An Automated Framework for Formal Verification of Timed Continuous Petri Nets
-draft-

Marius Kloetzer, Cristian Mahulea, Calin Belta, and Manuel Silva [*]

December 23, 2014

Abstract

In this paper, we develop an automated framework for formal verification of timed continuous Petri nets (*ContPN*). Specifically, we consider two problems: (1) given an initial set of markings, construct a set of unreachable markings, (2) given a Linear Temporal Logic (LTL) formula over a set of linear predicates in the marking space, construct a set of initial states such that all trajectories originating there satisfy the LTL specification. The starting point for our approach is the observation that a *ContPN* system can be expressed as a Piecewise Affine (PWA) system with a polyhedral partition. We propose an iterative method for analysis of PWA systems from specifications given as LTL formulas over linear predicates. The computation mainly consists of *polyhedral operations* and *searches on graphs*, and the developed framework was implemented as a freely downloadable software tool. We present several illustrative numerical examples.

Published as:
M. Kloetzer, C. Mahulea, C. Belta, and M. Silva, "An Automated Framework for Formal Verification of Timed Continuous Petri Nets," IEEE Transactions on Industrial Informatics, vol. 6, no. 3, pp. 460 - 471, August 2010. DOI: `http://doi.org/10.1109/TII.2010.2050001`

[*]M. Kloetzer is with the Department of Automatic Control and Applied Informatics at the Technical University "Gheorghe Asachi" of Iasi, Romania, `kmarius@ac.tuiasi.ro`
C. Mahulea and M. Silva are with the Aragón Institute for Engineering Research (I3A), University of Zaragoza, Maria de Luna 1, 50018 Zaragoza, Spain, {`cmahulea,silva`}`@unizar.es`
C. Belta is with the Department of Mechanical Engineering and the Division of Systems Engineering at Boston University, Boston, MA 02215, `cbelta@bu.edu`
This work was partially supported from grants NSF CNS-0410514 and NSF CNS-0834260 at Boston University and from grant CICYT - FEDER DPI2006-15390 at the University of Zaragoza. The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no 224498.

1 Introduction

Discrete Petri nets (PN) are a powerful mathematical formalism with an appealing graphical representation, suitable for modelling, analysis and synthesis of discrete-event systems. Their main feature is the capacity to graphically represent and visualize primitives such as parallelism, synchronization, and mutual exclusion. Petri nets are successfully used in multiple complex automated and distributed systems, such as manufacturing systems [2, 3, 4, 5, 6, 7], energy or railway transport networks [8], and petro-chemical plants [9]. Such complex systems have to satisfy a broad area of objectives, including safety and liveness requirements. Formal methods provide rich specification languages, such as temporal logics, to express such requirements, and algorithms, such as model checkers, to verify the satisfaction of the specifications. Despite its usefulness from a conceptual point of view, formal verification of Petri nets is in general a hard problem, mainly because most "realistic" Petri nets have very large the state-spaces.

A general approach when dealing with state explosion problems is to use abstraction techniques for constructing computationally manageable quotients. In the case of discrete PN with a time interpretation, the construction of a state space abstraction using the concept of "state-classes" have been introduced in [10, 11]. This technique allows to represent the state graph of a timed PN, while preserving marking and complete traces, and therefore it is suitable for reachability analysis and model checking. Another way of tackling the state explosion problem in discrete systems is the approximation by fluidification [12, 13], which leads to a so-called fluid Petri net. This approximation technique is not new and has been applied in many other discrete formalisms such as queuing networks leading to fluid queuing networks [14, 15, 16]. The fluid model has the advantage that many design and analysis techniques based on *integer linear programming problems* correspond to *linear programming problems*, hence they have polynomial time complexity.

Although fluidification proves its advantage by reducing the computation complexity in problems of practical interest [13, 17], even basic properties of continuous timed net models are undecidable [18]. For timed continuous PN, two firing semantics are mainly encountered in literature: *finite* and *infinite server semantics* [12, 13, 19]. The problem of formal verification of fluid PN was considered in the case of finite server semantics [20]. However, it was recently proven that continuous Petri nets systems with infinite server semantics provide, in general, a better approximation of the underlying discrete net [21]. Since a *ContPN* is a subclass of hybrid systems, it can be modelled as a *discrete hybrid automaton* [22]. However, for better exploiting the structural properties of Petri nets (e.g., continuous vector field at the region borders), we limit our attention only to *ContPN* systems instead of considering general hybrid systems. On the other hand, this implies that the obtained results cannot be extended to any hybrid system.

In this paper we develop an approach for performing formal analysis of timed continuous Petri nets with infinite server semantics. As far as we know, the proposed framework is the first one of this kind. More specifically, we provide algorithmic solutions to two general problems: (1) given an initial set of markings, construct a set of unreachable markings; and (2) given a Linear Temporal Logic (LTL) formula over a set of linear predicates in the marking space, construct a set of initial states such that all trajectories originating there satisfy the LTL specification. This paper extends the results from [1] by providing more technical details, a description of the software implementation, and by applying the developed framework to two very challenging Petri net problems: (1) finding timed implicit arcs, and (2) finding initial markings from where the system reaches deadlock.

Technically, the approach presented in this paper is based on the *Piecewise Affine (PWA) representation* of the dynamics of a *deterministically* timed continuous Petri net with infinite server semantics. As part of the solution, we develop an iterative procedure for analyzing PWA systems, which starts by embedding a PWA system into an infinite transition system, and continues by constructing a finite quotient of this transition system. Then, the obtained quotient is iteratively analyzed and refined until a termination condition is encountered. The formal analysis problem we solve for PWA systems relates to [23], where a richer class of hybrid affine systems is analyzed only against reachability properties. Temporal logic analysis problems for PWA systems are also studied in [24] (in continuous time) and [25] (in discrete time). However, in these works, the refinement is based on an (approximate) implementation of the *bisimulation* algorithm, and on the computation of the Pre image of sets through the vector fields of the system. In this paper, the iterative refinement is achieved through *simple cuts*, and resembles our previous work [26] for multi-affine systems and rectangular sets. Thus, our approach can be regarded as incorporating techniques from abstraction and analysis of continuous systems into tools for analyzing Petri nets. The framework described in this paper was implemented in Matlab as a freely downloadable software tool [27].

Formal analysis of hybrid systems using model checking techniques has been studied by the computer science community using the so called *linear hybrid automata* (LHA) [28, 29]. This is an autonomous non-deterministic model mainly used for simulation and verification of hybrid systems. Recently some results have been obtained related to the equivalence between PWA and LHA systems [30]. It is shown that every PWA system can be written as a LHA system and the obtained LHA system can generate all trajectories of the PWA system. Unfortunately more trajectories are obtained making the solution of the formal analysis based on the equivalent LHA an over-approximation of the solution obtained using the original PWA formulation.

The remainder of the paper is organized as follows. After some preliminaries concerning Petri nets, transition systems and temporal logic (Section 2), Section 3 formulates the addressed problems and outlines the main ideas for solving them. Section 4 translates the initial problems to PWA formulations, while Section 5 deals with formal analysis of PWA systems such that the proposed problems are solved. Some aspects regarding the implementation and complexity are given in Section 6. Section 7 applies the developed approaches to two important problems concerning timed continuous Petri nets, and Section 8 formulates some concluding remarks.

2 Preliminaries

2.1 Timed Continuous Petri Nets

Definition 2.1 *[Continuous Petri Net System] A Continuous Petri Net (*ContPN*) system is a pair $\langle \mathcal{N}, \boldsymbol{m}_0 \rangle$, where $\mathcal{N} = \langle P, T, \boldsymbol{Pre}, \boldsymbol{Post} \rangle$ is a* net structure *and $\boldsymbol{m}_0 \in \mathbb{R}_{\geq 0}^{|P|}$ is the* initial marking. *P is the set of places, T is the set of transitions, and $\boldsymbol{Pre}, \boldsymbol{Post} \in \mathbb{N}^{|P| \times |T|}$ are the pre and post incidence matrices, respectively.* ■

Let p_i, $i = 1, \ldots, |P|$ and t_j, $i = 1, \ldots, |T|$ denote the places and transitions. For a place $p_i \in P$ and a transition $t_j \in T$, Pre_{ij} and $Post_{ij}$ represent the weights of the arcs from p_i to t_j and from t_j to p_i, respectively. Each place p_i has a token load denoted by $m_i \in \mathbb{R}_{\geq 0}$. The vector of all token loads is called *marking*, and is denoted by $\boldsymbol{m} \in \mathbb{R}_{\geq 0}^{|P|}$. The *preset* and *postset* of a place or transition $x \in P \cup T$ are denoted by ${}^\bullet x$ and $x^\bullet$, and represent the input and output transitions and places of x, respectively. More specifically, if $t_j \in T$, ${}^\bullet t_j = \{p_i \in P | Pre_{ij} > 0\}$ and ${t_j}^\bullet = \{p_i \in P | Post_{ij} > 0\}$. Similarly, if $p_i \in P$, ${}^\bullet p_i = \{t_j \in T | Post_{ij} > 0\}$ and ${p_i}^\bullet = \{t_j \in T | Pre_{ij} > 0\}$.

It is important to note that the marking of a *ContPN* can take real non-negative values, while in discrete Petri Nets (PN) only natural values are possible. In fact, this is the only difference between a continuous and a discrete PN.

Example 2.2 *Let us consider the* ContPN *in Fig. 1. For this net, $P = \{p_1, p_2, p_3, p_4\}$, $T = \{t_1, t_2, t_3\}$,*

$$\boldsymbol{Pre} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 2 & 1 & 0 \end{bmatrix}, \quad \boldsymbol{Post} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 3 \end{bmatrix}.$$

$Pre_{41} = 2$ means that there exists an arc from p_4 to t_1 of weight 2*. $Post_{43} = $* 3 *signifies that there exists an arc from t_3 to p_4 of weight* 3*. ${}^\bullet p_1 = \{t_2, t_3\}$ and ${p_1}^\bullet = \{t_1\}$.* ■

A transition $t_j \in T$ is *enabled* at $\boldsymbol{m}$ if and only if $\forall p_i \in {}^\bullet t_j$, $m_i > 0$. Its enabling degree is

$$enab(t_j, \boldsymbol{m}) = \min_{p_i \in {}^\bullet t_j} \left\{ \frac{m_i}{Pre_{ij}} \right\}, \tag{1}$$

which represents the maximum amount in which t_j can fire. An enabled transition t_j can fire in any real amount $0 \leq \alpha \leq enab(t_j, \boldsymbol{m})$, leading to a new marking $\boldsymbol{m'} = \boldsymbol{m} + \alpha \boldsymbol{C}_{\cdot j}$, where $\boldsymbol{C} = \boldsymbol{Post} - \boldsymbol{Pre}$ is the token-flow matrix and $\boldsymbol{C}_{\cdot j}$ is its j^{th} column. If $\boldsymbol{m}$ is reachable from $\boldsymbol{m}_0$ through a finite sequence σ, a *state (or fundamental) equation* can be written:

$$\boldsymbol{m} = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma}, \tag{2}$$

where $\boldsymbol{\sigma} \in \mathbb{R}_{\geq 0}^{|T|}$ is the firing count vector, *i.e.*, σ_j is the cumulative amount of firing of t_j in the sequence σ.

Definition 2.3 *The set of all reachable markings from $\boldsymbol{m}_0$ is called* reachability set*, and it is denoted by $\mathcal{R}(\mathcal{N}, \boldsymbol{m}_0)$. For simplicity, notation $\mathcal{R}$ will be used when there is no confusion on $\mathcal{N}$ and $\boldsymbol{m}_0$. In the case of a* ContPN *system, $\mathcal{R}$ is a convex set [13].* ■

Definition 2.3 is trivially extended when a whole set Π_0 of initial markings is specified, by $\mathcal{R}(\mathcal{N}, \Pi_0) = \bigcup_{\boldsymbol{m}_0 \in \Pi_0} \mathcal{R}(\mathcal{N}, \boldsymbol{m}_0)$.

A *ContPN* is *bounded* when every place is bounded, *i.e.*, $\forall p_i \in P$, $\exists b_i \in \mathbb{R}_{\geq 0}$ with $m_i \leq b_i$ at any reachable marking $\boldsymbol{m}$. Right and left non negative annullers of the token flow matrix $\boldsymbol{C}$ are called T- and P-semiflows, respectively. If non negativity is not required, the annullers are called T- and P-flows.

If a timed interpretation is included in the model, the fundamental equation depends on time: $\boldsymbol{m}(\tau) = \boldsymbol{m_0} + \boldsymbol{C} \cdot \boldsymbol{\sigma}(\tau)$, which, through time differentiation, becomes $\dot{\boldsymbol{m}}(\tau) = \boldsymbol{C} \cdot \dot{\boldsymbol{\sigma}}(\tau)$. The derivative of the firing sequence $\boldsymbol{f}(\tau) = \dot{\boldsymbol{\sigma}}(\tau)$ is called the *(firing) flow*, and leads to the following equation for the dynamics of the *ContPN*:

$$\dot{\boldsymbol{m}}(\tau) = \boldsymbol{C}\boldsymbol{f}(\tau). \tag{3}$$

Definition 2.4 *[Timed Continuous Petri Net System] A Timed Continuous Petri Net system is a pair $\langle \mathcal{N}, \boldsymbol{\lambda}, \boldsymbol{m}_0 \rangle$, where $\langle \mathcal{N}, \boldsymbol{m}_0 \rangle$ is a* continuous Petri net system *and $\boldsymbol{\lambda} \in \mathbb{R}^{|T|}_{\geq 0}$ is the* firing rate vector*.* ■

From now on, we will refer only to timed continuous Petri Nets, and, with a slight abuse of notation, we will denote them by *ContPN*.

This paper deals with *infinite server semantics*, which was shown to provide a good approximation of the underlying discrete net for a broad class of systems [21]. Under this semantics, the flow of transition t_j is given by:

$$f_j(\tau) = \lambda_j \; enab(t_j, \boldsymbol{m}(\tau)), \tag{4}$$

where λ_j is its firing rate and the enabling function is given by (1). From (3), (4), and (1), it can be easily seen that a *ContPN* system with infinite server semantics is a piecewise linear system with polyhedral regions and everywhere continuous vector field. In other words, the dynamics of the markings are given by:

$$\dot{\boldsymbol{m}}(\tau) = \boldsymbol{A}_i \boldsymbol{m}(\tau), \; \boldsymbol{m} \in \mathcal{R}_i, \; i \in I, \tag{5}$$

where $\boldsymbol{A}_i \in \mathbb{R}^{|P| \times |P|}$, $\mathcal{R}_i$ is a polyhedral set, and I is a set of labels for the modes of the piecewise linear system (see [31] for more details).

Example 2.5 *Consider the* ContPN *in Fig. 1 with $\boldsymbol{m}_0 = \boldsymbol{m}(0) = [1, 5.25, 0.75, 5.5]^T$ and $\boldsymbol{\lambda} = [1, \ldots, 1]^T$. Transition t_1 has two input places, p_1 and p_4 and t_2 has also two input places: p_2 and p_4. According to* (4) *and* (1)*, the flows through the transitions of the system are given by*

$$f_1(\tau) = \begin{cases} \lambda_1 \cdot \frac{m_1(\tau)}{2}, & \text{if } \frac{m_1(\tau)}{2} \leq \frac{m_4(\tau)}{2} \\ \lambda_1 \cdot \frac{m_4(\tau)}{2}, & \text{if } \frac{m_1(\tau)}{2} \geq \frac{m_4(\tau)}{2} \end{cases},$$

$$f_2(\tau) = \begin{cases} \lambda_2 \cdot m_2(\tau), & \text{if } m_2(\tau) \leq m_4(\tau) \\ \lambda_2 \cdot m_4(\tau), & \text{if } m_2(\tau) \geq m_4(\tau) \end{cases},$$

and $f_3(\tau) = \lambda_3 \cdot m_3(\tau)$.

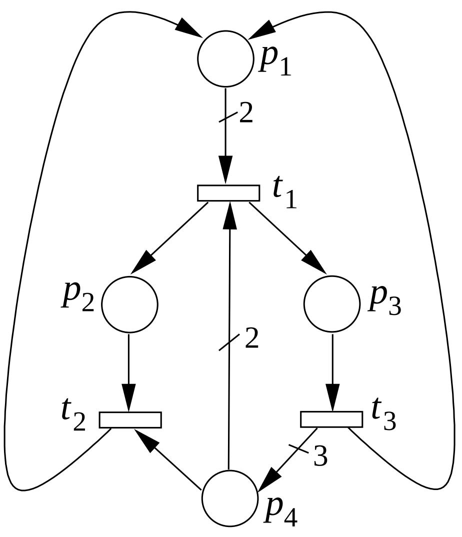


Figure 1: A (timed) continuous Petri Net *ContPN*.

This net has two token conservation laws (P-semiflows):

$$\begin{array}{l} m_1(\tau) + m_2(\tau) + m_3(\tau) = m_1(0) + m_2(0) + m_3(0) = 7 \\ m_1(\tau) + 4m_3(\tau) + m_4(\tau) = m_1(0) + 4m_3(0) + m_4(0) = 10 \end{array} \tag{6}$$

Since each place appears in at least one P-semiflow, the net system is bounded. Substituting (4) *into* (3) *leads to the piecewise linear representation* (5). *For example, one of the modes in this representation of the system is described by* $\mathcal{R}_1 : \{\frac{m_1}{2} \leq \frac{m_4}{2},\ m_2 \leq m_4\}$, *and*

$$A_1 = \begin{bmatrix} -1 & 1 & 1 & 0 \\ 0.5 & -1 & 0 & 0 \\ 0.5 & 0 & -1 & 0 \\ -1 & -1 & 3 & 0 \end{bmatrix}$$

■

The number of regions $\mathcal{R}_i$ of a *ContPN* system is upper bounded by $\prod_{t_i \in T} |^\bullet t_i|$, and in the case of a bounded net system they are closed polytopes. For a given initial marking, some places can be implicit [17] (given a *ContPN* system $\langle \mathcal{N}, \boldsymbol{m}_0 \rangle$, $p_j \in {}^\bullet t_i$ is implicit if and only if $\nexists \boldsymbol{m} \in \mathcal{R}(\mathcal{N}, \boldsymbol{m}_0)$ such that $\frac{m_j}{Pre_{ji}} < \frac{m_k}{Pre_{ki}}\ \forall p_k \in {}^\bullet t_i \setminus \{p_j\}$). For example, in the *ContPN* in Fig. 1 with $\boldsymbol{m}_0 = [15, 3, 1, 0]^T$, p_2 is an implicit place. Therefore, region $\mathcal{R}_1 = \{\frac{m_1}{2} \leq \frac{m_4}{2}, m_2 \leq m_4\}$ is included in $\mathcal{R}_3 = \{\frac{m_1}{2} \leq \frac{m_4}{2}, m_4 \leq m_2\}$ since $m_2 \leq m_4$ is satisfied only as equality. In fact, $\mathcal{R}_1$ is a frontier of $\mathcal{R}_3$. Also, $\mathcal{R}_2 = \{\frac{m_4}{2} \leq \frac{m_1}{2}, m_2 \leq m_4\}$ is included in $\mathcal{R}_4 = \{\frac{m_4}{2} \leq \frac{m_1}{2}, m_4 \leq m_2\}$ for the same reason. In our approach we consider only the regions that are full-dimensional polytopes in $\mathbb{R}^{rank(\boldsymbol{C})}$. Note that this is not a limitation, since at the common border of two regions, the corresponding linear systems provide the same vector field according to (4) and (1).

2.2 Transition systems and temporal logic

Definition 2.6 *[Transition system] A transition system is a tuple $\mathcal{T} = (Q, Q_0, \rightarrow, \Pi, \vDash)$, where Q is a (possibly infinite) set of states, $Q_0 \subseteq Q$ is a set of initial states, $\rightarrow \subseteq Q \times Q$ is a transition relation, Π is a finite set of atomic propositions, and $\vDash \subseteq Q \times \Pi$ is a satisfaction relation.* ■

For an arbitrary proposition $\pi \in \Pi$, we define $[\![\pi]\!] = \{q \in Q | q \vDash \pi\}$ as the set of all states satisfying it. Conversely, for an arbitrary state $q \in Q$, let $\Pi_q = \{\pi \in \Pi \,|\, q \vDash \pi\}$, $\Pi_q \in 2^{\Pi}$, denote the set of all atomic propositions satisfied at q. An initialized *trajectory* or *run* of $\mathcal{T}$ starting from $q \in Q_0$ is an infinite sequence $r = r(1)r(2)r(3)\ldots$ with the property that $r(1) = q$, $r(i) \in Q$, and $(r(i), r(i+1)) \in \rightarrow$, for all $i \geq 1$. A trajectory $r = r(1)r(2)r(3)\ldots$ generates a *word* $w = w(1)w(2)w(3)\ldots$, where $w(i) = \Pi_{r(i)}$. The set of all generated words is called the *language* of $\mathcal{T}$, and is denoted by $L(\mathcal{T})$.

An equivalence relation $\sim \subseteq Q \times Q$ over the state space of $\mathcal{T}$ is *proposition preserving* if for all $q_1, q_2 \in Q$ and all $\pi \in \Pi$, if $q_1 \sim q_2$ and $q_1 \vDash \pi$, then $q_2 \vDash \pi$. A proposition preserving equivalence relation naturally induces a *quotient transition system* $\mathcal{T}/_\sim = (Q/_\sim, Q_0/_\sim, \rightarrow_\sim, \Pi, \vDash_\sim)$. $Q/_\sim$ is the quotient space (the set of all equivalence classes), and the set of initial states is $Q_0/_\sim = \{P \in Q/_\sim \,|\, Q_0 \cap h_\sim(P) \neq \emptyset\}$, where $h_\sim : Q/_\sim \rightarrow 2^Q$ is the concretization map corresponding to $\sim$. The transition relation $\rightarrow_\sim$ is defined as follows: for $P_1, P_2 \in Q/_\sim$, $P_1 \rightarrow_\sim P_2$ if and only if there exist $q_1 \in h(P_1)$ and $q_2 \in h(P_2)$ such that $q_1 \rightarrow q_2$. The satisfaction relation is defined as follows: for $P \in Q/_\sim$, we have $P \vDash_\sim \pi$ if and only if there exists $q \in h(P)$ such that $q \vDash \pi$. It is easy to see that

$$L(\mathcal{T}) \subseteq L(\mathcal{T}/_\sim), \tag{7}$$

The quotient transition system $\mathcal{T}/_\sim$ is said to *simulate* the original system $\mathcal{T}$, which is written as $\mathcal{T}/_\sim \geq \mathcal{T}$.

In this work we consider system specifications given as formulas of a fragment of Linear Temporal Logic (LTL) [32], called LTL$_{-\mathrm{X}}$, which we will simply denote by LTL throughout the paper. A formal definition for the syntax and semantics of LTL formulas is beyond the scope of this paper. Informally, LTL formulas are recursively defined over a set of atomic propositions Π, by using the standard boolean operators and a set of temporal operators. The boolean operators are: $\neg$ (negation), $\vee$ (disjunction), $\wedge$ (conjunction), and the temporal operators that we use are: $\mathcal{U}$ ("until"), $\Box$ ("always"), $\Diamond$ ("eventually"). LTL formulas are interpreted over infinite words over the set 2^{Π}, such as those generated by the transition system $\mathcal{T}$ from Definition 2.6. If ϕ_1 and ϕ_2 are two LTL formulas over Π and w is a word produced by $\mathcal{T}$, then formula $\phi_1 \mathcal{U} \phi_2$ means that (over the word w) ϕ_2 will eventually become true, and ϕ_1 is true until this happens. Formula $\Diamond\phi_1$ means that ϕ_1 becomes eventually true, whereas $\Box\phi_1$ indicates that ϕ_1 is true at all positions of w. More expressiveness can be achieved by combining the mentioned operators.

Classical LTL allows for an additional temporal operator called "next". We do not allow for the "next" operator because, as shown in [33], it is meaningless when abstracting a continuous system to a finite discrete one, as considered in this paper. On the other hand, LTL$_{-\mathrm{X}}$ (LTL without the "next" operator) cannot distinguish between words with different numbers of finitely many consecutive repetitions of a symbol, *e.g.* $\pi_1\pi_2\pi_2\pi_3\ldots$ satisfies exactly the same formulas as $\pi_1\pi_2\pi_3 \ldots$.

Given a transition system $\mathcal{T}$ and an LTL formula ϕ over its set of propositions, checking whether $L(\mathcal{T})$ satisfies ϕ is called model checking. For finite transition systems, there exist off-the-shelf tools for model checking [34]. Note that if a proposition-preserving quotient $\mathcal{T}/_\sim$ satisfies ϕ, then by the language inclusion (7), the initial transition system $\mathcal{T}$ also satisfies the formula.

3 Problem formulation and approach

Consider a *ContPN* system and let Π be a user-defined set of strict linear inequalities over its marking $\boldsymbol{m}$, which will be simply called predicates. Formally, each element of Π has the form $\pi_i = \{\boldsymbol{m} \in \mathbb{R}^{|P|}_{\geq 0} | c_i^T \boldsymbol{m} + d_i < 0\}$, with $c_i \in \mathbb{R}^{|P|}$, $d_i \in \mathbb{R}$, $\forall i = 1, \ldots, |\Pi|$. Without restricting the generality of the problem, we assume that the set Π also includes all the affine functions in $\boldsymbol{m}$ necessary to define the full-dimensional regions $\mathcal{R}_i$.

Remark 3.1 *For technical reasons that go beyond the scope of this paper, we limit the specifications in the set of predicates* Π *to strict linear inequalities. Also, we will regard the negation of any predicate from set* Π *as meaning multiplication with* -1 *of the corresponding inequality. This means that we only include open halfspaces and full-dimensional polytopes. However, this assumption does not seem restrictive from an application point of view. If the predicates in* Π *model sensor information, it is unrealistic to check for the attainment of a specific value due to sensor noise. Moreover, if a specific value is of interest, it can be included in the interior of a polytope defined by other predicates. Our formalism ignores markings of* ContPN *that lie on the hyperplanes obtained by setting to zero the linear inequalities from* π_i, $i = 1, \ldots, |\Pi|$. *This fact leads to a slightly increased conservativeness when solving the problems formulated in this section only in the case when there are trajectories "disappearing" inside such hyperplanes. Reducing this conservativeness would require a much more complex and computationally slow embedding in Section 5, the gain being noticeable only in the very particular (and unrealistic) mentioned situations.*

Problem 3.2 (Construction of safe sets) *Given a set of initial markings defined as the conjunction of predicates from a set* $\Pi_0 \subseteq \Pi$, *find a subset of the reachability set that cannot be reached by trajectories of* ContPN *originating in the initial set.*

Problem 3.3 (Initial set satisfying LTL specification) *Given an LTL formula over* Π, *find a set of initial markings of* ContPN *from where all possible trajectories satisfy the formula.*

To illustrate the importance of the formulated problems, in Section 7 we will use the algorithm solving Problems 3.2 and 3.3 for providing solutions to two open questions in the *ContPN* area, namely finding timed implicit arcs and finding initial marking from where the system reaches a deadlock state. The algorithm for solving Problems 3.2 and 3.3 was implemented in Matlab as a freely downloadable software tool [27].

To fully specify Problems 3.2 and 3.3, we need to define the semantics of an LTL formula over a continuous trajectory. A formal definition is given in

Section 5 through an embedding into a transition system. However, an informal and intuitive definition can be given as follows: an evolving trajectory produces the set of predicates from Π that are true at the current marking, with no finite consecutive repetitions of the set of predicates, and with infinitely many repetitions of the set of predicates satisfied by a region that is an invariant for the trajectory. Note that this is consistent with our choice of LTL without the "next" operator. For example, in Fig. 2, if the regions $\mathcal{R}_i$ satisfy the sets of predicates $\Pi_i \subseteq \Pi$, $i = 1, \ldots, 4$, respectively, then the shown trajectory, starting from $\boldsymbol{m}_0$ and converging to $\boldsymbol{m}_f$, generates the word $\Pi_1\Pi_2\Pi_4\Pi_3\Pi_3\ldots$.

Our approach to solving Problems 3.2 and 3.3 consists of two main steps. The first step is required if the *ContPN* system has at least one P-flow, i.e., at least one left annuller of the incidence matrix. In this step, we compute a set of linearly independent P-flows of *ContPN* and then construct a reduced representation of the *ContPN* in the form of a piecewise affine system (PWA), as in Section 4. Second, we perform formal analysis of the corresponding PWA system based on discrete abstractions (finite quotients) and refinement, and by employing convexity properties of affine systems in full-dimensional polytopes [35, 36, 33], as shown in Section 5.

4 Derivation of the PWA for a *ContPN*

The token conservation laws (P-flows) introduce a number of $|P| - rank(\boldsymbol{C})$ dependent variables [31]. By removing these variables, a reduced system with a piecewise affine behavior is obtained.

Let $r = rank(\boldsymbol{C})$ and let $\boldsymbol{B}_y \in \mathbb{R}^{(|P|-r)\times(|P|)}$ be a matrix whose rows form a basis of P-flows, *i.e.*, $\boldsymbol{B}_y \cdot \boldsymbol{C} = \boldsymbol{B}_y \cdot \boldsymbol{A}_i = \boldsymbol{0}$. Since $\boldsymbol{B}_y$ is a basis, $rank(\boldsymbol{B}_y) = |P| - r$, and $\boldsymbol{B}_y$ can be written in the form:

$$\boldsymbol{B}_y = \left[\boldsymbol{R} \mid \boldsymbol{I}_{|P|-r}\right], \tag{8}$$

where $\boldsymbol{R} \in \mathbb{R}^{(|P|-r)\times r}$. By pre-multiplying the state equation (2) by $\boldsymbol{B}_y$, we obtain:

$$\boldsymbol{B}_y \cdot \boldsymbol{m} = \boldsymbol{B}_y \cdot \boldsymbol{m}_0 = \boldsymbol{b} \tag{9}$$

By considering $\boldsymbol{m} = [\boldsymbol{m}' \ \boldsymbol{m}'']^T$, with $dim(\boldsymbol{m}') = r$ and $dim(\boldsymbol{m}'') = |P| - r$, from (9) and (8) we obtain:

$$\boldsymbol{m}'' = \boldsymbol{b} - \boldsymbol{R}\boldsymbol{m}' \tag{10}$$

Let $\boldsymbol{Q} = \left[\begin{array}{c|c} \boldsymbol{I}_r & \boldsymbol{0} \\ \boldsymbol{R} & \boldsymbol{I}_{|P|-r} \end{array}\right]$

Equation (5) can be rewritten as:

$$\left[\begin{array}{c} \dot{\boldsymbol{m}}' \\ \dot{\boldsymbol{m}}'' \end{array}\right] = \left[\begin{array}{c|c} \boldsymbol{A}_i^{11} & \boldsymbol{A}_i^{12} \\ \boldsymbol{A}_i^{21} & \boldsymbol{A}_i^{22} \end{array}\right] \left[\begin{array}{c} \boldsymbol{m}' \\ \boldsymbol{m}'' \end{array}\right], \tag{11}$$

such that $rank\left(\left[\boldsymbol{A}_i^{11} \ \boldsymbol{A}_i^{12}\right]\right) = r$. By pre-multiplying (11) by $\boldsymbol{Q}$, we obtain:

$$\dot{\boldsymbol{m}}' = \boldsymbol{A}_i^{11}\boldsymbol{m}' + \boldsymbol{A}_i^{12}\boldsymbol{m}'', \tag{12}$$

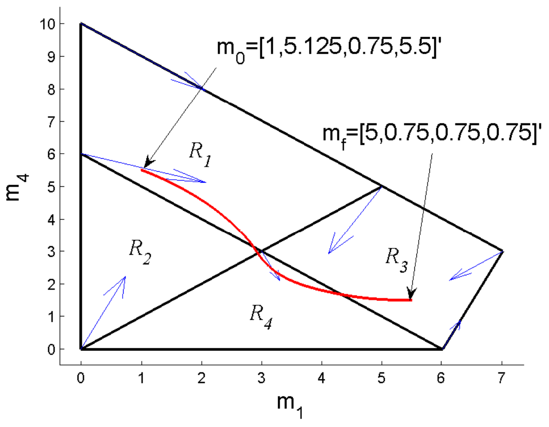


Figure 2: Reachability set of the *ContPN* in Fig. 1 with $\boldsymbol{m}_0 = [1, 5.25, 0.75, 5.5]^T$ and $\boldsymbol{\lambda} = \mathbf{1}$.

and according to (10):

$$\dot{\boldsymbol{m}}' = \boldsymbol{A}_i^{11}\boldsymbol{m}' + \boldsymbol{A}_i^{12}(\boldsymbol{b} - \boldsymbol{R}\boldsymbol{m}') = \begin{bmatrix} \boldsymbol{A}_i^{11} \\ -\boldsymbol{A}_i^{12}\boldsymbol{R} \end{bmatrix} \boldsymbol{m}' + \boldsymbol{A}_i^{12}\boldsymbol{b} \tag{13}$$

Therefore, the piecewise linear dynamics (5) are equivalent with the piecewise affine dynamics (13) in a reduced dimension, plus some equalities (10). For simplicity, we use a slight abuse of notation and denote the obtained PWA system by:

$$\dot{\boldsymbol{m}}(\tau) = \boldsymbol{A}_i\boldsymbol{m}(\tau) + \boldsymbol{b}_i,\ \boldsymbol{m} \in \mathcal{R}_i,\ i \in I, \tag{14}$$

with the implicit understanding that the state (marking) $\boldsymbol{m}$ has already been reduced and $\boldsymbol{A}_i$'s are the corresponding new system matrices. The regions $\mathcal{R}_i$ and the set I are the same as in (5), with the observation that $\mathcal{R}_i$ are now expressed using a smaller number of variables. The linear inequalities from the set of specification predicates Π are also transformed accordingly, while the predicate symbols remain the same. It is easy to see that, as in the piecewise linear representation, the vector field of (14) is continuous everywhere.

The trajectories of the PWA system (14) produce words according to the informal definition from Section 3. In the remainder of the paper, when we refer to Problems 3.2 and 3.3, we assume that they are formulated for the PWA representation (14) of the *ContPN* system.

Example 4.1 *The net in Fig. 1 has two token conservation laws (P-semiflows) given in (6), thus two variables are redundant. If m_1 and m_4 are chosen as free variables, then a planar PWA representation of the form (14) can be constructed. The reachability set in the reduced (m_1, m_4) - plane is sketched in Fig. 2. The dynamics corresponding to region $\mathcal{R}_1$ (defined in Example 2.5) are given by:*

$$\begin{bmatrix} \dot{m}_1 \\ \dot{m}_4 \end{bmatrix} = \begin{bmatrix} -2 & 0 \\ -1 & -1 \end{bmatrix} \begin{bmatrix} m_1(\tau) \\ m_4(\tau) \end{bmatrix} + \begin{bmatrix} 7 \\ 3 \end{bmatrix} \tag{15}$$

■

5 Formal analysis of PWA representations of *ContPN*

Assume there are M feasible sets of the form $\bigwedge_{i=1}^{|\Pi|}((-1)^{j_i}\pi_i)$, where $j_1, \dots j_{|\Pi|} \in \{0,1\}$. Since the affine functions necessary to define the regions $\mathcal{R}_k$ are among π_i, $i = 1, \dots, |\Pi|$, each of these sets is a full dimensional polytope included in the reachability set of the PWA system, and it corresponds to a feasible combination of predicates from Π inside each region $\mathcal{R}_k$. We denote these sets by $\mathcal{P}_1$,$\mathcal{P}_2$,...,$\mathcal{P}_M$.

Definition 5.1 *For the PWA system (14) and the set of predicates* Π*, the (infinite) embedding transition system is defined as*

$$\mathcal{T}_{emb} = \{Q_{emb}, Q_{emb_0}, \rightarrow_{emb}, \Pi_{emb}, \vDash_{emb}\}, \tag{16}$$

where $Q_{emb} = \bigcup_{i=1}^{M} \mathcal{P}_i$*,* $Q_{emb_0} = Q_{emb}$*, and* $\Pi_{emb} = \Pi$*. The satisfaction relation is obviously defined as* $\boldsymbol{m} \vDash_{emb} \pi_i$ *if and only if* $\boldsymbol{m}$ *verifies the strict linear inequality* π_i*. The transition relation is defined according to the following two rules: (1)* $(\boldsymbol{m'}, \boldsymbol{m''}) \in\rightarrow_{emb}$ *with* $\boldsymbol{m'} \in \mathcal{P}_i$*,* $\boldsymbol{m''} \in \mathcal{P}_j$*,* $\mathcal{P}_i \neq \mathcal{P}_j$ *if and only if the polytopes* $\mathcal{P}_i$ *and* $\mathcal{P}_j$ *are adjacent*[1] *and there exists a trajectory* $\boldsymbol{m}(\tau)|_{[0,T]}$ *of (14)* $(0 < T < \infty)$ *such that* $\boldsymbol{m}(0) = \boldsymbol{m'}$*,* $\boldsymbol{m}(T) = \boldsymbol{m''}$*, and* $\boldsymbol{m}(\tau)|_{[0,T]}$ *is included in the closure of* $\mathcal{P}_i \bigcup \mathcal{P}_j$*; (2)* $(\boldsymbol{m'}, \boldsymbol{m''}) \in\rightarrow_{emb}$ *with* $\boldsymbol{m'}, \boldsymbol{m''} \in \mathcal{P}_i$ *if and only if there exists a trajectory* $\boldsymbol{m}(\tau)|_{[0,\infty)}$ *of (14) such that* $\boldsymbol{m'} = \boldsymbol{m}(0)$ *and* $\boldsymbol{m''} = \lim_{\tau\rightarrow\infty} \boldsymbol{m}(\tau)$*.* ■

Note that the trajectories of $\mathcal{T}_{emb}$ satisfy the informal definition from Section 3. Formally, we have:

Definition 5.2 *The language* $L(\mathcal{T}_{emb})$ *of the transition system (16) is defined as the set of all words produced by trajectories of the PWA system (14) representing the* ContPN *system.* ■

The embedding transition system (16) has infinitely many states and cannot be model checked. To provide (conservative) solutions to Problems 3.2 and 3.3, we propose an iterative procedure that produces a finite quotient and then refines it if necessary. At each step, the language of the obtained quotient includes the language of $\mathcal{T}_{emb}$.

5.1 Construction and analysis of the quotients

Let $\sim$ be a polytopal proposition-preserving equivalence relation over Q_{emb} that does not violate the polytopes $\mathcal{P}_i$, $i = 1, \dots, M$. In other words, each equivalence class in $Q_{emb}/_\sim$ is a polytope included in exactly one of $\mathcal{P}_i$, $i = 1, \dots, M$. According to Definition 5.1, to compute the transitions of $\mathcal{T}_{emb}/_\sim$, we need to solve the following two problems: (i) for all pairs of equivalence classes corresponding to adjacent polytopes, decide if there is a trajectory of $\mathcal{T}_{emb}$ penetrating from one to another, and (ii) for all equivalence classes, decide if there exists a trajectory of $\mathcal{T}_{emb}$ for which the corresponding polytope is an invariant.

[1]Throughout the paper, we call two full dimensional polytopes in $\mathbb{R}^N$ adjacent if their closures share a facet that is a full dimensional polytope in $\mathbb{R}^{N-1}$.

For both problems (i) and (ii) above, we propose to use the computational framework developed in [36]. In [36], it is shown that an affine system has a trajectory contained in a full dimensional open polytope for all times if and only if the affine system has an equilibrium inside the polytope. Therefore, solving problem (ii) in a polytopal equivalence class reduces to checking the non-emptiness of the polyhedral set given by the equations of the polytope plus the equation setting the corresponding vector field to zero. In addition, in [36], it is shown that, given two adjacent polytopes, there exists a trajectory penetrating from one to another in finite time if and only if there exists a vertex on the common facet at which the projection of the vector field on the outer normal of the facet pointing from the first to the latter is strictly positive. Recall that the vector field of our system is continuous everywhere, so the vector fields of two affine systems on adjacent polytopes agree on the common facet. In conclusion, solving both problems (i) and (ii) reduces to checking non-emptiness of polyhedral sets, for which there exist several powerful tools [37].

Having a finite quotient $\mathcal{T}_{emb}/_\sim$, we can provide a (conservative) solution to Problem 3.2 as follows. First, we define the set of initial states $Q_{emb_0}/_\sim$ as the set of states of $\mathcal{T}_{emb}/_\sim$ that satisfy the predicates from Π_0. Then, by using a simple search on a graph, we find all states Q_{nr} of $\mathcal{T}_{emb}/_\sim$ that are not reachable from $Q_{emb_0}/_\sim$. Enabled by the language inclusion property (7), a solution to Problem 3.2 can be presented in the form $\{\boldsymbol{m} \in h_\sim(q) \,|\, q \in Q_{nr}\}$, where $h_\sim$ is the concretization map defined in Section 2.2.

Problem 3.3 can be solved by model checking $\mathcal{T}_{emb}/_\sim$ from each initial state using an off-the-shelf model checker. If the formula is satisfied at a state q of $\mathcal{T}_{emb}/_\sim$, then, by the language inclusion property (7), all trajectories of $\mathcal{T}_{emb}$ (and of *ContPN*) starting at $h_\sim(q)$ satisfy the formula. If we denote by Q_s the set of all initial states of $\mathcal{T}_{emb}/_\sim$ from which the formula is satisfied, then a set of initial states of $\mathcal{T}_{emb}$ (and of *ContPN*) from which the formula is satisfied is given by $\{\boldsymbol{m} \in h_\sim(q) \,|\, q \in Q_s\}$. In our implementation, we used the LTL planning tool developed in [33] and further improved in [38]. This is computationally more attractive, because our algorithm reuses some computations from the previously considered initial state, instead of completely reiterating a model checker for each new initial state (for details, see [38]).

5.2 Iterative analysis and refinement

We first construct and analyze the "roughest" quotient $\mathcal{T}_{emb}/_\sim$, which corresponds to partitioning with respect to predicates from the initial set Π, and to the equivalence relation defined by $\boldsymbol{m} \sim \boldsymbol{m'}$ if and only if there exists $\mathcal{P}_i$, $i = 1, \ldots, M$, such that $\boldsymbol{m}, \boldsymbol{m'} \in \mathcal{P}_i$. If the safe set is not large enough (or empty) in Problem 3.2, or if the set of initial states is not large enough (or empty) in Problem 3.3, then we construct "finer" quotients.

Example 5.3 *For the* ContPN *from Fig. 1 with $\boldsymbol{m}_0 = [1, 5.25, 0.75, 5.5]^T$ and $\boldsymbol{\lambda} = \boldsymbol{1}$, if the set Π contains only the linear predicates necessary to define the regions $\mathcal{R}_i$, $i = 1, 2, 3, 4$, then the first quotient is shown in Fig. 3. If we are interested in constructing a safe set (Problem 3.2), then it is easy to see that this set is empty. However, this set becomes non-empty through refinement, as shown below.* ■

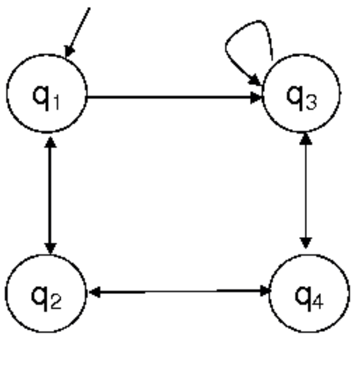


Figure 3: The first quotient of the PWA system from Fig. 2.

We construct finer quotients by adding to the current set Π some new predicates (from a set $\mathcal{H}$), and then recomputing the new feasible polytopes $\mathcal{P}_i$, as explained at the beginning of Section 5. Let us denote by $\mathcal{T}_{emb}/^{\mathcal{H}}_{\sim}$ the quotient obtained as in section 5.1, but corresponding to the set of predicates $\Pi \cup \mathcal{H}$ instead of Π (for simplicity and since no confusion is possible, we use the same notation $\sim$ for the polytopal proposition-preserving equivalence relation, even if it refers to a new partition). It is immediate to observe that $\mathcal{T}_{emb}/_{\sim} \geq \mathcal{T}_{emb}/^{\mathcal{H}}_{\sim}$, simply because the new partition[2] is a subpartition of the one corresponding to $\mathcal{T}_{emb}/_{\sim}$. Therefore, $L(\mathcal{T}_{emb}/_{\sim}) \supseteq L(\mathcal{T}_{emb}/^{\mathcal{H}}_{\sim}) \supseteq L(\mathcal{T}_{emb})$, which means that by using $\mathcal{T}_{emb}/^{\mathcal{H}}_{\sim}$ instead of $\mathcal{T}_{emb}/_{\sim}$ we can obtain less conservative solutions for Problems 3.2 and 3.3.

We start with $\mathcal{H} = \emptyset$, and for each pair of states $q_i, q_j \in Q_{emb}/_{\sim}$, $i < j$, such that $(q_i, q_j) \in \rightarrow_{emb} /_{\sim}$ and $(q_j, q_i) \in \rightarrow_{emb} /_{\sim}$, a new predicate α is added to $\mathcal{H}$. This α denotes the halfspace whose supporting hyperplane has the following property: it cuts the common facet of $h_{\sim}(q_i)$ and $h_{\sim}(q_j)$, such that it separates (on this facet) the points where the vector field projection on the outer normal of the common facet has positive and negative values, respectively. Assumption $i < j$ guarantees that we do not create two propositions for the same pair of states of $\mathcal{T}_{emb}/_{\sim}$. Results from [36] guarantee that such a separation is possible by a single linear predicate. For avoiding some new notations, we do not include the explicit equation of α, and we just mention that its computation requires only matrix multiplications. Our method of adding transitions between states of the discrete quotients implies that α can help in increasing the difference between $L(\mathcal{T}_{emb}/_{\sim})$ and $L(\mathcal{T}_{emb}/^{\mathcal{H}}_{\sim})$, as explained next.

Assume that $h_{\sim}(q_i)$ and $h_{\sim}(q_j)$ are each split by α in two subpolytopes, labelled in $\mathcal{T}_{emb}/^{\mathcal{H}}_{\sim}$ by q_i', q_i'', and q_j', q_j'', respectively. Note that q_i' and q_i'' are adjacent, and each of them is adjacent with only one of q_j', q_j'' (not with both), and vice-versa. Assume that q_i' is adjacent with q_j' and q_i'' is adjacent with q_j''. Then, the above mentioned sign separation provided by α, and the way of adding transitions from section 5.1, guarantee that in $\mathcal{T}_{emb}/^{\mathcal{H}}_{\sim}$ there exist either transitions (q_i', q_j') and (q_j'', q_i''), or transitions (q_j', q_i') and (q_i'', q_j''). Therefore,

[2]The regions induced by the proposition-preserving equivalence relation at each step do not really produce a partition of the state space. Because we consider only strict inequalities, we "lose" points at each step.

we hope that $\mathcal{T}_{emb}/^{\mathcal{H}}_{\sim}$ is less conservative than $\mathcal{T}_{emb}/_{\sim}$ (this fact cannot be guaranteed before testing transitions between q'_i and q''_i, q'_j and q''_j, respectively, and these transitions are not resulting from properties of α, but from tests as in section 5.1).

Note that there are infinitely many choices of predicates α yielding the same separation of the common facet of $h_{\sim}(q_i)$ and $h_{\sim}(q_j)$. Alternatively, one can focus on different splitting methods (instead of linear predicates), as long as the same sign separation is enforced. The motivation for our choice of cutting is three-fold. First, α is very easy to compute, and second, when splitting with some additional linear predicates we use the same algorithms as before, but with a larger input set Π. Third, we have the guarantee that the adjacent polytopes from the partition exactly share facets (as needed for adding transitions in the discrete quotients). The drawback is that α will not split only $h_{\sim}(q_i)$ and $h_{\sim}(q_j)$, but also other polytopes from the partition corresponding to $\mathcal{T}_{emb}/_{\sim}$, and thus the number of states of $\mathcal{T}_{emb}/^{\mathcal{H}}_{\sim}$ can increase significantly. Another way of cutting could involve a triangulation of $h_{\sim}(q_i)$ and $h_{\sim}(q_j)$ that preserves (contains as edge) the sign separating set we want. However, there are no algorithms for performing such a constrained triangulation in space dimensions higher than 2.

Even if the solutions to Problems 3.2 and 3.3 at a given step are not satisfactory, there are two situations when we do not perform refinement: either no more predicates are found, or a certain imposed complexity limit is reached (*e.g.*, a maximum number of states in the discrete quotient is reached). We note that, even if refinement in the current step does not produce a better solution to one of our problems, the refinement in the next step might yield an improvement, as it can be seen in the example concluding this section.

The above ideas are summarized in Algorithm 5.4, which presents the main steps to be taken for solving the discrete versions of Problems 3.2 and 3.3, respectively. By "discrete versions" we understand the problems of finding the discrete sets Q_{nr} and Q_s. Note that notation $\mathcal{T}_{emb}/^{\mathcal{H}}_{\sim}$ does not explicitly appear in Algorithm 5.4, since it just stands for the $\mathcal{T}_{emb}/_{\sim}$ to be constructed at the next iteration.

Algorithm 5.4 (Discrete solutions to Problems 3.2 and 3.3)

```
1: For Problem 3.2 skip lines 13-21, and for Problem 3.3 skip lines 8-12
2: $stop = 0$
3: **while** $\neg stop$ **do**
4:    Find feasible polytopes induced by predicates from $\Pi$ and construct $\mathcal{T}_{emb}/_{\sim}$
5:    **if** $|Q_{emb}/_{\sim}| \geq max\_num\_states$ **then**
6:       $stop = 1$
7:    **end if**
      {For **Problem 3.2**:}
8:    $Q_{emb_0}/_{\sim} = \{q \in Q_{emb}/_{\sim} | q \vDash \pi, \forall \pi \in \Pi_0\}$
9:    $Q_{nr} = \{q \in Q_{emb}/_{\sim} | q \text{ not reachable from } q_0, \forall q_0 \in Q_{emb_0}/_{\sim}\}$
10:   **if** $|Q_{nr} = Q_{emb}/_{\sim} \setminus Q_{emb_0}/_{\sim}$ **then**
11:      $stop = 1$
12:   **end if**
      {For **Problem 3.3**:}
13:   $Q_s = \emptyset$
```

```
14:     for all $q \in Q_{emb}/_\sim$ do
15:         if LTL formula is satisfied by any word of $\mathcal{T}_{emb}/_\sim$ starting from $q$ then
16:             Add $q$ in $Q_s$
17:         end if
18:     end for
19:     if $|Q_s = Q_{emb}/_\sim$ then
20:         $stop = 1$
21:     end if
22:     if stop=1 then
23:         Break “while” loop
24:     end if
        {Refinement}:
25:     $\mathcal{H} = \emptyset$
26:     for all $q_i, q_j \in Q_{emb}/_\sim$ s.t. $(q_i, q_j), (q_j, q_i) \in \rightarrow_{emb} /_\sim$ and $i < j$ do
27:         Construct predicate $\alpha$
28:         $\mathcal{H} = \mathcal{H} \cup \{\alpha\}$
29:     end for
30:     if $\mathcal{H} \setminus \Pi = \emptyset$ then
31:         $stop = 1$
32:     else
33:         $\Pi = \Pi \cup \mathcal{H}$
34:     end if
35: end while
```

Once the sets Q_{nr} and Q_s are found, the solutions to Problems 3.2 and 3.3 are immediate, by using the concretization map $h_\sim$ as shown at the end of subsection 5.1.

Example 5.5 *Consider the* ContPN *system in Fig. 1 with* $\boldsymbol{m}_0 = [1, 5.25, 0.75, 5.5]^T$, $\boldsymbol{\lambda} = \mathbf{1}$ *and the problem of constructing a safe set (Problem 3.2) for the initial region* $\mathcal{R}_1$. *It has been seen in Ex. 5.3 that at the first iteration, no safety regions are obtained (Fig. 4a). Through refinement, three new cutting predicates are obtained (the thin lines from Fig. 4b), and at the second step the transition system will contain 14 discrete states and a safety region depicted in Fig. 4b. At the next iteration, the number of discrete states of the transition system grows to 24, but the safety region is exactly the same as in previous step (Fig. 4c). Refining more, a number of 30 discrete states is obtained and the safety region is increased a little (Fig. 4d). Since no other cutting is possible, the procedure is finished.* ■

6 Software implementation, conservativeness and complexity

In this section we briefly present the software implementation of the proposed techniques, and we discuss the conservativeness and complexity of our approach for solving Problems 3.2 and 3.3.

We implemented our approach as a user friendly software package for formal verification of *ContPN* under Matlab. The tool takes as inputs the *ContPN* (defined by the ***Pre*** and ***Post*** matrices, as in Definition 2.1), the user-defined

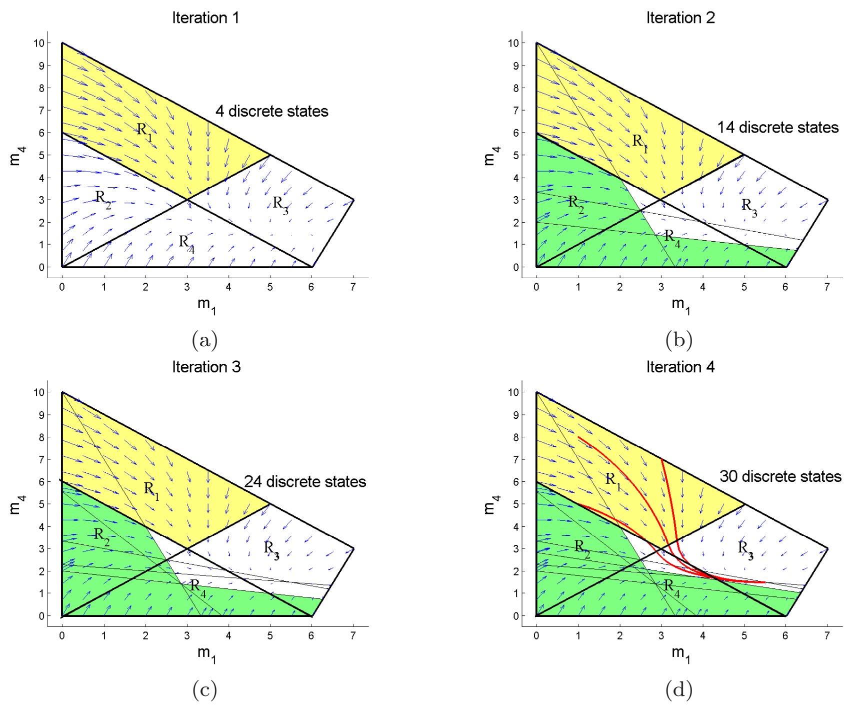


Figure 4: Iterative construction of a safe set for the initial region $\mathcal{R}_1$ shown in yellow. The safe set obtained at each iteration is shown in green.

propositions from set Π, and the set of initial markings defined by Π_0 (for Problem 3.2), respectively the LTL formula (for Problem 3.3). The initial *ContPN* is automatically projected into a PWA representation (together with the defined predicates), as described in Section 4, and then the approach from Section 5 is employed for solving the proposed problems. The software tool is freely downloadable from [27], and it also uses the next mentioned free packages. The first one is a mex-file calling CDD in Matlab [39], and it is used for finding the feasible polytopes induced by predicates from Π and for converting between edge representation and vertex representation of a polytope. For solving Problem 3.3, we embedded the LTL planning tool from [33, 38], which in turn uses LTL2BA [40], a free package that converts an LTL formula into a so-called Büchi automaton.

The approach we developed can be used for analyzing any bounded *ContPN*. Constructing a PWA representation of the *ContPN* does not introduce conservativeness, nor complex computations (as described in Section 4), and therefore our subsequent analysis on conservativeness and complexity will focus on the approach described in Section 5.

The abstraction of the PWA system to a finite quotient is a general source of conservativeness, because we look for whole polytopes instead of investigating distinct markings and trajectories in the reachability set. More specifically, the way we create transitions in the discrete quotient induces conservativeness in the following sense. The existence of a set of states $q, q', q" \in Q_{emb}/_\sim$, such that $(q, q'), (q', q") \in \rightarrow_{emb} /_\sim$ does not necessarily imply that there exists a continuous trajectory starting from a marking in $h_\sim(q)$ and crossing $h_\sim(q')$ and then $h_\sim(q")$. Such a situation can be called lack of transitivity, and it is fundamental in distinguishing between simulation (as in our case) and bisimulation relations among transition systems. The refinement aims to reduce this kind of conservativeness. However, the refinement that we develop is again conservative, because we restrict ourselves to linear cuts, as described in subsection 5.2.

From the complexity point of view, solving Problem 3.2 basically reduces to searches on a graph, where complexity is dependent on the number of nodes (states in our finite quotient). The complexity for solving Problem 3.3 depends on both the size of the LTL formula (chosen by user) and on the size of the finite quotient. We mention that although the upper bound complexity induced by the LTL formula (through the corresponding Büchi automaton) is exponential in the length of the formula, this limit is rarely reached in practice. Therefore, the necessary time for running Algorithm 5.4 strongly depends on the number of regions in our partition, and thus the bottleneck of our approach is resulting from the refinement procedure. As explained in subsection 5.2, each hyperplane we use in a refinement step cuts all the existing regions from the current partition (rather than cutting only those implied in finding the hyperplane), and this fact can significantly increase the number of states of the finite quotient from one refinement step to another. At each iteration of Algorithm 5.4, the resulted partition has at most $2^{|\Pi|}$ regions. However, if $|\Pi|$ is greater that the state space dimension (r) of the PWA system (which is usually the case, due to refinement), there will be much less than $2^{|\Pi|}$ regions. Also, it is worth mentioning that in our implementation we use an iterative procedure to construct the set of feasible polytopes, while at the same time taking into consideration new predicates. This way, especially for a large cardinality of $|\Pi|$, we end up with testing a number of predicate combinations much smaller than $2^{|\Pi|}$. Finally, to give a rough idea about the computation time, we can mention that the computation for any example presented here took less than 10 seconds.

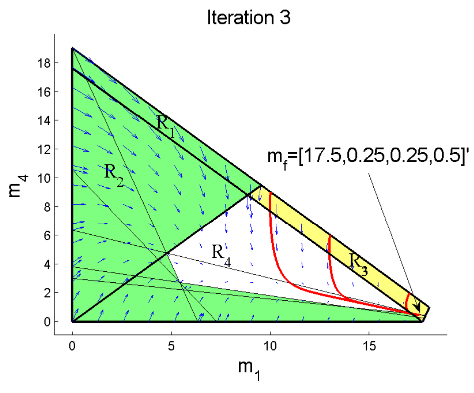
7 Formal analysis of *ContPN* systems

In this section we use the tools developed in this paper to answer some open questions in the analysis of timed continuous Petri Nets.

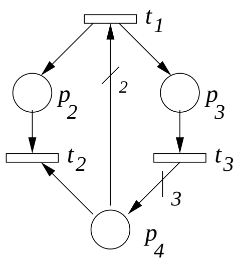
7.1 Timed implicit arc

Definition 7.1 *[Timed Implicit Arc] Given a timed* ContPN *system* $\langle \mathcal{N}, \boldsymbol{\lambda}, \boldsymbol{m}_0 \rangle$, *an input arc* (p_i, t_j) *is called* timed implicit *if and only if* $\frac{m_i(\tau)}{Pre_{ij}} \geq enab(t_j, \boldsymbol{m}(\tau))$ *for all* $\tau \geq 0$. ■

In other words, an input arc (p_i, t_j) is timed implicit if and only if the timed evolution of the *ContPN* system starting from $\boldsymbol{m}_0$ is such that $m_i(\tau)$ never gives the enabling degree of t_j, for all $\tau \geq 0$. In this case, the corresponding linear system is redundant and can be removed, since it will never govern the evolution of the *ContPN*. Moreover, if all output arcs of one place are timed implicit, that place can be removed, resulting in a reduced number of state variables. Therefore, any analysis technique inducing either a reduced set of linear systems or a reduced number of state variables is useful because of its



(a)



(b)

Figure 5: Using safety analysis to reduce the size of a *ContPN*: (a) the safe set for the yellow region $\mathcal{R}_3$ is shown in green; (b) the reduced *ContPN*.

direct impact on the computational complexity. Until now, this property has been structurally characterized only for the special case of *par-begin par-end nets* [41]. Using the solution to the safety Problem 3.2, Algorithm 7.2 can be used to determine if an arc is implicit.

Algorithm 7.2 (Check if (p_i, t_j) is a Timed Implicit Arc)

1. **let** $\mathcal{R}^* = \{\boldsymbol{m} \in \mathcal{R} \mid ||\boldsymbol{m} - \boldsymbol{m}_0||_\infty \leq \epsilon\}$, *i.e.*, a very small region near the initial marking
2. **let** Π_0 be the set of predicates necessary to define $\mathcal{R}^*$
3. **let** Π be the set of predicates necessary to define $\mathcal{R}_i$'s and $\mathcal{R}^*$
4. Use Algorithm 5.4 to obtain a solution to Problem 3.2
5. Check if all regions in which $\frac{m_i}{Pre_{ij}} = enab(t_j, \boldsymbol{m})$ are safe, i.e., non reachable.

Example 7.3 *Let us apply Alg. 7.2 for the* ContPN *in Fig. 1 with* $\boldsymbol{m}_0 = [13, 5, 0, 6]^T$, $\boldsymbol{\lambda} = \boldsymbol{1}$, *and initial set* $\mathcal{R}^* = \mathcal{R}_3$. *By applying the previous procedure, after three iterations, the safe set is shown in Fig. 5a. We also show three individual trajectories originating in* $\mathcal{R}_3$. *Note that all states in* $\mathcal{R}_1$ *and* $\mathcal{R}_2$ *are safe. Since* $m_1 \leq m_4$ *only in these two regions, the arc* (p_1, t_1) *will never constrain the enabling degree of* t_1 *during the evolution, and therefore it is a timed implicit arc [41]. Since it is the only output arc of* p_1, *it can be removed together with the place, and the equivalent obtained net is shown in Fig. 5b.* ■

7.2 Deadlock Analysis

In this subsection, we use the procedure of solving Problem 3.3 to provide a solution to the deadlock problem, *i.e.*, the total inactivity of the servers (transitions). Deadlock avoidance is a necessity for correct and safe functioning of a system. Therefore, it is an important problem for many engineering applications, and it has been extensively investigated in the last decades [7, 42, 43]. In this subsection we present a procedure for computing a set of "bad" initial states, starting from which the system eventually deadlocks. Obviously, this set can be used in the deadlock avoidance problem of timed systems. Even if the untimed system has a deadlock state, the time interpretation together with an initial state not in the "bad" set can induce that a steady-state different by the deadlock one is reached.

In the case of continuous Petri nets, the deadlock implies $\boldsymbol{f} = \mathbf{0}$ in steady state, hence, the corresponding LTL formula for computing the initial markings that brings the system to deadlock is:

$$\phi = \Diamond\Box\left((f_1 = 0) \wedge (f_2 = 0) \wedge \ldots \wedge (f_{|T|} = 0)\right),$$

meaning that from any initial marking, including a deadlock one, eventually ($\Diamond$) the evolution will end ($\Box$ - always) in a state in which the flow of all transitions is null ($f_i = 0, \forall i = 1, \ldots, |T|$). The null flow of a transition signifies the emptiness of at least one input place, and to codify it we define a predicate corresponding to a small region where the marking is close to zero. For example, the corresponding predicate for a place p_i approaching to zero is: $\pi' = \{\boldsymbol{m} \in \mathcal{R} | m_i < \epsilon\}$, where ϵ is a (very) small constant. Using these predicates, the following algorithm computes the initial states bringing the system to deadlock.

Algorithm 7.4 (Computes initial markings leading to deadlock)

1. **let** Π be the set of predicates necessary to define $\mathcal{R}_i$'s and the regions corresponding to the zero markings
2. Use Algorithm 5.4 to obtain a solution to Problem 3.3.

Example 7.5 *For the same* ContPN *of Ex. 7.3, but now with* $\boldsymbol{m}_0 = [15, 3, 1, 0]^T$ *and* $\boldsymbol{\lambda} = \mathbf{1}$*, we compute the initial set leading to deadlock using Alg. 7.4. It has been seen in section 2.1 that* p_2 *is an implicit place for this* $\boldsymbol{m}_0$*. Therefore, only two full-dimensional regions are possible:* $\mathcal{R}_1 = \{\frac{m_1}{2} < \frac{m_4}{2}\}$ *and* $\mathcal{R}_2 = \{\frac{m_4}{2} < \frac{m_1}{2}\}$ *(with the corresponding predicates included in set* Π*).*

Since the deadlock signifies the total inactivity of the servers, i.e, $\boldsymbol{f} = \mathbf{0}$*, let us define the following predicates:* $\pi_3 = \{\boldsymbol{m} \in \mathcal{R} | m_1 < \epsilon\}$*,* $\pi_4 = \{\boldsymbol{m} \in \mathcal{R} | m_2 < \epsilon\}$*,* $\pi_5 = \{\boldsymbol{m} \in \mathcal{R} | m_3 < \epsilon\}$ *and* $\pi_6 = \{\boldsymbol{m} \in \mathcal{R} | m_4 < \epsilon\}$ *where* ϵ *is a small constant (in Fig. 6a, these regions are the ones near the borders, where we chose* $\epsilon = 0.5$*).*

According to (4)*, the deadlock is: (i)* $\frac{m_1}{2} = 0$ *or* $\frac{m_4}{2} = 0$ *(*$f_1 = 0$*) and (ii)* $m_2 = 0$ *or* $m_4 = 0$ *(*$f_2 = 0$*) and (iii)* $m_3 = 0$ *(*$f_3 = 0$*). Hence, the LTL formula that computes the initial states bringing the* ContPN *system to deadlock is:*

$$\phi = \Diamond\Box\left((\pi_3 \vee \pi_6) \wedge (\pi_4 \vee \pi_6) \wedge (\pi_5)\right)$$

By applying our algorithm, the whole polytope is obtained after three iterations, as shown in Fig. 6. This means that from any initial marking, the system will eventually reach a deadlock state. In the same figure, two trajectories originating in $\mathcal{R}_1$ *are illustrated.* ■

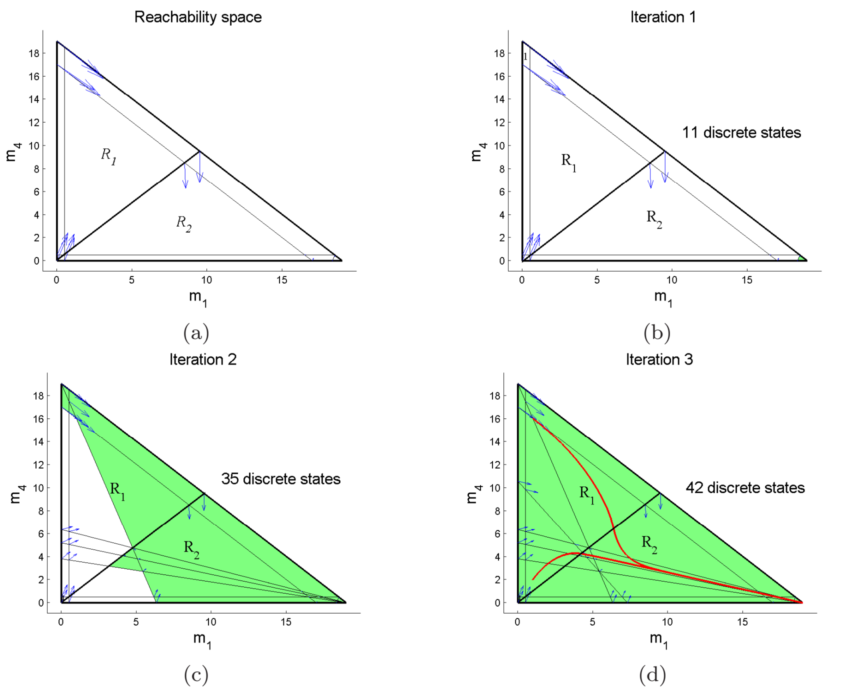


Figure 6: Computation of an initial set leading to deadlock: (a) the deadlock states; (b), (c), (d) successive iterations for the computation of an initial set leading to deadlock (green regions).

8 Conclusions

The focus of this paper was on developing an automated framework for formal analysis of timed continuous Petri nets. We addressed two important problems, namely (1) the construction of a safe region for a given initial set, and (2) the construction of an initial set such that an arbitrary LTL specification is satisfied by all trajectories originating in this set. The solutions to both these problems start with reducing the initial *ContPN* to an equivalent PWA system. Then, a finite (and conservative) abstraction of this PWA system was constructed by using computationally attractive results that mainly involve polyhedral operations. Intermediate solutions for the initial problems were obtained by using the discrete abstraction and standard tools as searches on graphs and model checking algorithms. Finally, a refinement procedure was developed, allowing us to iteratively reduce the modelling conservativeness and improve the solutions to the initial problems. The proposed framework was implemented as a freely downloadable software tool [27] and it was successfully used for providing solutions to two important problems concerning *ContPN*, namely finding timed implicit arcs and finding initial markings from where the system reaches deadlock.

Acknowledgement

This paper is written in memoriam of Prof. Laura Recalde, co-author of the conference version of the paper [1], who passed away in December 2008.

References

[1] M. Kloetzer, C. Mahulea, C. Belta, L. Recalde, and M. Silva, "Formal analysis of timed continuous Petri nets," in *47th IEEE Conference on Decision and Control (CDC 2008)*, Dec. 2008, pp. 245–250.

[2] M. C. Zhou and F. DiCesare, *Petri Net Synthesis for Discrete Event Control of Manufacturing Systems.* Kluwer Academic Publishers, 1993.

[3] M. Allam and H. Alla, "Modeling and simulation of an electronic component manufacturing system using hybrid Petri nets," *IEEE Trans. on Semiconductor Manufacturing*, vol. 11, no. 3, pp. 374–383, 1998.

[4] F. Balduzzi, A. Giua, and C. Seatzu, "Modelling and simulation of manufacturing systems with first-order hybrid Petri nets," *Int. J. of Production Research*, vol. 39, no. 2, pp. 255–282, 2001, special Issue on Modelling, Specification and analysis of Manufacturing Systems.

[5] A. Desrochers, Ed., *Modeling and Control of Automated Manufacturing Systems.* IEEE Computer Society Press, 1989.

[6] M. Dotoli, M. Fanti, A. Giua, and C. Seatzu, "First-Order Hybrid Petri nets. An application to distributed manufacturing systems," *Nonlinear Analysis: Hybrid Systems*, vol. 2, no. 2, pp. 408–430, June 2008.

[7] J. Ezpeleta, J. M. Colom, and J. Martínez, "A Petri net based deadlock prevention policy for flexible manufacturing systems," *IEEE Trans. on Robotics and Automation*, vol. 11, no. 2, pp. 173–184, 1995.

[8] A. Giua and C. Seatzu, "Modeling and Supervisory Control of Railway Networks Using Petri Nets," *IEEE Transactions on Automation Science and Engineering*, vol. 5, no. 3, pp. 431–445, July 2008.

[9] R. Zurawski and M. C. Zhou, "Petri nets and industrial applications: A tutorial," *IEEE Trans. on Industrial Electronics*, vol. 41, no. 6, pp. 567–583, 1994.

[10] B. Berthomieu and M. Diaz, "Modeling and verification of time dependent systems using time Petri nets," *IEEE Trans. on Software Engineering*, vol. 17, no. 3, pp. 259–273, 1991.

[11] B. Berthomieu and M. Menasche, "An enumerative approach for analyzing time Petri nets," in *Proceedings IFIP*. Elsevier Science Publishers, 1983, pp. 41–46.

[12] R. David and H. Alla, *Discrete, Continuous and Hybrid Petri Nets*. Springer-Verlag, 2005.

[13] M. Silva and L. Recalde, "On fluidification of Petri net models: from discrete to hybrid and continuous models," *Annual Reviews in Control*, vol. 28, no. 2, pp. 253–266, 2004.

[14] D. Bertsimas, D. Gamarnik, and J. Tsitsiklis, "Stability conditions for multiclass fluid queueing networks," *Automatic Control, IEEE Transactions on*, vol. 41, no. 11, pp. 1618–1631, Nov 1996.

[15] C. G. Cassandras, G. Sun, C. Panayiotou, and Y. Wardi, "Perturbation analysis of multiclass stochastic fluid models," in *15th IFAC World Congress*, 2002.

[16] H. Chen and D. D. Yao, *Fundamentals of queueing networks: Performance, asymptotics, and optimization*, ser. Stochastic Modelling and Applied Probability. Springer-Verlag, New York, 2001.

[17] M. Silva, E. Teruel, and J. M. Colom, "Linear algebraic and linear programming techniques for the analysis of net systems," in *Lectures in Petri Nets. I: Basic Models*, ser. LNCS, G. Rozenberg and W. Reisig, Eds. Springer, 1998, vol. 1491, pp. 309–373.

[18] L. Recalde, S. Haddad, and M. Silva, "Continuous Petri Nets: Expressive Power and Decidability Issues," in *Proc. of the 5th Int. Symp. on Automated Technology for Verification and Analysis (ATVA2007)*, vol. 4762. Springer, 2007, pp. 362–377.

[19] F. Balduzzi, G. Menga, and A. Giua, "First-order hybrid Petri nets: a model for optimization and control," *IEEE Trans. on Robotics and Automation*, vol. 16, no. 4, pp. 382–399, 2000.

[20] S. Troncale, J.-P. Comet, and G. Bernot, "Verification of biological models with Timed Hybrid Petri Nets," in *International Symposium on Computational Models of Life Sciences*, vol. 952, 2007, pp. 287–296.

[21] C. Mahulea, L. Recalde, and M. Silva, "Basic Server Semantics and Performance Monotonicity of Continuous Petri Nets," *Discrete Event Dynamic Systems: Theory and Applications*, vol. 19, no. 2, pp. 189–212, 2009.

[22] F. Torrisi and A. Bemporad, "HYSDEL — A tool for generating computational hybrid models," *IEEE Trans. Contr. Systems Technology*, vol. 12, no. 2, Mar. 2004, `http://control.ethz.ch/~hybrid/hysdel`.

[23] L. Habets, P. Collins, and J. van Schuppen, "Reachability and control synthesis for piecewise-affine hybrid systems on simplices," *IEEE Trans. Aut. Control*, vol. 51, pp. 938–948, 2006.

[24] A. Chutinan and B. H. Krogh, "Verification of infinite-state dynamic systems using approximate quotient transition systems," *IEEE Trans. Aut. Control*, vol. 46, no. 9, pp. 1401–1410, 2001.

[25] B. Yordanov, C. Belta, and G. Batt, "Model checking discrete time piesewise affine systems: application to gene networks," in *European Control Conference*, Kos, Greece, 2007.

[26] M. Kloetzer and C. Belta, "Reachability analysis of multi-affine systems," in *Hybrid Systems: Computation and Control: 9th International Workshop*, ser. LNCS, J. Hespanha and A. Tiwari, Eds. Springer Berlin / Heidelberg, 2006, vol. 3927, pp. 348 – 362.

[27] M. Kloetzer, C. Mahulea, C. Belta, and M. Silva, "Software tool for formal verification of timed continuous Petri nets," URL http://webdiis.unizar.es/∼cmahulea/research/formal_contPN.zip.

[28] R. Alur, C. Courcoubetis, N. Halbwachs, T. Henzinger, P. H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine, "The algorithmic analysis of hybrid systems," *Theoretical Computer Science*, vol. 138, no. 1, pp. 3–34, 1995.

[29] T. Henzinger, P. Ho, and H. Wong-Toi, "HyTech: A model checker for hybrid systems," *International Journal on Software Tools for Technology Transfer*, vol. 1, no. 1–2, pp. 110–122, 1997.

[30] S. D. Cairano and A. Bemporad, "An equivalence result between linear hybrid automata and piecewise affine systems," *IEEE Trans. Automatic Control*, vol. 55, no. 2, pp. 498–502, 2010.

[31] C. Mahulea, A. Ramírez, L. Recalde, and M. Silva, "Steady state control reference and token conservation laws in continuous Petri net systems," *IEEE Trans. on Autom. Science and Engineering*, vol. 5, no. 2, pp. 307–320, 2008.

[32] E. M. M. Clarke, D. Peled, and O. Grumberg, *Model checking.* MIT Press, 1999.

[33] M. Kloetzer and C. Belta, "A fully automated framework for control of linear systems from temporal logic specifications," *IEEE Trans. Aut. Control*, vol. 53, no. 1, pp. 287–297, 2008.

[34] G. Holzmann, *The SPIN Model Checker, Primer and Reference Manual.* Reading, Massachusetts: Addison-Wesley, 2004.

[35] C. Belta and L. Habets, "Constructing decidable hybrid systems with velocity bounds," in *43rd IEEE Conference on Decision and Control*, Paradise Island, Bahamas, 2004.

[36] L. Habets and J. van Schuppen, "A control problem for affine dynamical systems on a full-dimensional polytope," *Automatica*, vol. 40, pp. 21–35, 2004.

[37] K. Fukuda, "CDD/CDD+ package," URL http://www.cs.mcgill.ca/∼fukuda/soft/cdd_home/cdd.html, 1997.

[38] M. Kloetzer, "Symbolic motion planning and control," Ph.D. dissertation, Boston University, Boston, MA, 2008.

[39] F. Torrisi and M. Baotic, "Matlab interface for the CDD solver," URL http://control.ee.ethz.ch/∼hybrid/cdd.php.

[40] P. Gastin and D. Oddoux, "Fast LTL to Büchi automata translation," in *Proceedings of the 13th Conference on Computer Aided Verification (CAV'01)*, ser. Lecture Notes in Computer Science, H. C. G. Berry and A. Finkel, Eds., no. 2102. Springer, 2001, pp. 53–65.

[41] L. Recalde, C. Mahulea, and M. Silva, "Improving analysis and simulation of continuous Petri nets," in 2^{nd} *IEEE Conference on Automation Science and Engineering*, Shanghai, China, October 2006, pp. 7–12.

[42] S. Reveliotis, *Real-Time Management of Resource Allocation Systems: A Discrete Event Systems Approach.* Springer, 2005.

[43] Z. Li and M. C. Zhou, *Deadlock Resolution in Automated Manufacturing Systems: A Novel Petri Net Approach.* Springer Publishing Company, Incorporated, 2009.