Quantitative Evaluation of Time Petri Nets and Applications to Technical and Biochemical Networks

Louchka Popova-Zeugmann
Department of Computer Science
Humboldt University, Berlin, Germany
popova@informatik.hu-berlin.de

Monika Heiner
INRIA Rocquencourt, Projet Contraintes
BP 105, 78153 Le Chesnay Cedex - France
monika.heiner@inria.fr
on sabbatical leave from BTU Cottbus

Extended Abstract

1 Introduction

Among those methods, which aim at model-based system analysis, different kinds of Petri net based frameworks have attracted a lot of attention in the last three decades. At the beginning, only qualitative (i. e. time-free) models have been discussed. But because of the crucial impact of timing issues on concurrent systems, special emphasis has been laid more and more on the incorporation of time-dependent criteria, resulting into the consideration of quantitative models. Typical claims in a time-dependent system are meeting of deadlines, ensuring of response times, minimizing total execution times, or feasibility of a certain sequence of actions under given time constraints. If the evaluation of a model of a given system reveals undesirable behaviour, then it might be of interest to deeper understand the influence of time; to be more exact: Is it possible to change some time restrictions to avoid the undesirable behaviour, and if so, what are the new time restrictions. Obviously, answers to all of these questions above are important for the application of time-dependent models.

One of the standard concepts of time-dependent Petri nets is the one, where each transition gets a continuous time interval, specifying the range of the transition's reaction time. This extension of classical Petri nets is well-known under the term Time Petri nets. Its popularity might also be caused by the easiness to incorporate working times. In this paper we present algorithms allowing quantitative analyses of this kind of time-dependent system models.

The crucial point of any exhaustive analysis is an adequate state space representation. For this purpose we exploit basically two ideas of state space reduction: parametric state description and discretisation of the state space. Hereby, we consider algorithms which work for arbitrary systems, i.e., bounded as well as unbounded ones, and algorithms, which are suitable for bounded systems only. The first class of algorithms exploits the parametric description of the state space. The solutions of the problems are traced back to solutions of systems of linear inequalities or of linear programs, respectively. The second class takes advantage of the discretely reduced state space - the set of all reachable integer states. If this set of essential states is finite, then the reachability graph can be constructed. This allows to re-use efficient graph algorithms, which have been developed for directed weighted graphs.

Altogether, we introduce eight problems, which are characterised by their input/output relation, among them the ones mentioned above. To the best of our knowledge, these problems have not been solved anywhere else in this general setting. This paper is meant to give an overview of quantitative evaluation approaches. Therefore we restrict ourselves to a sketch of the solution ideas. To illustrate possible applications of the introduced solvable problems, we refer to special analysis questions, arising during the evaluation of technical as well as biochemical systems. The mentioned application scenarios are not supposed to be exhaustive.

This paper is organised as follows. At first we recall the concept of Time Petri nets and introduce some basic notions and fundamental properties of our state space reduction principles. Afterwards we sketch the type of time-dependent models we get, when applying Timed Petri nets for modelling technical or biochemical systems, respectively. Having that, we present in the next and main section eight problems and demonstrate how to solve them. We conclude with a summary of our contributions.

2 Basic Concepts

2.1 Basics

Time Petri Nets (TPN) are derived from classical Petri nets by assigning to each transition t a (continuous) time interval $[a_t, b_t]$. Here a_t and b_t are relative to the time, when t was enabled most recently. When t becomes enabled, it can not fire before a_t time units have elapsed, and it has to fire not later than b_t time units, unless t got disabled in between by the firing of another transition. The firing itself of a transition does not consume time. So, the given time intervals specify reaction times for the transition firings. The time intervals are designed by non-negative real numbers, but the interval bounds are given as nonnegative rational numbers. Rational numbers are sufficient to reflect any measuring accuracy required by a given application domain. Moreover, to support the normalisation of different time scales within a model, zero and ∞ are allowed as interval bounds, and it holds for obvious reasons $a_t \leq b_t$ or $b_t = \infty$.

Definition 1 (Time Petri net) *The structure $\mathcal{Z} = (P, T, F, V, m_o, I)$ is called a Time Petri net (TPN), if:*

- *$S(\mathcal{Z}) := (P, T, F, V, m_o)$ is a Petri nets, i.e.*
  P, T, F are finite sets with $P \cap T = \emptyset$, $P \cup T \neq \emptyset$, $F \subseteq (P \times T) \cup (T \times P)$,
  $V : F \longrightarrow \mathbb{N}^+$ (weight of the arcs), $m_o : P \longrightarrow \mathbb{N}$ (initial marking).
- *$I : T \longrightarrow \mathbb{Q}_0^+ \times (\mathbb{Q}_0^+ \cup \{\infty\})$ and $I_1(t) \leq I_2(t)$ for each $t \in T$, where $I(t) = (I_1(t), I_2(t))$.*

The Petri net $S(\mathcal{Z})$ is referred to as the *skeleton* of $\mathcal{Z}$. I is the *interval function* of $\mathcal{Z}$, $I_1(t)$ and $I_2(t)$ are the *earliest firing time of t* ($eft(t)$) and the *latest firing time of t* ($lft(t)$), respectively. A transition t is called *immediate*, if $eft(t) = lft(t) = 0$. A TPN is called *strong*, if there does not exist an infinite latest firing time in the net, i.e., $I : T \longrightarrow \mathbb{Q}_0^+ \times \mathbb{Q}_0^+$. In a strong TPN, each transition is forced to fire within a finite time interval.

It is not difficult to see (cf. [Pop99]) that considering TPNs with $I : T \longrightarrow \mathbb{N} \times (\mathbb{N} \cup \{\infty\})$ will not result in a loss of generality. Therefore, only such time functions I will be considered subsequently. Furthermore, two transitions are said to be in a *(structural) conflict*, if they share a pre-place.

Example 1 (An arbitrary Time Petri net)

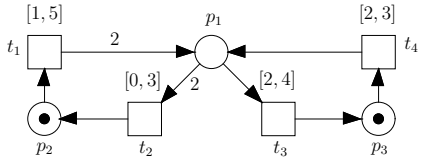

Figure 1: $\mathcal{Z}_1$ - a Time Petri net. The transitions t_2 and t_3 are in conflict.

Every possible situation in a given TPN can be described completely by a state $z = (m, h)$, consisting of a (place) marking m and a transition marking h. The (place) marking (p-marking), which is a place vector (i.e., the vector has as many components as places in the considered TPN), is defined as the standard marking notion in classical Petri nets, cf. Definition 1. The time marking (t-marking), which is a transition vector (i.e., the vector has as many components as transitions in the considered TPN), describes the current time circumstances in a certain situation. More exactly, each component of the time marking is either a real number or the sign $\sharp$. The real number indicates that the corresponding transition is enabled and gives the time elapsed since the latest enabling, whereas $\sharp$ means that the corresponding transition is disabled.

A transition t is enabled, if all its pre-places carry sufficiently many tokens. This can conveniently be tested, using the place vector t^- with $t^-(p) := V(p, t)$, if $(p, t) \in F$, and 0 else, for each place p.

Definition 2 (state) *Let $\mathcal{Z} = (P, T, F, V, m_o, I)$ be a TPN and $h : T \longrightarrow \mathbb{R}_0^+ \cup \{\#\}$. $z = (m, h)$ is called a state in $\mathcal{Z}$, if m is a reachable marking in $S(\mathcal{Z})$, $\forall t\ (\ (t \in T \wedge t^- \leq m) \longrightarrow h(t) \leq lft(t))$ and $\forall t\ (\ (t \in T \wedge t^- \not\leq m) \longrightarrow h(t) = \#)$.*

The state $z_o := (m_o, h_o)$ with $h_o(t) := \begin{cases} 0 & \text{iff } t^- \leq m_0 \\ \# & \text{iff } t^- \not\leq m_0 \end{cases}$ is set as the *initial state* of the TPN $\mathcal{Z}$.

Thus, the initial state in $\mathcal{Z}_1$, compare Fig. 1, is $z_0 = (\underbrace{(0,1,1)}_{p-marking}, \underbrace{(0,\sharp,\sharp,0)}_{t-marking})$.

The interpretation of the notion "state" is as follows: within the net, each transition t has a clock $h(t)$. If t is enabled at a marking m, its clock $h(t)$ shows the time elapsed since t became most recently enabled. If t is disabled at m, the clock is switched off (indicated by $h(t) = \#$). The state $z = (m, h)$ is called an *integer state*, if $h(t)$ is an integer for each enabled transition t in m.

The behaviour of a TPN is defined by changing from one state into another by *firing a transition* or by *time elapsing*. A formal definition can be found, e.g., in [Pop89], [Pop99], [Pop07a]. The consideration of all possible state changes leads to the following definition.

Definition 3 (reachable state, state space) *Let $\mathcal{Z} = (P, T, F, V, m_o, I)$ be a TPN.*

(a) The state $z = (m, h)$ is called reachable in $\mathcal{Z}$ (starting at z_0), if there exist states $z_1, z'_1, ..., z_n, z'_n$, transitions $t_1, ..., t_n$ and times $\tau_i \in \mathbb{R}_0^+, i = 0, 1, ..., n$ and it holds

$$z_0 \xrightarrow{\tau_0} z_1 \xrightarrow{t_1} z'_1 \xrightarrow{\tau_1} z_2 \xrightarrow{t_2} z'_2 \xrightarrow{\tau_2} \dots z_n \xrightarrow{t_n} z'_n \xrightarrow{\tau_n} z.$$

(b) The set $RS_\mathcal{Z}$ of all reachable states in $\mathcal{Z}$ (with the given initial state) is called the state space of $\mathcal{Z}$. The set of states, reachable in $\mathcal{Z}$ starting at a given state z', is denoted by $RS_\mathcal{Z}(z')$.

The sequence of transitions $\sigma = t_1 \dots t_n$ can fire in $\mathcal{Z}$ starting at z_0, because there is a sequence $\sigma(\tau) = \tau_0 t_1 \tau_1 \dots t_n \tau_n$. We call such a *transition sequence* σ a *feasible* one (or a *firing sequence* for short). The sequence $\sigma(\tau)$, which is a concrete execution of σ in $\mathcal{Z}$, is called a *(feasible) run* of σ. It is clear that in a given TPN the state changes generally consist of alternating series of time elapsing and transition firing. Obviously, for a given run the transition sequence is well defined, and for a given transition sequence there are infinitely many runs in general.

It is easy to see that the set of all reachable p-markings in a TPN $\mathcal{Z}$ (i.e., the p-marking space) is the set $\{m \mid (m, h) \in RS_\mathcal{Z}\}$, which will be denoted by $R_\mathcal{Z}$. The p-marking space of $\mathcal{Z}$ is a (not necessarily proper) subset of the reachable markings of the skeleton $S(\mathcal{Z})$. Therefore, a firing sequence in the skeleton $S(\mathcal{Z})$ is not necessarily a firing sequence in $\mathcal{Z}$. The set of p-markings, reachable in $\mathcal{Z}$ starting at a given p-marking m', is denoted by $R_\mathcal{Z}(m')$. A TPN is called *bounded*, if its set of reachable p-markings is finite, otherwise *unbounded*.

It is clear that the state space of a TPN is in general infinite and dense in terms of the time. On the one hand the set of reachable p-markings can be infinite. On the other hand, for a fixed p-marking the set of corresponding t-markings can be infinite. In the next subsection we consider two approaches for concise state space representations.

2.2 State Space Reduction

The central problem for the analysis of a given TPN is the adequate knowledge of its state space. We apply two approaches to get a finite description of infinite state spaces.

It can be shown that - despite the continuous nature of the time intervals - it is sufficient to pick up some "essential" states only to determine the entire behaviour of the net at any point in time so that qualitative and quantitative analyses remain possible. These essential states are the integer states. While the calculation of a single integer state is easy, cf. subsection 4.3, the proof that the knowledge of the integer states is sufficient for analysing a TPN is much more difficult. This problem is divided into a finite number of problems, which are solved recursively in a manner typical for the methodology of dynamic programming, see [Pop89] and [Pop99].

The state space can also be characterised parametrically, which has been introduced for strong TPN in [Pop99], and for arbitrary TPN in [Pop07a]. We sketch here the key ideas.

Let $\mathcal{Z}$ be an arbitrary TPN with the initial state $z_0 = (m_0, h_0)$. Let $\sigma = t_1 \dots t_n$ be a feasible transition sequence in $\mathcal{Z}$ and let $\sigma(\tau) = \tau_1 t_1 \dots \tau_n t_n$ be a corresponding run. Obviously, after firing $\sigma(\tau)$ a certain reachable state $z_{\sigma(\tau)}$ is obtained. When the times $\tau = \tau_1 \dots \tau_n$ are given parametrically by $X = x_1 \dots x_n$, then the reached state $z_{\sigma(X)} = (m_{\sigma(X)}, h_{\sigma(X)})$ is a parametric one. It is easy to see that the p-marking $m_{\sigma(X)}$ does not depend on X, it depends only on σ. However, the t-marking $h_{\sigma(X)}$ depends on σ as well as on the parameter X. Furthermore, it is clear that for a concrete value of X with the additional condition that the corresponding run is feasible, the t-marking $h_{\sigma(X)}$ is well-defined. Hence, an unique parametric state $z_\sigma := z_{\sigma(X)}$ can be assigned to each transition sequence σ, i.e., we can consider a function from the set of all transition sequences in $\mathcal{Z}$ into the set of all parametric states, syntactically appropriate to $\mathcal{Z}$. In order to describe the feasibility of σ, we consider the parametric state z_σ together with a set of conditions B_σ for the values of X. Actually, B_σ is a system of linear inequalities. If B_σ is solvable then σ is feasible in $\mathcal{Z}$.

The formal definitions can be found in [Pop99]. Here we restrict ourselves to an example for a parametric state reached in a TPN after firing a certain transition sequence.

Example 2 (parametric state description)
We consider $\mathcal{Z}_1$ again (cf. Fig. 1) with the parametric run $x_1t_4x_2t_3x_3t_4x_4$. The parametric state z_σ, which is reached after firing the transition sequence $\sigma := t_4t_3t_4$ with the time variables $X = x_1x_2x_3x_4$, is:

$$z_\sigma = \{\ (\underbrace{(1,1,0)}_{m_{\sigma(X)}}, \underbrace{(x_1+x_2+x_3+x_4, \sharp, x_4, \sharp)}_{h_{\sigma(X)}})\ |\ \underbrace{\begin{array}{lll} 2 \le x_1 \le 3, & 2 \le x_2 \le 4, & 0 \le x_1 + x_2 \le 5, \\ 2 \le x_3 \le 3, & 0 \le x_1 + x_2 + x_3 \le 5, & \\ 0 \le x_4 \le 4, & 0 \le x_1 + x_2 + x_3 + x_4 \le 5 & \end{array}}_{B_\sigma}\ \}.$$

3 Applications

In this section we motivate the application of Time Petri nets and sketch the type of time-dependent models we get, when applying them for modelling technical or biochemical systems, respectively.

Please note, our time model provides immediate transitions, although in reality nothing happens without consuming time. However, immediate transitions help to keep interval boundaries small. Often, system activities may be classified into activities with significant time consumption and those with non-significant (much less) time consumption. Without immediate transitions, such a difference had to be modelled by an appropriate absolute difference of time values. With immediate transitions, all time values can be scaled down relatively to a suitable time axis. Moreover, immediate transitions allow a straightforward incorporation of the working time concept.

3.1 Technical Systems

Technical systems are modelled at various abstraction levels, ranging from very fine-grained models, where transitions correspond to machine instructions, up to coarse-grained models, where transitions may represent subtasks in a production process plan (e.g., given as precedence graph). While fine-grained models are more popular for *dependability engineering* of safety-oriented systems (e.g., reactive manufacturing systems), see e.g. [Hei97b], coarse-grained models are commonly deployed for *task scheduling* under given resource constraints, see e.g. [Pop05b].

Dependability engineering of software-based systems needs to combine qualitative as well as quantitative analyses. As example let us consider the safety analysis of a software controller for pushing machines moving pieces within a production line [Hei97a]. At first, safety properties expressible in terms of legal local states have to be analysed, e.g., mutually exclusive use of shared regions by two adjacent pushing machines, which can be done using qualitative Petri nets. Afterwards, safety properties in terms of explicit error states and explicit error transitions into them might be of interest, e.g., if we want to show that a machine motion is stopped fast enough, before the machine has been moved beyond a given limit. Obviously, we need now a suitable notion of time in order to reason whether something happens fast enough. To describe and analyse the unreachability of such explicit error states (or equally - that the explicit error transitions are dead at the initial marking), we need the reaction time model.

Independently of the chosen abstraction level, the firing of a transition usually corresponds to the execution of the modelled actual system part, best expressed by the working time model. However, such an execution can be interrupted after being initiated by timers or exceptions, which calls again for the reaction time model.

Generally, execution times cannot be characterised adequately by constant time values because of data dependencies, operating systems influences or measurement variations. Therefore, these quantitative parameters are time intervals of the minimal and maximal duration to execute a given (sequential) part without resource limitations (no processor sharing, no communication delays, etc.). Of course, the interval may collapse to a constant duration in case of purely linear, non-interrupted control statement sequences. These execution time intervals can be measured by monitoring tools of the development environment, or - in special cases - calculated from machine instruction sequences. Occasionally, they are given by the system's requirement specification, sometimes they have to be assessed.

3.2 Biochemical Systems

Biochemical systems can be considered as networks of biochemical reactions, transforming input compounds via several intermediate compounds into output compounds. We have here an infinite

flux of chemical compounds realised by permanently occurring reactions. To derive a qualitative Petri net model of the biochemical network behaviour, each compound is assigned to a place, and each reaction to a transition. The arc multiplicities reflect the given stoichiometric numbers of the reactions' stoichiometric equations, if known. This straightforward modelling principle has been applied to a variety of biological networks. The Petri net structure mirrors the biochemical network topology, and the incidence matrix of the Petri net coincides with the stoichiometric matrix of the modelled biochemical system.

Following this construction principle, we get place-bordered models, where the input compounds appear as source nodes and the output compounds as sink nodes. To animate them, we transform the place-bordered models into transition-bordered ones by adding input transitions, generating the input compounds, and output transitions, consuming the output compounds. All input and output transitions work independently. Now, the Petri net behaviour reflects all partial order sequences of chemical reactions from the input to the output compounds respecting the given stoichiometric relations under any timing conditions. Transitions without pre-places may fire infinitely often; they are obviously live and all their immediate post-places are unbounded. Generally, the whole Petri net model is expected to be live and simultaneously unbounded in all places. Consequently, no analysis methods can be applied, which rely on exhaustive state space construction.

Living systems maintain - under normal conditions - a *steady state*, where all compounds have reached a dynamic concentration equilibrium, i.e. the concentrations of all compounds are permanently reproduced in a constant amount due to the constant reaction rates. The steady state and the steady state behaviour are fundamental characteristics of any network. Therefore, network evaluation typically starts with the steady state assumption, before taking into account also transient state behaviour, which might be caused, e.g., by a change of the living conditions.

In [Pop05a] it has been demonstrated, how to derive automatically quantitative steady state models of biochemical networks, starting from the underlying qualitative one. For this purpose the well-established structural Petri net analysis technique of transition invariants is exploited. Transition invariants may be interpreted as a characterisation of the system's steady state behaviour. By reading the entries of the transition invariants as relative firing rates and representing them by correspondingly adjusted transition working times, we get a quantitative model of the net behaviour in the steady state. The imposed time restrictions should make the model bounded.

4 Quantitative Evaluation

In this section we present algorithms supporting quantitative analyses of TPNs. Here, we consider algorithms which work for arbitrary systems, i.e., bounded as well as unbounded ones, and algorithms, which are suitable for bounded systems only. The first class of algorithms exploits the parametric description of state classes. The second class takes advantage of the discretely reduced state space - the set of all reachable integer states.

4.1 Preliminaries

First we formalise the notions *shortest/longest time path* between two states and between two p-markings.

Definition 4 (length of a run) *Let $\mathcal{Z} = (P, T, F, V, m_0, I)$ be a TPN and let $\sigma(\tau)$ be a run of the transition sequence σ in $\mathcal{Z}$. The length of the run $l\big(\sigma(\tau)\big)$ is the sum of all times while executing the run $\sigma(\tau)$, formally: $l\big(\sigma(\tau)\big) := \sum_{i=0}^{n} \tau_i$, whereby $n = l(\sigma)$ and $\tau = \tau_0\tau_1 \ldots \tau_n$.*

Definition 5 (minimal run) *Let $\mathcal{Z}$ be a TPN and let σ be a transition sequence in $\mathcal{Z}$. The feasible run $\sigma(\tau^*)$ of σ is the run of minimal length (for short: minimal run of σ), iff there is no further feasible run of σ, which length is shorter than the length of $\sigma(\tau^*)$.*

Evidently, for the minimal run $\sigma(\tau^*)$ of σ holds:

$$l\big(\sigma(\tau^*)\big) := \min_{\tau}\{\, l\big(\sigma(\tau)\big) \mid \sigma(\tau) \text{ is a feasible run of } \sigma \text{ in } \mathcal{Z}\,\}.$$

The notion *maximal run* can be introduced analogously, if the set of all feasible runs of σ is bounded above w.r.t. the length of the runs.

Definition 6 (maximal run) *Let $\mathcal{Z}$ be a TPN and let σ be a sequence of transitions in $\mathcal{Z}$. The feasible run $\sigma(\tau^*)$ of σ is the run of maximal length (for short: maximal run of σ), iff*

the set $\{\, l\big(\sigma(\tau)\big) \mid \sigma(\tau)$ is a feasible run of σ in $\mathcal{Z}\,\}$ is bounded above and

$$l\big(\sigma(\tau^*)\big) := \sup_{\tau}\{\, l\big(\sigma(\tau)\big) \mid \sigma(\tau) \ \textit{is a feasible run of}\ \sigma \ \textit{in}\ \mathcal{Z}\,\}.$$

Definition 7 (minimal time distance between two states) *Let $\mathcal{Z}$ be a TPN and let z_1 and z_2 be two reachable states in $\mathcal{Z}$ with $z_2 \in RS_{\mathcal{Z}}(z_1)$. The minimal time distance $d_{\min}(z_1, z_2)$ from z_1 to z_2 is the length of a minimal run of a firing sequence σ, which length is the shortest between all minimal runs from z_1 to z_2 in $\mathcal{Z}$, formally:*

$$d_{\min}(z_1, z_2) : \quad = \quad \min_{\sigma}\{\, l(\sigma(\tau)) \mid \sigma \in T^*, \sigma(\tau) \ \textit{is a minimal run of}\ \sigma \ \textit{and}\ z_1 \xrightarrow{\sigma(\tau)} z_2 \ \textit{in}\ \mathcal{Z}\,\}.$$

The notion *maximal time distance between two states* needs to consider the special situation where - starting at z_1 - we run into a cycle before reaching z_2. In this case we define the maximal time distance between the two states to be infinite.

Definition 8 (maximal time distance between two states) *Let $\mathcal{Z}$ be a TPN and let z_1 and z_2 be two reachable states in $\mathcal{Z}$ with $z_2 \in RS_{\mathcal{Z}}(z_1)$. The maximal time distance $d_{\max}(z_1, z_2)$ from z_1 to z_2 is defined by:*

$$d_{\max}(z_1, z_2) := \begin{cases} \infty & \textit{, if a cycle is reachable starting at } z_1 \\ & \textit{before reaching } z_2 \\ \max\limits_{\substack{\sigma,\\ z_1 \xrightarrow{\sigma(\tau)} z_2}} \sum\limits_{i=0}^{l(\tau)} \tau_i & \textit{, else.} \end{cases}$$

Here, the symbol ∞ means that - starting at the state z_1 - it is possible not to reach the state z_2, because of remaining in a cycle. In this case a run of arbitrary length exists, even if all edges of the cycle are weighted by zero.

In order to introduce the notion *distance between two p-markings* we define two sets of states, whereby the states of each set share the same p-marking. Let m_1 and m_2 be two p-markings with $m_1 \in R_{\mathcal{Z}}$ and $m_2 \in R_{\mathcal{Z}}(m_1)$. Than M_1 and M_2 are defined as follows:

$$\begin{aligned} M_1 &:= \{\, z \mid z \in RS_{\mathcal{Z}} \ \wedge\ z = (m_1, h)\} \\ M_2 &:= \{\, z \mid \exists z^* \big(\, z^* \in M_1 \ \wedge\ z \in RS_{\mathcal{Z}}(z^*) \ \wedge\ z = (m_2, h)\big)\,\} \end{aligned}$$

Definition 9 (minimal time distance between two p-markings) *Let $\mathcal{Z}$ be a TPN and let m_1 and m_2 be two reachable p-markings in $\mathcal{Z}$ with $m_2 \in R_{\mathcal{Z}}(m_1)$. The minimal time distance $d_{\min}(m_1, m_2)$ between m_1 and m_2 is $d_{\min}(m_1, m_2) = \min\{\, d_{\min}(z_1, z_2) \mid z_1 \in M_1 \ \wedge\ z_2 \in M_2\,\}$.*

Definition 10 (maximal time distance between two p-markings) *Let $\mathcal{Z}$ be a TPN and let m_1 and m_2 be two reachable p-markings in $\mathcal{Z}$ with $m_2 \in R_{\mathcal{Z}}(m_1)$. The maximal time distance $d_{\max}(m_1, m_2)$ between m_1 and m_2 is $d_{\max}(m_1, m_2) = \max\{\, d_{\max}(z_1, z_2) \mid z_1 \in M_1 \ \wedge\ z_2 \in M_2\,\}$.*

Remark 1 *In each TPN $\mathcal{Z}$ it holds:*

(1) For each firing sequence σ in $\mathcal{Z}$ is
$l\big(\sigma(\tau^*)\big) \in \mathbb{N}$ for the minimal run $\sigma(\tau^*)$, and
$l\big(\sigma(\tau^*)\big) \in \mathbb{N}$ for the maximal run $\sigma(\tau^*)$, if it exists.

(2) $d_{\min}(z_1, z_2), d_{\min}(m_1, m_2) \in \mathbb{N}$ and
$d_{\max}(m_1, m_2), d_{\max}(z_1, z_2) \in \big(\mathbb{N} \cup \{\infty\}\big)$.

Even more it holds: each time elapsing τ_i in a minimal as well as in a maximal run is also a natural number. The proof of this remark can be done by contraposition, see [Pop99].

4.2 Arbitrary Time Petri Nets

In this section we consider arbitrary Time Petri Nets. Hence, the state space can be infinite or far too huge to be managed within the given resource constraints. Therefore, any approach enumerating explicitly the state space is not useful here.

The next four problems deal with the time-dependent feasibility of transition sequences. The solutions of the problems are traced back to the solution of systems of linear inequalities or of linear programs, respectively. In particular, we always consider inequalities, the coefficients of which are natural numbers. We look for solutions, which are rational or real numbers, respectively. Due to the given inequalities, these solutions are always nonnegative ones. Real solutions can be found

in polynomial runtime (cf. [Pap98], [GLS93]), using the Chatchiyan algorithm. Finding rational solutions in polynomial runtime follows from the result by Tardos (1986) and a related corollary for rational polyhedrons (cf. [GLS93]).

Problem 1

Input: • A transition sequence σ in an arbitrary TPN $\mathcal{Z}$.

Output: (1) Is σ a firing sequence in $\mathcal{Z}$?

(2) A feasible run $\sigma(\tau)$ of σ, if the answer to (1) is *yes*.

Idea of the solution: The output requires to solve the linear inequality system B_σ in $\mathbb{R}$. The runtime of the solution of the inequality system is polynomial in the length of the transition sequence. (Here it holds: The number of the variables in B_σ is $l(\sigma)+1$. The number of the inequalities in B_σ is not greater than $2 \cdot (l(\sigma) \cdot |T| - 1)$. □

Applications: Even if a transition sequence is feasible in the time-free model, this has not to be true anymore under certain timing constraints. Hereby, the transition sequence can stand for a certain wanted/unwanted behaviour or it can be derived from a computed transition invariant, as it is the case in the context of modelling and analysis of biochemical networks, see [Pop05a].

Problem 1 allows to decide the time-dependent feasibility of a given transition sequence. The interval function is assumed to be completely known, i.e. totally defined. This is assumption is not always fulfilled, or the analysis objective actually consists in exploring the possible timing scope. Problem 3 provides a relaxation of problem 1 to support this kind of reasoning.

In the next problem we consider w.o.l.g. firing sequences: We are already able to check whether an arbitrary sequence is a feasible one, and the notion minimal/maximal run makes only sense for feasible transition sequences.

Problem 2

Input: • A firing sequence σ in an arbitrary TPN $\mathcal{Z}$.

Output: (1) A minimal run of σ.

(2) A maximal run of σ, if it exists.

Idea of the solution: We consider the linear inequality system B_σ and the parametric run $\sigma(x)$.

A solution of the linear program (P_1) and respectively a solution of the linear program (P_2) defines a minimal run of σ and respectively defines a maximal run of σ, if it exists. Thereby is $(P_1) = \min \{ \sum_{i=0}^{l(\sigma)} x_i \mid B_\sigma \}$ and $(P_2) = \max \{ \sum_{i=0}^{l(\sigma)} x_i \mid B_\sigma \}$. If the linear program (P_2) has no solution, then the maximal run is ∞ (the set B_σ is not empty because σ is a firing sequence in $\mathcal{Z}$). Both linear programs are solvable in polynomial time w.r.t $l(\sigma)$(cf. [Pap98]). □

Applications: The precise knowledge of the total time range to execute a given firing sequence, i.e., its minimal and maximal duration, helps to design technical systems which predictable timing behaviour. The firing sequence σ may stand for a subtask of some significance, e.g., the atomic steps, a server has to fulfill to provide a certain service; the actions, a controller has to go through, to react to an observed signal; or the transaction sequence, a data base system has to perform to respond to a received query. In all these cases, a typical non-functional system requirement will specify the expected timing behaviour of the critical system parts. Problem 2 helps to prove this type of requirements.

In a biochemical network, the firing sequence may stand for a subnetwork of chemical reactions to transform the input compound(s) into output compound(s), or – likewise – to propagate the input signal(s) into output signal(s). Problem 2 allows a model-based prediction of the network's reaction time, which might trigger new experiments to validate the prediction power of the model.

In cases, where the firing sequence is not explicitly known or hard to be specified, a critical system part might be better characterised by the initial and final system states or p-markings, respectively. To determine the time distance between two states or p-markings, see problems 5-8.

If the evaluation of a model of a given system reveals certain wanted or unwanted behaviour, then it might be of interest to deeper understand the influence of time; more exactly: Is it possible to change some time restrictions to keep the wanted behaviour or to avoid the unwanted behaviour,

respectively, and if so, what are the new time restrictions. Obviously, answers to these questions facilitate the application of TPNs. The following two problems deal with this kind of reasoning.

Problem 3

Input:
- A TPN $\mathcal{Z}$ with an *only* partially defined interval function I.
- A transition sequence σ and a number $\lambda \in \mathbb{R}_0^+$.

Output:
(1) Is it possible to complete I to a total function such that σ is a firing sequence in $\mathcal{Z}$ and $l\big(\sigma(\tau)\big) \leq \lambda$?

(2) A completed, totally defined function I, if the answer to (1) is *yes*.

(3) Is it possible to complete I to a total function such that σ is a firing sequence in $\mathcal{Z}$ and $l\big(\sigma(\tau)\big) \geq \lambda$?

(4) A completed, total defined function I, if the answer to (3) is *yes*.

Idea of the solution: First, we consider the interval function I for each $t \in T$: If $eft(t)$ is not defined, then assign to $eft(t)$ the parametric value a_t. Analogously, if $lft(t)$ is not defined, then let be $lft(t) := b_t$. Thus, I is defined for each $t \in T$. Now, we consider the parametric state (z_σ, B_σ) and the linear inequality system B_σ. The variables of B_σ are the parameters x_i defined by the parametric run $\sigma(x)$ and all the new parameters a_{t_i} and b_{t_i}. Now we consider the linear inequality system B_σ. For solving (1) we solve the linear inequality system B'_σ, which we get from B_σ by adding the linear inequality $l\big(\sigma(x)\big) \leq \lambda$. The variables are $x_0, \ldots, x_{l(\sigma)}$, and all a_{t_i} and b_{t_j}. The values of variables a_{t_i} and b_{t_j} complete the function I to a total one, if B'_σ has a solution. The output for (3) is analogue to (1), but B'_σ is now an extension of B_σ by $l\big(\sigma(\tau)\big) \geq \lambda$. The output of (4) is analogue to (2). □

Applications: Problem 3 may be read as a relaxation of problem 2. It not only requires less pre-knowledge of the given timing conditions (the interval function may be partially defined only), but allows also to specify some lower and upper limits for the expected timing behaviour (by two numbers λ) of a given subtask (represented by the transition sequence σ).

This is especially helpful for a systematic construction of technical systems with strong timing constraints. As soon as parts of the system under development have been implemented (in hardware or software), their timing behaviour (interval bounds) can be determined. For the other parts of system, not implemented yet, it is interesting to learn the time scope they have to follow, so that the total system has still a chance to fulfill the given timing requirements (specified by the two numbers λ). In the worst-case it will turn out that the implemented parts already contradict the given constraints. In such a model-guided step-wise system development, contradictions to timing requirements will be discovered early, i.e., in a development phase, where wrong design decisions can be corrected at a reasonable prize.

In biochemical networks, problem 3 supports a model-based drug prescription. Here, the undefined parts in the interval function correspond to disturbed reactions (accelerated, slowed down or totally switched off), causing too high/slow throughput of the network (characterised by the execution times λ of the subnetworks, transforming the input compounds into output compounds). The knowledge of a possible completion of the interval function to guarantee a certain throughput may be translated into the required drug amount to re-adjust the timing behaviour of the disturbed reactions.

Problem 4

Input:
- A TPN $\mathcal{Z}$ with an *only* partially defined interval function I.
- A transition sequence $\sigma_1 = \sigma t_1$, where σ is a transition sequence and t_1 is a transition in $\mathcal{Z}$.
- A further transition sequence $\sigma_2 = \sigma t_2$, where t_2 is a transition in $\mathcal{Z}$ such that $t_1 \neq t_2$.

Output:
(1) Is it possible to complete I to a total function such that σ_1 is a firing sequence in $\mathcal{Z}$ and σ_2 is *not* a firing sequence $\mathcal{Z}$?

(2) A completed, totally defined function I, if the answer to (1) is *yes*.

Idea of the solution: We complete the function I as in the solution of the problem 3. Then we consider the linear inequality system B_σ. Afterwards we obtain the parametric states $z_{\sigma_1} = (\sigma_1(x), B_1)$ and $z_{\sigma_2} = (\sigma_2(y), B_2)$. In order to solve the problem 4 we have to find values for the variables a_{t_i}, b_{t_j}, $x_0, \ldots, x_{l(\sigma_1)}$ and $y_0, \ldots, y_{l(\sigma_2)}$ such that the following inequality holds:

$$\underbrace{\max\{\sum_{i=0}^{l(\sigma_1)} x_i \mid B_1\}}_{(P_1)} < \underbrace{\min\{\sum_{i=0}^{l(\sigma_2)} y_i \mid B_2\}}_{(P_2)}.$$

Thereby, the values for the variables a_{t_i}, b_{t_j} have to be rational numbers. The values for $x_0, \ldots, x_{l(\sigma_1)}$ and $y_0, \ldots, y_{l(\sigma_2)}$ can be arbitrary nonnegative real numbers. The complete proof can be found in [Pop07b].

Problem 4 gives an answer to the question "Is it possible to restrict the behaviour of a Petri net by the influence of time?", but only for the special case of branching behaviour. This special question is easy to answer, if the transitions t_1 and t_2 share an input place (i.e., they are in conflict): we define the lft of the transition, which should fire, as smaller than the eft of the transition, which is expected not to fire. If the transitions do not share an input place, then the solution is more complicated, as we have just seen.

Applications: Problem 4 deals with the phenomenon of time-dependent feasibility of transitions. In technical systems, this is generally not an expected behaviour. So the knowledge of the non-feasibility of a given transition under certain timing conditions might help to find a better suitable design. In biochemical systems, problem 4 supports, similar to problem 3, a model-based drug prescription. The here given scenario helps to find timing constraints in order to downgrade unwanted reactions.

4.3 Bounded Time Petri Nets

This section deals with bounded TPNs only. If the TPN is bounded (i.e., the set of reachable p-markings is finite) and strong (i.e., the lft's for all transitions are finite), then the set of all reachable integer states is finite, too. Therefore, properties of such TPNs can be proved using this finite set of integer states. For this purpose we define step-wise a suitable reachability graph $\mathcal{RG}_\mathcal{Z}$ of a TPN $\mathcal{Z}$. We start with $\mathcal{RG}_\mathcal{Z} := (W, E, L)$. The vertices W are the reachable integer states. The labelled edges E are defined by triples (z, t, z') and $(z, 1, z')$, with $z \xrightarrow{t} z'$ and $z \xrightarrow{1} z'$, respectively. L is a function, defining the edge labels, i.e., $cod(L) = T \cup \{1\}$. This graph is finite for a strong TPN, iff the set of the reachable p-markings of the net is finite. The construction of $\mathcal{RG}_\mathcal{Z}$ is straightforward. Starting at the initial state all integer state successors of a reached integer state are derived successively. The formal definition can be found in [Pop07b].

Instead of L we often give directly the set $cod(L)$, i.e., we write $\mathcal{RG}_\mathcal{Z} := (W, E, T \cup \{1\})$. The reachability graph defined as far can be reduced. (1) Removal of all (transient) vertices, having only time-labelled input and output edges. Each input edge is merged with each output edge, labelled by the sum of both labels. (2) Removal of all vertices, having only time-labelled input edges. Each input edge is merged with each output edge, labelled by both labels (of the input and of the output edge). The second reduction decreases the number of vertices, but increases the complexity of the labels of the new edges. We get $\mathcal{RG}_\mathcal{Z} = (W, E, T \cup T \times \mathbb{N})$. For more details see e.g. [Pop91] or [Pop07a].

This reachability graph is not finite, if there is a transition t with $lft(t) = \infty$, even if the net is bounded. Therefore, we modify consistently the notion reachability graph in the following way. The time after reaching the $eft(t)$ is actually not important for t. Thus, when the clock $h(t)$ reaches the eft, i.e. $h(t) = eft(t)$, it is not necessary anymore to increment the time for this transition (even though the time is going on). For the modified reachability graph it holds now: It is finite for an arbitrary TPN, iff the set of the reachable p-markings of the net is finite. A formal proof can be found in [Pop07a].

Let $\mathcal{Z} = (P, T, F, V, m_0, I)$ be a TPN and $\mathcal{RG}_\mathcal{Z} = (W, E, T \cup T \times \mathbb{N})$ its reduced reachability graph. This is a directed labelled graph, where each edge $k \in E$ is labelled by $\kappa_k = t$ or by $\kappa_k = t, n$. Using the labels of $\mathcal{RG}_\mathcal{Z}$ we derive weights for the edges as follows:

$$w(k) := \begin{cases} 0 & \text{, if } \kappa_k = t \\ n & \text{, if } \kappa_k = t, n \end{cases} \quad \text{for each } k \in E.$$

Using these weights, we finally transform the reachability graph $\mathcal{RG}_\mathcal{Z} = (W, E, T \cup T \times \mathbb{N})$ into the directed weighted graph $\mathcal{RG}^w_\mathcal{Z} := (W, E, w)$. Hence, to solve our problems we are now able

to use various efficient graph algorithms, valid for directed weighted graphs. We use the graph $\mathcal{RG}^w_{\mathcal{Z}}$ to compute the minimal and the maximal distance between two integer states or between two p-markings, respectively. The path, which realises the minimal (maximal) distance specifies a minimal (maximal) run in the TPN.

The algorithms used for studying the directed weighted graphs have linear or polynomial runtime w.r.t. the size of the graph $\mathcal{RG}^w_{\mathcal{Z}}$ and therefore w.r.t. the size of $\mathcal{RG}_{\mathcal{Z}}$. Nevertheless, it has not to be forgotten that the construction of the reachability graph needs at least exponential runtime w.r.t. the size of the TPN (cf. [Pri03]). See [Cor01] for the notions and basic algorithms, applied here without further explanation.

Problem 5

Input: • Two integer states z_1 and z_2, reachable in an arbitrary TPN $\mathcal{Z}$.

Output: (1) Is $z_2 \in RS_{\mathcal{Z}}(z_1)$?

(2) The minimal time distance from z_1 to z_2 as well as the corresponding minimal run, if the answer to (1) is *yes*.

Idea of the solution: Question (1) asks for a path from z_1 to z_2 in the graph $\mathcal{RG}^w_{\mathcal{Z}}$. Question (2) asks for the shortest path from a vertex to a second one in a weighted directed graph (under the condition that there is a path). This is immediately following from Remark 1. Both questions can be answered using the well-known Dijkstra algorithm, because all weights are nonnegative numbers. The algorithm terminates in polynomial time and yields the shortest path, if there is a path from z_1 to z_2, or the answer is given that there is no path from z_1 to z_2 (cf. *shortest-path algorithm* in [Cor01]). The minimal distance is the sum of the weights of the shortest path. □

Applications: See the discussion of applications for problem 6.

Problem 6

Input: • Two integer states z_1 and z_2, reachable in an arbitrary TPN $\mathcal{Z}$.

Output: (1) Is $z_2 \in RS_{\mathcal{Z}}(z_1)$?

(2) The maximal time distance from z_1 to z_2 as well as the corresponding maximal run, if the answer to (1) is *yes*.

Idea of the solution: Let the graph $\mathcal{G}_{\mathcal{Z}} := (W', E', w')$ be derived from the graph $\mathcal{RG}^w_{\mathcal{Z}}$ as follows:

$$W' := W, \quad E' := E \setminus \{\, (z_2, z) \mid (z_2, z) \in E \,\}, \quad w' := -w_{|E'} \ .$$

The graph $\mathcal{G}_{\mathcal{Z}}$ is derived from the graph $\mathcal{RG}^w_{\mathcal{Z}}$ by eliminating all output edges of z_2 and by multiplying all weights by -1. The elimination ensures that each path, which passes z_2 (in $\mathcal{RG}^w_{\mathcal{Z}}$) ends in z_2 (in $\mathcal{G}_{\mathcal{Z}}$). Now, a maximal path from z_1 to z_2 in $\mathcal{RG}^w_{\mathcal{Z}}$ is a minimal path in $\mathcal{G}_{\mathcal{Z}}$. That is true because it holts: $\max\{\, f(x) \mid x \in L \,\} = -\min\{\, -f(x) \mid x \in L \,\}$. In this case we cannot use the fast Dijkstra algorithm because of the negative weights. Nevertheless, the Bellman-Ford algorithm finds a minimal path in a graph with arbitrary weights. This algorithm is slower as the Dijkstra algorithm, but finds a minimal path still in polynomial time w.r.t. the size of the graph. However, the problem 6 can be solved in linear time w.r.t. the size of the graph $\mathcal{G}_{\mathcal{Z}}$, see [Pop07b].

Applications: Similarly to problem 2, the precise knowledge of the minimal and maximal time distance between two arbitrary system states supports the systematic construction of technical systems with predictable timing behaviour. But complementary to problem 2, problem 5 and 6 help in those situations, where time-critical system parts are best specified by their initial and final system state. However, often we will only know the p-markings of these two states, which calls for the problems 7 and 8.

Problem 7

Input: • Two p-markings m_1 and m_2, reachable in an arbitrary TPN $\mathcal{Z}$.

Output: (1) Is $m_2 \in R_{\mathcal{Z}}(m_1)$?

(2) The minimal time distance from m_1 to m_2 as well as the corresponding minimal run, if the answer to (1) is *yes*.

Idea of the solution: We consider the graph $\mathcal{RG}^w_{\mathcal{Z}}$ and define two sets of vertices M_1 and M_2:

$$M_i := \{\, z \mid z = (m_i, h),\, z \in RS_{\mathcal{Z}}\,, z - \text{integer state}\}$$

for $i = 1, 2$. The questions (1) and (2) are equivalent to the questions, asking for the existence of a path or a minimal path, respectively, from the set M_1 to the set M_2 in the graph $\mathcal{RG}^w_{\mathcal{Z}}$. The existence is decidable in polynomial time w.r.t. the size of the graph (cf. *all-pairs shortest paths algorithm* in [Cor01]). Using the all-pairs shortest paths algorithm we get the values d_{ij}, $i, j = 1, \ldots, |RIS_{\mathcal{Z}}|$ and

$$d_{ij} = \begin{cases} d_{\min}(z_i, z_j) & \text{, if } z_2 \in RS_{\mathcal{Z}}(z_1) \\ \infty & \text{, else} \end{cases} .$$

The value $d_{\min}(m_1, m_2)$ is the minimal number of a finite set of numbers:

$$d_{\min}(m_1, m_2) = \min \{\, d_{ij} \mid z_i \in M_1 \ \wedge \ z_2 \in M_2 \,\}.$$

Thus, the question (2) is solvable in polynomial time w.r.t. the size of the graph, too. □

Applications: Complementing the application scenario of problem 5, problem 7 allows to characterise the critical system parts of interest by their initial and final p-marking, while the t-marking can be arbitrarily chosen. The exact determination of the minimal time distance between two arbitrary p-markings supports the best-case prediction of the timing behaviour of technical systems [Hei97b] as well as of biochemical systems. Similarly, task scheduling according a given precedence graph is supported by finding the best-case execution time of all possible schedules [Pop05b]. See also problem 8.

Problem 8

Input:
- Two p-markings m_1 and m_2, reachable in an arbitrary TPN $\mathcal{Z}$.

Output: (1) Is $m_2 \in R_{\mathcal{Z}}(m_1)$?

(2) The maximal time distance from m_1 to m_2 as well as the corresponding maximal run, if the answer to (1) is *yes*.

Idea of the solution: Analogue to the idea of solution of problem 7 we consider again the sets M_i, $i = 1, 2$. Then, an *all-pairs shortest-paths* algorithm, applied to the weighted directed graph $\mathcal{G}_{\mathcal{Z}}$, yields the solution. □

Applications: Complementing the application scenario of problem 6, problem 8 allows to characterise the critical system parts of interest by their initial and final p-marking, while the t-marking can be arbitrarily chosen. The exact determination of the maximal time distance between two arbitrary p-markings supports the worst-case prediction of the timing behaviour of technical systems [Hei97b] as well as of biochemical systems, which is necessary to ensure given deadlines or reaction times. Similarly, task scheduling according a given precedence graph is supported by finding the worst-case execution time of all possible schedules [Pop05b]. See also problem 7.

Finally, a straightforward generalisation of the last four problems allows also a graph-based solution of the notions *minimal time distance* and *maximal time distance* from an integer state to a p-marking as well as from a p-marking to an integer state. All these four problems are solvable in polynomial or linear runtime, respectively. The problem to search for a minimal/maximal time distance from an integer state to a p-marking is equivalent to the *single-source shortest-path problem*, while the problem to search for a minimal/maximal time distance from a p-marking to an integer state is equivalent to the *single-destination shortest-path problem*.

Problems 5-8 require the specification of complete p-markings. However, often we only know p-submarkings, leading to a further generalisation of path search between two sets of states.

5 Conclusions

In this paper we have presented algorithms to analyse quantitatively a well-known extension of standard Petri nets, the Time Petri nets, where each transition gets a continuous time interval,

specifying the range of the transition's reaction time. The crucial point in any exhaustive analysis is an adequate state space reduction. Our approaches exploit basically two ideas: parametric state description and discretisation of the state space.

The first class of considered algorithms relies on a parametric description of the state space. We have shown that the solutions of the problems can be traced back to solutions of systems of linear inequalities or of linear programs, respectively. The total avoidance of any state space construction in this approach allows their application even to unbounded systems. The second class of considered algorithms takes advantage of the discretely reduced state space - the set of all reachable integer states. For this purpose, we have introduced the reachability graph of a bounded Time Petri net, which allows to re-use efficient algorithms of directed weighted graphs.

We have presented eight problems, characterised by their input/output relation, and we have shown how these problems can be solved using the proposed state space reduction techniques. Suggested application scenarios cover technical as well as biochemical systems.

References

[Cor01] Cormen, T.H. and Leiserson, Ch.E. and Rivest, R.L. and Stein, C. *Introduction to Algorithms*. MIT Press, second edition, 2001.

[GLS93] Grötschel, M., Lovász, L., and Schrijver, A. *Geometric Algorithms and Combinatorial Optimization*. Springer-Verlag, second corrected edition, 1993.

[Hei97a] Heiner, M. On Exploiting the Analysis Power of Petri nets for the Validation of Discrete Event Systems. In *Proc. 2nd IMACS Symposium on Mathematical Modelling (MATHMOD VIENNA '97), Wien, Feb. 1997*, pages 171–176, 1997.

[Hei97b] Heiner, M. and Popova-Zeugmann, L. Worst-case Analysis of Concurrent Systems with Duration Interval Petri Nets. In *Tagungsband zur 5. Fachtagung "'Entwurf komplexer Automatisierungssysteme"'*, 1997.

[Pap98] Papadimitriou, Ch. and Steiglitz, K. *Combinatorial Optimization: Algorithms and Complexity*. Dover Publications, Inc., Mineola, New York, 1998.

[Pop89] Popova-Zeugmann, L. *Time Petri Nets (in German)*. Ph.D.Thesis, Humboldt University at Berlin, 1989.

[Pop91] Popova, L. On Time Petri Nets. *J. Inform. Process. Cybern. EIK 27(1991)4*, pages 227–244, 1991.

[Pop99] Popova-Zeugmann,L. and Schlatter, D. Analyzing Path in Time Petri Nets. *Fundamenta Informaticae (FI) 37, IOS Press, Amsterdam*, pages 311–327, 1999.

[Pop05a] Popova-Zeugmann, L. and Heiner, M. and Koch, I. Time Petri Nets for Modelling and Analysis of Biochemical Networks. *Fundamenta Informaticae (FI) 67, IOS Press, Amsterdam*, pages 149–162, 2005.

[Pop05b] Popova-Zeugmann, L. and Werner, M. Extreme Runtimes of Schedules Modelled by Time Petri Nets. *Fundamenta Informaticae (FI) 67, IOS Press, Amsterdam*, pages 163–174, 2005.

[Pop07a] Popova-Zeugmann, L. Time Petri Nets State Space Reduction Using Dynamic Programming. *Journal of Control and Cybernetics*, 35(3):721–748, 2007.

[Pop07b] Popova-Zeugmann,L. *Time and Petri Nets (in German)*. Habilitation Thesis, Humboldt University at Berlin, 2007.

[Pri03] Priese, L. and Wimmel, H. *Theoretical Informatics, Petri Nets (in German)*. Springer-Verlag, Berlin Heidelberg New York, 2003.

Remark: The full version of this extended abstract contains a related work section.