Noname manuscript No.
(will be inserted by the editor)

Shrinking of Time Petri nets

Didier Lime* · Claude Martinez · Olivier H. Roux***

the date of receipt and acceptance should be inserted later

Received: date / Accepted: date

Abstract The problem of the synthesis of time bounds enforcing good properties for reactive systems has been much studied in the literature. These works mainly rely on dioid algebra theory and require important restrictions on the structure of the model (notably by restricting to timed event graphs). In this paper, we address the problems of existence and synthesis of shrinkings of the bounds of the time intervals of a time Petri net, such that a given property is verified. We show that this problem is decidable for CTL properties on bounded time Petri nets. We then propose a symbolic algorithm based on the state class graph for a fragment of CTL. If the desired property "includes" k-boundedness, the proposed algorithm terminates even if the net is unbounded. A prototype has been implemented in our tool Romeo and the method is illustrated on a small case study from the literature.

Keywords Time Petri nets, time interval shrinking, model-checking, state-class graph.

1 Introduction

Production control problems in manufacturing are often modeled as Discrete Event System (DES) control problems and Petri nets are widely used to tackle such kind of control problems (Cassandras and Lafortune 1992; Li and Zhou 2009).

Some production systems are subject to strict time constraints in the duration of some particular tasks in their manufacturing process such as chemical

This work has been partially supported by project ImpRo ANR-2010-BLAN-0317.

*LUNAM Université, École Centrale de Nantes, IRCCyN UMR CNRS 6597
firstname.name@irccyn.ec-nantes.fr ·
** LUNAM Université, Université de Nantes, IRCCyN UMR CNRS 6597
firstname.name@univ-nantes.fr

or thermal treatment. These types of duration constraints that are to be taken into account appear for example in finishing (Spacek et al 1999; Manier and Bloch 2003; Bloch et al 2010), or in semi-conductor manufacturing (Kim and Lee 2003; Lee 2008; Wu et al 2008, 2011).

The management of automated production systems involves also the management of transportation facilities in workshops. Indeed, when parts transportation is handled by a fleet of Automated Guided Vehicles (AGV), the allocation of segments of the transportation path is yet another problem of constraint set to be taken into account (Park et al 2002). Maza and Castagna (2005), has transformed this problem in a time window allocation problem.

Achour and Rezg (2007) also pointed out that temporal specifications should be taken into account in the modeling and the control of industrial processes. They introduced Time Floating General Mutual Exclusion Constraints (TFGMEC), i.e. GMEC (Giua et al 1992) defined on a time interval. Together with the model of the plant (a Petri net), they define a global clock with the help of a p-timed Petri net to model the time intervals. Their approach consists therefore in avoiding forbidden states by adding some control places to the initial Petri net that models the plant.

Beyond the scope of manufacturing applications, one can cite the work of Houssin et al (2007) and the work of Katz (2007) that concern similar constraints set on transportation systems. Theses studies assume that an *a priori* schedule has been determined, providing a Timed Event Graph (TEG) to model the dynamic behavior of the system. The problem finally turns to the synthesis of a controller which is also a TEG, the controller synthesis consists in adding control places to the original TEG.

In the pioneer work of Berthomieu and Diaz (1991) on Time Petri nets to analyze DES subject to sojourn time constraints, the problem was formulated as a verification problem. More recently, in Bonhomme et al (2001) and Gardey et al (2006), this class of Petri net has been also used to tackle problems of control law synthesis to guaranty the meeting of temporal constraints. In these works, it is assumed that some inputs of the process are controllable (Holloway et al 1997).

Control, Model-checking, Interval Shrinking

The problem of *verification*, also known as *model-checking*, for a given system S and a given set of specifications ϕ, consists in verifying that S satisfies specifications ϕ. It is denoted $S \models \phi$. In a *control problem*, one assumes that one may restrict the behavior of S: some events of S are supposed to be *controllable* and therefore one can prevent or delay their occurrence. Hence, one intends to answer the question: is there a controller C such that $S \times C \models \phi$? One aggregates to this problem of existence the problem of *controller synthesis* for which one computes C. These problems may be treated with discrete, timed or hybrids models. The interval shrinking problem is more specific: given a model of a timed system S, containing a set of controllable events T_r subject to a time interval constraint, is there a shrinking of these time intervals (let

$[\![(S, T_r)]\!]_c$ denote the resulting model of this shrinking operation) such that $[\![(S, T_r)]\!]_c \models \varphi$? Again, one aggregates to this problem of existence the problem of *synthesis* for which one computes the whole set of admissible shrinkings.

Related works

The works done in the framework of TEG to meet temporal constraints are closely related to the interval shrinking problem presented in this article. In Amari et al (2005), the problem is formulated using $(min, +)$ algebra, assuming all delays in the TEG are integers, authors give a set of inequalities that specify the temporal constraints and they propose a method to synthesize a supervisor which guaranties that all constraints are respected. In Atto et al (2011), a method based on $(max, +)$ algebra is applied to control an industrial plant dedicated to the production of rubber parts for the automotive industry. In Spacek et al (1999), authors propose a method that use $(min, max, +)$ algebra, they apply their results on an electroplating finishing process. Finally, Kim and Lee (2003) propose to apply a strategy that lead to a periodic scheduling to control a cluster-tool for the semi-conductor industry. Their work is based on $(max, +)$ algebra.

In the works cited above, whenever the control problem admits a solution, authors give a set of delays to assign to control places that are added to the original TEG in order to comply with the desired behavior. Hence the behavior of the controlled systems is restricted.

Another way to handle the interval shrinking problem is to use the parametric model-checking approach. Alur et al (1993) have introduced Parametric Timed Automata (PTA) and they have proved that in general, the existence of particular values to be taken for the parameters of a PTA in order to verify

dioid algebras related to interval shrinking problem. Assuming the sequence of resources use is not *a priori* determined, one can consider all possible trajectories, the control would disable those that would lead to constraints violation. The problem is tackled here using a Time Petri net to model the process. The existence of a control law that guaranty the meeting of temporal constraints is translated into the existence of a set of interval shrinkings on the Time Petri net, without modification of its structure. It is shown that the interval shrinking problem is decidable for CTL properties on bounded Time Petri nets. A symbolic algorithm based on the state class graph is proposed for a fragment of CTL. If the property that is to be verified "includes" the k-boundedness of the net, then our algorithm terminates even if the initial net is unbounded. A prototype has been implemented in the tool Romeo.

Outline of the paper

Section 2 introduces the definitions and concepts used in the work. In section 3, the problem of interval shrinking is presented. Section 4 details the extension of the state class graph used to compute the shrinkings and section 5 presents explicitly the computation of the shrinkings. Finally, section 6 present the implementation in the tool Romeo and our method is applied to a case-study.

2 Definitions

Notations. Let B^A denote the set of mappings from A to B. If A is finite, and $|A| = n$, an element of B^A is also a vector in B^n. Operators $\{+, -, <, =\}$ on vectors of A^n, with $A = \mathbb{N}, \mathbb{Q}, \mathbb{R}$, are the pointwise extensions of their scalar definition in A. We call *valuation* of a set of n elements B a mapping ν from B to A. This mapping may be represented by a vector of A^n, therefore ν_i denotes the i^{th} component of ν. $\overline{0}$ denotes the valuation such that $\forall i \in [1..n], \overline{0}_i = 0$. $\mathbb{B} = \{\mathbf{true}, \mathbf{false}\}$ represent the booleans values *true* and *false*.

Consider a valuation $\nu \in A^n$ and $d \in A$, hence $\nu + d$ is the valuation (vector) such that $\forall 1 \leq i \leq n, (\nu + d)_i = \nu_i + d$, and for $A' \subseteq A$, $\nu[A' \mapsto 0]$ is the valuation ν' such that $\nu'(x) = 0$ for $x \in A'$ and $\nu'(x) = \nu(x)$ otherwise.

2.1 Time Petri nets

Time Petri nets (*TPN*), introduced in Merlin (1974), are an extension of autonomous Petri nets with time constraints on transition firings. Syntactically, for this model, a firing interval is associated with a transition but time can also be considered relative to places or arcs. See Boyer and Roux (2008) for a survey and a comparison of the expressiveness of these variants.

Definition 1 (Time Petri net) A *Time Petri net* $\mathcal{N}$ is a tuple $(P, T, {}^\bullet(.), (.)^\bullet, M_0, (\alpha, \beta))$ where $P = \{p_1, \cdots, p_m\}$ is a finite set of *places*, $T = \{t_1, \cdots, t_n\}$ is a finite set of *transitions*, ${}^\bullet(.) \in (\mathbb{N}^P)^T$ is the *backward* incidence mapping,

$(.)^\bullet \in (\mathbb{N}^P)^T$ the *forward* incidence mapping, $M_0 \in \mathbb{N}^P$ the *initial* marking, $\alpha \in (\mathbb{N})^T$ and $\beta \in (\mathbb{N} \cup \{\infty\})^T$ are mappings referring respectively to the *earliest* and *latest* firing times *i.e.* lower and upper bounds of the firing time interval.

A *marking* M of a *TPN* is a mapping in $\mathbb{N}^P$ and if $M \in \mathbb{N}^P$, $M(p_i)$ is the number of tokens in place p_i. A transition t is *newly enabled* by a marking M iff $M \geq^\bullet t$. $\uparrow enabled(t_k, M, t_i) \in \mathbb{B}$ is true if t_k is enabled by the firing of transition t_i from marking M, otherwise it is false. This definition of enableness is based on the one given in Berthomieu and Diaz (1991), the most common one. In this framework, a transition t_k is *newly enabled* from a marking M if it is not enabled by $M -^\bullet t_i$ and it is enabled by $M' = M -^\bullet t_i + t_i^\bullet$, (Berthomieu and Diaz 1991). Furthermore, transition t_k is also *newly enabled* by its own firing. Formally it gives:

$$\uparrow enabled(t_k, M, t_i) = \big(M -^\bullet t_i + t_i^\bullet \geq^\bullet t_k\big) \wedge \big((M -^\bullet t_i <^\bullet t_k) \vee (t_k = t_i)\big) \quad (1)$$

The semantics of TPNs can be given in term of Timed Transition Systems (TTS)(Larsen et al 1995) which are usual transition systems with two types of labels: discrete labels for events and positive real labels for time elapsing.

Definition 2 (Semantics of *TPNs*) The semantics of a *TPN* $\mathcal{N} = (P, T, {}^\bullet(.), (.)^\bullet, M_0, (\alpha, \beta))$ is a timed transition system $S_\mathcal{N} = (Q, q_0, \rightarrow)$ where:

- $Q = \mathbb{N}^P \times (\mathbb{R}_{\geq 0})^n$, with $n = \mid T \mid$; $\nu \in (\mathbb{R}_{\geq 0})^n$ is a *valuation* such that ν_i represents the elapsed time since the ultimate enabling of t_i.
- $q_0 = (M_0, \overline{0})$,
- $\longrightarrow \in Q \times (T \cup \mathbb{R}_{\geq 0}) \times Q$ is the transition relation composed of discrete and continuous transitions:
  - the discrete transition relation is defined by:

$$\forall t_i \in T, (M, \nu) \xrightarrow{t_i} (M', \nu') \text{ iff } \begin{cases} M \geq^\bullet t_i \wedge M' = M -^\bullet t_i + t_i^\bullet \\ \alpha(t_i) \leq \nu_i \leq \beta(t_i) \\ \nu'_k = \begin{cases} 0 & \text{if } \uparrow enabled(t_k, M, t_i), \\ \nu_k & \text{otherwize.} \end{cases} \end{cases}$$

  - the continuous transition relation is defined by:

$$\forall d \in \mathbb{R}_{\geq 0}, (M, \nu) \xrightarrow{\epsilon(d)} (M, \nu') \text{ iff } \begin{cases} \nu' = \nu + d \\ \forall k \in [1..n], \big(M \geq^\bullet t_k \Longrightarrow \nu'_k \leq \beta(t_k)\big) \end{cases}$$

A *run* of a *TPN* $\mathcal{N}$ is a path in $S_\mathcal{N}$ starting from q_0.

A marking M is *reachable* in $\mathcal{N}$ iff there exists a run $(M_0, \overline{0}) \stackrel{\sigma}{\Longrightarrow} (M, \nu)$ in $S_\mathcal{N}$. The set of *reachable markings* from $\mathcal{N}$ is denoted $Reach(\mathcal{N})$. If the set $Reach(\mathcal{N})$ is finite, the net $\mathcal{N}$ is *bounded*.

2.2 CTL for *TPNs*

In section 5, only a fragment of CTL formula is considered but we recall the grammar of CTL in the context of *TPNs*.

Definition 3 (CTL for *TPN*) The CTL grammar is:

$$CTL ::= \mathcal{P} \mid \neg\varphi \mid \exists X\varphi \mid \forall X\varphi \mid \varphi \wedge \psi \mid \exists \varphi \mathcal{U} \psi \mid \forall \varphi \mathcal{U} \psi$$

such that $\varphi, \psi \in CTL$, $\mathcal{P} \in GMEC$ where GMEC is the set of Generalized Mutual Exclusion Constraints (Giua et al 1992) i.e. the conjunction or disjunction of boolen evaluation of constraints: $(\sum_{i=1}^{m} a_i * M(p_i)) \bowtie c$ where $a_i \in \mathbb{Z}$, M is a marking, $p_i \in P$, $\bowtie \in \{<, \leq, =, >, \geq\}$.

CTL formula are interpreted on the states of a model $\mathcal{M} = (\mathcal{S}_{\mathcal{N}}, \mathcal{V})$, such that $\mathcal{S}_{\mathcal{N}}$ is the state space of *TPN* $\mathcal{N}$ and $\mathcal{V} : \mathcal{S}_{\mathcal{N}} \to 2^{PR}$ is a mapping that evaluates the marking of a state as follows: $\mathcal{V}(q) = \{\mathcal{P} \in PR \mid \mathcal{P}(M) = \textbf{true}\}$.

Being given a model $\mathcal{M} = (\mathcal{S}_{\mathcal{N}}, \mathcal{V})$, a proposition of makings $\mathcal{P} \in PR$ and a state $q = (M, I) \in \mathcal{S}_{\mathcal{N}}$, we use the notation $M \models \mathcal{P}$ if $\mathcal{P} \in \mathcal{V}(q)$ and $M \not\models \mathcal{P}$ if $\mathcal{P} \notin \mathcal{V}(q)$.

Finally, a *TPN* $\mathcal{N}$ satisfies a CTL formula φ, which is denoted $\mathcal{N} \models \varphi$, if and only if $q_0 \models \varphi$ (with $q_0 \in \mathcal{S}_{\mathcal{N}}$, the initial state of $\mathcal{N}$) .

3 The problem of time interval shrinking

The problem of model-checking consists in verifying that a model $\mathcal{N}$ satisfies to a property φ expressed in a particular logic. It is formally denoted $\mathcal{N} \models \varphi$. The answer to this problem is either **true** or **false**.

We are interested in the problem of *shrinking synthesis* of time intervals of a subset of transitions of a Petri net such that a particular property is verified.

Definition 4 (Shrinking of a *TPN*) Considering a Time Petri net $\mathcal{N} = (P, T, {}^\bullet(.), (.)^\bullet, M_0, (\alpha, \beta))$ and a set of transitions $T_r \subseteq \{t \in T \mid \beta(t) \neq \infty\}$, a shrinking of $\mathcal{N}$ on T_r is a Petri net $[\![(\mathcal{N}, T_r)]\!]_c = (P, T, {}^\bullet(.), (.)^\bullet, M_0, (\alpha^c, \beta^c))$ such that:

$$\forall t \in T \setminus T_r, \begin{cases} \alpha^c(t) = \alpha(t) \\ \beta^c(t) = \beta(t) \end{cases} \quad \text{and} \quad \forall t \in T_r,\ \alpha(t) \leq \alpha^c(t) \leq \beta^c(t) \leq \beta(t)$$

In the sequel, this shrinking will be described only by the valuation $c = \{\alpha^c(t), \beta^c(t) \mid t \in T_r\} \in (\mathbb{N} \times \mathbb{N})^{T_r}$ of T_r interval bounds.

The constraint $\alpha^c(t) \leq \beta^c(t)$ implies that no interval can be made empty as a result of shrinking. This ensures that shrinking intervals introduces no new deadlock in the system.

Definition 5 (Shrinkings set of a *TPN* with respect to a property) Considering a Time Petri net $\mathcal{N} = (P, T, {}^\bullet(.), (.)^\bullet, M_0, (\alpha, \beta))$, a set of transitions $T_r \subseteq \{t \in T \mid \beta(t) \neq \infty\}$, and a property φ, the set of shrinkings of $\mathcal{N}$ on T_r with respect to φ is the set $\chi(\mathcal{N}, T_r, \varphi)$ such that:

$$\forall c \in \chi(\mathcal{N}, T_r, \varphi),\ [\![(\mathcal{N}, T_r)]\!]_c \models \varphi$$

Definition 6 (Existence and synthesis of shrinkings of a *TPN*: SSP_φ) Considering a Time Petri net $\mathcal{N} = (P, T, {}^\bullet(.), (.)^\bullet, M_0, (\alpha, \beta))$, a set of transitions $T_r \subseteq \{t \in T \mid \beta(t) \neq \infty\}$, and a property φ, the problem of existence of a shrinking and the synthesis of the set of shrinkings of $\mathcal{N}$ on T_r with respect to φ is to determine the set $\chi(\mathcal{N}, T_r, \varphi)$. The existence of a shrinking is given by $\chi(\mathcal{N}, T_r, \varphi) \neq \emptyset$. These two problems are gathered and denoted SSP_φ.

Theorem 1 (SSP_{CTL} is decidable for bounded *TPN*) *Considering a bounded Time Petri net $\mathcal{N} = (P, T, {}^\bullet(.), (.)^\bullet, M_0, (\alpha, \beta))$, a set of transitions $T_r \subseteq \{t \in T \mid \beta(t) \neq \infty\}$, and a CTL property φ, the problem of existence of a shrinking and the synthesis of the set of shrinkings $\chi(\mathcal{N}, T_r, \varphi)$ is decidable.*

Proof The verification of a CTL property φ on a bounded *TPN* is decidable by building the graph of atomic classes Berthomieu and Vernadat (2003). As a consequence, for a particular shrinking c of $\mathcal{N}$ the verification of $[\![(\mathcal{N}, T_r)]\!]_c \models \varphi$ is decidable. Furthermore, the interval bounds of transitions T_r are in $\mathbb{N}$, hence the set of possible shrinkings $\chi(\mathcal{N}, T_r, \mathbf{true})$ on $\mathcal{N}$ is finite. The set $\chi(\mathcal{N}, T_r, \varphi)$ may therefore be obtained by the iterative calculus $\forall c \in \chi(\mathcal{N}, T_r, \mathbf{true})$ of the set:

$$\chi(\mathcal{N}, T_r, \varphi) = \{c \in \chi(\mathcal{N}, T_r, \mathbf{true}) \mid [\![(\mathcal{N}, T_r)]\!]_c \models \varphi\}$$

4 Graph of Shrinkable State Classes

As a first step, we intend to compute the state space of a Petri net $\mathcal{N} = (P, T, {}^\bullet(.), (.)^\bullet, M_0, (\alpha, \beta))$ assuming the transitions subset T_r is given. Time bounds of transitions in T_r may be constrained.

The number of states of a *TPN* in dense time is potentially infinite. Different abstractions of the state space of a *TPN* have been proposed in the literature (Berthomieu and Diaz 1991; Berthomieu and Vernadat 2003; Boucheneb et al 2009).

In the present work we adopt the concept of state class graph proposed in Berthomieu and Diaz (1991). A state class (M, D) refers to a marking M and a time domain D which is represented by a set of inequalities. Unlike for regions or zones, the variables used in state classes are not clocks, instead they are enabled transition firing time intervals. The graph of state classes preserves the markings and the untimed language of the state space. Therefore the graph of state classes may be used to verify marking reachability properties as well as LTL properties.

A *state class* C of a *TPN* is a pair (M, D) such that M is a marking of the net and D is a firing domain represented by a convex polyhedron of n variables, n being the number of transitions enabled by M. A point $x = (\theta_1, \theta_2, ..., \theta_n) \in D$ is composed of the values of variables $\theta_1, \theta_2, ..., \theta_n$ that refers to the firing instants in C of transitions that are enabled by M. In practice, only variables concerned by enabled transitions for marking M are kept in D in order to decrease the complexity when computing polyhedra.

A state class gather all states reachable by the same untimed firing sequence $s = t_0; t_1; \ldots; t_n$. Indeed, these states have the same marking. Nevertheless, a state class only describe the states entering the class, *i.e.* those states that are reachable with a run $\rho = q_0 \xrightarrow{d_0,t_0} q_1 \xrightarrow{d_1,t_1} \ldots \xrightarrow{d_n,t_n} q$. The firing domain corresponds therefore to the union of time intervals of the states that belong to the class.

Actually, we intend to solve the problem of synthesis of suitable time inter-

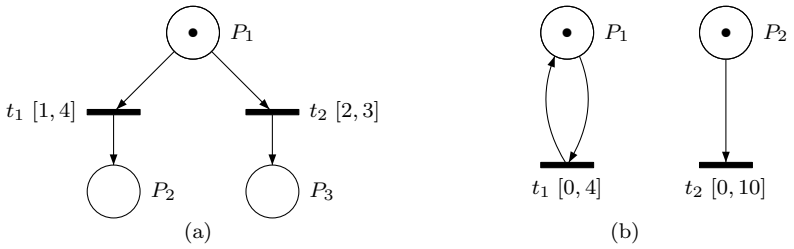

Fig. 1 Time Petri nets $\mathcal{N}_{1a}$ and $\mathcal{N}_{1b}$

shrinkable state class of $(\mathcal{N}, T_r)$ is the pair $C = (M_0, D_0)$ with

$$\begin{cases} \forall t_i \in T_r,\ \alpha(t_i) \leq \alpha_i \leq \beta_i \leq \beta(t_i) \\ \forall t_j \in T_r \cap \mathsf{enabled}\,(M_0)\,,\ \alpha_j \leq \theta_j \leq \beta_j \\ \forall t_j \in \mathsf{enabled}\,(M_0) \setminus T_r,\ \alpha(t_j) \leq \theta_j \leq \beta(t_j) \end{cases}$$

Definition 11 (Successor of a Shrinkable State Class) Assume $C = (M, D)$ is a shrinkable state class and assume transition t_f is firable from C. The shrinkable state class $C' = (M', D')$, the successor of C after firing t_f (denoted $C' = succ(C, t_f)$) is computed with the following algorithm:

- $M' = M -^{\bullet} t_f + t_f^{\bullet}$
- $D' = \mathsf{next}(D, t_f)$ is calculated from D using the following steps:
  1. intersection with other firability constraints: $\forall j,\ \theta_f \leq \theta_j$,
  2. substitution of variables $\forall t_j \neq t_f : \theta_j = \theta_f + \theta'_j$,
  3. elimination of variables θ that refers to disabled transitions after firing t_f (for example using Fourier-Motzkin method),
  4. addition of inequalities that refer to newly enabled transitions

$$\begin{array}{l} \forall t_k \in \uparrow \mathsf{enabled}\,(M, t_f) \cap T_r,\ \alpha_k \leq \theta'_k \leq \beta_k \\ \forall t_k \in \uparrow \mathsf{enabled}\,(M, t_f) \setminus T_r,\ \alpha(t_k) \leq \theta'_k \leq \beta(t_k) \end{array}$$

Like in Berthomieu and Diaz (1991), the variable changes correspond to a modification of the time origin, the new time origin being the firing instant of t_f.

One can note that in the Fourier-Motzkin method and henceforth in the determination of successor $\mathsf{next}(D, t_f)$, there is no coefficient applied on clock variables and therefore, as for parametrized DBM (Hune et al 2001), a domain may be encoded with a particular DBM whose inequalities are of the form:

$\theta_i - \theta_j \leq$ *linear expression with integer coefficients on variables* α *and* β *of* T_r

Example 1 Consider the time Petri net $\mathcal{N}_{1a}$ of figure 1.a with $T_r = \{t_1, t_2\}$.

The initial shrinkable state class and its successor after firing t_1 are the following:

$$\mathbf{C_0} = (\mathbf{M_0}, \mathbf{D_0}) : \quad M_0 = (P_1) \quad D_0 = \begin{cases} 1 \leq \alpha_1 \leq \beta_1 \leq 4 \\ 2 \leq \alpha_2 \leq \beta_2 \leq 3 \\ \alpha_1 \leq \theta_1 \leq \beta_1, \\ \alpha_2 \leq \theta_2 \leq \beta_2, \end{cases} \xrightarrow{t_1} \mathbf{C_1} = (\mathbf{M_1}, \mathbf{D_1}) : \quad M_1 = (P_2) \quad D_1 = \begin{cases} 1 \leq \alpha_1 \leq \beta_1 \leq 4 \\ 2 \leq \alpha_2 \leq \beta_2 \leq 3 \\ \beta_2 \geq \alpha_1 \end{cases}$$

One can note that the firing of transition t_1 implies that $\alpha_1 \leq \beta_2$. This prevents for example that time interval of transition t_1 would be narrowed to the point interval $[4, 4]$ which would force the firing of t_2 before firing of t_1.

State Class Graph:

The state class graph is generated by applying iteratively the calculation of a state class successor until it produces only previously generated state classes.

Definition 12 (Equivalence of Shrinkable State Classes) Two shrinkable state classes $C = (M, D)$ and $C' = (M', D')$ are said to be *equivalent* if $M = M'$ and $\mathsf{IH}(D) = \mathsf{IH}(D')$. This is denoted $C \equiv C'$.

Deciding the equivalence of two state classes requires to compute the integer hull of polyhedra. This can be done, for instance, using the method described in Cook et al (1992).

Definition 13 Assuming the *TPN* $\mathcal{N} = (P, T, {}^\bullet(.), (.)^\bullet, M_0, (\alpha, \beta))$ is shrinkable on set $T_r \subseteq \{t \in T \mid \beta(t) \neq \infty\}$, the *shrinkable states graph* of $(\mathcal{N}, T_r)$ is the transition system $\mathcal{G}((\mathcal{N}, T_r)) = \langle \mathcal{C}, \rightarrow, C_0 \rangle$ such that:

- $C_0 = (M_0, D_0)$ is the initial shrinkable state class of $(\mathcal{N}, T_r)$,
- $C \xrightarrow{t} C'$ iff $t \in \mathsf{firable}(C)$ is $C' = succ(C, t)$,
- $\mathcal{C} = \{C | C_0 \rightarrow^* C\}$, where $\rightarrow^*$ is the transitive and reflexive closure of $\rightarrow$.

The shrinkable state class graph $\mathcal{G}((\mathcal{N}, T_r))$ is therefore defined as the quotient $\mathcal{G}((\mathcal{N}, T_r))/\equiv$ of $\mathcal{G}$.

In practice, verifying that some newly computed class is equal to a previously computed one is a very expensive test. To alleviate this complexity, one can store classes in hash tables (for instance, the Parma Polyhedra Library of Bagnara et al (2008) provides a hash for each polyhedron), or binary search trees, or any efficient data structure. Inclusion of classes can also be considered instead of equality, as described in Traonouez et al (2009).

Termination of the Algorithm:

Theorem 2 *The shrinkable state class graph is finite for bounded TPNs.*

Proof Since the TPN is bounded, there is a finite number of state class markings. We now prove that there is a finite number of possible values for the integer hull of the domains.

Suppose that no transition has infinite time upper bound and let K be the greatest time upper bound in the net. Then all domains with n variables of state classes are included in the hypercube of $\mathbb{R}^n_{\geq 0}$ of vertex $\mathbf{0}$ and edge length K. By definition of the integer hull, we know that all integer hulls of domains have vertices with integer coordinates. Each of this coordinate is thus less or equal to K. Therefore, there is a finite number of such polyhedra. By definition, the transitions in the set T_r have finite time upper bound.

Suppose that some transition t not in T_r has an infinite time upper bound. In the state classes of Berthomieu and Diaz (1991), the only possible extremal rays belong to the canonical base of $\mathbb{R}^n$ where n is the dimension of the domain (Berthomieu et al 2007). Here since the added α_i and β_i variables are bounded, adding them to the polyhedra do not change these extremal rays. We further know that taking the integer hull preserves the extremal rays (Nemhauser and Wolsey 1988). Since every polyhedron can be expressed as (Schrijver 1986) the sum of its extremal rays (of which there can then only be a finite number) and of a bounded polyhedron (for which the previous paragraph applies), the number of integer hulls of domains is finite and so is the number of possible shrinkable state classes. □

Example 2 We illustrate the termination on the Time Petri net $\mathcal{N}_{1b}$ depicted in figure 1.b with $T_r = \{t_1\}$. Without taking the integer hull of D, firing t_1 repeatedly n times would lead to inequality $\theta_2 <= -n * \alpha_1 + 10$ and therefore to $n * \alpha_1 <= 10$ so that the algorithm would not terminate. The integer hull of D introduces a conjunction with $\alpha_1 <= \lfloor 10/n \rfloor$ and therefore for $n > 10$ we obtain $\alpha_1 = 0$.

Note that if some transition t is not firable from some class C the computation of $succ(C, t)$ gives an empty firing domain.

We now prove the following proposition which supports the choice of the convergence criterion in the sense that if two classes are equivalent then their successor by any transition will also be equivalent.

Proposition 1 *Let $C_1 = (M, D_1)$ and $C_2 = (M, D_2)$ be two shrinkable state classes with the same marking and $\mathsf{IH}(D_1) = \mathsf{IH}(D_2)$, then for any transition t, if $(M', D'_1) = succ(C_1, t)$ and $(M', D'_2) = succ(C_2, t)$, then $\mathsf{IH}(D'_1) = \mathsf{IH}(D'_2)$.*

Proof Let $x \in D_1 \backslash \mathsf{IH}(D_1)$. If t is not firable from $(M, \{x\})$, then $succ(M, \{x\}) = \emptyset$ and, in particular, it contains no integer point.

Now suppose that t is firable from $(M, \{x\})$. Suppose also that the coordinates of x corresponding to the α_i and β_i variables are integers. And let D_1^x be the subset of all points in D_1 that have the same projection on the α_i and

β_i variables as x. Then D_1^x is a state class of a TPN as in Berthomieu and Diaz (1991) (with additional variables but which have constant value over the polyhedron and can then be ignored) and therefore has integer vertices. These vertices thus belong to $\mathsf{IH}(D_1)$ and, by convexity of the integer hull, so does x, which is a contradiction.

Consequently the coordinates of x corresponding to the α_i and β_i variables are non integers and since the *succ* operator does not modify the values of these variables, we get that $succ(M, \{x\})$ contains no integer point.

So the integer points in $succ((M, D_1), t)$ are exactly those in $succ(M, \mathsf{IH}(D_1))$ and the same reasoning applies for D_2. And since, $\mathsf{IH}(D_2) = \mathsf{IH}(D_1)$, the integer points in $succ((M, D_1), t)$ are exactly those in $succ((M, D_2), t)$ and, finally, the two integer hulls are equal. □

5 Computing the Whole Shrinking Set of a Time Petri Net

In this section, we focus on a fragment of CTL in order to obtain an efficient algorithm. The method presented here could be extended to address the whole CTL logic by using the atomic state class graph of Berthomieu and Vernadat (2003) that preserves CTL properties but whose building is computationally expensive.

The fragment CTL_1 of CTL formulas considered here corresponds to non-nested formulas where operator X does not appear, *i.e.* $\exists\varphi\mathcal{U}\psi$ (EU), $\forall\varphi\mathcal{U}\psi$ (AU) for $\varphi \in GMEC$ and $\psi \in GMEC$ where $GMEC$ is (as in def. 3) the set of Generalized Mutual Exclusion Constraints. Classical abreviations $\exists\Diamond\varphi = \exists\mathbf{true}U\varphi$, $\forall\Diamond\varphi = \forall\mathbf{true}U\varphi$, $\exists\Box\varphi = \neg\forall\Diamond\neg\varphi$ and $\forall\Box\varphi = \neg\exists\Diamond\neg\varphi$ are included.

We now define the projection of a domain D on the set of the α_i and β_i variables of T_r.

Definition 14 (Reachability Condition of a Class) Assume the *TPN* $\mathcal{N} = (P, T, {}^\bullet(.), (.)^\bullet, M_0, (\alpha, \beta))$ is shrinkable on set $T_r \subseteq \{t \in T \mid \beta(t) \neq \infty\}$, Let $C = (M, D)$ be a shrinkable state class of $(\mathcal{N}, T_r)$. The *reachability condition* (or domain) $D_{|\alpha,\beta}$ of C is the projection of D on the set of variables α_i and β_i ($\forall t_i \in T_r$), *i.e.* removing variables θ in D.

In order to determine the set of shrinkings of a TPN on a set of transitions T_r that would allow a property ϕ to be verified, we propose to compute recursively, on each shrinkable state class $C = (M, D)$, a logical predicate $F_\phi(C)$. $F_\phi(C)$ corresponding to the verification of property ϕ on the current class C and on its successors. In the sequel, we make a slight abuse of notation using $F_\phi(C)$ to refer to the set of valuations that satisfy $F_\phi(C)$.

For a state class $C = (M, D)$, we immediately have the following properties: $F_{\mathcal{P}}(C) = \mathcal{P}(M)$, with $\mathcal{P} \in PR$, $F_{\varphi_1 \wedge \varphi_2}(C) = F_{\varphi_1}(C) \wedge F_{\varphi_2}(C)$ and $F_{\neg\varphi}(C) = \neg F_\varphi(C)$.

For $\exists\varphi\mathcal{U}\psi$ and $\forall\varphi\mathcal{U}\psi$ formulas, the value of $F(C)$ depends on its successors. We use the following two fixpoints on the shrinkable state class graph to

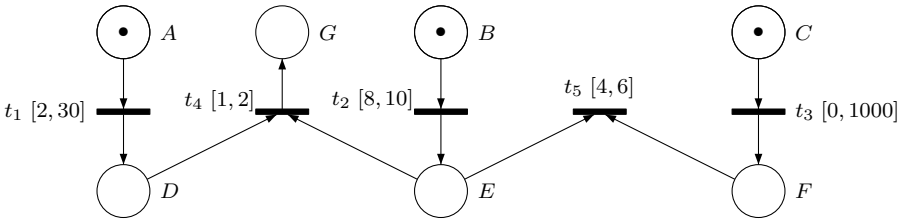

Fig. 2 The Time Petri net $\mathcal{N}_2$

compute its value. Note that $M \models \varphi$ denotes a polyhedron which is either the universe if indeed $M \models \varphi$ and empty otherwise.

Finally, in these algorithms (EU and AU), for any finite number of polyhedra $D_1, \ldots, D_m$, $\widetilde{\mathsf{IH}}(\bigcup_{i\in[1..m]} D_m) = \bigcup_{i\in[1..m]} \mathsf{IH}(D_m)$. There are many ways to express the operand of the $\widetilde{\mathsf{IH}}$ operator as a finite union of polyhedra. What is important here is that the integer points of the resulting application of $\widetilde{\mathsf{IH}}$ do not depend on this choice. In practice, the implementation will use whatever union is produced by the symbolic operators for union, complement, etc. The size of this union, in terms of the number of polyhedra involved, may unfortunately grow to large values, in particular due to the computation of the complement. We can however reduce that size by merging polyhedra when their union coincides with their convex hull. The Parma Polyhedra Library (Bagnara et al 2008), for instance, implements this operation.

5.1 EU Algorithm:

To be used on formulas of the form $\phi = \exists\varphi\mathcal{U}\psi$, with $\varphi \in PR$ and $\psi \in PR$. Assume $C = (M, D) \in \mathcal{G}_\phi(\mathcal{N})$, we compute:

$$F^0_{\exists\varphi\mathcal{U}\psi}(C) = \emptyset$$

$$F^{n+1}_{\exists\varphi\mathcal{U}\psi}(C) = (\mathsf{IH}(D)_{|\alpha,\beta} \cap M \models \psi) \cup \Big(M \models \varphi \cap M \not\models \psi \cap \widetilde{\mathsf{IH}}\Big(D_{|\alpha,\beta} \cap \bigcup_{\substack{t\in \mathsf{firable}(C)\\ C'=succ(C,t)}} F^n_{\exists\varphi\mathcal{U}\psi}(C')\Big)\Big)$$

Example 3 Considering the Time Petri net $\mathcal{N}_2$ depicted in figure 2 with $T_r = \{t_1, t_3\}$ and the formula $\forall\Box\varphi = \neg\exists\Diamond\neg\varphi = \neg\exists\mathbf{true}U\neg\varphi$ with $\varphi = (M(G) = 0)$ which express that place G is never marked, the result is: $\{\alpha_3 \leq \beta_3 \leq \alpha_1 - 6\} \wedge \{\alpha_1 \geq 16\} \wedge \{\alpha_1 \leq \beta_1 \leq 30\}$.

Constraint $\beta_3 \leq \alpha_1 - 6$ guarantees that t_3 will fire at least 6 time units before t_1. Constraint $\alpha_1 \geq 16$ imposes that t_1 will fire after instant 16 and therefore that t_3 will fire before instant 10. t_5 must then fire no later than $16 = 10 + 6$, t_4 would not fire before instant $17 = 16 + 1$ which is prohibited by firing t_5 that disable t_4 and prevents that place G would be marked.

5.2 AU Algorithm

To be used on formulas of the form $\phi = \forall\varphi\mathcal{U}\psi$. In this algorithm $\overline{D}$ is the complement of D.

Assume $C = (M, D) \in \mathcal{G}_\phi(\mathcal{N})$, we compute :

$$F^0_{\forall\varphi\mathcal{U}\psi}(C) = \emptyset$$

$$F^{n+1}_{\forall\varphi\mathcal{U}\psi}(C) = (\mathsf{IH}(D)_{|\alpha,\beta} \cap M \models \psi) \cup \Big(M \models \varphi \cap M \not\models \psi \cap \widetilde{\mathsf{IH}}\Big(D_{|\alpha,\beta} \cap \bigcap_{\substack{t \in \mathsf{firable}(C) \\ C'=(M',D')=succ(C,t)}} (F^n_{\forall\varphi\mathcal{U}\psi}(C') \cup \overline{D'_{|\alpha,\beta}})\Big)\Big)$$

The second disjunction needs the results of the calculus on the successors. Either the formula is verified on the successor, or condition $\neg D'_{|\alpha,\beta}$ disables the successor reachability.

Example 4 Consider again Petri net $\mathcal{N}_2$ of figure 2 with $T_r = \{t_1, t_3\}$, but with the following formula $\forall\Diamond\varphi = \forall\mathbf{true}U\varphi$ with $\varphi = (M(G) > 0)$.

The result is: $(\{\beta_1 \leq \alpha_3 + 1\} \vee \{\beta_1 \leq 9\}) \wedge \{2 \leq \alpha_1 \leq \beta_1 \leq 30\}$

The condition ensures that the firing of t_1 occurs no later than 1 time unit after enabling t_5 (obtained after firing t_2 and t_3). After this instant, t_4 must fire before 2 time units, and it remains at least 3 time units to wait before firing t_5 would be allowed, hence t_4 fires and place G is marked.

5.3 Correctness, completeness and termination

We now prove that the EU and AU algorithm effectively compute exactly the whole set of shrinkings for their respective properties and terminate.

Theorem 3 (Computation of the Shrinkings Set) *Let $\mathcal{N}$ be a TPN and T_r be a subset of its transitions with finite latest firing times. Let ϕ be a formula of CTL_1. The corresponding set of shrinkings is $\chi(\mathcal{N}, T_r, \phi) = \{c \in (\mathbb{N} \times \mathbb{N})^{T_r} | c \in F_\phi(C_0)\}$*

Proof In the context of parametric model checking, it has been proved in Traonouez et al (2009) that whenever the construction of the parametric state class graph terminates, which is not guaranteed, then the result of algorithm EU (resp. AU) is the set of the possible values of the parameters such that the corresponding EU (resp. AU) property is verified (Theorems 21 and 22 of that paper). Adapted to our problem, it means that whenever the construction of the shrinkable state class graph without integer hulls in the equality of classes terminates, then the result of algorithm EU (resp. AU) without integer hulls is the set of the possible (non necessarily integer) values of the variables α_i and β_i such that the corresponding EU (resp. AU) property is verified.

What we therefore need to prove is that, in addition to ensuring the finiteness of the state class graph, the integer hull preserves all the correct *integer* values for the α_i and β_i variables.

First, if we set integer values for the α_i and β_i variables in D, the projection of the obtained polyhedron on the θ variables is a classical state class as defined in Berthomieu and Diaz (1991). Since the bounds of the time intervals are integers, then the vertices of this projection are also integer points. And since all the time intervals in the net are closed this state class contains its vertices, and therefore at least one integer point. Consequently for any integer values of the α_i and β_i variables, there is at least a corresponding point in $\mathsf{IH}(D)$. And therefore, the integer points in $D_{|\alpha,\beta}$ are included in $\mathsf{IH}(D)_{\alpha,\beta}$. The converse inclusion is trivial and we then get that the integer points in $D_{|\alpha,\beta}$ and those in $\mathsf{IH}(D)_{\alpha,\beta}$ are exactly the same.

Second, for the same reason, the $\widetilde{\mathsf{IH}}$ operator preserves the set of integer points (and their projections on α_i and β_i variables) contained in its operand.

Finally, since algorithm EU only uses the domains of classes through their integer hulls, and thanks to Prop. 1 which says that if two classes are equivalent then their successors by the same actions will always be equivalent, we have $\forall n$, if $C_1 \equiv C_2$ then $F^n_{\exists\varphi\mathcal{U}\psi}(C_1) = F^n_{\exists\varphi\mathcal{U}\psi}(C_2)$. So computing $F^n_{\exists\varphi\mathcal{U}\psi}$ on the whole (generally infinite) shrinkable state class graph can be done by computing it on its quotient by $\equiv$. The same reasoning of course applies for algorithm AU. □

Theorem 4 *Algorithms EU and AU terminate for bounded TPNs.*

Proof By Thm. 2, if the TPN is bounded then its shrinkable state class graph is finite.

By a simple induction on n, we can prove that for all n and for all classes C, $F^n_{\exists\varphi\mathcal{U}\psi}(C)$ can be written as a finite union of polyhedra with integer vertices and that $F^n_{\exists\varphi\mathcal{U}\psi}(C) \subseteq F^{n+1}_{\exists\varphi\mathcal{U}\psi}(C) \subseteq D_{|\alpha,\beta}$.

Since there are only a finite number of polyhedra with integer vertices included in $D_{|\alpha,\beta}$, and a finite number of classes in the shrinkable state class graph, algorithm EU terminates.

The same reasoning applies to algorithm AU. □

5.4 The unbounded net case

If a net is not bounded, one can guarantee that the synthesis terminates in some cases. Assume we seek for a shrinking on net $\mathcal{N}$ that guarantees a property ϕ. A first possibility consists in researching either a shrinking on the net that guarantee $\psi = \phi_k \wedge \phi$, with $\phi_k = \forall\Box \bigwedge_{p\in P} M(p) \leq k$: ϕ_k is therefore the k-boundedness of the net. Hence we have $F_\psi(C_0) = F_{\phi_k}(C_0) \wedge F_\phi(C_0)$. The synthesis for ϕ_k terminates. Indeed, ϕ_k is also writen as $\neg(\exists\Diamond \bigvee_{p\in P} M(p) > k)$ and by algorithm EU, exploration stops as soon as a place is marked with more than k tokens. Constraints generated by ϕ_k guarantee henceforth that the net is k-bounded and by theorem 4, the synthesis on the whole set of shrinkings that guarantee ϕ taking into account these constraints terminates. We also obtain for a formula ϕ the set of bounded shrinkings $[\![(\mathcal{N}, T_r)]\!]_c$ verifying $[\![(\mathcal{N}, T_r)]\!]_c \models \phi$.

Another possibility, slightly more refined, consists in guaranteeing that for formulas of the type $\exists\varphi\mathcal{U}\psi$ or $\forall\varphi\mathcal{U}\psi$, we have $\varphi \Rightarrow \bigwedge_{p\in P} M(p) \leq k$. Algorithms EU and AU generate constraints that guarantee for each computed class, either φ is verified, and therefore that along the actual path the net is k-bounded, either ψ is verified and then in this case exploration stops. Indeed the algorithm terminates, with additional constraints that are necessary conditions for the global net to be k-bounded. In that way we obtain that for a formula ϕ the whole set of shrinkings $[\![(\mathcal{N}, T_r)]\!]_c$ not necessarily bounded, but verifying $[\![(\mathcal{N}, T_r)]\!]_c \models \phi$.

6 Implementation and Case study

We have implemented the algorithms in our tool ROMEO[1](Lime et al 2009), a software tool for time Petri nets analysis. ROMEO provides a graphical user interface for editing and simulating time Petri nets, and a computation module that performs state-space computation and TCTL model-checking. The Parma Polyhedra Library (Bagnara et al 2008) is used to represent the firing domains of the Shrinkable State Classes.

6.1 Comparison with naive enumerating approach

For this comparison, we adapt the version of the train-gate controller with multiple tracks described in Berthomieu and Vernadat (2003) with 3 trains. We are looking for a shrinking of the following transitions: the transition IN, associated with the approach time of a train, with interval $[0, max]$ where max is an input of our experiments; the transition L that gives the lowering time of the gate, also with interval $[0, max]$. The property we want to ensure is: *when a train is in front of the gate, the gate is always closed*. This property is satisfied iff for all trains we have $\beta_L < \alpha_{IN}$.

Table 1 provides some insight on the performance of our algorithm and a comparison to the naive enumerative algorithm (which checks the property for all possible values of the bounds of the intervals) described in the proof of theorem 1. For both our approach and the naive enumeration, in order to make the analysis more tractable, we use an additional constraint to make sure the three copies of the transition IN (one for each train) share the same bounds. For this comparison, we take $max \in \{10, 30, 100, 500, \infty\}$.

The experiments show that our algorithm is less sensitive to the value of max than the naive enumerative one. The latter is more efficient when max (and then, the number of cases to enumerate) is low whereas our method is useful when max (and then the possible bounds of the intervals) is higher. Finally, note that when the intervals are opened on infinity, the termination of our algorithm is no longer guaranteed but, for this example, it terminates and

[1] available at `http://romeo.rts-software.org/`

	$max = 10$	$max = 30$	$max = 100$	$max = 500$	$max = \infty$
Our Algo.	31.23 s	1 m 59 s	2 m 35 s	6 m 51 s	1.75 s
Naive Algo.	4.45 s	44.93 s	8 m 23 s	3 h 41 m	NA

Table 1 Comparison of the execution times for our algorithm and a naive enumeration.

the execution ends much faster because the removal of constraints improves the convergence achieved by inclusion of polyhedra. In this case, the naive enumeration is not even possible.

6.2 Case Study

We have applied our method to a single arm cluster-tool model described in Wu et al (2008). Cluster-tools are used to cyclically produce a set of silicon wafers contained in a cassette. Cluster-tools are composed of several Processing Modules (PM), a robot, and gripping and aligning docks where the robot loads (or unloads) wafers from (or to) the cassette. The problem that engineers have to face concerns the control of sojourn time of wafers in the PMs. On each PM, the sojourn time has to belong to a strict time interval, the quality of the silicon wafer directly depends on the respect of these constraints. The modeling of these constraints violation is provided through additional places BAD and BAD2 that one appends to the initial *TPN*. Transition T_{14} leading to place BAD2 should fire only if transition T_7 does not fire on time, *i.e.* if the robot does not unload the wafer from the PM on time.

Wu *et al* propose a method to elaborate a schedule that takes into account the critical tasks duration of a cluster-tool (Wu et al 2008). The model in figure 3 is adapted from the configurations they present as examples, it corresponds to a schedule of a two steps production involving three PMs and dispatched as $(2, 1)$. Configuration (i, j) of a cluster-tool means that i PMs are devoted to the first step of the production process and j PMs are devoted to the second step. We replace here the delays of three transitions (T_7, T_9, T_{15}) by some intervals that have to be determined in order to verify that process durations in PMs are respected. In our model, transition T_7 represents the operation of unloading a wafer from the third PM, transition T_9 models the gripping and aligning operation from the dock, and T_{15} refers to the process nominal duration of the second step, see Wu et al (2008). Then the set of shrinkable transition is $T_r = \{T_7, T_9, T_{15}\}$ (green transitions of figure 3) and our experiment was done with the following initial constraints on timed intervals: $\alpha_7 \geq 5$, $\beta_7 \leq 200$, $\beta_9 \leq 100$, $\beta_{15} \leq 100$. The problem is therefore, on the *TPN* of figure 3, to find suitable shrinked intervals on T_r, that satisfy the following property:

$$\forall\Box((M(\mathsf{BAD}) = 0) \text{ and } (M(\mathsf{BAD2}) = 0)$$

meaning that places BAD and BAD2 are never marked. As a result, the property is satisfied if and only if the intervals of transitions T_7, T_9 and T_{15} are

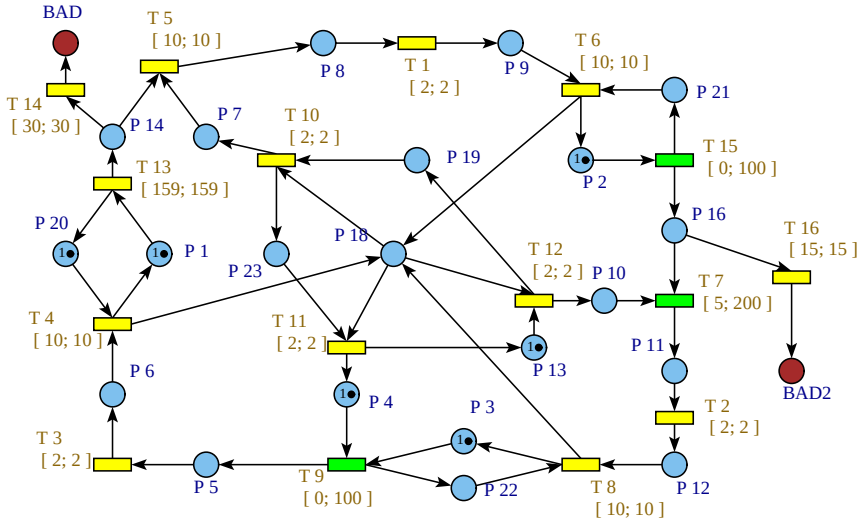

Fig. 3 A *TPN* model of a cluster tool with a (2,1) configuration

chosen with respect to the following domain:

$$\{-\beta_9 + \alpha_{15} \geq 16 \texttt{ AND } \beta_7 \leq 14\}$$
$$\texttt{OR } \{-\beta_9 + \alpha_{15} \geq 15 \texttt{ AND } \beta_7 \leq 13\}$$
$$\texttt{OR } \{\beta_7 \leq 13 \texttt{ AND } \beta_7 + \beta_9 - \alpha_{15} \leq -2\}$$

One can remark that the maximal duration of loading and unloading (given by β_9 and β_7) operations are directly linked to the lower bound of the nominal process duration (α_{15}) in the second step of the production process. The bounds computed with our method define a domain that includes the values given in Wu et al (2008), providing a wider set of values to tune the cluster tool.

7 Conclusion

In this article, we have formalized the problems of existence and synthesis of time interval shrinkings in *TPNs*, in order to verify CTL properties. We have proved that both problems are decidable and we have proposed, for a fragment of CTL, an algorithm for the synthesis of the shrinkings. We have proved that this algorithm, that may be viewed as a "decidable" specialization of a parametric model-checking semi-algorithm, terminates in the case of bounded nets. Furthermore, in the case of unbounded nets, for CTL formulas that"include" boundedness. The method has been implemented in the tool for time Petri nets Romeo and we have showed its usefulness on a small case-study from the literature.

Our approach offers an alternative to previous works, based on dioid algebra. It is set in a more general framework, considering *TPNs* without structure restrictions and CTL properties.

Perspectives consist in the extension of the algorithm to consider efficiently more expressive formula classes: including imbrication in formula and timed formula.

References

Achour Z, Rezg N (2007) Time floating general mutual exclusion constraints. Journal Studies in Informatics and Control 16(1):57–66

Alur R, Henzinger TA, Vardi MY (1993) Parametric real-time reasoning. In: ACM Symposium on Theory of Computing, pp 592–601

Amari S, Demongodin I, Loiseau J (2005) Méthode formelle de commande sous contraintes de temps dans les dioides. Journal européen des systèmes automatisés - Numéro spécial sur la Modélisation des Systèmes Réactifs, (MSR'05) 39(1-2-3):pp 319–334

André E, Chatain T, Encrenaz E, Fribourg L (2009) An inverse method for parametric timed automata. International Journal of Foundations of Computer Science 20(5):819–836

Atto AM, Martinez C, Amari S (2011) Control of Discrete Event Systems with Respect to Strict Duration: Supervision of an Industrial Manufacturing Plant. Computers and Industrial Engineering 61(4):Pages 1149–1159, DOI 10.1016/j.cie.2011.07.004

Bagnara R, Hill PM, Zaffanella E (2008) The Parma Polyhedra Library: Toward a complete set of numerical abstractions for the analysis and verification of hardware and software systems. Science of Computer Programming 72(1–2):3–21

Berthomieu B, Diaz M (1991) Modeling and verification of time dependent systems using time Petri nets. IEEE trans on soft eng 17(3):259–273

Berthomieu B, Vernadat F (2003) State class constructions for branching analysis of time Petri nets. In: TACAS 2003, Lecture Notes in Computer Science, vol 2619, pp 442–457

Berthomieu B, Lime D, Roux OH, Vernadat F (2007) Reachability problems and abstract state spaces for time Petri nets with stopwatches. Journal of Discrete Event Dynamic Systems - Theory and Applications (DEDS) 17(2):133–158

Bloch C, Manier MA, Baptiste P, Varnier C (2010) Hoist Scheduling Problem, ISTE, pp 193–231. DOI 10.1002/9780470611050.ch8

Bonhomme P, Aygalinc P, Calvez S (2001) Systèmes à contraintes de temps : Validation, évaluation et contrôle. In: Modélisation des Systèmes Réactifs, MSR 01, Toulouse, France

Boucheneb H, Gardey G, Roux OH (2009) TCTL model checking of time Petri nets. Journal of Logic and Computation 19(6):1509–1540

Boyer M, Roux OH (2008) On the compared expressiveness of arc, place and transition time Petri nets. Fundamenta Informaticae 88(3):225–249

Cassandras C, Lafortune S (1992) Introduction to discrete event systems. Kluwer Academic

Cook W, Hartmann M, Kannan R, McDiarmid C (1992) On integer points in polyhedra. Combinatorica 12(1):27–37

Gardey G, Roux OF, Roux OH (2006) Safety control synthesis for time Petri nets. In: 8th International Workshop on Discrete Event Systems (WODES'06), IEEE Computer Society Press, Ann Arbor, USA, pp 222–228

Giua A, DiCesare F, Silva M (1992) Generalized mutual exclusion constraints on nets with uncontrollable transitions. In: IEEE Int. Conf. on SMC

Holloway LE, Krogh BH, Giua A (1997) A survey of petri net methods for controlled discrete event systems. Discrete Event Dynamic Systems 7(2):151–190

Houssin L, Lahaye S, Boimond J (2007) Just in time control of constrained (max, +)-linear systems. Journal of Discrete Event Dynamic Systems 17:pp 159–178

Hune T, Romijn J, Stoelinga M, Vaandrager FW (2001) Linear parametric model checking of timed automata. In: TACAS'01, Springer, LNCS, vol 2031

Katz RD (2007) Max-plus (a,b)-invariant spaces and control of timed discrete event systems. IEEE Transactions on Automatic Control 52:229–241

Kim J, Lee T (2003) Schedule stabilization and robust timing control for time-constrained cluster tools. In: IEEE International Conference on Robotics and Automation, Taipei, Taiwan, pp pp 1039–1044

Larsen KG, Pettersson P, Yi W (1995) Model-checking for real-time systems. In: Fundamentals of Computation Theory, pp 62–88

Lee TE (2008) A review of scheduling theory and methods for semiconductor manufacturing cluster tools. In: Winter Simulation Conference, pp 2127–2135

Li ZW, Zhou MC (2009) Deadlock Resolution in Automated Manufacturing Systems: A Novel Petri Net Approach, 1st edn. Springer Publishing Company, Incorporated

Lime D, Roux OH, Seidner C, Traonouez LM (2009) Romeo: A parametric model-checker for Petri nets with stopwatches. In: Kowalewski S, Philippou A (eds) 15th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2009), Springer, York, United Kingdom, Lecture Notes in Computer Science, vol 5505, pp 54–57

Manier MA, Bloch C (2003) A classification for hoist scheduling problems. International Journal of Flexible Manufacturing Systems 15:37–55

Maza S, Castagna P (2005) A performance-based structural policy for conflict-free routing of bi-directional automated guided vehicles. Computers in Industry 56(7), DOI DOI: 10.1016/j.compind.2005.03.003

Merlin PM (1974) A study of the recoverability of computing systems. PhD thesis, Dep. of Information and Computer Science, Univ. of California, Irvine, CA

Nemhauser GL, Wolsey LA (1988) Integer and combinatorial optimization. Wiley-Interscience, New York, NY, USA

Park J, Reveliotis SA, Bodner DA, McGinnis LF (2002) A distributed, event-driven control architecture for flexibly automated manufacturing systems. International Journal of Computer Integrated Manufacturing 15:109–126

Schrijver A (1986) Theory of linear and integer programming. John Wiley & Sons, Inc., New York, NY, USA

Spacek P, Manier M, Moudni A (1999) Control of an electroplating line in the max and min algebras. International Journal of Systems Science 30(7):pp 759–778

Traonouez LM, Lime D, Roux OH (2009) Parametric model-checking of stopwatch petri nets. Journal of Universal Computer Science 15(17):3273–3304

Virbitskaite I, Pokozy E (1999) Parametric behaviour analysis for time petri nets. In: PaCT'99, Springer-Verlag, London, UK, pp 134–140

Wang F (1996) Parametric timing analysis for real-time systems. Inf Comput 130(2):131–150, DOI http://dx.doi.org/10.1006/inco.1996.0086

Wu N, Chu C, Chu F, Zhou M (2008) A petri net method for schedulability and scheduling problems in single-arm cluster tools with wafer residency time constraints. IEEE Transactions on Semiconductor Manufacturing 21(2):224–237

Wu N, Chu F, Chu C, Zhou M (2011) Petri net-based scheduling of single-arm cluster tools with reentrant atomic layer deposition processes. Automation Science and Engineering, IEEE Transactions on 8(1):42 –55, DOI 10.1109/TASE.2010.2046736