Behaviour Equivalent Max-Plus Automata for a Class of Timed Petri Nets ⋆

Lukas Triska * **Thomas Moor** *

* *Lehrstuhl für Regelungstechnik, Friedrich-Alexander Universität-Erlangen-Nürnberg, Erlangen, Germany (e-mail: {lukas.triska, thomas.moor}@fau.de)*

Abstract: Timed Petri nets and max-plus automata are well known modelling frameworks for timed discrete-event systems. In this paper we present an iterative procedure that constructs a max-plus automaton from a timed Petri net while retaining the timed behaviour. Regarding the Petri net, we essentially impose three assumptions. First, the reachability graph must be finite; i.e., the Petri net must be bounded. Second, the Petri net operates according to the so called race policy. And third, we interpret the Petri net with single server semantics. Under these assumptions we show that the proposed procedure terminates with a finite deterministic max-plus automaton that realises the same timed behaviour as the Petri net. We demonstrate by example how our procedure can be applied in the context of supervisor controller design for timed discrete-event systems.

Keywords: Petri nets, max-plus automata, timed discrete-event systems, supervisory control.

1. INTRODUCTION

Max-plus automata are defined as a generalisation of plain automata by assigning minimum durations to individual transitions, and, hence, can be used to model the behaviour of timed discrete-event systems; see Gaubert (1995). Although max-plus automata are not as expressive as general timed automata introduced by Alur and Dill (1994), they can be conveniently analysed within an algebraic setting, e.g., considering power series with coefficients from the idempotent semi ring over the reals with addition and maximum as the two binary operations. Max-plus automata must not be confused with linear max-plus dynamic systems and variations thereof; see e.g. Baccelli et al. (1992) or Hardouin et al. (2018).

Max-plus automata have also been utilised in the context of supervisory control. Here, a given max-plus automaton represents the plant behaviour and one seeks to synthesise a supervisory controller such that the closed-loop behaviour satisfies a prescribed specification; e.g. (Komenda et al., 2009; Su et al., 2012). As with un-timed supervisory control introduced by Ramadge and Wonham (1989), the basic case of complete observation corresponds to a deterministic plant automaton. However, it is well known that not all max-plus automata are determinisable. Algorithms that enable determinisation of max-plus automata under restrictive conditions have been presented in Gaubert (1995) and Mohri (1997), where weighted automata are applied to speech recognition and their determinisation is possible if the so-called twin property holds. The classical algorithm for determinisation based on normalisation of the state vector has been extended in Kirsten (2008) for polynomially ambiguous max-plus automata, where a more general clone property guarantees the determinisation. Nevertheless, for non polynomially ambiguous max-plus automata even the decidability of determinisation is still open. Weighted automata have also been used in image processing (Culik and Kari, 1997).

⋆ This work was supported by the German Research Foundation (DFG) under grant MO 1697/4-1.

Similar to max-plus automata, timed Petri nets are introduced by assigning durations to individual transitions in a Petri net; see Ramchandani (1973). There are a number alternative firing rules commonly applied to plain Petri nets, and this variety is inherited by timed Petri nets. For example, Gaubert and Mairesse (1999) and Lahaye et al. (2015) consider so called *safe timed Petri nets under preselection policy* and show how they can be converted to behaviour equivalent max-plus automata. Although this is a relevant result for the purpose of analysis, the obtained automata in general fail to be deterministic. In contrast, Komenda et al. (2016) consider so called *bounded Petri nets under race policy* and provide a semi-algorithm that in the case of termination generates behaviour equivalent deterministic max-plus automata. To this end, Komenda et al. (2016, p. 427) impose a fairness condition on the Petri net and show that this condition is sufficient to imply termination of the semi-algorithm.

The present paper follows the line of thought of Komenda et al. (2016) and derives a similar algorithm. However, we do propose a number of variations that allow to drop the fairness requirement. In particular, our algorithm terminates for all bounded Petri nets under race policy with rational timing parameters, and thereby generalises the earlier results.

The paper is organised as follows. In Section 2 we denote necessary notations. The following section contains fundamentals about max-plus automata and timed Petri nets. Furthermore, we derive a formal representation for the behaviour of timed Petri nets. Section 4 is devoted to the construction of behaviour equivalent max-plus automata and it is shown that the behaviour of any timed Petri net satisfying the stated requirements can be realised by a finite, deterministic max-plus automaton. An example for application of this procedure to supervisor design is given in Section 5.

2. NOTATION

The positive integers are denoted $\mathbb{N}$ and we let $\mathbb{N}_0 := \mathbb{N} \dot\cup \{0\}$. The rationales are denoted $\mathbb{Q}$ and the reals $\mathbb{R}$. The non-negative rationales are denoted $\mathbb{Q}_{\geq 0}$ and the non-negative reals $\mathbb{R}_{\geq 0}$. For a neutral element ε regarding the max operation, we also consider $\mathbb{Q}_{\max} := \mathbb{Q} \dot\cup \{-\infty\}$ and $\mathbb{R}_{\max} := \mathbb{R} \dot\cup \{-\infty\}$ with $\varepsilon := -\infty$ and the convention that $x + \varepsilon = \varepsilon$ for all $x \in \mathbb{R}_{\max}$.

An *alphabet* A is a finite set of symbols. We denote A^* the Kleene-closure of A, i.e., the set of finite-length words composed from symbols in A, including the empty word $\lambda \in A^*$, $\lambda \notin A$. Subsets of A^* are referred to as *languages over* A.

Throughout this paper we identify a map $f : X \to Y$ with the associated vector $g = (g_x)_{x \in X} \in Y^X$, $g_x = f(x) \in Y$ for all $x \in X$, i.e., we do not distinguish between f and g and use either notation whenever convenient.

3. MAX-PLUS AUTOMATA AND TIMED PETRI NETS

In this section we first recall fundamental definitions about max-plus automata and timed Petri nets. Notation conventions are kept in line with Komenda et al. (2016). A more extensive introduction to this topic is given by Gaubert (1995); Seatzu et al. (2012). Subsequently we illustrate how a representation of the semantical state of a timed Petri net can be obtained that can be transferred directly to a state automaton.

3.1 Deterministic max-plus automata

Max-plus automata are introduced as a generalisation of plain automata with durations assigned to transitions. We regard the timing component in $\mathbb{R}_{\max}$ with the binary operations max and $+$, which entail the respective neutral elements $\varepsilon := -\infty$ and $e := 0$. [1]

Definition 1. A *max-plus automaton* is defined as a quadruple $G = (Q, A, Q_0, \delta)$, where

- Q is the set of states,
- A is the alphabet of event symbols,
- Q_0 is the set of initial states, and
- $\delta : Q \times A \times Q \to \mathbb{R}_{max}$ is the transition function.

The max-plus automaton G is finite if Q is a finite set. □

The transition function in a max-plus automaton associates with each transition a non-negative duration and thereby generalises the common transition relation of plain automata. Technically, we require that for q, $q' \in Q$ and $a \in A$ either

$$\delta(q, a, q') = d \geq 0 \tag{1}$$

to indicate that the respective transition takes no more than d time units, or that

$$\delta(q, a, q') = \varepsilon \tag{2}$$

to indicate that the respective transition can not take place at all.

A *path* or *run* in the max-plus automaton G is defined as a sequence

$$\pi = q_0 a_1 q_1 a_2 q_2 \cdots a_n q_n \tag{3}$$

such that $q_0 \in Q_0$ and $q_i \in Q$, $a_i \in A$, $\delta(q_i, a_{i+1}, q_{i+1}) \neq \varepsilon$ for all $i \in \mathbb{N}_0$, $i < n$. With the run π we associate the word $w = a_1 a_2 \cdots a_n$. A word $w \in A^*$ is *recognised* by G if there exists at least one run π with associated word w. Note that the empty word λ is recognised by any max-plus automaton via the trivial run $\pi = q_0$. The *logical behaviour* $\mathrm{L}(G) \subseteq A^*$ of G is then defined as the set of all words recognised by G.

A max-plus automaton is *deterministic* if it has exactly one initial state and if, given a state and an event symbol, there can be at most one successor state. Throughout this paper, we only consider deterministic automata. Technically, we then have that $Q_0 = \{q_0\}$ and that for all $q, q', p' \in Q$ and for all $a \in A$

$$\delta(q, a, q') \in \mathbb{R}_{\geq 0} \text{ and } \delta(q, a, p') \in \mathbb{R}_{\geq 0} \quad \Rightarrow \quad q' = p' \,. \tag{4}$$

In particular, we have that for each $w \in \mathrm{L}(G)$ there exists exactly one run of G with w the associated word.

The *timed behaviour* of max-plus automata is defined via a *dater function* that returns the date at which a sequence of events has definitely been executed. For the situation of deterministic max-plus automata, the technical construction simplifies considerably.

Definition 2. The *behaviour* $y_G : A^* \to \mathbb{R}_{\max}$ of a deterministic max-plus automaton G is given by

$$y_G(w) := \delta(q_0, a_1, q_1) + \delta(q_1, a_2, q_2) + \cdots + \delta(q_{n-1}, a_n, q_n) \,, \tag{5}$$

where $\pi = q_0 a_1 q_1 a_2 q_2 \cdots a_n q_n$ is the unique run with associated word $w \in \mathrm{L}(G)$, including the special case of $w = \lambda$ with the empty sum, i.e., $y_G(\lambda) = 0$. For $w \notin \mathrm{L}(G)$, we let $y_G(w) = \varepsilon$. □

Referring to the transition durations as *weights*, the timed behaviour amounts to the sum of the transition weights along the unique run associated with each individual recognised word. Taking this perspective, we define a weighted transition relation by

$$q \xrightarrow{a/d} q' \tag{6}$$

for q, $q' \in Q$ and $a \in A$ if and only if $d = \delta(q, a, q') \geq 0$. This transition relation is commonly extended to words in A^*. Technically, we begin with the empty word $\lambda \in \mathcal{T}^*$ and define the transitions

$$q \xrightarrow{\lambda/0} q \,, \tag{7}$$

for all $q \in Q$. We then iteratively define further transitions

$$q \xrightarrow{wa/d} q'' \tag{8}$$

with $q, q'' \in Q$, $w \in A^*$ and $a \in A$ if and only if

$$q \xrightarrow{w/d'} q' \quad \text{and} \quad q' \xrightarrow{a/d''} q'' \tag{9}$$

are both defined and $d = d' + d''$. Note that, given $q \in Q$ and $w \in A^*$, determinism of the underlying max-plus automaton implies that there exists at most one duration $d \in \mathbb{R}_{\geq 0}$ and one successor state $q' \in Q$ such that

$$q \xrightarrow{w/d} q' \,. \tag{10}$$

Considering the initial state $q = q_0$ and a word $w \in A^*$, it is evident from the construction, that there exist d and q' that qualify for Eq. (10) if $y_G(w) = d \geq 0$, and vice versa.

3.2 Petri nets

A Petri net is a bipartite graph with places and transitions as nodes. Places can hold any number of tokens and the token configuration determines which transitions are enabled. We recall the formal definition and comment on aspects relevant for the present paper.

Definition 3. A *Petri net* is a quadruple $\mathcal{G} = (\mathcal{P}, \mathcal{T}, \mathcal{F}, M_0)$, where

[1] Our main technical results are restricted to timing constraints expressible in terms of rational deadlines from $\mathbb{Q}_{\max}$. However, we formally refer to the more general case of $\mathbb{R}_{\max}$ when citing basic definitions from the literature.

- $\mathcal{P}$ is a finite set of *places*,
- $\mathcal{T}$ is a finite set of *transitions*,
- $\mathcal{F} \subseteq (\mathcal{P} \times \mathcal{T}) \cup (\mathcal{T} \times \mathcal{P})$ is the *incidence relation*, and
- $M_0 : \mathcal{P} \to \mathbb{N}_0$ is the *initial marking*. □

The configuration of a Petri net is given by a *marking* $M : \mathcal{P} \to \mathbb{N}_0$, which specifies the number of tokens present at each place. Whenever convenient, we identify the function M with the corresponding vector in $\mathbb{N}_0^{\mathcal{P}}$. The evolution of the marking over logic time adheres to the following rules:

(S1) Transition $t \in \mathcal{T}$ is *enabled* by a marking M if each input place of t has at least one token; i.e., if $M(p) > 0$ for all $p \in \mathcal{P}$ with $(p,t) \in \mathcal{F}$. This is denoted $M \xrightarrow{t}$.

(S2) An enabled transition $t \in \mathcal{T}$ can *fire*. The firing of t transforms the marking M into M', where one token is removed from each input place of t and one additional token is generated for each output places; i.e., $M'(p) := M(p) - 1$ for all $p \in \mathcal{P}$ with $(p,t) \in \mathcal{F}$ and $M'(p) := M(p) + 1$ for all $p \in \mathcal{P}$ with $(t,p) \in \mathcal{F}$. This is denoted $M \xrightarrow{t} M'$.

Conforming with the above rules, a *firing sequence* is specified by markings $M_i : \mathcal{P} \to \mathbb{N}_0$ and transitions $t_i \in \mathcal{T}$ such that $M_{i-1} \xrightarrow{t_i} M_i$ for $i = 1, \dots, n \in \mathbb{N}$, and we associate the word $w := t_1 t_2 \cdots t_n \in \mathcal{T}^*$ with this firing sequence. The *logical behaviour* $\mathrm{L}(\mathcal{G}) \subseteq \mathcal{T}^*$ of the Petri net $\mathcal{G}$ is then defined as the set of all words associated with some *firing sequence*.

A Petri net is called *bounded* if for all markings reachable by some firing sequence the token count at each place does not exceed a uniform bound.

Definition 4. The *reachability graph* or *marking graph* of a bounded Petri net $\mathcal{G}$ is the deterministic finite automaton $\mathit{Reach}(\mathcal{G}) = (\mathcal{M}, M_0, \mathcal{T}, t_r)$, where

- the state set $\mathcal{M}$ is the set of markings reachable by some firing sequence,
- the alphabet is the set of transitions $\mathcal{T}$,
- the initial state is the initial marking M_0, and
- the partial transition function $t_r : \mathcal{M} \times \mathcal{T} \to \mathcal{M}$ is defined for $M \in \mathcal{M}$ and $t \in \mathcal{T}$ by $t_r(M,t) := M'$ if and only if $M \xrightarrow{t}$ and where $M' \in \mathcal{M}$ is the unique marking with $M \xrightarrow{t} M'$. □

Note that the determinism of the reachability graph crucially depends on the direct use to the set of transitions as alphabet. When applying the same approach in the presence of explicit transition labels, such a labeling needs to be injective for us to obtain a deterministic reachability graph. Since an explicit labeling is quite common in the literature, we refer to our setting as *injectively labeled*.

3.3 Timed Petri nets

We consider a class of timed Petri nets that is obtained from plain Petri nets by associating with each individual transition a duration. This style of generalisation is similar to when moving from plain automata to max-plus automata.

Definition 5. A *timed Petri net* is a pair $(\mathcal{G}, \tau)$, where $\mathcal{G}$ is a Petri net with set of transitions $\mathcal{T}$ and where $\tau = (\tau_a)_{a \in \mathcal{T}} \in \mathbb{R}_{\geq 0}^{\mathcal{T}}$ is a parameter vector representing the durations associated with each individual transition. □

In contrast to the setting with max-plus automata, the duration τ_a here is interpreted as the firing time of some transition $a \in \mathcal{T}$. Since holding times are not considered throughout this paper, the duration τ_a correlates to the minimum delay between enabling and firing of a. This interpretation leads to the following informal extension of firing rules.

(S3) The tokens belonging to the initial marking become available at time instant zero.

(S4) All transitions are considered to be *single server*, meaning that a transition can only process one token from each input place at a time.

(S5) If multiple transitions are enabled, the one that can fire the earliest has priority. This rule is also known as *race policy*. In case several transition have exactly the same remaining delay multiple options are viable.

(S6) Transitions are fired as soon as possible, which corresponds to the *earliest functioning firing rule*.

At this point we are looking to find a formal representation of the timed behaviour. Our approach here is to extend the discrete state set

$$\mathcal{M} \subseteq \mathbb{N}_0^{\mathcal{P}} \tag{11}$$

from the reachability graph by a continuous component $\mathcal{C}$ to strategically encode clock values in order to address the timing rules (S3)–(S6). Technically, we let

$$\mathcal{C} := (\mathbb{R}_{\geq 0} \cup \{\ddagger\})^{\mathcal{T}} \tag{12}$$

to maintain one clock per transition that shows the time for which the transition has been continuously enabled or, alternatively, the distinguished symbol ‡ to explicitly indicate that the respective transition is disabled and, hence, the clock is inactive. While the initial marking M_0 is specified by the Petri net, we define the initial clock vector as $C_0 := (c_{0,t})_{t \in \mathcal{T}} \in \mathcal{C}$ with

$$c_{0,t} := \begin{cases} 0 & \text{if } M_0 \xrightarrow{t}, \text{and} \\ \ddagger & \text{else}. \end{cases} \tag{13}$$

Thus, the overall set of *semantic states* amounts to the product $\mathcal{M} \times \mathcal{C}$ with initial state $(M_0, C_0) \in \mathcal{M} \times \mathcal{C}$. We define a number of operations on these states that turn out useful in the subsequent discussion.

- As a means to update the values of all clocks after the elapse of some finite amount of time we define the operation *Inc* for a clock vector $C = (c_t)_{t \in \mathcal{T}} \in \mathcal{C}$ and a duration $d \in \mathbb{R}_{\geq 0}$, that is $\mathit{Inc}(C,d) \in \mathcal{C}$ with

$$\mathit{Inc}(C,d)_t := \begin{cases} c_t + d & \text{if } c_t \in \mathbb{R}_{\geq 0}, \textit{and} \\ \ddagger & \text{else}, \end{cases} \tag{14}$$

  for $t \in \mathcal{T}$.
- To reset the value of specific clocks we define the operation *Reset* for a set of transitions $S \subseteq \mathcal{T}$ and a clock vector $C = (c_t)_{t \in \mathcal{T}} \in \mathcal{C}$. The clocks corresponding to transitions in S are reset and other clocks are not effected; that is $\mathit{Reset}(C,S) \in \mathcal{C}$ with

$$\mathit{Reset}(C,S)_t := \begin{cases} 0 & \text{if } t \in S, \text{and} \\ c_t & \text{else}, \end{cases} \tag{15}$$

  for $t \in \mathcal{T}$. The single server transition semantics are then implemented by resetting the respective clock value after every firing of a transition.
- Since the timing is only relevant for enabled transitions, we deactivate a clock referring to disabled transitions using the indicator symbol ‡. This operation is performed by *Sub*,

defined for $S \subseteq \mathcal{T}$ and $C = (c_t)_{t\in\mathcal{T}} \in \mathcal{C}$ with $Sub(C,S) \in \mathcal{C}$ and

$$Sub(C,S)_t := \begin{cases} c_t, & \text{if } t \in S, \text{and} \\ \ddagger & \text{else}. \end{cases} \tag{16}$$

for $t \in \mathcal{T}$.

- In order to adequately reset and start relevant clocks, we need to identify *newly enabled* transitions. Given a marking $M \in \mathcal{M}$ with $M \xrightarrow{t}$ for some transition $t \in \mathcal{T}$ and the unique successor marking $M' \in \mathcal{M}$, a transition $t' \in \mathcal{T}$ is *obviously newly enabled* if it is enabled in M' but not in M. In this case, the corresponding entry in the clock vector shall be set to 0.

  However, we also need to account for the situation where the elimination of tokens as required by firing t temporally disables a transition t' which otherwise is enabled by both markings M and M'. Technically, we define the *intermediate marking* $M^+ \in \mathbb{N}_0^{\mathcal{P}}$

$$M^+(p) := \begin{cases} M(p) - 1 & \text{if } (p,t) \in \mathcal{F}, \text{and} \\ M(p) & \text{else}, \end{cases} \tag{17}$$

  and consider a transition $t' \in \mathcal{T}$ to be newly enabled if and only if it is enabled in M' but not in M^+. Referring to the parameters M, $M' \in \mathcal{M}$ and $t \in \mathcal{T}$, the set of all newly enabled transitions is denoted

$$NewEn(M,t,M') \subseteq \mathcal{T}. \tag{18}$$

  Note that systematically resetting clocks of newly enabled transitions ensures that clocks of enabled transitions are always active, and hence, show a real value as opposed to the distinguished symbol ‡.

  We want to emphasise that the removal of one token in Eq. (17) is sufficient due to the restriction on single server transition semantics. If multiple tokens are allowed to fire simultaneously, determining which transitions are newly enabled becomes more involved.

- The race policy guarantees that among enabled transitions only the one(s) with the minimal remaining firing delay can be fired. With the set of transitions enabled by a marking M denoted

$$En(M) := \{\, t \in \mathcal{T} \mid M \xrightarrow{t} \,\} \tag{19}$$

  we define $FirstFired(M,C) \subseteq En(M)$ by

$$FirstFired(M,C) = \{\, a \in En(M) \mid \forall\, b \in En(M) \,:\, \tau_a - c_a \le \tau_b - c_b \}. \tag{20}$$

  In this regard the expression $d = \tau_a - c_a$ for $a \in FirstFired(M,C)$ represents the minimal remaining firing delay among transitions enabled by the marking M. Hence, before the elapse of d time units, no transition can fire and after the elapse of d time units some transition $a \in FirstFired(M,C)$ will fire provided that $En(M) \neq \emptyset$.

We are now in the position to formally define the overall timed Petri nets semantics by introducing weighted transitions

$$(M,C) \xrightarrow{a/d} (M',C'), \tag{21}$$

between two states with M, $M' \in \mathcal{M}$, C, $C' \in \mathcal{C}$, $a \in \mathcal{T}$ and $d \in \mathbb{R}_{\ge 0}$ if and only if

(i) $M \xrightarrow{a} M'$,
(ii) $a \in FirstFired(M,C)$,
(iii) $d = \tau_a - c_a$,
(iv) $C^+ = Inc(C,d)$,
(v) $C^{++} = Reset(C^+, NewEn(M,a,M') \cup \{a\})$,
(vi) $C' = Sub(C^{++}, En(M'))$.

Note that the above transition relation is deterministic by construction in the sense that

$$(M,C) \xrightarrow{a/d'} (M',C') \quad \text{and} \quad (M,C) \xrightarrow{a/d''} (M'',C'') \tag{22}$$

imply $d'' = d'$, $M'' = M'$ and $C'' = C'$. In particular, the transitions can be interpreted as state transitions in a deterministic max-plus automata with state set $\mathcal{M} \times \mathcal{C}$ and we will follow this up in the subsequent section. To this end, we refer to the common extension of weighted transition relations to words as presented at the end of Section 3.1 for a formal definition of the timed behaviour of the Petri net. Technically, we begin with the empty word $\lambda \in \mathcal{T}^*$ and define the transitions

$$(M,C) \xrightarrow{\lambda/0} (M,C) \tag{23}$$

for all $M \in \mathcal{M}$, $C \in \mathcal{C}$. Referring to Eq. (21), we then iteratively introduce further transitions

$$(M,C) \xrightarrow{wa/d} (M'',C'') \tag{24}$$

with $w \in \mathcal{T}^*$, $a \in \mathcal{T}$ and $d \in \mathbb{R}_{\ge 0}$ whenever there exist $M' \in \mathcal{M}$, $C' \in \mathcal{C}$ and d', $d'' \in \mathbb{R}_{\ge 0}$ such that

$$(M,C) \xrightarrow{w/d'} (M',C') \quad \text{and} \quad (M',C') \xrightarrow{a/d''} (M'',C''). \tag{25}$$

and $d = d' + d''$. Note that the determinism as observed above carries over to the extension to words. In particular, given a word $w \in \mathcal{T}^*$ there exists at most one matching sequence of transitions beginning at the initial state (M_0, C_0) and, if so, a unique corresponding overall duration d.

Definition 6. The *behaviour* of a timed Petri net $(\mathcal{G},\tau)$ is defined as the dater function $y_{\mathcal{G}} : \mathcal{T}^* \to \mathbb{R}_{max}$ with $y_{\mathcal{G}}(w) = d$ if

$$(M_0,C_0) \xrightarrow{w/d} (M',C') \tag{26}$$

for some $M' \in \mathcal{M}$ and $C' \in \mathcal{C}$, and referring to the transition relation defined by Eqs. (21)–(25), or, else, $y_{\mathcal{G}}(w) = \varepsilon$. □

4. FINITE STATE REPRESENTATION

Based on the semantics defined in the previous section, we are now looking to obtain a max-plus automaton with equal behavior as a bounded, timed Petri net operating under race policy with single server semantics. For our main result, Theorem 8, we show that a suitable automaton with a finite number of states always exists and we demonstrate how to obtain it.

4.1 Behaviour Considerations

As a first step in constructing the desired max-plus automaton we propose an initial candidate with an infinite state set that realizes the same behavior as the respective timed Petri net. This is done be re-interpreting the transition relation on the semantic states, Section 3.3, Eq. (21), as the transition function of a max-plus automaton with state set $Q = \mathcal{M} \times \mathcal{C}$, i.e., the set of all pairs of markings and clock vectors with regard to a given bounded timed Petri net $(\mathcal{G},\tau)$, $\mathcal{G} = (\mathcal{P},\mathcal{T},M_0,\mathcal{F})$. Given a state $q = (M,C) \in Q$, we ask for all possible successor states under the restriction of the race policy and with single server semantics. Referring to determinism of the transition relation on semantic states, Eq. (21), recall that given $q = (M,C) \in Q$ and $t \in \mathcal{T}$ there exist at most one duration $d \in \mathbb{R}_{\ge 0}$ and one successor state $q' = (M',C') \in Q$ such that

$$q \xrightarrow{t/d} q'. \tag{27}$$

Hence, we can define the the max-plus transition function $\delta : Q \times \mathcal{T} \times Q \to \mathbb{R}_{\max}$ by

$$\delta(q,t,q') = \begin{cases} d, & \text{if } q \xrightarrow{t/d} q' \text{ for some } d \in \mathbb{R}_{\geq 0} \text{ and} \\ \varepsilon, & \text{else,} \end{cases} \tag{28}$$

and consider the deterministic max-plus automaton

$$G = (Q, \mathcal{T}, \{(M_0, C_0)\}, \delta)\,. \tag{29}$$

It is evident from the construction that the weighted transition relation, Eq. (28), associated with G matches the transition relation on the semantic states defined in Section 3.3, Eq. (21). Hence, we have that

$$\forall\, w \in \mathcal{T}^*\,.\, y_{\mathcal{G}}(w) = y_G(w)\,. \tag{30}$$

In other words, the timed behaviour of the max-plus automaton G equals the behaviour of the timed Petri net $(\mathcal{G}, \tau)$.

4.2 Restriction to a Finite Automaton

Although the state set $Q = \mathcal{M}\times\mathcal{C}$ of G is technically infinite due to the $\mathcal{C}$-component, our conjecture is that the set of reachable states is only finite. In support of a formal argument, we conduct a forward-reachability analysis on G. Consider the operator

$$\begin{aligned} &\mathit{NextState}(P) := \\ &\quad \{\, q' \in Q \,|\, \exists\, t \in \mathcal{T},\, q \in P\,.\, \delta(q,t,q') \geq 0 \,\} \end{aligned} \tag{31}$$

defined for sets of states $P \subseteq Q$. Then the set of reachable states in G is obtained by the following iteration

$$Q_0 := \{(M_0, C_0)\}\,, \tag{32}$$

$$Q_{i+1} := Q_i \cup \mathit{NextState}(Q_i)\,, \tag{33}$$

$$Q_* := \cup\{\, Q_i \,|\, i \in \mathbb{N}_0 \,\}\,, \tag{34}$$

i.e., there exists a path π in G that ends in a state $q \in Q$ if and only if $q \in Q_*$. Since non-reachable states do not contribute to the behaviour, we can restrict G to the state set Q_*. Technically, we consider the max-plus automaton

$$G_* = (Q_*, \mathcal{T}, \{(M_0, C_0)\}, \delta_*)\,, \tag{35}$$

where the transition function $\delta_* : Q_* \times \mathcal{T} \times Q_* \to \mathbb{R}_{\max}$ equals δ on the restricted domain, i.e.,

$$\forall\,(q,t,q') \in Q_* \times \mathcal{T} \times Q_*\,.\; \delta_*(q,t,q') = \delta(q,t,q')\,. \tag{36}$$

and we conclude that $y_{\mathcal{G}}(w) = y_G(w) = y_{G_*}(w)$ for all $w \in \mathcal{T}^*$.

From the boundedness assumption of the Petri net $\mathcal{G}$ it follows that the set of reachable markings $\mathcal{M}$ in $\mathit{Reach}(\mathcal{G})$ is finite. In order to establish that Q_* is finite, we show that the range of the $\mathcal{C}$-component over all states in Q_* is finite, too. To this end, consider the following Lemma.

Lemma 7. The entries of the clock vector C in every state $(M, C) \in Q_*$ of the automaton G_* are bounded by the respective entry in the vector of transition durations $\tau \in \mathbb{R}^{\mathcal{T}}_{\geq 0}$, i.e.

$$\forall\,(M,C) \in Q_*\,, t \in \mathcal{T}\,.\, c_t \neq \ddagger \;\Rightarrow\; c_t \leq \tau_t\,. \tag{37}$$

Proof. For the case of $Q_* = \{(M_0, C_0)\}$ the claim is trivially true since each entry $c_{0,t}$ of C_0 by definition either equals $\ddagger$ or $0 \leq \tau_t$, with the latter inequality as a consequence of τ_t being non negative.

For a proof by contradiction, suppose there exists a state $q' = (M', C') \in Q_*$ different from the initial state, and such that $c'_t > \tau_t$ for some transition $t \in \mathcal{T}$. Since q' is not the initial state, there exists $i \in \mathbb{N}_0$ such that $q' \notin Q_i$ but $q' \in Q_{i+1}$. Hence, by Eq. (33), we have that $q' \in \mathit{NextState}(Q_i)$. By the definition of *NextState*, Eq. (31), we can choose a predecessor state $q = (M, C) \in Q_i \subseteq Q_*$ such that $\delta(q, a, q') = d \geq 0$ for some $a \in \mathcal{T}$. With Eq. (28) this implies

$$(M, C) \xrightarrow{a/d} (M', C')\,, \tag{38}$$

and we can refer to conditions (i)–(vi) below Eq. (21) to derive further consequences. In order for c'_t to be greater than zero, transition t has to be enabled for marking M' as well as the predecessor marking M, otherwise transition t would be considered *obviously newly enabled* or *disabled* and the clock value c'_t is reset; see conditions (v) and (vi), respectively. In particular, we have $t \in \mathit{En}(M)$. Referring to condition (i), we have $a \in \mathit{En}(M)$ and, since transition a was chosen according to the race policy semantics, condition (ii), we also have $a \in \mathit{FirstFired}(M, C)$, i.e.,

$$d = \tau_a - c_a \leq \tau_b - c_b\,, \tag{39}$$

for all $b \in \mathit{En}(M)$. From the definition of clock values via conditions (iv)–(vi), we obtain $c_t + d = c'_t$ to conclude

$$c_t + d = c'_t \tag{40}$$

$$\Leftrightarrow \quad c_t + \tau_a - c_a = c'_t > \tau_t \tag{41}$$

$$\Leftrightarrow \quad \tau_a - c_a > \tau_t - c_t\,. \tag{42}$$

As $t \in \mathit{En}(M)$ holds true, Eq. (42) constitutes a contradiction with Eq. (39). □

Regarding the above lemma we consider the restricted range of clock values

$$\mathcal{C}_* := ([0, \tau_{\max}] \cup \{\ddagger\})^{\mathcal{T}} \subseteq \mathcal{C}\,, \tag{43}$$

with $\tau_{\max} = \max_{t\in\mathcal{T}} \tau_t$ to observe that

$$Q_* \subseteq \mathcal{M} \times \mathcal{C}_*\,. \tag{44}$$

If all entries in the timing vector τ are non-negative integers, i.e., $\tau \in \mathbb{N}_0^{\mathcal{T}}$, then the clock vector C of any state $q = (M, C) \in Q_*$ is also in $\mathbb{N}_0^{\mathcal{T}}$. Since the intersection of $\mathcal{C}_*$ with $\mathbb{N}_0^{\mathcal{T}}$ is a finite set, this implies that Q_* is a finite set, too. For the case of a rational timing $\tau \in \mathbb{Q}^{\mathcal{T}}_{\geq 0}$, we uniformly scale clocks to refer to the least common denominator of all entries in τ to again obtain a finite set Q_* by the same argument. Note that this observation does not carry over to general real-valued timings $\tau \in \mathbb{R}^{\mathcal{T}}_{\geq 0}$. We now state our main result.

Theorem 8. For every bounded, injectively labeled, timed Petri net $(\mathcal{G}, \tau)$ with rational timing vector $\tau \in \mathbb{Q}^{\mathcal{T}}_{\geq 0}$ there exists a finite deterministic max-plus automaton with equal behaviour. One such an automaton is given by

$$G_* = (Q_*, \mathcal{T}, \{(M_0, C_0)\}, \delta_*)\,, \tag{45}$$

as defined in (35) via the iteration (32)–(34). In particular, the iteration attains a fixpoint after finitely many steps, i.e., we have $Q_* = Q_i$ for some $i \in \mathbb{N}_0$.

Proof. Finiteness of Q_* is a consequence of Lemma 7 and the discussion for rational timings thereafter. Behavioural equivalence has been discussed in Section 4.1 concluding with Eq. (30). Attaining a fixpoint $Q_* = Q_i$ for some $i \in \mathbb{N}_0$ is a consequence of finiteness of Q_* and monotonicity $Q_i \subseteq Q_{i+1} \subseteq Q_*$ in the iteration (32)–(34). □

4.3 Algorithm and Example

With the intent to have the necessary design steps be clearly accessible we present an equivalent representation in the form of an algorithm.

Algorithm: Taking as an input a bounded, injectively labeled, timed Petri net $(\mathcal{G}, \tau)$ with $\mathcal{G} = (\mathcal{P}, \mathcal{T}, M_0, \mathcal{F})$.

Construct marking graph $\mathit{Reach}(\mathcal{G}) = (\mathcal{M}, \mathcal{T}, M_0, t_r)$.
Initialise with $Q_0 := \{(M_0, C_0)\}$, $\delta_* :\equiv \varepsilon$, and $i := 0$.
Repeat

$Q_{i+1} := Q_i$
For all $(M, C) \in Q_i$
For all $a \in FirstFired(M, C)$
$d := \tau_a - c_a$
$M' := t_r(M, a)$
$C^+ := Inc(C, d)$
$C^{++} := Reset(C^+, NewEn(M, a, M') \cup \{a\})$
$C' := Sub(C^{++}, En(M'))$
$Q_{i+1} := Q_{i+1} \cup \{(M', C')\}$
$\delta_*((M, C), a, (M', C')) := d$
End For all
End For all
If $Q_{i+1} = Q_i$ then
Return $G_* := (Q_{i+1}, \mathcal{T}, \{(M_0, C_0)\}, \delta_*)$
End If
$i := i + 1$
End Repeat

In order to illustrate the application of the proposed procedure we will detail a few construction steps. Consider the Petri net depicted in Figure 1. The initial state is defined as $(M_0, C_0) = (201, 0‡0)$. As a result we obtain $Q_0 = \{(201, 0‡0)\}$. Following the algorithmic procedure

- $FirstFired(201, 0‡0) = \{a\}$,
- $d = \tau_a - c_a = 2 - 0 = 2$,
- $t_r(201, a) = 111$,
- $NewEn(201, a, 111) = \{b\}$,
- $Sub(Reset(Inc(0‡0, 2), \{a, b\}), \{a, b, c\}) = 002$,

we obtain $NextState(Q_0) = \{(111, 002)\}$ and thus

$$Q_1 = \{(201, 0‡0), (111, 002)\}. \tag{46}$$

For the next iteration we additionally need to consider successors of the newly obtained state $(111, 002)$.

- $FirstFired(111, 002) = \{b, c\}$,
- $d_b = \tau_b - c_b = 1 - 0 = 1$,
- $t_r(111, b) = 201$,
- $NewEn(111, b, 201) = \{c\}$,
- for b: $Sub(Reset(Inc(002, 1), \{b, c\}), \{a, c\}) = 1‡0$,
- $d_c = \tau_c - c_c = 3 - 2 = 1$,
- $t_r(111, c) = 111$,
- $NewEn(111, c, 111) = \{b, c\}$,
- for c: $Sub(Reset(Inc(002, 1), \{b, c\}), \{a, b, c\}) = 100$,

yields

$$NextState(\{(111, 002)\}) = \{(201, 1‡0), (111, 100)\}$$

and therefore

$$Q_2 = \{(201, 0‡0), (111, 002), (201, 1‡0), (111, 100)\}. \tag{47}$$

Continuing in the same vein until the termination condition is reached at $Q_5 = Q_4$, results in the max-plus automaton portrayed in Figure 2.

5. APPLICATION TO SUPERVISOR DESIGN

With the computational procedure obtained from Theorem 8 we can convert a bounded, timed Petri net to an max-plus automaton with equivalent behaviour. In the situation where the Petri net represents a *plant model* and if its behaviour fails to satisfy a prescribed *specification*, we seek a *supervisor* to enforce the specification in closed-loop configuration. In this section we demonstrate by example how our computational procedure can be utilised for the design of a suitable supervisor.

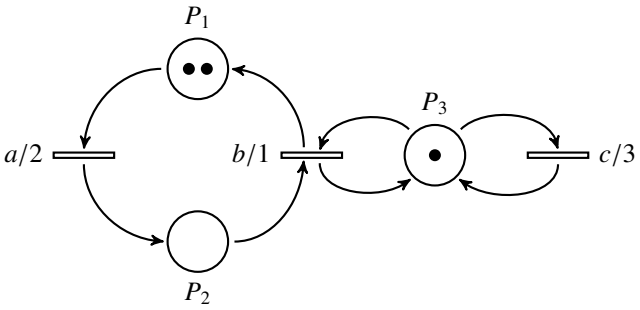

Fig. 1. Timed Petri net $(\mathcal{G}, \tau)$

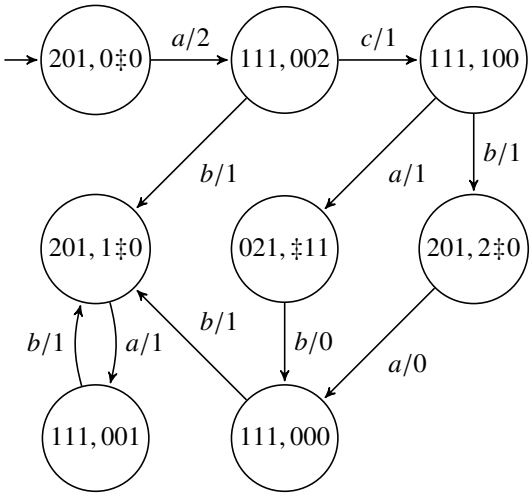

Fig. 2. Resulting max-plus automaton G_*

A common approach to formalise the effect that a supervisor can take on the plant is to partition the set of transitions $\mathcal{T}$ into controllable transitions $\mathcal{T}_c$ and uncontrollable transitions $\mathcal{T}_{uc}$, i.e., $\mathcal{T} = \mathcal{T}_c \dot\cup \mathcal{T}_{uc}$. Clearly, this needs to be followed up by adapting the firing rules accordingly. To this end, we propose the following rules for controllable and uncontrollable transitions.

(C1) The firing of a transition $t \in \mathcal{T}_{uc}$ can not be prevented and it will fire as soon as its duration has elapsed.

(C2) A enabled transition $t \in \mathcal{T}_c$ can be selected for firing if it has a smaller remaining delay than all enabled uncontrollable transitions.

(C3) The supervisor can influence only the logical aspect but not the timing aspect among controllable transitions.

(C4) The net can not idle, which means that for any situation the firing of a feasible transition has to be initialised.

(C5) If transitions $t_1, t_2 \in \mathcal{T}_{uc}$ are enabled and have the same remaining delay both firing options are viable.

(C6) If transitions $t_1 \in \mathcal{T}_{uc}$ and $t_2 \in \mathcal{T}_c$ are both enabled and have the same remaining delay the uncontrollable transition t_1 has priority, as firing of t_2 can not be guaranteed.

The restriction of the supervisor to logical choices solely is due to states depending on clock values. As a result delaying a transition alters the correct successor state. For this example we will only regard the simpler case of logical decisions.

For further illustration we consider a toy example motivated by an engineering application; see Figure 3. Here, the displayed timed Petri net $(\mathcal{G}, \tau)$ is interpreted as a model of a thermal cycle, where work pieces are alternatively heated or cooled. The work pieces are represented by tokens in the left loop and in compliance with single server semantics only one piece can be processed at a time. In this regard the heating process b is only possible if a work piece has arrived and the waste heat valve is closed for two time units, indicated by a token in P_3 used by b. Thereafter the work piece has reached its desired temperature and has to leave the oven in order to avoid being damaged. We represent this relation by choosing $b \in \mathcal{T}_{\mathrm{uc}}$. As a means to reset the temperature in the oven, waste heat can be removed over one time unit and be used for other purposes. The token in P_3 acting as a shared resource implements this behaviour. Furthermore, a work piece getting cooled for longer than two time units does no harm and the valve position can be set as desired, i.e. $a, c \in \mathcal{T}_{\mathrm{c}}$. As a result we obtain $\mathcal{T}_{\mathrm{c}} = \{a, c\}$ and $\mathcal{T}_{\mathrm{uc}} = \{b\}$.

With this distinction and the above mentioned transition properties in mind we can now modify the procedure from Section 4.3 to build a max-plus automaton with a timed behaviour that represents feasible sequences of transitions in the timed Petri net. To this end we extend possible transitions and successor states for a given marking and clock vector to additionally include all enabled controllable transitions with smaller remaining delay than all enabled uncontrollable transitions. Furthermore, to obtain a finite state set we introduce an upper bound on clock values given by the associated duration in the timed Petri net. This is necessary as Lemma 7 does not hold without race policy. However, under the given firing rules we can stop counting when the clock value has reached its respective limit without influencing the timed behaviour. These adjustment yield the max-plus automaton $F_{G_*}^{\mathcal{T}_{\mathrm{c}}}$ depicted in Figure 4. A path in $F_{G_*}^{\mathcal{T}_{\mathrm{c}}}$ represents a firing sequence of transitions and the associated duration that is possible in the timed Petri net with regard to the chosen controllability of transitions. If one were to choose $\mathcal{T}_{\mathrm{uc}} = \mathcal{T}$ the resulting automaton is equal to the one obtained by race policy.

Let us now illustrate the usage of the constructed automaton by introducing a few specifications for the thermal cycle, namely

1) The cooling station only has room for one work piece, i.e. capacity of P_1 is one.
2) Two work pieces have to remain in the left loop at all times.
3) The waste heat valve can be used without restrictions.
4) The heating and cooling processes have to alternate and complete three iterations with minimal duration.
5) The procedure has to end in the same state as it begins.

In order to fulfil 1) we disregard the state $(201, 2 \ddagger 0)$ in the automaton. Condition 2) is achieved by virtue of the chosen initial marking. The desired language to guarantee the first part of 4) is given by $L_{\mathrm{spec}} = (c^* b c^* a c^*)^3$.

With this in mind we observe that the automaton $F_{G_*}^{\mathcal{T}_{\mathrm{c}}}$ seems to be more accessible with regard to upholding the specification than the original timed Petri net. A close examination reveals that one possible path to uphold all specifications disregarding minimal duration is given by

$$w_1 = (bcac)^3, \; d_1 = y_{\mathcal{G}}(w_1) = 12 \tag{48}$$

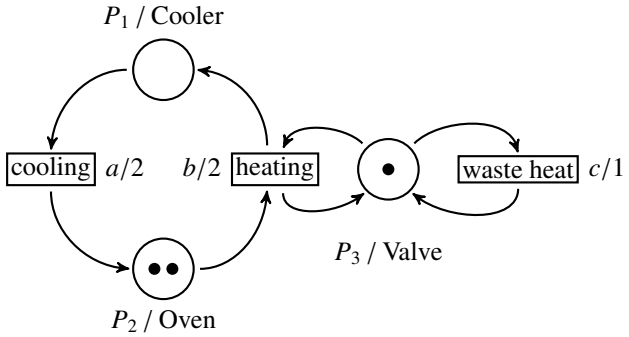

Fig. 3. A thermal cycle application modelled by $(\mathcal{G}, \tau)$.

However, there is a faster alternative possible, that is more effective in running the heating and cooling simultaneously, given by

$$w_2 = b(cab)^2 cac, \; d_2 = y_{\mathcal{G}}(w_2) = 10 \tag{49}$$

Using the graphical representation of $F_{G_*}^{\mathcal{T}_{\mathrm{c}}}$ we conclude that the fastest firing sequence to guarantee success is given by $w_2 = b(cab)^2 cac$ with a duration of ten time units.

6. CONCLUSION

The main technical contribution of this paper, Theorem 8, established a terminating procedure for the conversion of a timed Petri net to a deterministic finite max-plus automaton while retaining the behaviour. The assumptions imposed on the Petri net are boundedness, injective labelling, operation under race policy, and rational timing parameters. Although our argument follows the same line of thought as Komenda et al. (2016), our result is more general in that we do not need to impose fairness requirements on the Petri net.

By the determinism of the resulting max-plus automaton we envisage the application of our procedure in the context of supervisory controller synthesis. A first demonstration has been given by a simple engineering application. By its scale, the example turned out manageable by inspection and we were able to utilise the feasibility automaton in our controller design. Future research topics include the development of a systematic supervisory controller synthesis for more general specifications and/or alternative approaches to incorporate controllability aspects in the construction of the max-plus automaton.

ACKNOWLEDGEMENTS

We greatly acknowledge valuable comments by Jan Komenda on an early draft of this document.

REFERENCES

Alur, R. and Dill, D.L. (1994). A theory of timed automata. *Theor. Computer Science*, 126(2), 183 – 235.

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

Culik, K. and Kari, J. (1997). Handbook of formal languages, vol. 3. chapter Digital Images and Formal Languages, 599–616. Springer-Verlag New York, Inc., New York, NY, USA.

Gaubert, S. and Mairesse, J. (1999). Modeling and analysis of timed petri nets using heaps of pieces. *IEEE Transactions on Automatic Control*, 44(4), 683–697.

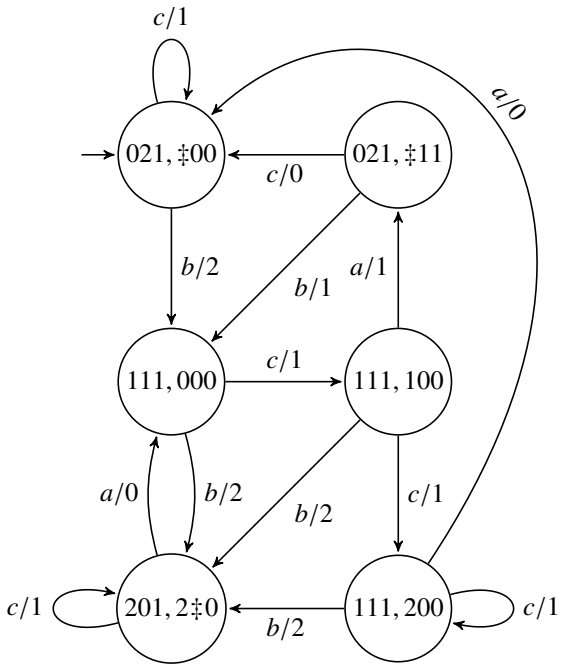

Fig. 4. Feasibility automaton $F^{\mathcal{T}_c}_{G_*}$.

Gaubert, S. (1995). Performance evaluation of (max,+) automata. *IEEE Transactions on Automatic Control*, 40(12), 2014–2025.

Hardouin, L., Cottenceau, B., Shang, Y., and Raisch, J. (2018). Control and state estimation for max-plus linear systems. *Foundations and Trends® in Systems and Control*, 6(1), 1–116.

Kirsten, D. (2008). A burnside approach to the termination of Mohri's algorithm for polynomially ambiguous min-plus-automata. *RAIRO - Theoretical Informatics and Applications - Informatique Théorique et Applications*, 42(3), 553–581.

Komenda, J., Lahaye, S., and Boimond, J.L. (2009). Supervisory control of (max,+) automata: A behavioral approach. *Discrete Event Dynamic Systems*, 19(4), 525–549.

Komenda, J., Lahaye, S., and Boimond, J.L. (2016). Determinization of timed petri nets behaviors. *Discrete Event Dynamic Systems*, 26(3), 413–437.

Lahaye, S., Komenda, J., and Boimond, J.L. (2015). Compositions of (max, +) automata. *Discrete Event Dynamic Systems*, 25(1-2), 323–344.

Mohri, M. (1997). Finite-state transducers in language and speech processing. *Comput. Linguist.*, 23(2), 269–311.

Ramadge, P.J. and Wonham, W.M. (1989). The control of discrete event systems. *Proceedings of the IEEE*, 77, 81–98.

Ramchandani, C. (1973). Analysis of asynchronous concurrent systems by timed petri nets. *Ph.D. Thesis, M.I.T.*

Seatzu, C., Silva, M., and van Schuppen, J.H. (2012). *Control of Discrete-Event Systems: Automata and Petri Net Perspectives*. Springer Publishing Company, Incorporated.

Su, R., van Schuppen, J.H., and Rooda, J.E. (2012). The synthesis of time optimal supervisors by using heaps-of-pieces. *IEEE Transactions on Automatic Control*, 57(1), 105–118. doi:10.1109/TAC.2011.2157391.