\sim)$ and $t_U(s_\sim, s'_\sim)$ are in general unions of finitely many DBM. The following result holds:

Theorem 7: For any $s_\sim \in S/_\sim$, $\bigcup_{(s_\sim, \cdot, s'_\sim) \in \bar{\delta}^L_\sim} t_X(s_\sim, s'_\sim) = s_\sim$ and $\bigcup_{(s_\sim, \cdot, s'_\sim) \in \bar{\delta}^L_\sim} t_U(s_\sim, s'_\sim) = U$.

Proof: From the definition of $t_X(s_\sim, s'_\sim)$, we know that $\bigcup_{(s_\sim, \cdot, s'_\sim) \in \bar{\delta}^L_\sim} t_X(s_\sim, s'_\sim) = \{x \in s_\sim : \exists u \in U \text{ s.t. } \bar{A} \otimes [x^T\ u^T]^T \in \mathbb{R}^n\}$. The inclusion $\bigcup_{(s_\sim, \cdot, s'_\sim) \in \bar{\delta}^L_\sim} t_X(s_\sim, s'_\sim) \subseteq s_\sim$ is then evident. Recall that $\bar{A}$ is a row-finite max-plus matrix because $\bar{A} = [A\ B]$ and A is assumed to be row-finite. Thus, $\bar{A} \otimes [x^T\ u^T]^T \in \mathbb{R}^n$, for each $x \in \mathbb{R}^n$ and $u \in \mathbb{R}^m$, which proves the other direction of the inclusion. A similar argument holds for the union over t_U. □

Looking back at Proposition 6, the sufficient conditions raised to establish a bisimulation relation can be restated as follows: for each $(s_\sim, s'_\sim) \in \bar{\delta}_\sim$, $s_\sim \subseteq t_X(s_\sim, s'_\sim)$.

The labels in t_U can be useful to construct a TS correspond-

input: $s_\sim, s'_\sim$, partitioning regions s.t. $(s_\sim, s'_\sim) \in \bar{\delta}_\sim$;
$\bar{\mathsf{R}} \leftarrow$ MPL2PWA($\bar{A}$), a cover of $\mathbb{R}^{n+m}$ where $|\bar{\mathsf{R}}| < \infty$;
U, the set of inputs
output: $\bar{l}_\sim, t_X, t_U$, labels of the given transition
1: $\bar{l}_\sim \leftarrow \emptyset$, $t_X \leftarrow \emptyset$, $t_U \leftarrow \emptyset$
2: **for all** $\bar{R} \in (s_\sim \times U) \cap \bar{\mathcal{I}}^{-1}(s'_\sim)$ **do**
3: $\quad \bar{l}_\sim \leftarrow \bar{l}_\sim \cup \{\textsc{Labels}(\bar{R})\}$ ▷ Section IV-A
4: $\quad t_X \leftarrow t_X \cup \{\Pi_X(\bar{R})\}$, $t_U \leftarrow t_U \cup \{\Pi_U(\bar{R})\}$
5: **end for**

Fig. 12. Computation of the labels associated with a transition of a nonautonomous model via backward-reachability analysis

ing to any subset $U' \subseteq U$. Recall that t_U is a union of finitely many DBM in $\mathbb{R}^m$ and consider for the sake of argument that the original set of inputs U is selected as $\mathbb{R}^m$. Select an arbitrary DBM $U' \subset \mathbb{R}^m$ as the new set of inputs. The new set of transitions is made up of the existing transitions intersecting U', namely those for which $t_U \cap U'$ is not empty. This procedure is faster than the one discussed in Section III-C2, since its worst-case complexity is $\mathcal{O}(m^3 q |S/_\sim|^2|)$, where q is the maximum number of DBM in $t_U(s_\sim, s'_\sim)$ for any $(s_\sim, s'_\sim) \in \bar{\delta}_\sim$.

Algorithm 12 presents a procedure to compute the labels of a given transition from $s_\sim$ to $s'_\sim$, and is part of the general labeling procedure for the whole LTS. (Notice that the state labels are not computed in Algorithm 12, since they are directly characterized by the partitioning regions in the finite-state LTS.)

Let us quantify the worst-case complexity of Algorithm 12. Notice that the maximum number of iterations in step 2 is $|\bar{\mathsf{R}}|$. This, compounded with the complexity of checking the emptiness of a DBM and of intersecting DBM, leads to $\mathcal{O}((n+m)^3|\bar{\mathsf{R}}|)$. Furthermore, the worst-case complexity of the complete procedure is $\mathcal{O}((n+m)^3|S/_\sim|^2|\bar{\mathsf{R}}|)$, since the maximum number of transitions is $|S/_\sim|^2$.

Example: Consider the transition from $s^3_\sim$ to $s^2_\sim$. From Algorithm 12, t_X and t_U are calculated by projecting the region w.r.t. $\{x_1, x_2\}$ and $\{u_1, u_2\}$, obtaining respectively $s^3_\sim$ and U. □

V. Computational Benchmark

In order to test the practical efficiency of the proposed algorithms, we compute the runtime required to perform the abstraction of an MPL system into a finite-state LTS, for increasing dimensions n of the given MPL model. We furthermore keep track of the number of states and of transitions of the obtained LTS, which is directly related to the memory requirement of the technique.

For any given n, we generate row-finite matrices A with 2 finite elements placed uniformly at random in each row, as well as matrices B as column vectors where all elements are finite. The finite elements are uniformly generated integers taking values between 1 and 100. The input space U is conservatively selected to be equal to $\mathbb{R}$.

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. Over 10 independent experiments, Tables II and III report the (mean and maximum values for the) time needed to construct the LTS, broken down over the three successive procedures for the generation of the states, the transitions, and the labels of the LTS, respectively. The total number of states and of transitions in the LTS are also reported.

Recall that the first step of the procedure (generation of states) consists of the partitioning of the state space (Algorithm 6) and, for nonautonomous models, of the construction of a PWA system over the augmented space (Algorithm 2), whereas the second step (generation of transitions) uses forward-reachability analysis to determine transitions between abstract states. Finally, with regards to the generation of labels, the results focus on the computation of the labels over the transitions, since those over the states can be directly (and computationally more efficiently) derived from the first step.

With regards to autonomous models, as confirmed by Table II, the bottleneck of the abstraction procedure resides on the generation of transitions and depends on the number of partitioning regions that is in the worst case exponential w.r.t. the dimension of the state space. On the other hand, for nonautonomous models, as reported in Table III, the labeling part consumes more time compared to the generation of transitions – this is due to the computation step based on backward reachability, which as discussed is in general computationally more expensive than forward reachability. The computation time is higher than in the autonomous case, since its complexity depends on $|\bar{\mathsf{R}}|$ (cf. Algorithm 2), which in the worst case is $\mathcal{O}((n+m)^n)$. For nonautonomous models, the figures refer to the computation of transition labels and of the corresponding t_X and t_U.

We have also performed similar computations for the case of autonomous models with full matrices A (in a max-plus sense), which is likely to generate abstract models with more states. Elements are again uniformly distributed integers taking values between 1 and 100. Analogously to the above results, the bottleneck of the abstraction procedure also resides in the generation of the transitions. For an 8-dimensional MPL model over 10 independent experiments, the maximum time needed to compute the LTS amounts to 21.19 minutes, which is made up of 6.90, 13.21, and 1.08 minutes for generating the partitions, transitions, and labels, respectively.

VI. Formal Verification of MPL Models

This section argues that the presented abstraction approach enables the study of general properties of a given MPL model by formally verifying, by use of a model checker, related logical specifications over its LTS abstraction. Unlike classical results based on the global algebraic [9] or geometric [10] features of MPL models, the approach in this work allows to prove dynamical properties (expressed via logical constructs) over sets of trajectories of the model.

In order to specify properties of trajectories of the MPL model, we use a modal logic known as LTL [15, Ch. 5]. LTL formulae are recursively defined, over a set of atomic propositions, by Boolean and temporal operators. Boolean

TABLE II
NUMERICAL BENCHMARK – AUTONOMOUS MODEL – {MEAN;MAXIMAL} VALUES

size of MPL model	time for generation of states	time for generation of transitions	time for generation of labels	total number of states of LTS	total number of transitions of LTS
3	{0.16;0.23} [sec]	{0.47;0.97} [sec]	{0.05;0.09} [sec]	{3.60;6.00}	{4.30;13.00}
4	{0.21;0.37} [sec]	{0.50;0.89} [sec]	{0.10;0.21} [sec]	{6.20;12.00}	{11.40;35.00}
5	{0.26;0.33} [sec]	{0.46;1.06} [sec]	{0.08;0.17} [sec]	{8.60;24.00}	{13.80;90.00}
6	{0.43;0.51} [sec]	{0.47;0.98} [sec]	{0.15;0.26} [sec]	{19.40;36.00}	{68.50;191.00}
7	{0.90;1.05} [sec]	{0.49;0.91} [sec]	{0.33;0.84} [sec]	{37.20;84.00}	{289.30;1278.00}
8	{1.58;1.83} [sec]	{0.58;0.97} [sec]	{0.60;1.75} [sec]	{58.00;160.00}	{512.30;1927.00}
9	{4.09;4.83} [sec]	{0.83;1.44} [sec]	{1.62;3.12} [sec]	{120.00;208.00}	$\{1.75;4.35\}\times10^3$
10	{9.49;12.85} [sec]	{3.14;15.47} [sec]	{7.88;39.34} [sec]	{283.60;768.00}	$\{1.31;8.35\}\times10^4$
11	{24.85;32.13} [sec]	{15.17;46.56} [sec]	{16.13;33.61} [sec]	{613.20;1104.00}	$\{1.87;4.82\}\times10^4$
12	{1.19;1.94} [min]	{1.52;3.61} [min]	{42.92;106.10} [sec]	$\{1.20;2.03\}\times10^3$	$\{4.76;14.08\}\times10^4$
13	{3.53;5.04} [min]	{5.49;15.52} [min]	{2.77;11.06} [min]	$\{1.92;3.81\}\times10^3$	$\{1.91;8.50\}\times10^5$
14	{12.03;29.65} [min]	{28.21;86.35} [min]	{12.65;54.76} [min]	$\{4.16;8.13\}\times10^3$	$\{7.83;34.50\}\times10^5$
15	{53.58;78.31} [min]	{1.98;9.45} [hr]	{39.43;219.61} [min]	$\{7.42;19.71\}\times10^3$	$\{2.05;11.60\}\times10^6$

TABLE III
NUMERICAL BENCHMARK – NONAUTONOMOUS MODEL – {MEAN;MAXIMAL} VALUES

size of MPL model	time for generation of states [sec]	time for generation of transitions	time for generation of labels	total number of states of LTS	total number of transitions of LTS
3	{0.22;0.29}	{0.52;1.00} [sec]	{0.23;0.57} [sec]	{3.60;6.00}	{7.20;16.00}
4	{0.39;0.44}	{0.51;0.99} [sec]	{0.17;0.24} [sec]	{6.20;12.00}	{15.30;38.00}
5	{0.88;1.04}	{0.78;1.28} [sec]	{0.19;0.53} [sec]	{8.60;24.00}	{21.80;120.00}
6	{2.11;2.63}	{1.84;3.39} [sec]	{1.11;3.54} [sec]	{19.40;36.00}	{107.20;364.00}
7	{5.92;8.46}	{8.93;21.63} [sec]	{27.49;128.60} [sec]	{37.20;84.00}	{485.00;2520.00}
8	{12.66;18.33}	{30.55;107.43} [sec]	{1.98;7.20} [min]	{58.00;160.00}	{730.30;2578.00}
9	{39.06;55.94}	{5.39;14.71} [min]	{34.40;123.64} [min]	{120.00;208.00}	{2819.40;8742.00}
10	{98.42;141.97}	{43.21;156.55} [min]	{3.75;11.11} [hr]	{206.80;432.00}	{6211.60;16996.00}

operators are $\neg$ (negation), $\wedge$ (conjunction), and $\vee$ (disjunction), whereas temporal operators are $\bigcirc$ (next), U (until), $\Box$ (always), and $\Diamond$ (eventually). A formula ϕ, which in general is determined by application of the above operators, is interpreted over traces [15, Definition 3.8] generated by the LTS. In particular it is of interest to check whether (the trajectories of) an LTS satisfies a given formula (or specification) – this procedure is known as model checking and can be performed automatically [15].

With focus on the verification over state labels, let us consider a set of atomic propositions that corresponds to the collection of the partitioning regions defining the LTS. More precisely, the inverse image w.r.t. the labeling function [15, Definition 2.1] of each atomic proposition equals to a union of partitioning regions. Thus the partition leading to the LTS states is proposition preserving, which implies that the labeling function for the TS abstraction is well defined and that the simulation or the bisimulation relations between the abstract and concrete model are retained. If the abstract LTS bisimulates the MPL model, one can verify the general class of specifications expressed via LTL formulae [15, Ch. 5]. Otherwise, if the LTS simulates the MPL model, one can exclusively verify safety properties, a strict subset of LTL [15, Sec. 7.4].

With focus on the verification over transition labels, let us consider a set of atomic propositions based on their collection. Again, the inverse image w.r.t. the labeling function of each atomic proposition equals to a union of transition labels. In order to obtain a proposition preserving partition the transition labels have to be pairwise disjoint, which is usually the case for autonomous MPL models (see example in this work). If this is not the case (as it usually happens for nonautonomous models), one can attempt state space refinement procedures that can enable unique association of atomic propositions to labels. Alternatively, one can declare that a transition satisfies an atomic proposition depending on the particular specification under study. Computationally, this can be done by exploiting the DBM inclusion property [16, Sec. 4.1], or by checking non-emptiness of the intersection among DBM (cf. Section II-C).

We use the SPIN model checker [21] to verify general LTL specifications. Given an MPL system expressed within the MATLAB environment, we first abstract the MPL system into an LTS within MATLAB (cf. previous sections), then export the obtained data structure into the PROMELA language and feed it, along with an LTL formula expressing a specification

of interest for the model, to SPIN. SPIN deals with both deterministic and nondeterministic LTS. The outcome of the model checking procedure is the set of states of the LTS satisfying the given formula, which is known as the satisfiability set. The satisfiability set corresponds to partitioning regions on the original MPL model, which comprise points that also verify the given formula. Whenever this set is empty, the model checker returns a counterexample, namely a trajectory that does not satisfy the formula.

A. Autonomous Case

In the following example we discuss the verification procedure over the MPL model in (2).

Example: The given MPL model is abstracted into the LTS shown in Fig. 11 (transitions $\delta_\sim$ are marked with bold lines and labels $L_\sim$ referring to time delay between consecutive events, cf. Section IV). Notice that the transition labels are non-intersecting.

Recall that Section II has looked at the concept of cyclicity and at the eigenspace. In order to identify the eigenspace, we can use the formula $\bigvee_{l_\sim \in L_\sim}((\Box l_\sim) \ \wedge \ C(l_\sim))$, where $C(l_\sim)$ is a Boolean function returning true if $l_\sim$ is a vector of finite constants, and false otherwise. This LTL formula is not verified by any state (the satisfiability set is empty), because no partitioning regions correspond to the eigenspace (which is indeed contained in region $s^6_\sim$).

Let us consider an autonomous MPL model with system matrix A and focus on its whole periodic behavior. By direct inspection the LTS in Fig. 11, we conclude that $E(A^{\otimes c})$ corresponds to the set $\{s^2_\sim, s^6_\sim, s^7_\sim\}$. If we are interested in computing it, then we can express the formula $\Psi = \bigvee_{l_\sim \in L_\sim} \Box(l_\sim \wedge \bigcirc^c l_\sim)$, where $\bigcirc^c$ denotes the application of the next operator c times [15, Remark 5.15]. In this example, $c = 2$ and the LTL formula is verified by the set $\{s^2_\sim, s^6_\sim, s^7_\sim\}$. We can furthermore characterize the set $\{x \in \mathbb{R}^n : k_0(x) \leq k\}$, for $k \in \mathbb{N} \cup \{0\}$, by computing the satisfiability set of the LTL formula $\Diamond^{\leq k} \Psi$ [15, Remark 5.15]. By extension, if there exists an $x \in \mathbb{R}^n$ such that $k_0(x) = k_0(A)$, we can formulate the value of $k_0(A)$ as a function of the previous LTL formula, as $k_0(A) = \arg\min_k \left\{\Diamond^{\leq k} \Psi\right\}$.

Along with the above properties related to the periodic regime of the MPL model, we may be interested in model checking general formulas, such as the following reach-avoid specification: $\Diamond\psi_1 \wedge \Box\neg\psi_2$, where ψ_1 denotes the incoming label of $s^2_\sim$, whereas ψ_2 the union of those of $s^3_\sim$ and $s^4_\sim$. This formula can be expressed as "the trajectory will eventually reach set $s^2_\sim$, while never entering set $s^3_\sim \cap s^4_\sim$." The satisfiability set results in $\{s^1_\sim, s^2_\sim, s^5_\sim, s^7_\sim\}$. □

B. Nonautonomous Case

If the abstract LTS bisimulates the nonautonomous MPL model, one can verify the general class of specifications expressed via LTL formulae [15, Ch. 5]. Otherwise, if the LTS simulates the nonautonomous MPL model, one can verify safety properties, a strict subset of LTL [15, Sec. 7.4]. Nonautonomous MPL models allow for controller synthesis. In the first case, an approach to synthesis is to negate the specification of interest and retrieve the successful control policy from a counterexample generated by the model checker. In the latter case, we can instead leverage a control synthesis approach based on games, as described in [31].

Example: In [4, Sec. 0.1], the autonomous two-dimensional MPL system (2) is used to model a simple railway network with 2 stations. Let us focus on the LTS in Fig. 10a, which is obtained for $U = s^6_\sim$, but where properties are defined over the state labels (describing the difference in timings of the same event, and characterized by the partitions in Fig. 8a). Suppose that there is a requirement on the departure times at station 1 to be at least 2 time steps before those at station 2, and at most the same times as those at station 2. From Fig. 11, the collection of states $s^2_\sim \cup s^6_\sim \cup s^7_\sim$ verifies this requirement – we thus introduce the corresponding LTL formula $\Box(s^2_\sim \vee s^6_\sim \vee s^7_\sim)$. The LTL formula is verified within the whole $\{s^2_\sim, s^6_\sim, s^7_\sim\}$. This allows to conclude that, as long as the initial condition is in the safe set, the trajectory will always reside there. Similarly, in the case where $U = s^2_\sim \cup s^6_\sim$, the same safety property over the LTS in Fig. 10b (with labels on the states) can be model checked, and in this instance the LTL formula admits the same satisfiability set. □

VII. Conclusions

This work has introduced a new technique to generate finite abstractions of autonomous and nonautonomous Max-Plus-Linear (MPL) models, characterized as finite-state Labeled Transition Systems (LTS). The procedure is based on the partitioning (covering) of the state (input) space, on the study of the one-step dynamics to relate partitioning regions, and on the use of the timing of events to associate labels on the discrete abstraction. The resulting finite LTS abstraction has been shown to either simulate or bisimulate the original MPL model.

The computational complexity of the approach has been fully quantified and its performance has been tested on a numerical benchmark, which has displayed a bottleneck that mainly depends on the number of generated partitioning regions. Still, the abstraction procedure comfortably manages models with reasonable size (15-dimensional, in the autonomous case) and can then be employed to study properties of the original MPL model in an original manner. Along this line, the authors are interested in extending the analysis and control synthesis over the MPL models by means of formal verification techniques applied over finite LTS abstractions. Considering Metric Temporal Logic [32] as an extension of LTL represents a first meaningful goal.

With the objective of obtaining an LTS that only simulates the original MPL model, it is clear that partitioning procedures obtaining a quotient set with lower cardinality lead to a tradeoff between computability and abstraction precision. The authors are also interested in employing new data structures and operations to improve the developed software toolbox [22]. Formula-based abstractions and counterexample-guided refinements represent avenues to explore. Finally, the authors plan to further connect the proposed abstraction technique to related approaches developed for PWA models [18], as well as to results developed for Petri Nets [33].

REFERENCES

[1] D. Adzkiya, B. De Schutter, and A. Abate, "Abstraction and verification of autonomous max-plus-linear systems," in *Proc. 31st Amer. Control Conf. (ACC'12)*, Montreal, CA, Jun. 2012, pp. 721–726.

[2] F. Baccelli, G. Cohen, and B. Gaujal, "Recursive equations and basic properties of timed Petri nets," *Discrete Event Dynamic Systems: Theory and Applications*, vol. 1, no. 4, pp. 415–439, Jun. 1992.

[3] H. P. Hillion and J. P. Proth, "Performance evaluation of job-shop systems using timed event graphs," *IEEE Trans. Autom. Control*, vol. 34, no. 1, pp. 3–9, Jan. 1989.

[4] B. Heidergott, G. 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, 2006.

[5] B. Roset, H. Nijmeijer, J. van Eekelen, E. Lefeber, and J. Rooda, "Event driven manufacturing systems as time domain control systems," in *Proc. 44th IEEE Conf. Decision and Control and European Control Conf. (CDC-ECC'05)*, Dec. 2005, pp. 446–451.

[6] J. van Eekelen, E. Lefeber, and J. Rooda, "Coupling event domain and time domain models of manufacturing systems," in *Proc. 45th IEEE Conf. Decision and Control (CDC'06)*, Dec. 2006, pp. 6068–6073.

[7] 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*, vol. 303, no. 0, pp. 128–140, Jun. 2012.

[8] F. Baccelli, G. Cohen, G. Olsder, and J.-P. Quadrat, *Synchronization and Linearity, An Algebra for Discrete Event Systems*. John Wiley and Sons, 1992. [Online]. Available: http://www-rocq.inria.fr/metalau/cohen/SED/book-online.html

[9] S. Gaubert and R. Katz, "Reachability and invariance problems in max-plus algebra," in *Positive Systems*, ser. Lecture Notes in Control and Information Science, L. Benvenuti, A. De Santis, and L. Farina, Eds. Springer Berlin Heidelberg, Apr. 2003, vol. 294, ch. 4, pp. 15–22.

[10] R. D. Katz, "Max-plus (A, B)-invariant spaces and control of timed discrete-event systems," *IEEE Trans. Autom. Control*, vol. 52, no. 2, pp. 229–241, Feb. 2007.

[11] B. De Schutter, "On the ultimate behavior of the sequence of consecutive powers of a matrix in the max-plus algebra," *Linear Algebra and its Applications*, vol. 307, no. 1-3, pp. 103–117, Mar. 2000.

[12] G. J. Pappas, "Bisimilar linear systems," *Automatica*, vol. 39, no. 12, pp. 2035–2047, Dec. 2003.

[13] R. Milner, *Communication and Concurrency*. Englewood Cliffs, NJ: Prentice-Hall, 1989.

[14] E. D. Sontag, "Nonlinear regulation: The piecewise-linear approach," *IEEE Trans. Autom. Control*, vol. 26, no. 2, pp. 346–358, Apr. 1981.

[15] C. Baier and J.-P. Katoen, *Principles of Model Checking*. The MIT Press, 2008.

[16] D. L. Dill, "Timing assumptions and verification of finite-state concurrent systems," in *Automatic Verification Methods for Finite State Systems*, ser. Lecture Notes in Computer Science, J. Sifakis, Ed. Springer Berlin Heidelberg, 1990, vol. 407, ch. 17, pp. 197–212.

[17] R. Alur, T. Henzinger, G. Lafferriere, and G. J. Pappas, "Discrete abstractions of hybrid systems," *Proc. IEEE*, vol. 88, no. 7, pp. 971–984, Jul. 2000.

[18] B. Yordanov and C. Belta, "Formal analysis of discrete-time piecewise affine systems," *IEEE Trans. Autom. Control*, vol. 55, no. 12, pp. 2834–2840, Dec. 2010.

[19] W. Heemels, B. De Schutter, and A. Bemporad, "Equivalence of hybrid dynamical models," *Automatica*, vol. 37, no. 7, pp. 1085–1091, Jul. 2001.

[20] P. Abdulla and A. Nylén, "Timed Petri Nets and BQOs," in *Applications and Theory of Petri Nets*, ser. Lecture Notes in Computer Science, J.-M. Colom and M. Koutny, Eds. Springer Berlin Heidelberg, 2001, vol. 2075, pp. 53–70.

[21] G. Holzmann, *The SPIN Model Checker: Primer and Reference Manual*. Addison-Wesley, 2003.

[22] D. Adzkiya and A. Abate, "VeriSiMPL: Verification via biSimulations of MPL models," in *Int. Conf. Quantitative Evaluation of SysT. (QEST'13)*, ser. Lecture Notes in Computer Science, K. Joshi *et al.*, Eds., vol. 8054. Springer Berlin Heidelberg, Sep. 2013, pp. 253–256.

[23] R. Cuninghame-Green, "Minimax algebra and applications," *Fuzzy Sets and Systems*, vol. 41, no. 3, pp. 251–267, Jun. 1991.

[24] J. Cochet-Terrasson, G. Cohen, S. Gaubert, M. McGettrick, and J.-P. Quadrat, "Numerical computation of spectral elements in max-plus algebra," in *IFAC Conf. Syst. Structure and Control*, Nantes, FR, Jul. 1998, pp. 699–706.

[25] B. Charron-Bost, M. Függer, and T. Nowak, "On the transience of linear max-plus dynamical systems," *CoRR*, vol. abs/1111.4600, 2011.

[26] M. Hartmann and C. Arguelles, "Transience bounds for long walks," *Mathematics of Operations Research*, vol. 24, no. 2, pp. 414–439, May 1999.

[27] M. Péron and N. Halbwachs, "An abstract domain extending difference-bound matrices with disequality constraints," in *Verification, Model Checking, and Abstract Interpretation*, ser. Lecture Notes in Computer Science, B. Cook and A. Podelski, Eds. Springer Berlin Heidelberg, 2007, vol. 4349, ch. 20, pp. 268–282.

[28] R. W. Floyd, "Algorithm 97: Shortest path," *Commun. ACM*, vol. 5, no. 6, p. 345, Jun. 1962.

[29] R. Bellman, "On a routing problem," *Quart. Appl. Math.*, vol. 16, pp. 87–90, 1958.

[30] A. H. Land and A. Doig, "An automatic method of solving discrete programming problems," *Econometrica*, vol. 28, no. 3, pp. 497–520, Jul. 1960.

[31] M. Kloetzer and C. Belta, "Dealing with nondeterminism in symbolic control," in *Hybrid Systems: Computation and Control (HSCC'08)*, ser. Lecture Notes in Computer Science, M. Egerstedt and B. Mishra, Eds. Springer Berlin Heidelberg, 2008, vol. 4981, ch. 21, pp. 287–300.

[32] Y. Lakhneche and J. Hooman, "Reasoning about durations in metric temporal logic," in *Formal Techniques in Real-Time and Fault-Tolerant Systems*, ser. Lecture Notes in Computer Science, H. Langmaack, W.-P. Roever, and J. Vytopil, Eds. Springer Berlin Heidelberg, 1994, vol. 863, ch. 24, pp. 488–510.

[33] M. Kloetzer, C. Mahulea, C. Belta, and M. Silva, "An automated framework for formal verification of timed continuous Petri nets," *IEEE Trans. Ind. Informat.*, vol. 6, no. 3, pp. 460–471, 2010.