

Technical report 12-037

On the Synchronization of Cyclic Discrete-Event Systems*

G. A. D. Lopes, B. De Schutter, and T. J. J. van den Boom

To cite this work, please refer to the published version:

G. A. D. Lopes, B. De Schutter, and T. J. J. van den Boom, “On the synchronization of cyclic discrete-event systems,” *Proceedings of the 51st IEEE Conference on Decision and Control*, Maui, Hawaii, pp. 5810–5815, Dec. 2012. doi:[10.1109/CDC.2012.6426345](https://doi.org/10.1109/CDC.2012.6426345)

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

* This report can also be downloaded via <https://dpub.eu/12-037>

On the synchronization of cyclic discrete-event systems

G.A.D. Lopes, B. De Schutter, and T.J.J. van den Boom

Abstract—Max-plus linear systems are a powerful modeling tool for many applications that involve scheduling and synchronization, such as manufacturing, traffic, and legged locomotion. In this paper we investigate how to systematically construct synchronization controllers for multiple cyclic discrete-event systems modeled in the max-plus framework. We consider that a synchronization specification is given to the control designer as a set of ordering pairs of events that need to occur for the same event counter. We introduce a synchronization controller that verifies the feasible specifications and show that unfeasible specifications are automatically detected. We present simulation results for the evolution of controlled synchronized cyclic systems.

I. INTRODUCTION

The synchronization of cyclic processes is crucial in many applications, such as railroad and urban traffic networks [1]; production systems [2]; queuing systems and array processors [3]; genomics [4]; and legged locomotion [5], [6]. In these applications, the max-plus modeling framework plays an important role, since under the right assumptions it transforms the inherently nonlinear structure of the dynamic equations of cyclic systems into linear expressions in the max-plus algebra [3], [7]. Timed event graphs [8], [9], a subclass of Petri nets, are a class of timed discrete-event systems that can be modeled by max-plus linear equations. Due to the parallels between max-plus linear systems theory and the traditional linear systems theory, many tools are available for analysis of max-plus linear systems.

In this paper we address systems with a nominal behavior consisting in multiple concurrent cyclic processes that can be described in the max-plus algebra. These are to be synchronized via a set of predefined ordering specifications. Given such a list we aim to generate feasible synchronization controllers, such that no deadlock occurs. As such, this paper aims at automatic control synthesis for the canonical max-plus linear systems as presented in Cohen et al. [10], where the control specification naturally results in an implicit form. We assume that the control specification gives rise to constraints that cannot all be met, i.e. it is *desired* to meet all specifications, but not strictly necessary. The goal is thus to minimize the number of specifications that cannot be fulfilled. This synthesis problem appears in scenarios such as schedule creation for the rail road or in legged locomotion as the example presented in this paper. This approach differs from the traditional view of strictly meeting a set of time constraint as in [11] or Just-in-time control as in [12].

This paper is organized as follows: in Section II we briefly review relevant concepts from the theory of max-plus algebra. In Section III we present the structure of the discrete-event systems to be controlled and in Section

IV a synchronization controller is proposed. We conclude in Section V with a synchronization example in legged locomotion. Simulations are presented.

II. MAX-PLUS ALGEBRA

Cuninghame-Green [13], [14] and Giffler [15], [16], [17] discovered independently in the sixties that certain classes of discrete-event systems could be described by equations that only contain the max and + operations. Although nonlinear in traditional algebra, such classes of systems have a linear structure when written in the max-plus algebra [3], [7], [9], a type of tropical algebra with max and + operations as the basic structural elements. For systems modeled in the max-plus algebra, synchronization is driven by operations starting immediately after all preceding operations complete (equivalent to events firing immediately after enabled in the Petri net language) and the duration of the operations is driven by addition: the finishing time of one operation is equal to its starting time plus its duration (equivalent to a holding time in timed Petri nets). In this paper we use the standard notation for the operator $x \oplus y := \max(x, y)$ and $x \otimes y := x + y$. Please see [7], [9] for a complete treatment of the material and the definitions of max-plus nilpotency, precedence graphs, strongly connected graphs, and irreducibility.

Theorem 1 (see [9], Th 3.17): Consider the following system of linear equations in the max-plus algebra:

$$x = A \otimes x \oplus b \quad (1)$$

with $A \in \mathbb{R}_{\max}^{n \times n}$ and $b, x \in \mathbb{R}_{\max}^{n \times 1}$. Now let

$$A^* := \bigoplus_{p=0}^{\infty} A^{\otimes p} .$$

If A^* exists then $x = A^* \otimes b$ solves the system of max-plus linear equations (1). If the greatest eigenvalue of A is negative then the solution is unique.

III. CYCLIC SYSTEMS

We now consider discrete-event systems that consist of multiple circuits that we aim to synchronize, as illustrated in Figure 1. Let $a_i(k) \in \mathbb{R}_{\max}$ represent the time at which the event $a_i \in \mathcal{Q}$ fires for the k -th turn, where \mathcal{Q} is the set of all events. A firing precedence order in a cycle indexed by k is represented by the following equation in the traditional algebra:

$$a_j(k) \geq a_i(k) + \tau_{i,j}$$

meaning that event a_j can only fire $\tau_{i,j}$ time units after event a_i has fired, within the same cycle k . Now consider a system

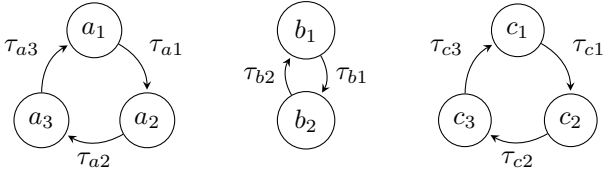


Fig. 1. A system to be synchronized: multiple disconnected circuits of events. The parameters τ_i represent the “holding times”, i.e. the time that must elapse before a transition occurs.

whose nominal dynamics consists of m circuits, with $m > 1$. Let $r(i)$ be the number of events in circuit i . Then the total event state is defined by:

$$x(k) = [\bar{a}_1^T(k) \cdots \bar{a}_m^T(k)]^T \quad (2)$$

where each event vector $\bar{a}_i(k)$ describes a circuit:

$$\bar{a}_i(k) = [a_{h(i,1)}(k) \cdots a_{h(i,r(i))}(k)]^T, \quad (3)$$

$$a_{h(i,j)}(k) \in \mathbb{R}_{\max}, \quad \forall i, j : 1 \leq i \leq m, 1 \leq j \leq r(i),$$

where $h(i, j)$ is an indexing function that maps the event j of circuit i to an index on the total set of events \mathcal{Q} . We consider that the following ordering relations exist for all events in the circuits, with all strictly positive holding times $\tau_{i,j} > 0$:

$$a_{h(i,j+1)}(k) \geq a_{h(i,j)}(k) + \tau_{i,j} \quad \text{for } 1 \leq j < r(i) \quad (4)$$

$$a_{h(i,1)}(k) \geq a_{h(i,r(i))}(k-1) + \tau_{i,r(i)} \quad (5)$$

The graph in Figure 1 shows an example of three concurrent circuits with 3, 2 and 3 events each, for the total set of events $\mathcal{Q} = \{a_1, a_2, a_3, b_1, b_2, c_1, c_2, c_3\}$. Equations (4) and (5) can be written compactly in the max-plus algebra by the expression (here replacing the inequalities by equalities, meaning that each event fires as soon as all conditions are satisfied):

$$\bar{a}_i(k) = A_{0,i} \otimes \bar{a}_i(k) \oplus A_{1,i} \otimes \bar{a}_i(k-1)$$

where the matrix $A_{0,i}$ is max-plus lower-band triangular and $A_{1,i}$ is max-plus upper-band triangular, both with max-plus zeros in the diagonal:

$$A_{0,i} = \begin{bmatrix} \varepsilon & & \cdots & & \varepsilon \\ \tau_{i,1} & \varepsilon & & & \\ \varepsilon & \tau_{i,2} & \varepsilon & & \vdots \\ \vdots & & \ddots & \ddots & \\ \varepsilon & \cdots & \varepsilon & \tau_{i,r(i)-1} & \varepsilon \end{bmatrix}$$

$$A_{1,i} = \begin{bmatrix} \varepsilon & \cdots & \varepsilon & \tau_{i,r(i)} \\ & & \varepsilon & \varepsilon \\ & \ddots & & \vdots \\ \varepsilon & & & \varepsilon \end{bmatrix}$$

The total uncontrolled system for the event state defined in (2) is then written as:

$$x(k) = A_0 \otimes x(k) \oplus A_1 \otimes x(k-1) \quad (6)$$

with block diagonal matrices

$$A_0 = \begin{bmatrix} A_{0,1} & \varepsilon & \cdots & \varepsilon \\ \varepsilon & A_{0,2} & & \vdots \\ \vdots & & \ddots & \varepsilon \\ \varepsilon & \cdots & \varepsilon & A_{0,m} \end{bmatrix}$$

$$A_1 = \begin{bmatrix} A_{1,1} & \varepsilon & \cdots & \varepsilon \\ \varepsilon & A_{1,2} & & \vdots \\ \vdots & & \ddots & \varepsilon \\ \varepsilon & \cdots & \varepsilon & A_{1,m} \end{bmatrix}$$

It can be shown that system (6) always admits the solution

$$x(k) = A_0 \otimes x(k) \oplus A_1 \otimes x(k-1) = A_0^* \otimes A_1 \otimes x(k-1)$$

since A_0^* can always be computed due to A_0 being nilpotent. Since, by construction, each circuit is disconnected from each other, the precedence graph of the matrix $A_0^* \otimes A_1$ is not strongly connected and as such it is not irreducible. The precedence graph of the matrix $A_0 \oplus A_1$ represents m disconnected circuits, by construction, as illustrated in Figure 1

We now consider the controlled system by introducing the signal $u(k)$ that can insert delays:

$$x(k) = A_0 \otimes x(k) \oplus A_1 \otimes x(k-1) \oplus u(k)$$

The signal $u(k)$ is now used to enforce synchronization between the circuits. Assume that a synchronization specification \mathcal{S} is supplied to the control designer as a set of precedence order of events of the form:

$$\mathcal{S}_v : a_j \succ a_i + \sigma_{i,j} \quad (7)$$

This specification, indexed by v , represents that the event a_j should only fire $\sigma_{i,j} > 0$ time units after event a_i has fired. Note that the event counter k is omitted in this specification since it is not known yet if it is possible to construct a correct synchronization controller that verifies all the given pairs. The goal of the next section is to find the largest set of specification pairs that can be implemented within the same cycle k while the nominal dynamics, represented by equations (4) and (5) are verified.

For the state vector $x = [a_1 \cdots a_n]^T$, given a specification \mathcal{S}_v one can build the matrix $W_v(a_i, a_j, \sigma_{i,j})$, with $1 \leq i, j \leq n$, as:

$$[W_v]_{lp} = \begin{cases} \sigma_{i,j} & \text{if } l = j \text{ and } p = i \\ \varepsilon & \text{otherwise} \end{cases}$$

such that the expression

$$x \succ W_v \otimes x$$

is equivalent to expression (7). Now consider that the control signal $u(k)$ takes the form:

$$u(k) = S_0 \otimes x(k) \oplus S_1 \otimes x(k-1)$$

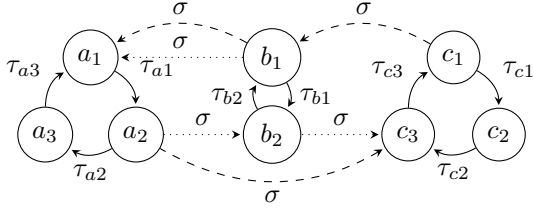


Fig. 2. Example of a partially synchronized precedence graph, represented by the solid arcs together with the dotted arcs (each circuit has at least one event with an incoming arc from a different circuit); and of a fully synchronized precedence graph, represented by the solid arcs together with the dashed arcs (each circuit has at least one event with an incoming arc and one event with an outgoing arc, from and to different circuits; and graph is weakly connected)

We aim to encode the synchronization specifications into the matrices S_0 and S_1 such that the resulting closed-loop max-plus linear system equation can be transformed from an implicit form to an explicit one, i.e. it is solvable. Note that ideally all specifications should be encoded in the matrix S_0 , since that represents that synchronizations occur between events in the same event cycle k . Placing all the specifications in matrix S_1 trivially allows for computing the explicit form, but in practice does not result in a system that verified the specifications (although it might be synchronized). Replacing the input $u(k)$ into equation (6) we obtain the closed-loop system:

$$x(k) = (A_0 \oplus S_0) \otimes x(k) \oplus (A_1 \oplus S_1) \otimes x(k-1) \quad (8)$$

Definition 2: We call the system represented by equation (8) *partially synchronized* if for every circuit (encoded by the matrices A_0 and A_1) of the precedence graph of the matrix $A_0 \oplus A_1 \oplus S_0 \oplus S_1$ there is at least one event with an incoming arc from a different circuit (encoded in the matrices S_0 and S_1). The system is called *fully synchronized* or simply *synchronized* if it is partially synchronized, there is at least an event on every circuit with an outgoing arc to a different circuit, and the precedence graph is weakly connected. We call a system *unsynchronized* if it is not partially synchronized. See the example in Figure 2.

Proposition 3: The matrix $A_0 \oplus A_1 \oplus S_0 \oplus S_1$ of a fully synchronized system is irreducible.

Proof: Since each circuit has events with incoming and outgoing arcs to other circuits, and the precedence graph is weakly connected, then there exists a path from any node to any other node, i.e. the graph is strongly connected, thus $A_0 \oplus A_1 \oplus S_0 \oplus S_1$ is irreducible. ■

IV. NON-BLOCKING CONTROLLERS

By construction the precedence graph of the matrix A_0 has no cycles due to its max-plus lower-band triangular structure with max-plus zeros in the diagonal. That is a sufficient condition for the matrix A_0^* to be well defined. In the controlled case, one needs to make sure that each arc introduced by the synchronization pairs admits a solution for equation (8), i.e. the matrix $(A_0 \oplus S_0)^*$ must be well defined. Given an arbitrary specification of synchronization

pairs, as in (7), there can be situations where some pairs are conflicting. In this situation the controller must be designed such that all conflicting pairs are addressed. This is done by placing the conflicting synchronization constraints in the matrix S_1 . Before we present systematic methodologies for designing such controllers we introduce an example: let the state vector x be

$$x = [a_1 \ a_2 \ a_3 \ b_1 \ b_2 \ c_1 \ c_2 \ c_3]^T$$

with paths:

$$p_1 : a_1 \rightarrow a_2 \rightarrow a_3 \rightarrow a_1$$

$$p_2 : b_1 \rightarrow b_2 \rightarrow b_1$$

$$p_3 : c_1 \rightarrow c_2 \rightarrow c_3 \rightarrow c_1$$

as illustrated in Figure 1. Now consider the following list of desired synchronization pairs:

$$\begin{aligned} \mathcal{S}_1 : c_1 \succ a_1 + \sigma_1 & & \mathcal{S}_2 : b_1 \succ c_1 + \sigma_2 \\ \mathcal{S}_3 : c_3 \succ b_1 + \sigma_3 & & \mathcal{S}_4 : a_2 \succ c_3 + \sigma_4 \\ \mathcal{S}_5 : c_2 \succ a_2 + \sigma_5 & & \mathcal{S}_6 : a_3 \succ c_2 + \sigma_6 \\ \mathcal{S}_7 : b_2 \succ a_3 + \sigma_7 & & \end{aligned}$$

From this list, one can see that specifications \mathcal{S}_4 and \mathcal{S}_5 are conflicting, since in the same cycle k both cannot occur simultaneously without breaking the natural order of each circuit. Let $S_{\text{all}} = W_1 \oplus W_2 \oplus \dots \oplus W_7$. For this example we get

$$S_{\text{all}} = \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \sigma_4 \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \sigma_6 & \varepsilon \\ \hline \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \sigma_2 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \sigma_7 & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \hline \sigma_1 & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \sigma_5 & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \sigma_3 & \varepsilon & \varepsilon & \varepsilon & \varepsilon \end{bmatrix}$$

A sufficient condition for the matrix $(A_0 \oplus S_0)^*$ to exist is that it is max-plus lower-band diagonal with max-plus zeros on the diagonal, i.e. it is nilpotent. As such, if we let $S_0 = W_1 \oplus W_3 \oplus W_5 \oplus W_7$ and $S_1 = W_2 \oplus W_4 \oplus W_6$, then the matrix $A_0 \oplus S_0$ is max-plus lower-band diagonal, and equation (8) admits a solution. However, since the holding times σ_2 , σ_4 , and σ_6 were placed in the matrix S_1 we obtain the resulting equations for the event cycle k :

$$\begin{aligned} \mathcal{S}_1 : c_1(k) &\geq a_1(k) + \sigma_1 & \mathcal{S}_2 : b_1(k) &\geq c_1(k-1) + \sigma_2 \\ \mathcal{S}_3 : c_3(k) &\geq b_1(k) + \sigma_3 & \mathcal{S}_4 : a_2(k) &\geq c_3(k-1) + \sigma_4 \\ \mathcal{S}_5 : c_2(k) &\geq a_2(k) + \sigma_5 & \mathcal{S}_6 : a_3(k) &\geq c_2(k-1) + \sigma_6 \\ \mathcal{S}_7 : b_2(k) &\geq a_3(k) + \sigma_7 & & \end{aligned}$$

This result is not optimal, in the sense enforcing the maximum number of synchronization constraints within the same event cycle k , it is weaker than the original desired specification. In fact, for this example, one can find a permutation matrix P , such that all of the non-max-plus zero elements of $P \otimes A_0 \otimes P^T$ stay in the lower diagonal and all but one of the non-max-plus zero elements of $P \otimes S_0 \otimes P^T$ stay in the lower diagonal. This poses the question of how to assign the minimum number of synchronization holding times to the matrix S_1 such that the matrix $(A_0 \oplus S_0)^*$ is well defined. This problem can be approached from different angles.

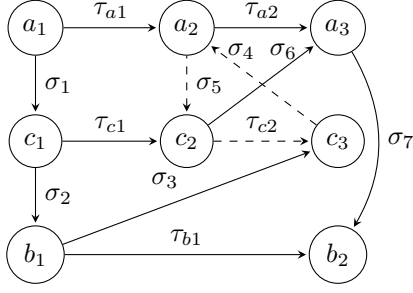


Fig. 3. The precedence graph of the matrix $A_0 \oplus S_{\text{all}}$. The undesirable cycle is represented by the dashed lines.

A. Trimming arcs in the precedence graph

Since all the non ε -parameters in the matrix $A_0 \oplus S_0$ are assumed to be positive numbers, a sufficient and necessary condition for the matrix $(A_0 \oplus S_0)^*$ to exist is that the precedence graph of the matrix $A_0 \oplus S_0$ has no cycles [3]. Figure 3 illustrates the precedence graph of the matrix $A_0 \oplus S_{\text{all}}$ from the previously described example. Due to the conflicting specification a cycle is present via the path

$$c_2 \xrightarrow{\tau_{c2}} c_3 \xrightarrow{\sigma_4} a_2 \xrightarrow{\sigma_5} c_2$$

represented in Figure 3 by the dashed lines. To break the cycle, it is sufficient then to either remove the $c_3 \rightarrow a_2$ arc by setting $S_0 = W_1 \oplus W_2 \oplus W_3 \oplus W_5 \oplus W_6 \oplus W_7$ and $S_1 = W_4$ or by removing the $a_2 \rightarrow c_2$ arc by setting $S_0 = W_1 \oplus W_2 \oplus W_3 \oplus W_4 \oplus W_6 \oplus W_7$ and $S_1 = W_5$. Detecting cycles in directed graphs is a well known problem in graph theory. In [18], [19] Knuth et al. presented a method based on incrementally deleting “sources”, i.e. vertices in the graph with no incoming arcs. Tarjan [20] presented a depth-first search algorithm. These methods find cycles in $O(n + q)$ time, where n is the number of vertices (event states) and q is the number of arcs. In our case we have that $q = n - m + p$ where m is the number of circuits and p is the number of specification pairs (note that we are enumerating the arcs in the precedence graph of $A_0 \oplus S_{\text{all}}$). Once a cycle is detected, it must be eliminated. Since the detection of an arc causing a cycle is a function of the “initial topology representation” of the graph supplied to the algorithm, only sub-optimal solutions are guaranteed to be found.

B. Incremental topological ordering

A *topological ordering* [21] of a directed graph is defined to be a total ordering of the vertices (events in precedence graphs) such that for every arc from a_i to a_j we get an order $a_i < a_j$. Ordering thus preclude cycles in the graph, since their existence would invalidate the order of vertices. Incremental topological ordering methods [22], [21] are designed to detect cycles when new arcs are added to an existing graph. These methods typically assume a fixed vertex structure, an initial topological ordering is given for a graph, and new arcs are added incrementally. Haeupler et al. [21] presented an algorithm that processes p arc additions in $O(p^{3/2})$ time, where in our case p is the number of

specification pairs. Using this algorithm, starting with the trivial topological ordering $O_0 = \{a_1 \ a_2 \ a_3 \ b_1 \ b_2 \ c_1 \ c_2 \ c_3\}$ (the order of the state vector x), and the set of arcs $\{a_1 \rightarrow a_2, a_2 \rightarrow a_3, b_1 \rightarrow b_2, c_1 \rightarrow c_2, c_2 \rightarrow c_3\}$, we obtain the following topological orderings as new arcs are added:

$$\begin{array}{ll} O_0 = \{a_1 \ a_2 \ a_3 \ b_1 \ b_2 \ c_1 \ c_2 \ c_3\} \\ S_1 : a_1 \rightarrow c_1 & O_1 = \{\underline{a_1} \ a_2 \ a_3 \ b_1 \ b_2 \ \underline{c_1} \ c_2 \ c_3\} \\ S_2 : c_1 \rightarrow b_1 & O_2 = \{a_1 \ a_2 \ a_3 \ \underline{c_1} \ \underline{b_1} \ b_2 \ c_2 \ c_3\} \\ S_3 : b_1 \rightarrow c_3 & O_3 = \{a_1 \ a_2 \ a_3 \ c_1 \ \underline{b_1} \ b_2 \ c_2 \ \underline{c_3}\} \\ S_4 : c_3 \rightarrow a_2 & O_4 = \{a_1 \ c_1 \ b_1 \ b_2 \ c_2 \ \underline{c_3} \ \underline{a_2} \ a_3\} \\ \cancel{S_5} : a_2 \rightarrow c_2 & O_5 = \{a_1 \ c_1 \ b_1 \ b_2 \ c_2 \ c_3 \ a_2 \ a_3\} \\ S_6 : c_2 \rightarrow a_3 & O_6 = \{a_1 \ c_1 \ b_1 \ b_2 \ \underline{c_2} \ c_3 \ a_2 \ \underline{a_3}\} \\ S_7 : a_3 \rightarrow b_2 & O_7 = \{a_1 \ c_1 \ b_1 \ c_2 \ c_3 \ a_2 \ \underline{a_3} \ \underline{b_2}\} \end{array}$$

The algorithm detects that no topological ordering is possible if specification S_5 is added, since it creates a cycle. As such, S_5 is discarded from S_0 but added to S_1 . Since the order at which arcs are introduced affects the final result, this procedure is not guaranteed to always discard the minimum number of arcs. As such, as it was the case in the previous methodology, sub-optimal solutions are returned. One interesting property of the topological ordering method is that it can be used to construct a permutation matrix P_i that renders the matrix $P_i \otimes (A_0 \oplus S_0) \otimes P_i^T$ max-plus lower-band diagonal, which simplifies the computation of $(A_0 \oplus S_0)^*$. For example, construct a new vector \bar{x} with the same ordering as O_7 :

$$\bar{x} = [a_1 \ c_1 \ b_1 \ c_2 \ c_3 \ a_2 \ a_3 \ b_2]^T$$

Then, a permutation matrix P_7 can be readily computed such that $\bar{x} = P_7 \otimes x$:

$$P_7 = \begin{bmatrix} 0 & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & 0 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & 0 & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & 0 & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & 0 \\ \varepsilon & 0 & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 0 & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & 0 & \varepsilon & \varepsilon & \varepsilon \end{bmatrix}$$

We can now use the permutation matrix to observe the structure of the matrix $A_0 \oplus S_{\text{all}}$ written in the \bar{x} coordinates:

$$P_7 \otimes (A_0 \oplus S_{\text{all}}) \otimes P_7^T = \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \sigma_1 & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \sigma_2 & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \tau_{c1} & \varepsilon & \varepsilon & \varepsilon & \sigma_5 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \sigma_3 & \tau_{c2} & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \tau_{a1} & \varepsilon & \varepsilon & \varepsilon & \sigma_4 & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \sigma_6 & \varepsilon & \tau_{a2} & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \tau_{b1} & \varepsilon & \varepsilon & \varepsilon & \sigma_7 & \varepsilon \end{bmatrix}$$

From the last matrix it is clear that by eliminating the specification S_5 from S_0 the matrix $P_7 \otimes (A_0 \oplus S_0) \otimes P_7^T$ becomes max-plus lower-band diagonal.

Figure 4 illustrates a sample execution in time of equation (8) with $S_0 = W_1 \otimes W_2 \otimes W_3 \otimes W_4 \otimes W_6 \otimes W_7$ and $S_1 = W_5$, and with all $\tau_{i,j} = 1$ and all $\sigma_{i,j} = 1/2$. The execution follows the order O_7 .

C. Mixed Integer Linear Programming

To compute the maximal set of specification pairs that can be included in the matrix S_0 one can recast the max-plus linear equations described by (8) into a mixed integer linear programming problem (MILP). The important property of equation (8) is that it is written in an implicit form. If the equation

$$x(k) = (A_0 \oplus S_0) \otimes x(k) \oplus \gamma \quad (9)$$

admits a solution for some vector γ , with no max-plus zero entries in γ , then equation (8) will also admit a solution. As such, it is sufficient to analyze the solutions of (9), which allows us to drop the index k , since only $x(k)$ is referenced in (9). So we get:

$$x = (A_0 \oplus S_0) \otimes x \oplus \gamma \quad (10)$$

Equation (10) contains the max operator, but it can be recast as a set of inequalities for the synchronization of each circuit:

$$a_{h(i,j+1)} \geq a_{h(i,j)} + \tau_{i,j} \quad \text{for } 1 \leq j < r(i) \quad (11)$$

and for the synchronization between different circuits:

$$a_j(k) \geq a_i(k) + \sigma_{i,j} \quad (12)$$

Since imposing all specifications may fail to admit a solution of (10), we modify equation (12) such that new binary parameters $\delta_{i,j} \in \{0, 1\}$ are included to in effect “turn on” or “turn off” specifications:

$$a_j(k) \geq a_i(k) + \sigma_{i,j} \delta_{i,j} + \beta(1 - \delta_{i,j}) \quad (13)$$

When $\delta_{i,j} = 1$ equation (12) is recovered. When $\delta_{i,j} = 0$, by making the parameter β very negative (the largest allowed value of β can be computed based on properties of the system, such as the total cycle time) then the term $a_i(k) + \beta$ becomes very small so that it always verifies the inequality. Consider the new vector z :

$$z = [x^T \quad \delta^T]^T$$

Equations (11) and (13) are linear in the parameters x and δ , and can be combined into the system:

$$Cz \geq b, \quad (14)$$

for appropriate parameters C and b . The goal is then to maximize the sum of the parameters $\delta_{i,j}$, i.e., to include the maximum number of specifications in the matrix S_0 :

$$\max \sum \delta_{i,j} = \max g^T z, \quad (15)$$

with $g = [1 \cdots 1 \quad 0 \cdots 0]^T$. Equation (14) together with (15) forms a standard mixed integer linear programming problem. The solution of this optimization problem yields the (global) optimal solution, at the cost of increased computational cost. Several efficient branch-and-bound algorithms [23] are

available for mixed integer linear programming problems. In addition, there exist several commercial and free solvers for mixed integer linear programming problems such as CPLEX, Xpress-MP, GLPK, or Ip_solve (see [24], [25] for an overview).

V. EXAMPLE: LEGGED LOCOMOTION

An example of circuits that need to be synchronized appears in designing controllers for legged robots [5]. Each leg i is abstracted into a two event circuit $\{t_i, l_i\}$, with events t_i representing the instant the leg touches the ground and l_i representing the moment the leg lifts off. Let τ_f be the swing time (leg in the air) and τ_g the stance time (leg touching the ground). The nominal equations for leg i are:

$$\begin{aligned} t_i(k) &\geq l_i(k) + \tau_f \\ l_i(k) &\geq t_i(k-1) + \tau_g \end{aligned}$$

A gait (a manner of walking) can be constructed by synchronizing the leg circuits. Consider a robot with legs numbered from left to right and front to back, as illustrated in Figure 5.a). A trotting gait on a quadruped robot means that legs 1 and 4 are synchronized, and have the opposite phase of legs 2 and 3 (i.e. diagonal legs are synchronized, see [5] for more details). Such gait is equivalent to the following list of specifications imposing constraints on the lift off times of legs as a function of touchdown times (i.e. legs are only allowed to lift off the ground after others have touched down to avoid the robot from falling from lack of support), where τ_Δ is called the “double stance time”:

$$\begin{aligned} S_1 : l_2 > t_1 + \tau_\Delta & \quad S_2 : l_3 > t_1 + \tau_\Delta \\ S_3 : l_2 > t_4 + \tau_\Delta & \quad S_4 : l_3 > t_4 + \tau_\Delta \\ S_5 : l_1 > t_2 + \tau_\Delta & \quad S_6 : l_4 > t_2 + \tau_\Delta \\ S_7 : l_1 > t_3 + \tau_\Delta & \quad S_8 : l_4 > t_3 + \tau_\Delta \end{aligned}$$

These specifications are conflicting but describe the gait correctly: legs 2 and 3 should lift off only after legs 1 and 4 have touched down, and legs 1 and 4 should only lift off after legs 2 and 4 have touched down. Computing the matrix $A_0 \oplus S_{\text{all}}$ for the state vector $x = [l_1 \ t_1 \ l_2 \ t_2 \ l_3 \ t_3 \ l_4 \ t_4]$ results in:

$$A_0 \oplus S_{\text{all}} = \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon & \tau_\Delta & \varepsilon & \varepsilon & \varepsilon & \tau_\Delta \\ \tau_f & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \tau_\Delta & \varepsilon & \varepsilon & \varepsilon & \tau_\Delta & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \tau_f & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \tau_\Delta & \varepsilon & \varepsilon & \varepsilon & \tau_\Delta \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \tau_f & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \tau_\Delta & \varepsilon & \varepsilon & \varepsilon & \tau_\Delta & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \tau_f & \varepsilon \end{bmatrix}$$

Using the incremental topological ordering method we obtain the following new ordering:

$$O = \{l_1 \ l_4 \ t_1 \ t_4 \ l_2 \ l_3 \ t_2 \ t_3\}$$

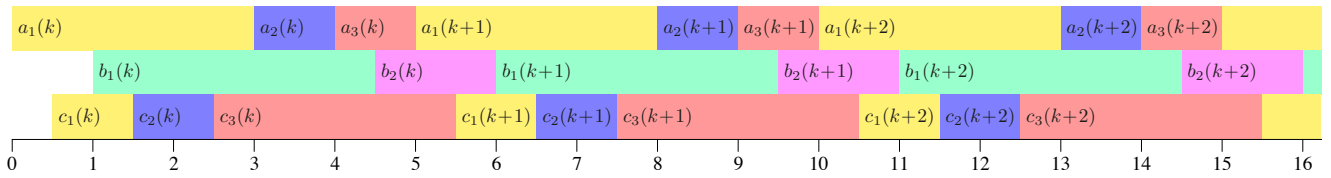


Fig. 4. Sample execution in time of equation (8) for the example presented in the beginning of Section IV. With matrices $S_0 = S_1 \otimes S_2 \otimes S_3 \otimes S_4 \otimes S_6 \otimes S_7$ and $S_1 = S_5$, and with all $\tau_{i,j} = 1$ and all $\sigma_{i,j} = 1/2$. Execution follows the order $O_7 = \{a_1 \ c_1 \ b_1 \ c_2 \ c_3 \ a_2 \ a_3 \ b_2\}$.

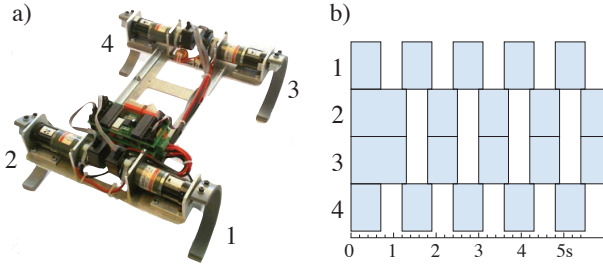


Fig. 5. a) RQuad, a quadruped robot developed at DCSC. Legs are numbered; b) the resulting trotting gait: blue boxes represent legs in stance and white boxes represent legs in swing. In this example $\tau_f = 0.5s$, $\tau_g = 0.7s$, and $\tau_\Delta = 0.1s$.

By constructing a permutation matrix P based on the previous ordering we obtain:

$$P \otimes A_0 \oplus S_{\text{all}} \otimes P^T = \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \tau_\Delta & \tau_\Delta \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \tau_\Delta & \tau_\Delta \\ \tau_f & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \tau_f & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \tau_\Delta & \tau_\Delta & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \tau_\Delta & \tau_\Delta & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \tau_f & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \tau_f & \varepsilon & \varepsilon \end{bmatrix}$$

A non-blocking controller is then achieved by making $S_0 = W_1 \oplus W_2 \oplus W_3 \oplus W_4$ and $S_0 = W_5 \oplus W_6 \oplus W_7 \oplus W_8$. Figure 5.b) illustrates a sample simulation of the resulting trotting gait.

VI. CONCLUSIONS

We have proposed a systematic way to design synchronization controllers for discrete-event systems with concurrent circuits via a list of synchronization specifications. When the desired specifications cannot be fully fulfilled, a subset of them must be relaxed. We propose three avenues for dealing with unfeasible specifications: Trimming arcs in a precedence graph, incremental topological ordering, and solving a mixed integer linear program. The first two methods are time efficient but return sub-optimal results, while the third is known to be NP-hard to solve, but returns optimal solutions. This trade-off needs to be considered with regards to the application in mind.

REFERENCES

- [1] B. Heidergott and R. de Vries, "Towards a (max,+) control theory for public transportation networks," *Discrete Event Dynamic Systems*, vol. 11, pp. 371–398, 2001.
- [2] M. Zhou, F. DiCesare, and A. Desrochers, "A hybrid methodology for synthesis of Petri net models for manufacturing systems," *IEEE Trans. on Robotics and Automation*, vol. 8, no. 3, pp. 350–361, 1992.
- [3] B. Heidergott, G. Olsder, and J. van der Woude, *Max Plus at Work: Modeling and Analysis of Synchronized Systems*. Kluwer, 2006.
- [4] K. Shedden, "Analysis of cell-cycle gene expression in *Saccharomyces cerevisiae* using microarrays and multiple synchronization methods," *Nucleic Acids Research*, vol. 30, no. 13, pp. 2920–2929, 2002.
- [5] B. Kersbergen, G. Lopes, B. De Schutter, T. van den Boom, and R. Babuška, "Optimal gait switching for legged locomotion," in *Proc. of the IEEE/RSI Int. Conf. on Intelligent Robots and Systems*, San Francisco, USA, 2011, pp. 2729–2734.
- [6] G. Lopes, B. Kersbergen, T. van den Boom, B. De Schutter, and R. Babuška, "On the eigenstructure of a class of max-plus linear systems," in *Proc. of the IEEE Conf. on Decision and Control*, Orlando, USA, 2011, pp. 1823–1828.
- [7] R. A. Cuninghame-Green, *Minimax Algebra*, ser. Lecture Notes in Econ. and Math. Systems. Springer-Verlag, 1979, vol. 166.
- [8] J. L. Peterson, *Petri Net Theory and the Modeling of Systems*. Englewood Cliffs, New Jersey: Prentice Hall, Inc., 1981.
- [9] F. Baccelli, G. Cohen, G. Olsder, and J. Quadrat, *Synchronization and Linearity: an Algebra for Discrete Event Systems*. Wiley, 1992.
- [10] G. Cohen, S. Gaubert, and J. Quadrat, "Max-plus algebra and system theory: Where we are and where to go now," *Annual Reviews in Control*, no. 23, pp. 207–219, 1999.
- [11] S. Amari, I. Demongodin, J. Loiseau, and C. Martinez, "Max-Plus Control Design for Temporal Constraints Meeting in Timed Event Graphs," *IEEE Trans. Auto. Control*, vol. 57, no. 2, pp. 462–467, 2012.
- [12] L. Houssin, S. Lahaye, and J. L. Boimond, "Just in time control of constrained (max,+)-linear systems," *Discrete Event Dynamic Systems*, vol. 17, no. 2, 2007.
- [13] R. Cuninghame-Green, "Process synchronisation in a steelworks – A problem of feasibility," in *Proc. of the 2nd Int. Conf. on Operational Research*, J. Banbury and J. Maitland, Eds. Aix-en-Provence, France: London: English Universities Press, 1961, pp. 323–328.
- [14] —, "Describing industrial processes with interference and approximating their steady-state behaviour," *Operational Research Quarterly*, vol. 13, no. 1, pp. 95–100, 1962.
- [15] B. Giffler, "Mathematical solution of production planning and scheduling problems," IBM ASDD, Tech. Rep., 1960.
- [16] —, "Scheduling general production systems using schedule algebra," *Naval Research Logis. Quart.*, vol. 10, no. 3, pp. 237–255, 1963.
- [17] —, "Schedule algebra: A progress report," *Naval Research Logis. Quart.*, vol. 15, no. 2, pp. 255–280, 1968.
- [18] D. Knuth, *The Art of Computer Programming, Vol. 1: Fundamental Algorithms*. Addison-Wesley, 1973.
- [19] D. Knuth and J. Schwarzfiter, "A structured program to generate all topological sorting arrangements," *Information Processing Letters*, vol. 2, no. 6, pp. 153–157, 1974.
- [20] R. Tarjan, "Depth-first search and linear graph algorithms," in *Symposium on Switching and Automata Theory*, 1971, pp. 114–121.
- [21] B. Haeupler, T. Kavitha, R. Mathew, S. Sen, and R. E. Tarjan, "Incremental Cycle Detection, Topological Ordering, and Strong Component Maintenance," *ACM Trans. on Algorithms*, vol. 8, no. 1, 2012.
- [22] D. J. Pearce and P. H. J. Kelly, "A dynamic topological sort algorithm for directed acyclic graphs," *ACM J. of Exp. Algor.*, vol. 11, 2007.
- [23] R. Fletcher and S. Leyffer, "Numerical experience with lower bounds for MIQP branch-and-bound," *SIAM J. on Optimization*, vol. 8, no. 2, pp. 604–616, 1998.
- [24] A. Atamtürk and M. Savelsbergh, "Integer-programming software systems," *Annals of Oper. Research*, vol. 140, no. 1, pp. 67–124, 2005.
- [25] J. Linderoth and T. Ralphs, "Noncommercial software for mixed-integer linear programming," *Optimization Online*, 2005.