Computation of the Transient in Max-Plus Linear Systems via SMT-Solving

Alessandro Abate[1 [0000−0002−5627−9093]], Alessandro Cimatti[2 [0000−0002−1315−6990]], Andrea Micheli[2 [0000−0002−6370−1061]], and Muhammad Syifa'ul Mufid[1 [0000−0003−0817−1106]]

[1] Department Computer Science, University of Oxford, UK
{alessandro.abate,muhammad.syifaul.mufid}@cs.ox.ac.uk
[2] Fondazione Bruno Kessler, Italy
{cimatti,amicheli}@fbk.eu

Abstract. This paper proposes a new approach, grounded in Satisfiability Modulo Theories (SMT), to study the transient of a Max-Plus Linear (MPL) system, that is the number of steps leading to its periodic regime. Differently from state-of-the-art techniques, our approach allows the analysis of periodic behaviors for subsets of initial states, as well as the characterization of sets of initial states exhibiting the same specific periodic behavior and transient. Our experiments show that the proposed technique dramatically outperforms state-of-the-art methods based on max-plus algebra computations for systems of large dimensions.

1 Introduction

Max-Plus Linear (MPL) systems are a class of discrete-event systems (DES) that are based on the max-plus algebra, an algebraic system using the two operations of maximisation and addition. MPL systems are employed to model applications with features of synchronization without concurrency, and as such are widely used for applications in transportation networks [4], manufacturing [14] and biological systems [6,10]. In MPL models, the states correspond to time instances related to discrete events.

A fundamental and well-studied property of MPL systems is related to the periodic behavior of its states: from an initial vector, the trajectories of an MPL system are eventually periodic (in max-plus algebraic sense) starting from a specific event index called the *transient*, and with a specific period called *cyclicity* [4]. As explained in [14, Section 3.1], the transient is closely related to the notion of *cycle-time* vector, which governs the asymptotic behaviour of MPL systems.

The transient is key to solve a number of fundamental problems of MPL systems, such as reachability analysis [17] and bounded model checking [18]: it plays a crucial role as the "completeness threshold" (namely, the maximum iteration that is sufficient for the termination of the algorithm) [9] for those two problems. The computation of the transient is an interesting problem, as it is in general not correlated to the dimension of the MPL system. Thus, it is possible

for the resulting transient to be relatively large for a small-dimensional MPL system. There are several known upper bounds [8,16,19,20] for the transient, which are mostly computed via the corresponding precedence graph and are, in practice, much larger than the actual values.

This paper has two specific contributions. The first is to provide a novel procedure to compute the transient by means of Satisfiability Modulo Theory (SMT) solving [5]. The main idea underpinning the new method is to transform the problem instance into a formula in difference logic, and then passing the formula into an SMT solver, which outputs the transient. More precisely, in order to check the validity of the formula, we check the unsatisfiability of its negation. If the SMT solver reports "satisfied", then the original formula admits a counterexample, from which we can refine the formula. On the other hand, if SMT solver reports "unsatisfied", then from the formula we obtain the transient and the corresponding cyclicity. The second contribution of this work is to provide a procedure to synthesize the subset of the state space of an MPL system that corresponds to a specific transient/cyclicity pair. We show that one can partition the state space into sets corresponding to different transient/cyclicity pairs.

The rest of the paper is structured as follows. Section 2 describes the basics of MPL systems, including the key notion of cycle-time vector. In Section 3, we provide the formal definition of transient over MPL systems and also a standard linear algebra procedure, based on matrix multiplication, to compute the transient (cf. Algorithm 1), which is later used as a benchmark. Section 4 is divided into four parts. The first part provides the background on SMT and including the underlying relevant theory. The translation of inequalities over max-plus algebra to formulae in difference logic is explained in the second part. In the third part, we provide SMT-based methods (cf. Algorithms 2 and 3) to compute the transient. The spatial synthesis problem is discussed in the last part. The comparison of the performance of the novel algorithm against the standard linear algebra procedure is presented in Section 5. The paper is concluded with Section 6. The proofs of the propositions and theorems are presented in a longer version of this paper [1]. The developed code and generated data can be found in `https://es.fbk.eu/people/amicheli/resources/formats20/`.

2 Preliminaries

2.1 Max-Plus Linear Systems

Max-plus algebra is a modification of linear algebra derived over the max-plus semiring $(\mathbb{R}_{\max}, \oplus, \otimes)$ where $\mathbb{R}_{\max} := \mathbb{R} \cup \{\varepsilon := -\infty\}$ and $a \oplus b := \max\{a, b\}$, $a \otimes b := a + b$, for all $a, b \in \mathbb{R}_{\max}$. The zero and unit elements of $\mathbb{R}_{\max}$ are ε and 0, respectively. The max-plus algebraic operations can be extended to matrices

and vectors in a natural way. For $A, B \in \mathbb{R}_{\max}^{m\times n}, C \in \mathbb{R}_{\max}^{n\times p}$ and $\alpha \in \mathbb{R}_{\max}$,

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

Given $A \in \mathbb{R}_{\max}^{n\times n}$ and $t \in \mathbb{N}$, $A^{\otimes t}$ denotes $A \otimes \ldots \otimes A$ (t times). For $t = 0$, $A^{\otimes 0}$ is an n-dimensional max-plus identity matrix where all diagonal and non-diagonal elements are 0 and ε, respectively.

Given $V = \{v_1, \ldots, v_p\}$ as a set of vectors in $\mathbb{R}_{\max}^n$, we use the same notation to denote a matrix where all columns are in V i.e., $V(\cdot, i) = v_i$ for $1 \le i \le p$. A vector $v \in \mathbb{R}^n$ is *a max-plus linear combination* of V if $v = \alpha_1 \otimes v_1 \oplus \ldots \oplus \alpha_p \otimes v_p$ for some scalars $\alpha_1, \ldots, \alpha_p \in \mathbb{R}$ or equivalently there exists $w \in \mathbb{R}^p$ such that $V \otimes w = v$. The set of all max-plus linear combinations of V is called *max-plus cone*[1] and is denoted by $\mathsf{cone}(V)$ [7]. It is formally expressed as

$$\mathsf{cone}(V) = \{V \otimes w \mid w \in \mathbb{R}^p\}. \tag{1}$$

Furthermore, we denote as $v_1, \ldots, v_p$ the basis of $\mathsf{cone}(V)$. Notice that the max-plus cone is closed under the operations $\oplus$ and $\otimes$: if v, w are in $\mathsf{cone}(V)$, then so is $\alpha \otimes v \oplus \beta \otimes w$ for $\alpha, \beta \in \mathbb{R}$. Max-plus cones are the analogues of vector subspaces in classical linear algebra.

A dynamical system over the max-plus algebra is called a Max-Plus Linear (MPL) system and is defined as

$$\mathbf{x}(k+1) = A \otimes \mathbf{x}(k), \quad k = 0, 1, \ldots \tag{2}$$

where $A \in \mathbb{R}_{\max}^{n\times n}$ is the system matrix, and vector $\mathbf{x}(k) = [x_1(k) \ \ldots \ x_n(k)]^\top$ encodes the state variables [4]. For example, $\mathbf{x}$ can be used to represent the time stamps associated to the discrete events, while k corresponds to the events counter. Hence, it is more convenient to consider $\mathbb{R}^n$ (instead of $\mathbb{R}_{\max}^n$) as the state space. Applications of MPL systems are significantly found on systems where the time variable is essential, such as transportation networks [14], scheduling or [3] manufacturing [15] problems, or biological systems [6,10].

Definition 1 (Precedence Graph [4]). The precedence graph of $A \in \mathbb{R}_{\max}^{n\times n}$, denoted by $\mathcal{G}(A)$, is a weighted directed graph with nodes $1, \ldots, n$ and an edge from j to i with weight $A(i,j)$ for each $A(i,j) \neq \varepsilon$. □

Definition 2 (Regular Matrix [14]). A matrix $A \in \mathbb{R}_{\max}^{n\times n}$ is called *regular* if A contains at least one finite element in each row. □

Definition 3 (Irreducible Matrix [4]). A matrix $A \in \mathbb{R}_{\max}^{n\times n}$ is called *irreducible* if $\mathcal{G}(A)$ is strongly connected. □

[1] Unlike in [7,13], we require each max-plus cone to be a subset of $\mathbb{R}^n$.

Recall that a directed graph is strongly connected if, for any two different nodes i, j, there exists a path from i to j. The weight of a path $p = i_1 i_2 \dots i_k$ is equal to the sum of the edge weights in p. A circuit, namely a path that begins and ends at the same node, is called *critical* if it has maximum average weight, which is the weight divided by the length of the path [4].

Each irreducible matrix $A \in \mathbb{R}_{\max}^{n \times n}$ admits a unique max-plus eigenvalue $\lambda \in \mathbb{R}$ and a corresponding max-plus eigenspace $E(A) = \{x \in \mathbb{R}^n \mid A \otimes x = \lambda \otimes x\}$[2]. The scalar λ is equal to the average weight of critical circuits in $\mathcal{G}(A)$, and $E(A)$ can be computed from $A_\lambda^+ = \bigoplus_{k=1}^{n}((-\lambda) \otimes A)^{\otimes k}$. More specifically, $E(A)$ is the max-plus linear combination of the i^{th} column of A_λ^+, for i such that $A_\lambda^+(i,i) = 0$ [4]. Thus, the eigenspace $E(A)$ is a max-plus cone. A reducible matrix may have multiple eigenvalues, where the maximum one equals to the average weight of critical circuits of $\mathcal{G}(A)$.

Example 1. Consider a two-dimensional MPL system $\mathbf{x}(k+1) = A \otimes \mathbf{x}(k)$, with

$$A = \begin{bmatrix} 2 & 5 \\ 3 & 3 \end{bmatrix},$$

which represents a simple railway network between two cities [4, Sec. 0.1], as shown in Figure 1. The dynamics w.r.t. (2) can be expressed as

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} \max\{x_1(k)+2, x_2(k)+5\} \\ \max\{x_1(k)+3, x_2(k)+3\} \end{bmatrix}.$$

For $1 \le i, j \le 2$, the element $A(i,j)$ corresponds to the time taken to travel from station S_j to S_i, while $x_i(k)$ is the time of the k-th departure at station S_i.

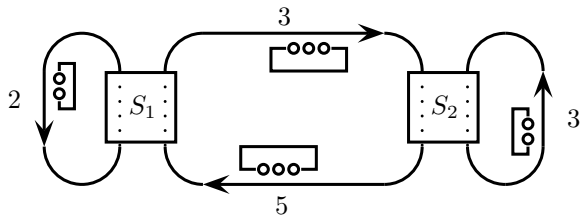

Fig. 1: A simple railway network represented by an MPL system.

From an initial vector, say $\mathbf{x}(0) = [0\ 0]^\top$, one can compute vectors denoting the next departure times, as follows

$$\begin{bmatrix} 5 \\ 3 \end{bmatrix}, \begin{bmatrix} 8 \\ 8 \end{bmatrix}, \begin{bmatrix} 13 \\ 11 \end{bmatrix}, \begin{bmatrix} 16 \\ 16 \end{bmatrix}, \dots.$$

Leaving the details aside, the matrix A has eigenvalue $\lambda = 4$ and eigenspace $E(A) = \{\mathbf{x} \in \mathbb{R}^2 \mid x_1 - x_2 = 1\}$. □

[2] Because we regard $\mathbb{R}^n$ to be the state space of the MPL system (2), we only consider eigenvectors with finite elements.

2.2 Cycle-Time Vector

This section presents the definition of cycle-time vector of MPL systems. The computation of the cycle-time vector is indeed important, as it can shed light on the asymptotic behavior of MPL systems. In this section, we show its relationship with the eigenspace and eigenvalue of the underlying state matrix. Furthermore, as it will be clear in Section 3, the cycle-time vector can be used to determine whether the states of an MPL system are eventually periodic.

Definition 4 (Cycle-Time Vector [14]). Consider a regular MPL system (2), and assume that for all $j \in \{1, \ldots, n\}$ the quantity η_j, defined by

$$\eta_j = \lim_{k \to +\infty} (x_j(k)/k),$$

exists. Then the vector $\chi = [\eta_1 \ \ldots \ \eta_n]^\top$ is called the the cycle-time vector of the given sequence $\mathbf{x}(k)$ with respect to A. □

It has been shown in [14, Theorem 3.11] that if the cycle-time vector of A exists for at least one initial vector then it exists for any initial vector. Instead of computing the limit as in Definition 4, the cycle-time vector can be generated using a procedure [12, Algorithm 31].

Theorem 1 ([12]). Suppose we have a regular MPL system (2). For each $\mathbf{x}(0) \in \mathbb{R}^n$ there exist natural numbers p, q such that $\mathbf{x}(k+q) = (q \times \chi) + \mathbf{x}(k)$ for all $k \geq p$, where $\chi = [\eta_1 \ \ldots \ \eta_n]^\top$ is the cycle-time vector of A and the multiplication $q \times \chi$ is defined in the classical algebra. □

By Theorem 1, the trajectories of a regular MPL system (2) starting from any initial vector is governed by the corresponding cycle-time vector χ. In general, the elements of χ may be different, as shown in [12, Example 1]. However, if $E(A) \neq \emptyset$ then the elements of χ are all equal.

Proposition 1. Suppose a regular MPL system (2) has maximum eigenalue λ. The eigenspace $E(A)$ is not empty iff $\chi = [\lambda \ \ldots \ \lambda]^T \in \mathbb{R}^n$. □

3 Transient in Max-Plus-Linear Systems

The transient of MPL systems is related to the sequence of the powers of matrix A, namely $A^{\otimes k}$ for $k \geq 0$.

Proposition 2 (Transient [4,14]). For an irreducible matrix $A \in \mathbb{R}_{\max}^{n \times n}$ and its max-plus eigenvalue $\lambda \in \mathbb{R}$, there exist $k_0, c \in \mathbb{N}_0$, such that $A^{\otimes(k+c)} = (\lambda \times c) \otimes A^{\otimes k}$ for all $k \geq k_0$. The smallest such k_0 and c are called the *transient* and the *cyclicity* of A, respectively. □

For the rest of this paper, we denote the transient and the cyclicity of A as $\mathsf{tr}(A)$ and $\mathsf{cyc}(A)$, respectively. While $\mathsf{cyc}(A)$ is related to critical circuits in

the precedence graph $\mathcal{G}(A)$ (see [4, Definition 3.94] for more details[3]), $\mathsf{tr}(A)$ is unrelated to the dimension of A. Even for a small n, the transient of $A \in \mathbb{R}_{\max}^{n\times n}$ can be large. Upper bounds of the transient have been discussed in [8,16,19,20].

By Proposition 2, each *irreducible* MPL system enjoys a *periodic* behaviour with a rate λ: for each initial vector $\mathbf{x}(0) \in \mathbb{R}^n$ we have $\mathbf{x}(k + \mathsf{cyc}(A)) = (\lambda \times \mathsf{cyc}(A)) \otimes \mathbf{x}(k)$ for all $k \geq \mathsf{tr}(A)$ where the vectors $\mathbf{x}(1), \mathbf{x}(2)$ are computed recursively by (2). A similar condition may be found on reducible MPL systems: we denote the corresponding transient and cyclicity as global, as per Proposition 2. The local transient and cyclicity for a specific initial vector $\mathbf{x} \in \mathbb{R}^n$ and for a set $X \subseteq \mathbb{R}^n$ is defined as follows.

Definition 5. Given $A \in \mathbb{R}_{\max}^{n\times n}$ with maximum eigenvalue λ and an initial vector $\mathbf{x} \in \mathbb{R}^n$, the local transient and cyclicity of $\mathbf{x}(0)$ w.r.t. A are respectively the smallest $k_0, c \in \mathbb{N}_0$ such that $\mathbf{x}(k + c) = \lambda c \otimes \mathbf{x}(k)$ for all $k \geq k_0$. We denote those scalars as $\mathsf{tr}(A, \mathbf{x})$ and $\mathsf{cyc}(A, \mathbf{x})$, respectively. Furthermore, for $X \subseteq \mathbb{R}^n, \mathsf{tr}(A, X) = \max\{\mathsf{tr}(A, \mathbf{x}(0)) \mid \mathbf{x}(0) \in X\}$ and $\mathsf{cyc}(A, X) = \mathtt{lcm}\{\mathsf{cyc}(A, \mathbf{x}(0)) \mid \mathbf{x}(0) \in X\}$, where $\mathtt{lcm}$ stands for the "least common multiple". □

By definition, we have $\mathsf{tr}(A, \mathbb{R}^n) = \mathsf{tr}(A)$. For a max-plus cone $X = \mathsf{cone}(V)$, we show that the local cyclicity and transient can be computed from the corresponding bases, provided that $\mathsf{tr}(A, v_i)$ exists for all $1 \leq i \leq p$.

Proposition 3. Given a max-plus cone $X = \mathsf{cone}(V)$ where $V = \{v_1, \ldots, v_p\}$, we have $\mathsf{tr}(A, X) = \mathsf{tr}(A, V) = \max\{\mathsf{tr}(A, v) \mid v \in V\}$, and $\mathsf{cyc}(A, X) = \mathsf{cyc}(A, V) = \mathtt{lcm}\{\mathsf{cyc}(A, v) \mid v \in V\}$. □

Definition 6. Suppose we have a regular matrix $A \in \mathbb{R}_{\max}^{n\times n}$. The underlying MPL system (2) is classified into three categories as follows:

i. *never periodic*: $\mathsf{tr}(A, \mathbf{x}(0))$ does not exist for all $\mathbf{x}(0) \in \mathbb{R}^n$,
ii. *boundedly periodic*: $\mathsf{tr}(A, \mathbf{x}(0))$ exists for all $\mathbf{x}(0) \in \mathbb{R}^n$ and $\mathsf{tr}(A)$ exists,
iii. *unboundedly periodic*: $\mathsf{tr}(A, \mathbf{x}(0))$ exists for all $\mathbf{x}(0) \in \mathbb{R}^n$ but $\mathsf{tr}(A)$ does not.

We call (2) *periodic* if it is either *unboundedly periodic* or *boundedly periodic*. □

We show that the periodic behavior of an MPL system is indeed related to the eigenspace and cycle-time vector of its corresponding state matrix.

Theorem 2. Suppose we have a regular matrix $A \in \mathbb{R}_{\max}^{n\times n}$ with a maximum eigenvalue λ and cycle-time vector χ. The following statements are equivalent.

a. The underlying MPL system (2) is *periodic*.
b. The corresponding cycle-time vector is $\chi = [\lambda \ \ldots \ \lambda]^\top \in \mathbb{R}^n$.
c. The eigenspace $E(A)$ is not empty. □

Proposition 4. Suppose we have a regular matrix $A \in \mathbb{R}_{\max}^{n\times n}$ with maximum eigenvalue λ and non-empty eigenspace $E(A)$. If there exist $i \in \{1, \ldots, n\}$ and natural numbers k_0', c' such that $A^{\otimes k+c'}(\cdot, i) = \mu c' \otimes A^{\otimes k}(\cdot, i)$ for all $k \geq k_0'$ with $\mu < \lambda$ then (2) is *unboundedly periodic*. □

[3] In this reference, one can find the cyclicity for reducible and irreducible matrices using graph-theoretical approaches.

We now will provide the procedure to compute the transient of MPL systems. As per Proposition 2, the common method to obtain the (global) transient of $A \in \mathbb{R}_{\max}^{n\times n}$ is by computing the power of the matrix $A^{\otimes 0}, A^{\otimes 1}, \ldots$ until we find $k_0 \geq 0$ such that $A^{\otimes(k_0+c)} = \lambda^{\otimes c} \otimes A^{\otimes k_0}$ where λ, c is respectively the max-plus eigenvalue and cyclicity of A. Similarly, to find the transient of A w.r.t. a max-plus cone $X = \mathsf{cone}(V)$ one needs to compute $A^{\otimes 0} \otimes V, A^{\otimes 1} \otimes V, \ldots$.

Algorithm 1 illustrates the procedure to compute transient (and cyclicity) for a max-plus cone $\mathsf{cone}(V)$ w.r.t. $A \in \mathbb{R}_{\max}^{n\times n}$. While originally designed for irreducible matrices, it also can be applied to find the transient of reducible matrices (if any). For this reason, we assign a maximum bound as termination condition. It is important to note that Algorithm 1 can also be used to compute the local transient and cyclicity for a vector: that is, when V has only one column. The algorithm starts by computing the cycle-time vector χ of the state matrix. If the entries of χ are not all the same then the transient for $\mathsf{cone}(V)$ does not exist. In line 11, we perform equality checking w.r.t. a scalar between $A^{\otimes it-m} \otimes V$ and $A^{\otimes it} \otimes V$.

By Theorem 2 and Proposition 4, one can classify an MPL system (2) into a category in Definition 6. As a result, determining the existence of global transient is a decidable problem. For *boundedly periodic* MPL systems, computing the global transient is also a decidable problem. This is because they ensure the existence of a finite transient, meaning that Algorithm 1 eventually terminates. However, Algorithm 1 is sound but does not necessarily terminate (in general) for *unboundedly periodic* MPL systems.

Algorithm 1 Computation of cyclicity and transient of A w.r.t. $\mathsf{cone}(V)$

```
1: function TransCone(A, V, N)
2:     M ← EmptyVector()                          ▷ empty vector used to store
3:     M.push_back(V)                               A^0 ⊗ V, A^1 ⊗ V, ...
4:     it ← 0                                                ▷ number of iterations
5:     χ ← CycleTimeVector(A)                     ▷ computing cycle-time vector
6:     if elements of χ are all equal then
7:         while (it ≤ N) do
8:             M.push_back(A ⊗ M[it])
9:             it ← it + 1
10:            for 1 ≤ m < it do
11:                if (M[it] = (λ × m) ⊗ M[it − m]) then
12:                    return ⟨it − m, m⟩
13:        if (it > N) then
14:            print "terminated after reaching maximum bound"
15:    else
16:        print "the transient does not exist"
```

Remark 1. The procedure in Algorithm 1 only employs matrix operations in max-plus algebra. It can be improved by computing the cyclicity of the matrix from the corresponding precedence graph. If the resulting cyclicity is c then the range in line 10 of Algorithm 1 can be taken between 1 and c. □

Example 2. Suppose we have a regular and reducible MPL system $\mathbf{x}(k+1) = B \otimes \mathbf{x}(k)$, where

$$B = \begin{bmatrix} 2 & 8 & \varepsilon \\ 10 & 5 & \varepsilon \\ 3 & \varepsilon & a \end{bmatrix}, \tag{3}$$

and where $a \geq 8$. The corresponding eigenvalue for B is $\lambda = 9$ if $8 \leq a \leq 9$; $\lambda = a$ otherwise. Taking the power of the matrix, we have

$$B^{\otimes 2} = \begin{bmatrix} 18 & 13 & \varepsilon \\ 15 & 18 & \varepsilon \\ a+3 & 11 & 2a \end{bmatrix}, B^{\otimes 3} = \begin{bmatrix} 23 & 26 & \varepsilon \\ 28 & 23 & \varepsilon \\ b & a+11 & 3a \end{bmatrix}, B^{\otimes 4} = \begin{bmatrix} 36 & 31 & \varepsilon \\ 33 & 36 & \varepsilon \\ a+b & 2a+11 & 4a \end{bmatrix},$$

where $b = \max\{21, 2a+3\}$. One can check that, for $a > 9$, the matrix does not admit an eigenvector over $\mathbb{R}^3$ (but it still has eigenvector over $\mathbb{R}^3_{\max}$). As a result, B is *never periodic*.

On the other hand, for $8 \leq a \leq 9$, the corresponding $E(B)$ is not empty. Thus, B is *periodic*. Furthermore, for $k \geq 2$, we have

$$[B^{\otimes k+2}](i,\cdot) = \begin{cases} 18 \otimes [B^{\otimes k}](\cdot, i), & \text{if } i \in \{1,2\}, \\ 2a \otimes [B^{\otimes k}](\cdot, i), & \text{if } i = 3, \end{cases}$$

which shows that B is *boundedly periodic* with global transient $\mathsf{tr}(B) = 2$ if and only if $a = 9$. Thus, when $8 \leq a < 9$ B is *unboundedly periodic*. □

4 Computation of Transient of MPL Systems with SMT

This section describes a new procedure to compute the transient of MPL systems by means of Satisfiability Modulo Theories (SMT). We first mention some basic notions on SMT.

4.1 Background on SMT

Given a first-order formula ψ in a background theory T, the Satisfiability Modulo Theory (SMT) problem consists in deciding whether there exists a model (i.e. an assignment to the free variables in ψ) that satisfies ψ [5]. For example, consider the formula $(x \leq y) \wedge (x+3 = z) \vee (z \geq y)$ within the theory of real numbers. The formula is satisfiable and a valid model is $\{x := 5,\ y := 6,\ z := 8\}$.

SMT solvers can support different theories. A widely used theory is Linear Real Arithmetic (LRA). A formula in LRA is an arbitrary Boolean combination, or universal ($\forall$) and existential ($\exists$) quantification, of atoms in the form $\sum_i a_i x_i \bowtie c$ where $\bowtie \in \{>, <, \geq, \leq, \neq, =\}$, every x_i is a real variable, and every a_i and c are rational constants. Difference logic (RDL) is the subset of LRA in which all atoms are restricted to the form $x_i - x_j \bowtie c$. Both theories are decidable [5, Section 26.2.2.2].

4.2 From Max-Plus Algebra to Difference Logic

Before providing the main contribution, we show that the inequalities in max-plus algebra can be expressed as a formula in difference logic. For the rest of this paper, $\sim$ is either $\geq$ or $>$. We write $\neg(a \sim b)$ if it is not the case that $a \sim b$.

Proposition 5. Given $a_1, \ldots, a_p, a, b \in \mathbb{R}_{\max}$, real-valued variables $\mathtt{x}_1, \ldots, \mathtt{x}_p$, and $1 \leq j \leq p$, we have

$$\bigoplus_{i=1}^{p} (\mathtt{x}_i + a_i) \sim a \equiv \bigvee_{i=1}^{p} (\mathtt{x}_i + a_i \sim a), \tag{4}$$

$$a \sim \bigoplus_{i=1}^{p} (\mathtt{x}_i + a_i) \equiv \bigwedge_{i=1}^{p} (a \sim \mathtt{x}_i + a_i), \tag{5}$$

$$\bigoplus_{i=1}^{p} (\mathtt{x}_i + a_i) \sim \mathtt{x}_j + b \equiv \begin{cases} \mathtt{true}, & \text{if } (a_j \sim b), \quad (6) \\ \bigvee\limits_{\substack{i=1 \\ i \neq j}}^{p} (\mathtt{x}_i + a_i \sim \mathtt{x}_j + b), & \text{otherwise}, \quad (7) \end{cases}$$

$$\mathtt{x}_j + b \sim \bigoplus_{i=1}^{p} (\mathtt{x}_i + a_i) \equiv \begin{cases} \bigwedge\limits_{\substack{i=1 \\ i \neq j}}^{p} (\mathtt{x}_j + b \sim \mathtt{x}_i + a_i), & \text{if } (b \sim a_j), \quad (8) \\ \mathtt{false}, & \text{otherwise}. \quad (9) \end{cases}$$

Proposition 6 (Reduced Formula). Given real valued variables $\mathtt{x}_1, \ldots, \mathtt{x}_p$ and $a_1, \ldots, a_p, b_1, \ldots, b_p \in \mathbb{R}_{\max}$, the inequality

$$F \equiv \bigoplus_{i=1}^{p} (\mathtt{x}_i + a_i) \sim \bigoplus_{j=1}^{p} (\mathtt{x}_j + b_j) \tag{10}$$

is equivalent to

$$F^* \equiv \bigoplus_{i \in S_1} (\mathtt{x}_i + a_i) \sim \bigoplus_{j \in S_2} (\mathtt{x}_j + b_j), \tag{11}$$

where $S_1 = \{1, \ldots, p\} \setminus \{1 \leq k \leq p \mid a_k = \varepsilon \text{ or } \neg(a_k \sim b_k)\}$ and $S_2 = \{1, \ldots, p\} \setminus \{1 \leq k \leq p \mid b_k = \varepsilon \text{ or } a_k \sim b_k\}$, respectively. □

Proposition 6 ensures that any inequality expression in max-plus algebra can be reduced to a simpler one in which no a variable appears on both sides i.e., $S_1 \cap S_2 = \emptyset$. However, S_1 and S_2 cannot be both empty if there exists at least one finite scalar in both sides of (10). We call (11) as a non-trivial reduced formula if both $S_1 \neq \emptyset$ and $S_2 \neq \emptyset$.

Proposition 7. Given a non-trivial reduced formula in (11), then

$$F^* \equiv \bigwedge_{j \in S_2} \left(\bigvee_{i \in S_1} (\mathtt{x}_i - \mathtt{x}_j \sim b_j - a_i) \right) \equiv \bigvee_{i \in S_1} \left(\bigwedge_{j \in S_2} (\mathtt{x}_i - \mathtt{x}_j \sim b_j - a_i) \right). \tag{12}$$

If $S_1 = \emptyset$ then $F^* \equiv \mathtt{false}$. On the other hand, if $S_2 = \emptyset$ then $F^* \equiv \mathtt{true}$. □

Proposition 7 shows that any non-trivial formula of (11) can be expressed as a difference logic formula in disjunctive and conjunctive normal forms.

4.3 Procedure to Compute Transient of MPL Systems with SMT

We now will discuss the procedure to compute the transient of an MPL system via SMT-solving. The idea behind the SMT-based procedure is to transform the equality checking in line 11 of Algorithm 1 into a formula in difference logic. Notice that the quantity $\mathbf{M}[it]$ in Algorithm 1 corresponds to $A^{\otimes it} \otimes V$ next, and $\mathsf{cone}(V)$ can be expressed as matrix V. Thus, it can be equivalently written as

$$(A^{\otimes it} \otimes V) \otimes \mathbf{x} = (\lambda \times m) \otimes (A^{\otimes it-m} \otimes V) \otimes \mathbf{x}, \quad \forall \mathbf{x} \in \mathbb{R}^p, \tag{13}$$

where p is the number of columns of V. By denoting $R = A^{\otimes it} \otimes V$ and $S = (\lambda \times m) \otimes A^{\otimes it-m} \otimes V$, (13) can be expressed as

$$\bigwedge_{k=1}^{n} \left(\left(\bigoplus_{i=1}^{p} (\mathtt{x}_i + r_{ki}) \geq \bigoplus_{j=1}^{p} (\mathtt{x}_j + s_{kj}) \right) \wedge \left(\bigoplus_{i=1}^{p} (\mathtt{x}_i + s_{ki}) \geq \bigoplus_{j=1}^{p} (\mathtt{x}_j + r_{kj}) \right) \right), \tag{14}$$

where r_{ki} (resp. s_{ki}) is the element of R (resp. S) at row k and column i. For simplicity, we denote (14) as $\mathtt{EqFunc}(R, S)$. By Proposition 7, each disjunct in (14) can be expressed as a formula in difference logic.

Algorithm 2 summarizes the SMT-based version of Algorithm 1. If the corresponding eigenspace of the matrix is not empty, we set the value for transient and cyclicity respectively to $k_0 = 0$ and $c = 1$ (the smallest possible for both). Then, we generate the corresponding difference logic formula F w.r.t. (13) in line 10. To check the validity of F, we use an SMT solver to check the unsatisfiability of the negation. If it is not satisfiable then the original formula is valid, and then we obtain the transient and cyclicity from the current value of k_0 and c.

On the other hand, if it is satisfiable then there exists a counterexample falsifying formula F. We express the counterexample from a satisfying assignment of $\neg F$ as a real-valued vector $w \in \mathbb{R}^p$ (line 15). Vector $v = V \otimes w$ corresponds to the counterexample: its transient is greater than k_0 or its cyclicity is greater than c. The resulting transient and cyclicity of v become the updated value for (k_0, c). This process is repeated until either the SMT solver reports "unsatisfiable" in line 12 or $k_0 + c$ exceeds the maximum bound N. (which corresponds to the termination condition of Algorithm 1).

Unlike Algorithm 1, which only works on max-plus cones, Algorithm 2 can be modified (into Algorithm 3) so that it can be applied on any set of initial conditions $X \subseteq \mathbb{R}^n$. Although (14) is can be translated exclusively to RDL, we can extend X as an LRA formula. In line 9 of Algorithm 3, we generate a formula F which corresponds to the equality checking between $A^{\otimes k_0}$ and $A^{\otimes k_0+c}$. If $X \rightarrow F$ is valid then for all $\mathbf{x}(0) \in X$ we have $\mathsf{tr}(A, \mathbf{x}(0)) \leq k_0$ and $\mathsf{cyc}(A, \mathbf{x}(0)) \leq c$. Again, to check the validity of $X \rightarrow F$, we check the unsatisfiability of its negation.

Algorithm 2 Computation of transient and cyclicity of A w.r.t. $\mathsf{cone}(V)$ via SMT-solving

function TransConeSMT(A, V, N)
  $\chi \leftarrow$ CycleTimeVector(A)
  if elements of χ are all equal **then**
    $n \leftarrow$ NrRows(A)
    $p \leftarrow$ NrCols(V)
    for $i \in \{1 \cdots p\}$ **do**
      $x[i] \leftarrow$ MakeSMTRealVar( ) ▷ symbolic variables
    $k_0 \leftarrow 0, c \leftarrow 1$
    while $((k_0 + c) \leq N)$ **do**
      $F \leftarrow \mathtt{EqFunc}(A^{\otimes k_0 + c} \otimes V, (\lambda \times c) \otimes A^{\otimes k_0} \otimes V)$
      $model \leftarrow$ GetSMTModel($\neg F$)
      if $model = \bot$ **then** ▷ formula is unsatisfiable
        return $\langle k_0, c \rangle$
      else ▷ formula is satisfiable
        $w \leftarrow \langle model(x[1]), \cdots model(x[p]) \rangle$ ▷ vector in $\mathbb{R}^p$
        $v \leftarrow V \otimes w$ ▷ vector in $\mathbb{R}^n$
        $\langle k_0', c' \rangle \leftarrow$ TransCone(A, $A^{\otimes k_0} \otimes v$) ▷ computed by Algorithm 1
        $k_0 \leftarrow k_0 + k_0'$
        $c \leftarrow \mathtt{LCM}(c, c')$
    if $((k_0 + c) > N)$ **then**
      print "terminated after reaching maximum bound"
  else
    print "the transient does not exist"

Algorithm 3 Computation of transient and cyclicity of A w.r.t. a set of initial conditions X via SMT-solving

function TransSMT(A, X, N)
  $\chi \leftarrow$ CycleTimeVector(A)
  if elements of χ are all equal **then**
    $n \leftarrow$ Row(A) ▷ number of rows of A
    for $i \in \{1 \cdots n\}$ **do**
      $x[i] \leftarrow$ MakeSMTRealVar( ) ▷ symbolic variables
    $k_0 \leftarrow 0, c \leftarrow 1$
    while $(k_0 + c) \leq 1000$ **do**
      $F \leftarrow \mathtt{EqFunc}(A^{\otimes k_0 + c}, (\lambda \times c) \otimes A^{\otimes k_0})$
      $model \leftarrow$ GetSMTModel($X \wedge \neg F$)
      if $model = \bot$ **then** ▷ formula is unsatisfiable
        return $\langle k_0, c \rangle$
      else ▷ formula is satisfiable
        $v \leftarrow \langle model(x[1]), \cdots model(x[N]) \rangle$
        $\langle k_0', c' \rangle \leftarrow$ TransCone(A, $A^{\otimes k_0} \otimes v$)
        $k_0 \leftarrow k_0 + k_0'$
        $c \leftarrow \mathtt{LCM}(c, c')$
    if $((k_0 + c) > N)$ **then**
      print "terminated after reaching maximum bound"
  else
    print "the transient does not exist"

4.4 A Synthesis Problem

In addition to computing the transient and cyclicity of $A \in \mathbb{R}_{\max}^{n\times n}$ w.r.t. a set of initial conditions, we show that by means of difference logic and SMT, one can synthesise sets of states corresponding to specific transient (and cylicity) defined as follows

$$\mathcal{S}_{p,q}(A) = \{x \in \mathbb{R}^n \mid \mathsf{tr}(A,x) = p, \mathsf{cyc}(A,x) = q\}, \tag{15}$$
$$\mathcal{S}_{p}(A) = \{x \in \mathbb{R}^n \mid \mathsf{tr}(A,x) = p\}. \tag{16}$$

On the one hand, the computation of (16) has been discussed in [2, Section 4.2] by applying backward reachability analysis. On the other hand, to the best of the authors' knowledge, there is no approach to generate (15). The following proposition shows that both (15) and (16) can be computed symbolically by expressing them as difference logic formulae: the set (15) (resp. (16)) is not empty if and only if the corresponding formula (17) (resp. (18)) is satisfiable.

Proposition 8. Given $A \in \mathbb{R}_{\max}^{n\times n}$ with global cyclicity c and maximum eigenvalue λ, we have

$$\mathcal{S}_p(A) = \begin{cases} \mathsf{EqFunc}(A^{\otimes p+c}, \lambda c \otimes A^{\otimes p}), & \text{if } p = 0, \\ \mathsf{EqFunc}(A^{\otimes p+c}, \lambda c \otimes A^{\otimes p}) \wedge \\ \quad \neg\mathsf{EqFunc}(A^{\otimes p-1+c}, \lambda c \otimes A^{\otimes p-1}), & \text{if } p > 0, \end{cases} \tag{17}$$

and

$$\mathcal{S}_{p,q}(A) = \begin{cases} \mathsf{EqFunc}(A^{\otimes p+q}, \lambda q \otimes A^{\otimes p}) \wedge \\ \quad \bigwedge_{d \in \mathtt{Div}(q)-\{q\}} \neg\mathsf{EqFunc}(A^{\otimes p+d}, \lambda d \otimes A^{\otimes p}), & \text{if } p = 0, \\ \mathsf{EqFunc}(A^{\otimes p+q}, \lambda q \otimes A^{\otimes p}) \wedge \neg\mathsf{EqFunc}(A^{\otimes p-1+q}, \lambda q \otimes A^{\otimes p-1}) \wedge \\ \quad \bigwedge_{d \in \mathtt{Div}(q)-\{q\}} \neg\mathsf{EqFunc}(A^{\otimes p+d}, \lambda d \otimes A^{\otimes p}), & \text{if } p > 0, \end{cases} \tag{18}$$

where $\mathtt{Div}(q)$ is a set of divisors of q. □

As both (15) and (16) can be expressed as formulae in difference logic, the problem of determining the emptiness of both sets is decidable. By definition, for *never periodic* MPL system, $\mathcal{S}_{p,q} = \mathcal{S}_p = \emptyset$ for all p, q. Furthermore, for irreducible MPL systems the emptiness of (15) and (16) is related to the global transient and cyclicity of A.

Proposition 9. For an irreducible matrix $A \in \mathbb{R}_{\max}^{n\times n}$ with global transient k_0 and cyclicity c we have $\mathcal{S}_0(A) = E(A^{\otimes c})$ and $\mathcal{S}_{0,1}(A) = E(A)$. Furthermore,

i. $\mathcal{S}_p(A) \neq \emptyset$ iff $p \leq k_0$,
ii. If $p > k_0$ or q is not a divisor of c then $\mathcal{S}_{p,q}(A) = \emptyset$,

iii. If $\mathcal{S}_{p,q}(A)$ is empty then so is $\mathcal{S}_{p+1,q}(A)$. □

Example 3. Let us recall the 3×3 MPL system in Example 2 with $a = 8$. From the precedence graph $\mathcal{G}(B)$, the global cyclicity is $c = 2$. Leaving details aside, for $p \geq 1$, we have

$$\mathsf{EqFunc}(B^{p+2}, 18 \otimes B^{\otimes p}) \equiv \begin{cases} (\mathtt{x}_1 - \mathtt{x}_3 \geq 6 - p) \vee (\mathtt{x}_2 - \mathtt{x}_3 \geq 8 - p), \text{ if } p \text{ is odd,} \\ (\mathtt{x}_1 - \mathtt{x}_3 \geq 7 - p) \vee (\mathtt{x}_2 - \mathtt{x}_3 \geq 7 - p), \text{ if } p \text{ is even.} \end{cases}$$

Thus, by Proposition 8, for $p \geq 2$ we have

$$\mathcal{S}_p(B) = \begin{cases} \{\mathbf{x} \in \mathbb{R}^3 | (6 - p \leq x_1 - x_3 < 8 - p) \wedge (x_2 - x_3 < 8 - p)\}, \text{ if } p \text{ is odd,} \\ \{\mathbf{x} \in \mathbb{R}^3 | (x_1 - x_3 < 7 - p) \wedge (7 - p \leq x_2 - x_3 < 9 - p)\}, \text{ if } p \text{ is even.} \end{cases}$$

An illustration of the above sets is depicted in Figure 2. From an initial vector $\mathbf{x}(0) = [4\ 2\ 0]^\top \in \mathcal{S}_3(B)$ one can compute $\mathbf{x}(k)$ for $k = 1, \ldots, 5$ as follows:

$$\begin{bmatrix} 10 \\ 14 \\ 8 \end{bmatrix}, \begin{bmatrix} 22 \\ 20 \\ 16 \end{bmatrix}, \begin{bmatrix} 28 \\ 32 \\ 25 \end{bmatrix}, \begin{bmatrix} 40 \\ 38 \\ 33 \end{bmatrix}, \begin{bmatrix} 46 \\ 50 \\ 43 \end{bmatrix}.$$

Notice that, $\mathbf{x}(5) = 18 \otimes \mathbf{x}(3)$ which confirms that the local transient for $\mathbf{x}(0)$ is $\mathsf{tr}(B, \mathbf{x}(0)) = 3$. It is straightforward to conclude that the global transient for B does not exist. □

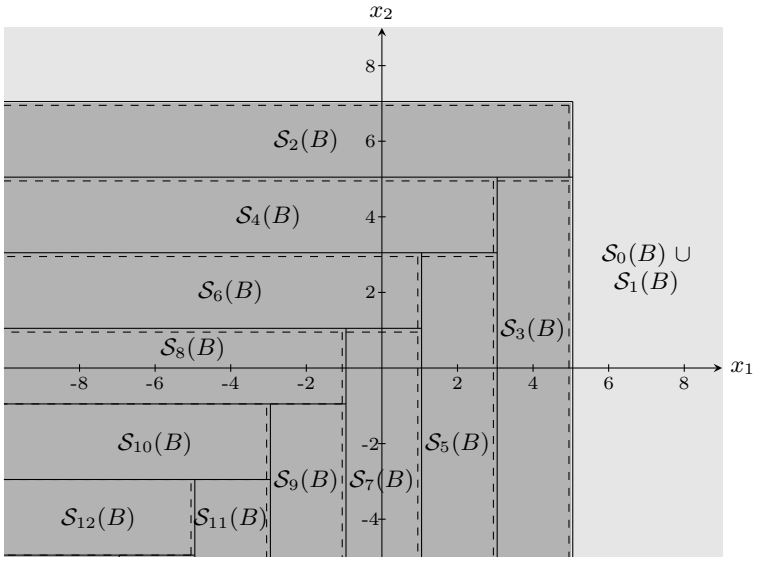

Fig. 2: Plots of the synthesized sets projected on the plane $x_3 = 0$. The solid and dashed lines represent $\geq$ and $>$, respectively.

5 Computational Benchmarks

We compare the performance of Algorithms 1 and 3, to compute the transient of MPL systems. The experiments for both procedures are implemented in Python. For the SMT solver, we use Yices 2.2 [11]. The computational benchmark has been implemented on an Intel® Xeon® CPU E5-1660 v3, 16 cores, 3.0GHz each, and 16GB of RAM. For the experiments, we generate 1000 irreducible matrices of dimension n, with m finite elements in each row, where the values of the finite elements are rational numbers $\frac{p}{q}$ with $1 \leq p \leq 100$ and $1 \leq q \leq 5$. The locations of the finite elements are chosen randomly. We focus on irreducible matrices to ensure the termination of the algorithms. Algorithm 1 is initialised by setting V to be a max-plus identity matrix, while for Algorithm 3 the set of initial conditions is expressed as $X \equiv \texttt{true}$. For all experiments, we choose $N = 10000$ as the maximum bound. The benchmarks are stored at `https://es-static.fbk.eu/people/amicheli/resources/formats20/`, where we have chosen $n \in \{4, 6, 8, 10, 20, 30, 40\}$ and three different values of m for each n.

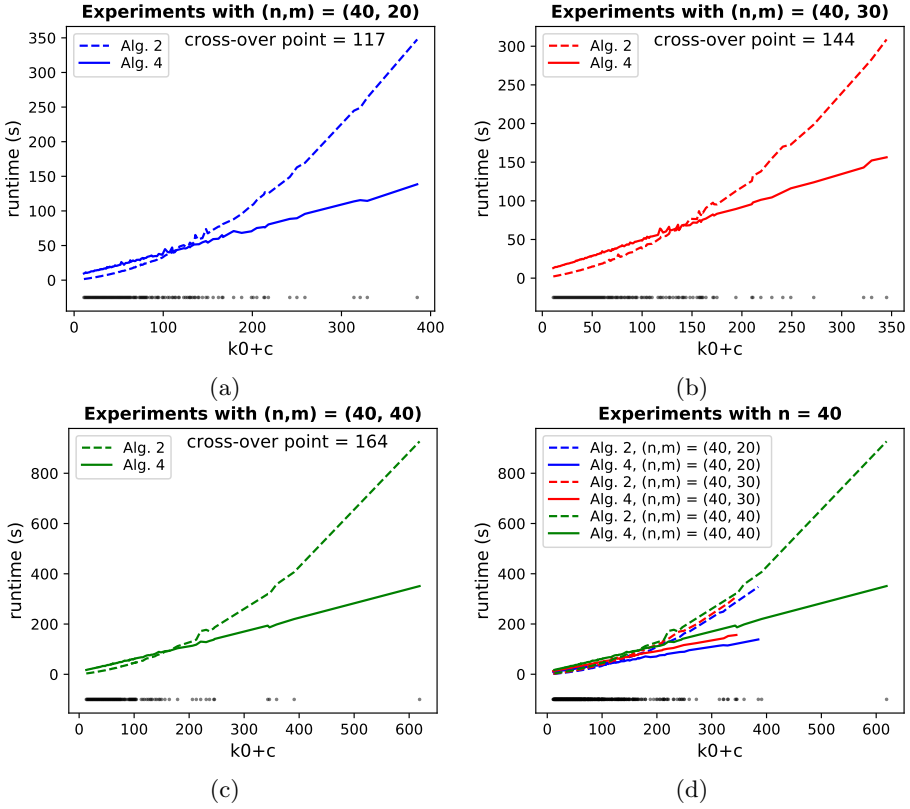

Fig. 3: The plots of running time of Algorithms 1 and 3 from 1000 experiments with $n = 40$ and $m \in \{20, 30, 40\}$. A "cross-over point" is the smallest value of $k_0 + c$ when Algorithm 3 is faster.

Figure 3(a)-(c) illustrate the experiments for $n = 40$ and $m \in \{20, 30, 40\}$ (the experiments for other pairs (n, m) are presented in [1]). They show the plots of the running times of Algorithm 1 (dashed lines) and of Algorithm 3 (solid lines) against the resulting transient k_0 and cyclicity c - the scattered plots (in black) correspond to the resulting $k_0 + c$. If there are several experiments with the same value of $k_0 + c$ then we display the average running time among those experiments. It is evident that most of the experiments result in small $k_0 + c$.

With regards to the running time, the matrix-multiplication algorithm is faster when the values of $k_0 + c$ are quite small. On the other hand, the larger the value of $k_0 + c$, the better the performance of the SMT-based algorithm is. We argue that this is because in Algorithm 3 there may be a large increment from the current guess of transient and cyclicity to the new ones. Whereas in Algorithm 1, the next candidate of transient and cyclicity is increased by one at each iteration.

As depicted in Figure 3(d), the number of finite elements m clearly affects the running time of the algorithms. We recall that the running time of Algorithm 3 depends on the satisfaction checking of a difference logic formula in line 11. The more are the finite elements, the more likely the formula is complex, and therefore the slower is the associated running time. Interestingly, based on the outcomes of the benchmarks which are presented in [1], the finite elements also affect the cross-over points, which tend to increase gradually as the number of finite elements grows larger.

6 Conclusions and Future Work

In this paper, we have introduced a novel, SMT-based approach to compute the transient of MPL systems: our technique encodes the problem as a sequence of satisfiability queries over formulae in difference logic, which can be solved by standard SMT solvers. We have also presented a procedure to partition the state-space of MPL systems w.r.t. a given transient and cyclicity pair. The procedure has been thoroughly tested on computational benchmarks and the results show how the SMT-based algorithm is much faster that state-of-the-art techniques to compute large values of transient and cyclicity. Furthermore, we highlight that the SMT-based method can be applied to compute the transient for any initial condition, as long as it is expressible as an LRA formula.

For future research, we are interested in exploring and developing SMT-based procedures for the general model checking of MPL systems.

References

1. Abate, A., Cimatti, A., Micheli, A., Mufid, M.S.: Computation of the Transient in Max-Plus Linear Systems via SMT-Solving. arXiv e-prints (Jul 2020), `https://arxiv.org/abs/2007.00505v2`
2. Adzkiya, D., De Schutter, B., Abate, A.: Backward reachability of autonomous max-plus-linear systems. IFAC Proceedings Volumes **47**(2), 117–122 (2014)

3. Alirezaei, M., van den Boom, T.J., Babuska, R.: Max-plus algebra for optimal scheduling of multiple sheets in a printer. In: Proc. 31st American Control Conference (ACC), 2012. pp. 1973–1978 (June 2012)
4. Baccelli, F., Cohen, G., Olsder, G.J., Quadrat, J.P.: Synchronization and linearity: an algebra for discrete event systems. John Wiley & Sons Ltd (1992)
5. Barrett, C.W., Sebastiani, R., Seshia, S.A., Tinelli, C.: Satisfiability modulo theories. In: Biere, A., Heule, M., van Maaren, H., Walsh, T. (eds.) Handbook of Satisfiability, Frontiers in Artificial Intelligence and Applications, vol. 185, pp. 825–885. IOS Press (2009). https://doi.org/10.3233/978-1-58603-929-5-825, `https://doi.org/10.3233/978-1-58603-929-5-825`
6. Brackley, C.A., Broomhead, D.S., Romano, M.C., Thiel, M.: A max-plus model of ribosome dynamics during mRNA translation. Journal of Theoretical Biology **303**, 128–140 (2012)
7. Butkovič, P., Schneider, H., et al.: Generators, extremals and bases of max cones. Linear algebra and its applications **421**(2-3), 394–406 (2007)
8. Charron-Bost, B., Függer, M., Nowak, T.: New transience bounds for max-plus linear systems. Discrete Applied Mathematics **219**, 83–99 (2017)
9. Clarke, E., Kroening, D., Ouaknine, J., Strichman, O.: Completeness and complexity of bounded model checking. In: International Workshop on Verification, Model Checking, and Abstract Interpretation. pp. 85–96. Springer (2004)
10. Comet, J.P.: Application of max-plus algebra to biological sequence comparisons. Theoretical computer science **293**(1), 189–217 (2003)
11. Dutertre, B.: Yices 2.2. In: Intl. Conf. on Computer Aided Verification *(CAV'14)*. LNCS, vol. 8559, pp. 737–744 (2014)
12. Fahim, K., van der Woude, J., et al.: On a generalization of power algorithms over max-plus algebra. Discrete Event Dynamic Systems **27**(1), 181–203 (2017)
13. Gaubert, S., Katz, R.D.: Minimal half-spaces and external representation of tropical polyhedra. Journal of Algebraic Combinatorics **33**(3), 325–348 (2011)
14. Heidergott, B., Olsder, G.J., Van der Woude, J.: Max Plus at work: modeling and analysis of synchronized systems: a course on Max-Plus algebra and its applications. Princeton University Press (2014)
15. Imaev, A., Judd, R.P.: Hierarchial modeling of manufacturing systems using max-plus algebra. In: Proc. American Control Conference, 2008. pp. 471–476 (June 2008)
16. Merlet, G., Nowak, T., Sergeev, S.: Weak csr expansions and transience bounds in max-plus algebra. Linear Algebra and its Applications **461**, 163–199 (2014)
17. Mufid, M.S., Adzkiya, D., Abate, A.: Symbolic reachability analysis of high dimensional max-plus linear systems. arXiv e-prints (Jul 2020), `https://arxiv.org/abs/2007.04510`, accepted at the International Workshop on Discrete Event Systems (WODES)
18. Mufid, M.S., Adzkiya, D., Abate, A.: Bounded model checking of max-plus linear systems via predicate abstractions. In: International Conference on Formal Modeling and Analysis of Timed Systems. pp. 142–159. Springer (2019)
19. Nowak, T., Charron-Bost, B.: An overview of transience bounds in max-plus algebra. Tropical and Idempotent Mathematics and Applications **616**, 277–289 (2014)
20. Soto Y Koelemeijer, G.: On the behaviour of classes of min-max-plus systems. Ph.D. thesis, Delft University of Technology (2003)