Computations on Zones using Max-Plus Algebra

Jesper Dyhrberg, Qi Lu, Michael Madsen, and Søren Ravn

Aalborg University, Department of Computer Science
Selma Lagerlöfs Vej 300
9220 Aalborg East, Denmark
{jdyhrb08,qlu09,msma09,sravn06}@student.aau.dk
Group room: 3.1.12
Supervisor: Ulrich Fahrenberg
Send peer review comments to: d406a@cs.aau.dk

Abstract. We describe how to use max-plus algebra for model checking. We provide conversions from zone constraints to max-plus algebra. We give efficient algorithms for the basic operations required–delay, reset, intersection–as well as union, all of which run in $O(n)$ time. Finally, we compare the capabilities of max-plus algebra to those of difference bound matrices, the approach currently used by tools such as UPPAAL.

1 Introduction

Model checking is the process of taking a model of some system–expressed using some formalism, such as timed automata–and verifying that the system has some desirable property. Several tools, such as UPPAAL [1], can be used to perform model checking.

In UPPAAL, model checking is reduced to a reachability problem: we must be able to get to a state in our model where our desired property holds. However, this is not easy to quantify, as our state is dependent on one or more clocks: real numbers that represent time passing. This means there are infinitely many states to consider, and it is not possible to perform model checking if we have to do this. By using integer constraints and dividing our set of states into zones, however, the problem becomes decidable, and we can perform the model checking.

The current industry standard, used by tools such as UPPAAL, is to use *difference bound matrices*–DBMs for short–to represent the current set of states as a zone. However, this approach has some limitations; in particular, zones have to be convex, which becomes a problem when there are multiple DBMs which can bring us to the current state - the union of two DBMs is required, but this will not give us a DBM. This problem can be worked around by first checking an overapproximation of the intended zone, and, if necessary, performing the detailed analysis with each of the

possible DBMs, but this brings on the risk of state-space explosion: if there are many possible ways to get to our desired state, then we will need to look at every one of them. This is not ideal, as the number of paths can grow exponentially: if two paths lead into the same state, we get twice as many possible paths to analyze.

Consequently, it is desirable to find a different approach which will be less likely to require this detailed analysis of each possible path–in other words, a smaller oveapproximation, leading to fewer false positives. One such approach is to represent the zones using *max-plus algebra*, an approach which has already been shown capable of providing large performance boosts for some problems in static analysis. [2,3]

Contributions. We show how max-plus algebra can be used to express the constraints of our model. We show how the basic operations required to perform model checking can be applied using our approach, and provide evidence that our approach is better at handling unions than DBMs.

Related Work. Timed automata were introduced by Dill and Alur [4,5]; for surveys on the topic we recommend [6,7,8].

Max-plus algebra dates back more than 50 years, under various different names. In 1997, Gaubert et al. provided a survey of the topic, which includes a section on its history [9].

2 Max-plus algebra

Max-plus algebra is the semiring $\mathbb{R}_{max}$ defined as $\mathbb{R} \cup \{-\infty\}$ with two operations: addition $x \oplus y := \max(x, y)$ and multiplication $x \otimes y := x + y$, with multiplication taking precedence over addition. Like in regular algebra, we may use the multiplication operator implicitly; $5x$ is equivalent to $5 \otimes x$ in max-plus algebra. Addition is both associative and commutative, while multiplication is both associative and distributive over $\oplus$, and commutative for individual numbers–but not in the general case, e.g. multiplication of matrices.

$-\infty$ is the zero element, in that $-\infty \oplus x = x \oplus -\infty = x$. For this reason, we also write this as $\mathbb{0}$. 0 is the unit element, also written as $\mathbb{1}$, in that $0 \otimes x = x \otimes 0 = x$. Finally, $\mathbb{0}$ is absorbing for multiplication, since $-\infty \otimes x = x \otimes -\infty = -\infty$.

2.1 Expressing constraints using max-plus algebra

Zones. A zone is an efficient representation for timing information in symbolic state space exploration for timed systems [7]. A set of atomic

constraints form a zone over a set of clocks C. The collection of atomic constraints are of the form $x \sim n$ and $x - y \sim n$, where $x, y \in C$, and $\sim \in \{\leq, <, =, >, \geq\}$ and $n \in \mathbb{N}$. However, we only cover the non-strict inequalities $\leq$ and $\geq$, as handling $<$ and $>$ can be done simply by maintaining separate information on the strictness.

Max-plus Polyhedra. The notation $\mathbf{x}$ will be used to represent a vector of the form $\begin{bmatrix} x_1 \; x_2 \cdots x_n \end{bmatrix}^T$, where ^{T} is the matrix transposition operator.

Allamigeon et al. describe that a max-plus polyhedron can be expressed as a system of constraints of the form $A\mathbf{x} \oplus \mathbf{b} = C\mathbf{x} \oplus \mathbf{d}$, or, since inequalities and equalities have the same expressive power in max-plus algebra, $A\mathbf{x} \oplus \mathbf{b} \leq C\mathbf{x} \oplus \mathbf{d}$ where $A, C \in \mathbb{R}_{max}^{s \times n}$, $\mathbf{b}, \mathbf{d} \in \mathbb{R}_{max}^{s}$ and $\mathbf{x} \in \mathbb{R}_{max}^{n}$, s is the number of constraints in the system and n is the number of clocks [10].

We can also think of each individual constraint as being expressed in such a form, in which case $A, C \in \mathbb{R}_{max}^{1 \times n}$ and $\mathbf{b}, \mathbf{d} \in \mathbb{R}_{max}$.

Representing Zones using Max-Plus Polyhedra. In order to use max-plus polyhedra to represent zones, we must be able to convert zone constraints to a max-plus polyhedra. We have derived the values to be used for A, $\mathbf{b}$, C, and $\mathbf{d}$ in systems of constraints in order to represent any regular atomic constraints expressible by zones.

Because the equality $x - y = n$ can be converted into two inequalities, $x - y \leq n$ and $x - y \geq n$, we can use these two inequalities for representing this equality. The values are seen in Table 1, where $n \in \mathbb{N}$.

Atomic constraints	$A_{i,x}$	$A_{i,y}$	$A_{i,z}$	b_i	$C_{i,x}$	$C_{i,y}$	$C_{i,z}$	d_i
$x \leq n$	0	-∞	-∞	-∞	-∞	-∞	-∞	n
$x \geq n$	-∞	-∞	-∞	n	0	-∞	-∞	-∞
$x - y \leq n$	0	-∞	-∞	-∞	-∞	n	-∞	-∞
$x - y \geq n$	-∞	n	-∞	-∞	0	-∞	-∞	-∞

Table 1: Atomic constraints and values for max-plus matrix inequalities

Each vector $A_i, C_i \in \mathbb{R}_{max}^{1 \times n}$ is one of the two matrices (A, C) respectively, where i is the index of the rows, i.e. the constraint index. Table 1 shows all values to be used for $A_{i,k}$ and $C_{i,k}$, where k is the clock index. Index z represents all clocks not affected by the constraint.

To prove that these values hold, we will show that the solution for each of the four max-plus equations correspond to the atomic constraints exactly. Finally we will show that inserting the value -∞ in A and C for all clocks not affected by the constraint will make the constraints in the table valid for any arbitrary number of clocks.

We start by proving that our mapping holds for the atomic constraint $x \leq n$ when there is only one clock. By rewriting the constraint, we get the following:

$$x \leq n \Leftrightarrow 0\, x \oplus \text{-}\infty \leq \text{-}\infty\, x \oplus n \Leftrightarrow \max(0 + x, -\infty) \leq \max(-\infty + x, n)$$

showing it exactly encodes the intended constraint. Similarly, we can rewrite $x - y \leq n$ to prove that this constraint holds for two clocks:

$$\begin{aligned} x - y \leq n \Leftrightarrow x \leq n + y \Leftrightarrow 0\, x \oplus \text{-}\infty\, y \oplus \text{-}\infty \leq \text{-}\infty\, x \oplus n \otimes y \oplus \text{-}\infty \Leftrightarrow \\ \max(0 + x, \text{-}\infty + y, \text{-}\infty) \leq \max(\text{-}\infty + x, n + y, \text{-}\infty) \end{aligned}$$

By solving the inequalities for $x \geq n$ and $x - y \geq n$ in the same way as above, it is seen that the values encode exactly the constraints in Table 1.

Extending the constraints to more clocks is simple, due to the way max-plus matrix multiplication is calculated:

$$\begin{bmatrix} a_{1,1} & \cdots & a_{1,n} \\ a_{2,1} & \cdots & a_{2,n} \\ \vdots & \ddots & \vdots \\ a_{m,1} & \cdots & a_{m,n} \end{bmatrix} \otimes \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} a_{1,1} \otimes x_1 \oplus \cdots \oplus a_{1,n} \otimes x_n \\ a_{2,1} \otimes x_1 \oplus \cdots \oplus a_{2,n} \otimes x_n \\ \vdots \\ a_{m,1} \otimes x_1 \oplus \cdots \oplus a_{m,n} \otimes x_n \end{bmatrix} \tag{1}$$

Since max-plus addition is commutative and -∞ is the $\mathbb{0}$ element, Equation 1 can be used to see that using -∞ for all values associated with a clock will prevent the clock from having any influence on the final result, and that using -∞ for a clock in just one constraint will prevent it from influencing that particular constraint.

3 Max-plus polyhedra and Extreme points

According to Allamigeon et al., a max-plus polyhedron can be represented in two different ways. One way is by a set of constraints, as mentioned in Section 2.1, while the other way is by the Minkowski sum of a max-plus convex combination and a max-plus linear combination. The convex combination $\mathbf{co}(\mathbf{v})$ and linear combination $\mathbf{cone}(\mathbf{w})$ are two sets of vectors, referred to as generators [10].

Throughout the rest of the paper, we will refer to the representation using constraints as external, and the representation using generators as internal.

A vector $\mathbf{v}$ is a *convex combination* of vectors $\mathbf{v}_1, \cdots, \mathbf{v}_p \in \mathbb{R}^n_{max}$ if $\mathbf{v} = \alpha_1\mathbf{v}_1 \oplus \cdots \oplus \alpha_p\mathbf{v}_p$ for some scalars $\alpha_1, \cdots, \alpha_p \in \mathbb{R}_{max}$, satisfying $\bigoplus_{i=1}^{p} \alpha_i = \mathbb{1}$. A vector $\mathbf{w}$ is a *linear combination* of vectors $\mathbf{w}_1, \cdots, \mathbf{w}_q \in \mathbb{R}^n_{max}$ if $\mathbf{w} = \beta_1\mathbf{w}_1 \oplus \cdots \oplus \beta_q\mathbf{w}_q$. Unlike the convex combination, there are no restrictions on the scalars for the linear combination.

We will see later that in some sense, the linear combinations are used for representing infinite bounds, while the convex combinations represent actual finite bounds.

The representation of a max-plus polyhedron by a max-plus convex and linear sets is not unique, but there exists a unique minimal representation of any given max-plus polyhedron. The vectors in this minimal set are called the extreme points.

Figure 1 depicts a bounded max-plus polyhedron P in $\mathbb{R}^2_{max}$. Determining whether a vector is an extreme point will be discussed in Section 4.1, however we can prove that the points a, b, and c can be represented as max-plus linear combinations of the points e_1, e_2, e_3, and e_4. Indeed, it can be shown that the minimal max-plus convex set of P, i.e. the set of extreme points, consists of e_1, e_2, e_3, and e_4.

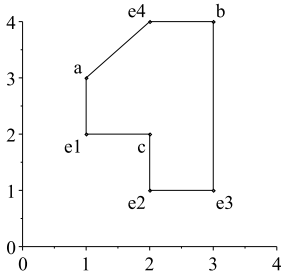


Fig. 1: A max-plus polyhedron P in $\mathbb{R}^2_{max}$

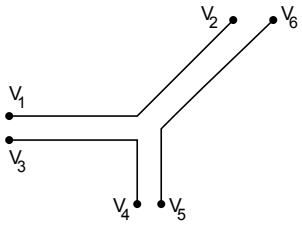


Fig. 2: The three kinds of generic max-plus segments in $\mathbb{R}^2_{max}$

Figure 2 shows the types of generic segments that can be derived between each pair of extreme points with a convex combination in $\mathbb{R}^2_{max}$, see also Allamigeon et al. [11]. Looking at Figure 1, we see that the boundary of the polyhedron is created by exactly these three kinds of generic max-plus segments between the extreme points.

4 Operations

For doing forward reachability analysis of timed automata using zones, three different operations are needed. These basic operations are intersection ($\cap$), delay ($\uparrow$), and reset ($_{R[x_i=val]}$) [6]. We will describe how each of these operations can be applied on max-plus polyhedra. Additionally, we will explain how to calculate the union ($\cup$) of two polyhedra, and discuss how our approach relates to the overapproximation that is done when using DBMs.

Conversions Between Constraints and Extreme Points It is possible to convert between the external and internal representations. The constraints of zones are expressed by using a max-plus inequality $A\mathbf{x} \oplus \mathbf{b} \leq C\mathbf{x} \oplus \mathbf{d}$, which can then be converted to the homogeneous system $(A\mathbf{b})\mathbf{z} \leq (C\mathbf{d})\mathbf{z}$ of $\mathbb{R}_{max}^{n+1}$ by letting the values of the first n elements of $\mathbf{z}$ are the same as $\mathbf{x}$, and setting the value of the last element of $\mathbf{z}$ to $\mathbb{1}$. Afterwards, we compute the extreme points. An efficient way of doing this is the ComputeExtreme algorithm as described by Allamigeon et al. [3]

After applying our operations on the extreme points, we can convert the new extreme points back to a max-plus equation. The detailed process of the conversion from internal to external representation is given by Allamigeon et al. [10]

It is worth noting that there is a fairly high computational complexity associated with the conversions, especially with the conversion from internal to external representation.

Intersection As mentioned previously, the max-plus equations encode one constraint in each row. Consequently, the intersection between two equations can be performed simply by concatenating the rows as mentioned in the work of Allamigeon et al. [10]. This can be done in $O(n)$ by simply taking one constraint at a time, and adding it to the resulting equation–each row can be added like this in constant time, and there are n rows to add. This algorithm can add redundant constaints, so it may be desirable to convert the constraints to the internal representation and perform cleanup as discussed in Section 4.1.

Delay Delay is the operation of letting time pass. This, in effect, is the same as adding $c \in \mathbb{R}_+$ to all clocks, thereby changing the current state of the system.

In the internal representation, one algorithm for delaying is to delay each of the extreme points. This is done by copying all α scaled points

and adding these copies to the linear combination with a new scalar β_i. As mentioned in Section 3 unlike the α's there is no restriction on these β scalars. This can be done in $O(n)$, as copying a single point can be done in constant time and there are n points to copy.

Since the β scalars have no restriction, it is easy to see that using positive scalars will extend the points upward diagonally. The β scalars can also be negative, but our α scalars will then prevent extending the lines downward as $\bigoplus_{i=1}^{n} \alpha_1 = \mathbb{1}$ must still be satisfied, thereby making sure the minimal values from the convex set is also the minimal values of the resulting, delayed set.

Example 1. Let us consider the polyhedron P, in Figure 3a, expressed by the extreme points (1,1), (3,1) and (1,3). Delaying P would result in:

$$P\uparrow = \alpha_1 \begin{bmatrix}1\\1\end{bmatrix} \oplus \alpha_2 \begin{bmatrix}3\\1\end{bmatrix} \oplus \alpha_3 \begin{bmatrix}1\\3\end{bmatrix} \oplus \beta_1 \begin{bmatrix}1\\1\end{bmatrix} \oplus \beta_2 \begin{bmatrix}3\\1\end{bmatrix} \oplus \beta_3 \begin{bmatrix}1\\3\end{bmatrix}$$

For a visual view of $P\uparrow$, see Figure 3b

Reset The reset operation is performed by setting the value for a specific clock to some specific value - the reset value. This results in a projection of the polyhedron to a line on the axis that represents the clock or is parallel to the axis.

The internal representation of polyhedron is very useful for the reset operation. For example, assume we have a polyhedron Z, expressed by the extreme points (1,1), (3,1) and (1,3). Resetting the x_1 clock to zero would be written as $Z_{R[x_1=0]}$ and result in the extreme points (0,1), (0,1) and (0,3).

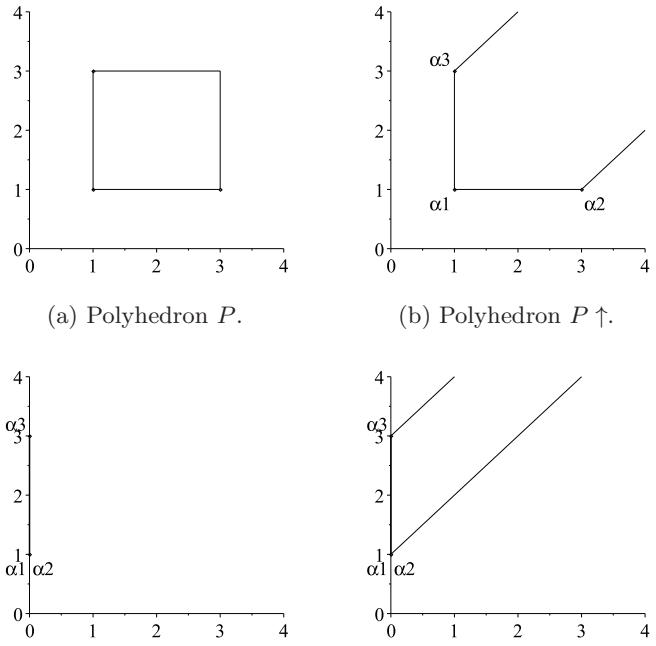
However, this approach will not work when resetting infinite polyhedra. If we reset the affected coordinate in all points–including the delayed points–in this way, it will lead to a wrong result. A correct approach is to reset all extreme points scaled by α as described above, while for all β scaled points, the value should be -∞.

This works because setting the respective coordinate of the points scaled by α scalars to the new value effectively projects all points onto the respective axis, and then offsetting these points if the new value is not 0. The points with β scalars have the respective coordinate removed from the equation, in effect, since $\beta \otimes$ -∞ = -∞ for any $\beta \in \mathbb{R}_{max}$, ensuring that they will not allow us to obtain a different value than the value we have enforced by the reset operation.

Example 2. Recall the polyhedron $P \uparrow$ from Example 1, also shown in Figure 3b. Using the proposed reset algorithm gives us exactly the polyhedron we expect from the definition of the reset operation. $P \uparrow_{R[x_1=0]}$ is shown in Figure 3d.

$$P \uparrow_{R[x_1=0]} = \alpha_1 \begin{bmatrix} 0 \\ 1 \end{bmatrix} \oplus \alpha_2 \begin{bmatrix} 0 \\ 1 \end{bmatrix} \oplus \alpha_3 \begin{bmatrix} 0 \\ 3 \end{bmatrix} \oplus \beta_1 \begin{bmatrix} -\infty \\ 1 \end{bmatrix} \oplus \beta_2 \begin{bmatrix} -\infty \\ 1 \end{bmatrix} \oplus \beta_3 \begin{bmatrix} -\infty \\ 3 \end{bmatrix}$$

Although correct, this linear combination does contain some redundant points. Removing the redundant points will be discussed in Section 4.1.



(a) Polyhedron P.

(b) Polyhedron $P \uparrow$.

(c) Polyhedron $P_{R[x_1=0]}$.

(d) Polyhedron $P_{R[x_1=0]} \uparrow$.

Fig. 3: Examples of the operations

4.1 Cleanup

As all operations ($\cap$, $\uparrow$, $_{R[x_k=v]}$, $\cup$) may introduce redundant information, a cleanup algorithm is needed to keep the memory usage down. Additionally, the algorithm described by Allamigeon et al. for conversion from

internal to external representation has a high computational complexity– $O(p \times n \times c(2n+2,p)^4)$, where p is the number of points, n is the number of clocks and $c(2n-2,p)$ is the maximal number of generators of the set of solutions of a system of p constraints in $\mathbb{R}^{2n+2}_{max}$. [10]. This high complexity means that keeping the set of points to a minimum is useful.

The way to clean up any redundant point is to check whether a given point can be represented as a linear combination of the other points, where the remaining α scalars still satisfy $\bigoplus_{i=1}^{n} \alpha_i = \mathbb{1}$. Any point which is such a linear combination can be removed without losing any information. The resulting set of extreme points is unique for any polyhedron, and can, as shown by the work of Butkovič et al., be computed from an arbitrary set p of generators in $O(n \times p^2)$ operations [12].

When cleaning up points for a polyhedron with infinite bounds, i.e. points with β scalars, the algorithm must be modified slightly: a point with a β scalar can only be removed if it can be expressed as a linear combination of the remaining points with β scalars.

Example 3. Let us consider the polyhedron $P\uparrow$ from Example 1.

$$P\uparrow = \alpha_1 \begin{bmatrix}1\\1\end{bmatrix} \oplus \alpha_2 \begin{bmatrix}3\\1\end{bmatrix} \oplus \alpha_3 \begin{bmatrix}1\\3\end{bmatrix} \oplus \beta_1 \begin{bmatrix}1\\1\end{bmatrix} \oplus \beta_2 \begin{bmatrix}3\\1\end{bmatrix} \oplus \beta_3 \begin{bmatrix}1\\3\end{bmatrix}$$

This representation for $P\uparrow$ contains some redundant information, as some points can be expressed as linear combinations of other points. By choosing the correct scalars, we can eliminate certain points. For example, the point (3,1) can be removed by choosing the scalars $\alpha_1 = 0$, $\alpha_2 = \text{-}\infty$, $\beta_1 = \text{-}\infty$, $\beta_2 = 0$, $\beta_3 = \text{-}\infty$, since:

$$\begin{bmatrix}3\\1\end{bmatrix} = 0 \begin{bmatrix}1\\1\end{bmatrix} \oplus \text{-}\infty \begin{bmatrix}1\\3\end{bmatrix} \oplus \text{-}\infty \begin{bmatrix}1\\1\end{bmatrix} \oplus 0 \begin{bmatrix}3\\1\end{bmatrix} \oplus \text{-}\infty \begin{bmatrix}1\\3\end{bmatrix}$$

Points with scalars α_3 and β_1 can be removed in the same way, reducing the representation of the zone to the minimal representation.

4.2 Union

Similar to delay and reset, union is most easily done with the internal representation.

One algorithm for taking the union of two polyhedra is to take the union of the extreme points for both of them, which will give us an over-approximating polyhedron containing both of the original polyhedra–in fact, it will give us the smallest overapproximating polyhedron, as proved

in Theorem 1. This can be done in $O(n)$, as copying each of the n points to the new set of points can be done in constant time.

Theorem 1 (Union of points gives smallest overapproximation). *Given two max-plus polyhedra, P_1 and P_2 with extreme points $\{v_1, \ldots, v_n\}$ and $\{w_1, \ldots, w_m\}$, the polyhedron P generated by the extreme points $\{v_1, \ldots, v_n\} \cup \{w_1, \ldots, w_m\}$ is the smallest overapproximation of $P_1 \cup P_2$.*

Proof. It is trivially seen that all points used to generate P are part of either P_1 or P_2; no new points are added, so we only have the points we started with.

We can simplify our set of points by removing redundant points. Once redundant points have been removed, we are left with the points which define the set of segments which are still visible in the unioned polyhedron. It is no longer possible to remove any more points without affecting the shape of the polyhedron, as the shape is determined by the linear combinations of all possible pairs of points. We also cannot change any of the points, since that would affect the linear combinations, creating a different shape. Since this gives us the minimal set of points required to represent the shape in question, all of which are part of the original polyhedra, this means we have created the smallest overapproximating polyhedron of the union of P_1 and P_2. □

Corollary 1. *If the combined area covered by P_1 and P_2 can be expressed exactly by a max-plus polyhedron, our algorithm will generate exactly that polyhedron.*

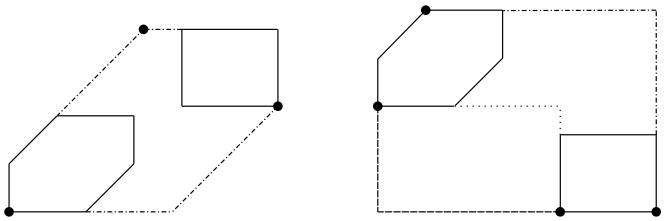
Proof. This directly follows from the fact that we generate the smallest overapproximation–if the algorithm didn't generate that polyhedron, then it wouldn't be the smallest overapproximation, as the exact polyhedron would be smaller and still contain all of P_1 and P_2. □

Corollary 2. *The max-plus polyhedron representing the smallest overapproximation of the union of two max-plus polyhedra will never have more false positives–points included in the overapproximation, but not in the two original polyhedra–than the DBM representing the smallest overapproximation of the union of two equivalent DBMs.*

Proof. Given that any DBM can be expressed as a max-plus polyhedron, it follows that the overapproximating DBM can also be expressed as a max-plus polyhedron. If the overapproximating DBM were smaller than the overapproximating max-plus polyhedron, then the max-plus polyhedron could not be the smallest overapproximation of the two polyhedra:

the polyhedron expressing the DBM would be a smaller overapproximation. Thus, we are guaranteed to never create more false positives than the DBM approach. □

This only proves that we won't do any worse than the DBM approach; it doesn't guarantee that we will ever do better. However, it can easily be shown with an example that there are indeed scenarios where max-plus polyhedra will generate a smaller overapproximation than the DBM. Fig. 4a shows an example where the two approaches generate the same overapproximation, while Fig. 4b shows an example where the max-plus approach generates a smaller overapproximation. The dashed lines show the area covered by the overapproximating DBM, while the dotted lines show the area covered by the overapproximating max-plus polyhedron. Lines which are both dotted and dashed show lines which are used for both overapproximations. The extreme points of the overapproximating polyhedron are also shown in the figures.



(a) Same approximation.

(b) Different approximations.

Fig. 4: Examples of overapproximating max-plus polyhedra and DBMs

5 Conclusion and future work

As we have demonstrated, max-plus algebra is able to provide fewer false positives for overapproximations when compared to DBMs, due to the smaller overapproximation provided by the higher expressiveness of max-plus polyhedra. This, in turn, leads to more efficient model checking. We have also provided efficient algorithms for the three basic operations, as well as union, all running in $O(n)$. Unfortunately, these algorithms currently rely on converting between the internal and external representations as necessary, and these conversions are very computationally expensive.

Additionally, we have provided a starting point for generating the max-plus polyhedra through the conversion from constraints to a max-plus matrix equation.

Since the conversions prevent us from showing the computational feasibility of this approach, this would be an obvious problem to tackle in future work. One approach for doing this would be to construct an algorithm for intersection which works directly on the internal representation, since this would allow all of the operations to be performed on that representation, eliminating the need to convert back and forth.

It would also be highly relevant to create a prototype implementation and compare the performance with that of DBMs. Max-plus polyhedra have been shown capable of providing a large performance gain in other fields [2], so it is plausible that similar results are achievable here.

References

1. Uppsala University, Aalborg University: UPPAAL. `http://www.uppaal.com/`
2. Allamigeon, X., Gaubert, S., Goubault, É.: The tropical double description method. In Marion, J.Y., Schwentick, T., eds.: STACS. Volume 5 of LIPIcs., Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2010) 47–58
3. Allamigeon, X., Gaubert, S., Goubault, É.: Computing the extreme points of tropical polyhedra (2009)
4. Alur, R., Dill, D.L.: Automata for modeling real-time systems. In Paterson, M., ed.: ICALP. Volume 443 of Lecture Notes in Computer Science., Springer (1990) 322–335
5. Alur, R., Dill, D.L.: The theory of timed automata. In de Bakker, J.W., Huizing, C., de Roever, W.P., Rozenberg, G., eds.: REX Workshop. Volume 600 of Lecture Notes in Computer Science., Springer (1991) 45–73
6. Aceto, L., Ingólfsdóttir, A., Larsen, K.G., Srba, J.: Reactive Systems. Cambridge University Press (2007)
7. Bengtsson, J.: Clocks, DBMs, and States in Timed Systems. PhD thesis, Uppsala University (June 2002)
8. Fahrenberg, U., Larsen, K.G., Thrane, C.R.: Verification, performance analysis and controller synthesis for real-time systems. In Arbab, F., Sirjani, M., eds.: FSEN. Volume 5961 of Lecture Notes in Computer Science., Springer (2009) 34–61
9. Gaubert, S., Plus, M.: Methods and applications of (max, +) linear algebra. In Reischuk, R., Morvan, M., eds.: STACS. Volume 1200 of Lecture Notes in Computer Science., Springer (1997) 261–282
10. Allamigeon, X., Gaubert, S., Goubault, É.: Inferring min and max invariants using max-plus polyhedra. In Alpuente, M., Vidal, G., eds.: SAS. Volume 5079 of Lecture Notes in Computer Science., Springer (2008) 189–204
11. Gaubert, S., Katz, R.D.: The minkowski theorem for max-plus convex sets. Linear Algebra and its Applications **421**(2-3) (2007) 356 – 369
12. Butkovič, P., Schneider, H., Sergeev, S.: Generators, extremals and bases of max cones. Linear Algebra and its Applications **421**(2-3) (2007) 394 – 406