Controlling Petri Net Behavior Using Time Constraints

Irina A. Lomazova[1]*, Louchka Popova-Zeugmann[2]

[1] National Research University Higher School of Economics (HSE), Moscow, 101000, Russia,
ilomazova@hse.ru
[2] Department of Computer Science, Humboldt University Berlin, Germany,
popova@informatik.hu-berlin.de

Abstract. In this paper we study how it is possible to control Petri net behavior using time constrains. Controlling here means forcing a process to behave in a stable way by associating time intervals to transitions and hence transforming a classic Petri net into a Time Petri net.
For Petri net models stability is often ensured by liveness and boundedness. These properties are crucial in many application areas, e.g. workflow modeling, embedded systems design, and bioinformatics. This paper deals with the problem of transforming a given live, but unbounded Petri net into a live and bounded one by adding time constraints. We specify necessary conditions for the solvability of this problem and present an algorithm for adding time intervals to net transitions in such a way that the resulting net becomes bounded while staying live.

1 Introduction

Distributed systems range in almost all areas: from technical systems to biological systems or to systems of business processes. Although such systems are very different in their subject matter they all have common properties, such as reiteration of all subprocesses or returning to some initialization in the system, or containing finitely or infinitely many different states etc. Petri nets are widely used for modeling and analysis of distributed systems. The first two properties concern the liveness of the model, the second two are the subject of boundedness studies. In most of the practical systems the infiniteness of all reachable states is an undesired property.

A typical example is a business process model, represented by a workflow net — a special kind of a Petri net. The essential property for workflow nets is soundness, also called proper termination [1]. Soundness is intensively studied in the literature [1, 2, 7, 11, 13, 16]. Checking soundness of workflow nets can be reduced to checking liveness and boundedness for the extended net obtained by connecting the source place with the sink place through a new transition in the initial workflow net. Thus ensuring liveness and boundedness of a model can be applied for asserting soundness of workflow nets. In biological systems liveness and boundedness ensure system stability [8, 9]. In embedded systems scheduling is often necessary due to the resource limitations [10].

In practice it may often happen that a given live Petri net is not bounded. Then it would be helpful to "repair" the model by adding priorities or time to transitions so, that the net becomes bounded staying live. In other words, the question is whether we can "repair" the model with the help of priority or time constrains. In [5] J. Desel proposed an approach for a brute-force-scheduling to ensure bounded behavior, employing transitions of a given subset infinitely often. Here we study when and how time constraints can ensure boundedness of a given live Petri net, retaining its liveness. In contrast to brute-force approach, time constraints allow local and more flexible control — not just forcing one 'good' execution.

In [12] we have considered a way to add priorities in order to transform a live and unbounded model, represented by a Petri nets, into a live and bounded one. For this reason we have defined a sub-tree of the reachability tree, the so-called "spine tree". The spine tree contains all minimal cyclic runs together with prefixes leading to the cycles. In this paper we use the spine-based

* This study was supported by the National Research University Higher School of Economics' Academic Fund.

coverability tree, derived from the spine tree, in order to add time intervals to the transitions in such a way that the resulting Time Petri net is live and bounded. Of course, this is not always possible.

Priority scheduling is an appropriate solution for workflow systems. For biological systems it is not so good, since there is no mechanism to assign priorities to events in biological systems. Scheduling biological systems with the help of time constraints would provide a much more natural solution [18].

In this work we show how to associate time constrains to a given live and unbounded Petri net in such a way that the resulting time-dependent net is live and bounded. Thereby we want to fully preserve the structure of the given net. For that we use the spine tree and the spine-coverability tree, introduced in [12] and compute a parametrical state space for a Time Petri net, for which the underlying timeless Petri net is the given one.

The paper is organized as follows. In Section 2 we first give a more detailed motivation for this work and then recall some basic definitions in the theory of Petri nets. Section 3 presents the main contributions of the paper: a sufficient condition for transforming a live and unbounded Petri net into a live and bounded one by adding time intervals and an algorithm for computing these intervals, as well as, an example illustrating the algorithm. Section 5 contains some conclusions.

2 Preliminaries

2.1 Motivation

Liveness in bounded Petri nets is considered in numerous works. In [19] Ridder and Lautenbach considered the relationship between liveness in bounded nets and T-invariants. For marked graph Petri nets characterization of liveness and boundedness in terms of reachability graphs was done in [3]. Scedulability analysis of Petri nets, aimed at ensuring infinitely repeated firing sequences within a bounded stated space, was studied in [10].

In [9] and [8] M. Heiner considered the problem of transforming live and unbounded Petri nets into live and bounded nets by adding time durations to transitions. It is shown in these works, that when a Petri net is covered by T-invariants (i.e. each transition enters into at least one T-invariant with a non-zero component), T-invariants can be used for computing time durations for transitions, making the net bounded. In other words, this method allows to transform a live and unbounded Petri net, covered by T-invariants, into a live and bounded Timed Petri net [18] with the same structure. Unfortunately this method does not always work, as it was shown in [9].

Furthermore, as it is shown in [12], a possibility to transform a live and unbounded net into a bounded one can depend not only on T-invariants, but on initial markings as well. So, the algorithm for making a live Petri net also unbounded with the help of transition priorities, which we presented in [12], essentially takes into account initial states. This is the case also for the algorithm assigning time intervals to the transitions and represented in this article.

Live and unbounded Petri nets were considered also in [6]. The notion of *weak boundedness* was introduced there. A Petri net $\mathcal{N}$ is called weakly bounded, iff it is unbounded, but for every reachable marking m in $\mathcal{N}$ a bounded run is enabled in m, i.e. from every reachable marking we can find a way to continue the execution in such a way, that the number of tokens in each place will be not greater that some fixed value. The distinction between bounded, weakly bounded and not weakly bounded Petri nets is very important for applications. However, till now, there is no algorithm for distinguishing weakly bounded and not weakly bounded Petri nets. There is a reason to believe that these notions are connected with a possibility to transform an unbounded Petri net into a bounded one by adding some time, or priority constraints.

2.2 Basics

Let $\mathbb{N}$ denote the set of natural numbers (including zero) and let $\mathbb{Q}_0^+$ be the set of all non-negative rational numbers including zero. All notions and notations used here are generally known and can be found in [4].

Let P and T be disjoint sets of *places* and *transitions* with $P \cup T \neq \emptyset$ and let $F \subseteq (P \times T) \cup (T \times P) \to \mathbb{N}$ be a *flow relation*. Then $\mathcal{N} = (P, T, F)$ is a *(unmarked) Petri net*. A *marking* in a Petri net is a function $m : P \to \mathbb{N}$, mapping each place to some natural number (possibly zero). A (marked) *Petri net* $(\mathcal{N}, m_0)$ is an unmarked Petri net $\mathcal{N}$ with its initial marking m_0. Further we call marked Petri nets just Petri nets and use vector notation for marking by fixing some ordering of places in a Petri net.

Pictorially, P-elements are represented by circles, T-elements by boxes, and the flow relation F by directed arcs. Places may carry tokens represented by filled circles. A current marking m is designated by putting $m(p)$ tokens into each place $p \in P$.

For a transition $t \in T$ an arc (x, t) is called an *input arc*, and an arc (t, x) – an *output arc*.

A transition $t \in T$ is *enabled* in a marking m iff $\forall p \in P\ m(p) \geq F(p, t)$. An enabled transition t may *fire* yielding a new marking m', such that $m'(p) = m(p) - F(p, t) + F(t, p)$ for each $p \in P$ (denoted $m \xrightarrow{t} m'$, or just $m \to m'$). Then we say that a marking m' is *directly reachable* from a marking m.

A marking m is called *dead* iff it enables no transition.

A *run* in $\mathcal{N}$ is a finite or infinite sequence of firings $m_1 \xrightarrow{t_1} m_2 \xrightarrow{t_2} \dots$. An *initial run* in $(\mathcal{N}, m_0)$ is a run, starting from the initial marking m_0. A *cyclic run* is a finite run starting and ending at the same marking. A *maximal run* is either infinite, or ends with a dead marking.

We say that a marking m is *reachable from a marking* m' in $\mathcal{N}$ iff there is a run $m' = m_1 \to m_2 \to \dots \to m_n = m$; m is *reachable in* $(\mathcal{N}, m_0)$ iff m is reachable from the initial marking. By $\mathcal{R}(\mathcal{N}, m)$ we denote the set of all markings reachable in $\mathcal{N}$ from the marking m. A run σ in $(\mathcal{N}, m)$ is called *feasible* iff σ starts from a reachable marking.

A T-invariant in a Petri net with n transition $t_1, \dots, t_n$ is an n-dimensional vector $\alpha = (\alpha_1, \cdots, \alpha_n)$ with $\alpha_i \in Nat$ such that after firing of every transition sequence containing exactly α_i occurrences of each transition t_i in an arbitrary marking m (if possible) leads to the same marking m.

A *reachability graph* of a Petri net $(\mathcal{N}, m_0)$ presents detailed information about the net behavior. It is a labeled directed graph, where vertices are reachable markings in $(\mathcal{N}, m_0)$, and an arc labeled by a transition t leads from a vertex v, corresponding to a marking m, to a vertex v', corresponding to a marking m' iff $m \xrightarrow{t} m'$ in $\mathcal{N}$.

A reachability graph may be also represented in the form of a *reachability tree*, which can be defined in a constructive form. We start from the initial marking as a root. If for a current leaf v labeled with a marking m, there is already a node $v' \neq v$ lying on the path from the root to v and labeled with the same marking m, we notify v to be a leaf in the reachability tree. If not, nodes directly reachable from m and the corresponding arcs are added. Note, that in a reachability tree run cycles are represented by finite paths from nodes to leafs.

A place p in a Petri net is called *bounded* iff for every reachable marking the number of tokens residing in p does not exceed some fixed bound $\kappa \in \mathbb{N}$. A marked Petri net is bounded iff all its places are bounded.

It is easy to see, that a Petri net $(\mathcal{N}, m_0)$ is bounded iff its reachability set $\mathcal{R}(\mathcal{N}, m_0)$, and hence its reachability graph, are finite.

A marking m' *covers* a marking m (denoted $m' \geq m$) iff for each place $p \in P$, $m'(p) \geq m(p)$. The relation $\geq$ is a partial ordering on markings in N. By the firing rule for Petri net, if a sequence of transitions is enabled in a marking m, and $m' \geq m$, then this sequence of transitions is also enabled also in m'. A marking m' *strictly covers* a marking m (denoted $m' > m$) iff $m' \geq m$ and $m \neq m$. redundantly For an unbounded Petri net, a *coverability tree* gives a partial information about the net behavior. It uses the notion of a *generalized marking*, where the special symbol ω designates an arbitrary number of tokens in a place. Formally, a generalized marking is a mapping $m : P \to \mathbb{N} \cup \{\omega\}$. A coverability tree is defined constructively. It is started from the initial marking and is successively constructed as a reachability tree. The difference is that when a marking m' of a current leaf v' in a reachability tree strictly covers a marking m of a node v, lying on the path from the root to v', then in a coverability tree the node v' obtains a marking m_ω, where $m_\omega(p) = \omega$, if $m'(p) > m(p)$, and $m_\omega(p) = m'(p)$, if $m'(p) = m(p)$. For generalized markings enabling of a transition and a firing rule is defined as for usual markings except that ω-marked

places are ignored. Each place p, which was marked by ω, remains ω-marked for all possible run continuations.

Let $\mathcal{N} = (P, T, F)$ be an unmarked Petri net and let $I : T \longrightarrow \mathbb{Q}_0^+ \times (\mathbb{Q}_0^+ \cup \{\infty\})$ be a function such that for each $t \in T$ holds: $I(t) = (a_t, b_t)$ and $a_t \leq b_t$. Thus, the function I associates an interval $[a_t, b_t]$ with each transition t in T. We notate a_t with *eft*(t) (earliest firing time for t) and b_t with *lft*(t) (latest firing time for t).

Here a_t and b_t are relative to the time, when t was enabled last. 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 takes no time. The time interval is designed by real numbers, but the interval bounds are nonnegative rational numbers. It is easy to see (cf. [18]) that w.l.o.g. the interval bounds can be considered as integers only. Thus, the interval bounds a_t and b_t of any transition t are natural numbers, including zero and $a_t \leq b_t$ or $b_t = \infty$. A comprehensive introduction can be found in [18].

$\mathcal{Z} = (\mathcal{N}, m_0, I)$ is called Time Petri net (TPN) and it was first introduced by Merlin [14]. The marked Petri net $(\mathcal{N}, m_0) := S(\mathcal{Z})$ is called the skeleton and I - the interval function of $\mathcal{Z}$.

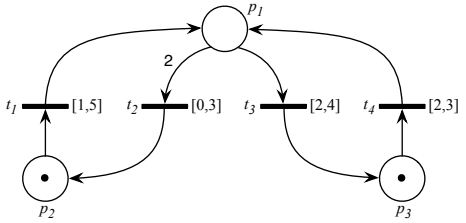

Fig. 1. The Time Petri net $\mathcal{Z}$

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, which is a place vector (i.e. the vector has as many components as places in the considered TPN), is defined as the marking notion in classic Petri nets. The time marking, which is a transition vector (i.e. the vector has as many components as transitions in the considered TPN), describes the time circumstances in the considered situation. In general, each TPN has infinite number of states.

The state space of a TPN can be characterized parametrically and it is shown that knowledge of the integer-states, i.e. states whose time markings are (nonnegative) integers, is sufficient to determine the entire behavior of the net at any point in time (cf. [17] and [18]. Thus, a reachability graph for a TPN can be defined so that the nodes of the graph are the reachable integer-states and a directed edge connects two nodes, from z_1 to z_2 if there is possible to change from z_1 to z_2, considered as states in the TPN. And finally, a reachability tree can be then also defined for TPN considering the reachable p-markings.

In this paper we use the parametric states in order to restrict the behavior of a live and unbounded PN to a live and bounded one. The notions *parametric state* and *parametric run* can be easily defined by recursion. Let $\mathcal{Z} = (P, T, F, V, m_0, I)$ be a Time Petri net and let $\sigma = t_1 \cdots t_n$ be a firing sequence in $\mathcal{Z}$. Then, the parametric run $(\sigma(x), B_\sigma)$ of σ in $\mathcal{Z}$ with $\sigma(x) = x_0 t_1 x_1 \cdots x_{n-1} t_n x_n$ and the parametric state (z_σ, B_σ) in $\mathcal{Z}$ are recursively defined as follows:

Basis: $\sigma = \varepsilon$, i.e., $\sigma(x) = x_0$.

Then $z_\sigma = (m_\sigma, h_\sigma)$ and B_σ are defined as follows:

1. $m_\sigma := m_o$,
2. $h_\sigma(t) := \begin{cases} x_0 & \text{if } t^- \leq m_\sigma \\ \sharp & \text{otherwise} \end{cases}$,
3. $B_\sigma := \{\ 0 \leq h_\sigma(t) \leq \mathit{lft}(t) \ \mid \ t \in T \ \wedge \ t^- \leq m_\sigma \ \}$

Step: Assume that z_σ and B_σ are already defined for the sequence $\sigma = t_1 \cdots t_n$.
For $\sigma = \underbrace{t_1 \cdots t_n}_{:=w}\ t_{n+1} = w t_{n+1}$ we set

1. $m_\sigma := m_w + \Delta t_{n+1}$,

2. $h_\sigma(t) := \begin{cases} \sharp & \text{if } t^- \not\leq m_\sigma \\ h_w(t) + x_{n+1} & \text{if } t^- \leq m_\sigma \ \wedge\ t^- \leq m_w \ \wedge \\ & \quad {}^\bullet t_{n+1} \cap {}^\bullet t = \emptyset \\ x_{n+1} & \text{otherwise} \end{cases}$,

3. $B_\sigma := B_w \ \cup\ \{\ \mathit{eft}(t_{n+1}) \leq h_w(t_{n+1})\ \} \ \cup\ \{\ 0 \leq h_\sigma(t) \leq \mathit{lft}(t)\ \mid\ t \in T \wedge t^- \leq m_\sigma\ \}$.

A short example should illustrate the calculation of parametric states. We use K_σ as a shorthand for $\{z_\sigma \mid B_\sigma\}$.

Let us consider the Time Petri net $\mathcal{Z}$ in Fig. 1.

It is easy to see that

$$K_\varepsilon = \{(\underbrace{(0,1,1)}_{m_\varepsilon}, \underbrace{(x_0, \sharp, \sharp, x_0)}_{h_\varepsilon}) \mid \underbrace{\{0 \leq x_0 \leq 3\}}_{B_\varepsilon}\}.$$

After firing the sequence $\sigma = t_4$ the net $\mathcal{Z}_2$ is in a state belonging to $K_\sigma = K_{t_4}$.

$$K_{t_4} = \{(\underbrace{(1,1,0)}_{m_{t_4}}, \underbrace{(x_0 + x_1, \sharp, x_1, \sharp)}_{h_{t_4}}) \mid \underbrace{\{2 \leq x_0 \leq 3, x_0 + x_1 \leq 5, 0 \leq x_1 \leq 4\}}_{B_{t_4}}\}.$$

The set of conditions B_{t_4} is the union of the three sets

$$B_\varepsilon,\ \{\mathit{eft}(t_4) \leq h_\varepsilon(t_4)\} = \{2 \leq x_0\} \text{ and } \{0 \leq h_\sigma(t) \leq \mathit{lft}(t)\ \mid\ t^- \leq m_\sigma\} = \left\{ \begin{matrix} x_0 + x_1 \leq 5, \\ 0 \leq x_1 \leq 4 \end{matrix} \right\}.$$

By repeatedly firing the transitions t_3 and t_4 we obtain the parametric states $z_{t_4 t_3}$ and $z_{t_4 t_3 t_4}$ and $K_{t_4 t_3}$ and $K_{t_4 t_3 t_4}$:

$$K_{t_4 t_3} = \{((0,1,1), (x_0 + x_1 + x_2, \sharp, \sharp, x_2)) \mid \begin{matrix} 2 \leq x_0 \leq 3, & x_0 + x_1 \leq 5, \\ 2 \leq x_1 \leq 4, & x_0 + x_1 + x_2 \leq 5, \} \\ 0 \leq x_2 \leq 3 \end{matrix}$$

$$K_{t_4 t_3 t_4} = \{((1,1,0), (x_0 + x_1 + x_2 + x_3, \sharp, x_3, \sharp)) \mid \begin{matrix} 2 \leq x_0 \leq 3, & 2 \leq x_1 \leq 4, & 2 \leq x_2 \leq 3, \\ 0 \leq x_3 \leq 4, & x_0 + x_1 \leq 5, & x_0 + x_1 + x_2 \leq 5, \} \\ & x_0 + x_1 + x_2 + x_3 \leq 5 \end{matrix}.$$

Obviously, some of the inequalities are redundant. For instance, the inequalities of the set $B_{t_4 t_3 t_4}$ can be reduced to the set

$$\left\{ \begin{matrix} 2 \leq x_0 \leq 3, & 2 \leq x_1 \leq 4, & 2 \leq x_2 \leq 3, \\ 0 \leq x_3 \leq 4, & x_0 + x_1 + x_2 + x_3 \leq 5 \end{matrix} \right\}.$$

In general, the number of inequalities in B_σ is at most $\min\{2 \cdot \big(n \cdot |T| + 1\big), (n+1) \cdot (\frac{n}{2} + 2)\}$ (cf. [18]).

Liveness can be defined in several ways for Petri nets [15]. We will use the standard "L4-live" variant, which states that every transition in a PN is potentially enabled in any reachable marking. More exactly, a transition t in a Petri net $(\mathcal{N}, m_0)$ is called *live* in $(\mathcal{N}, m_0)$ iff for every reachable marking m in $(\mathcal{N}, m_0)$ there exists a sequence of firings starting from m, which includes t.

3 Time Constrains for Boundedness

Let $(\mathcal{N}, m_0)$ be a live and unbounded Petri net. We would like to check, whether it is possible to make this net bounded, not losing its liveness, by transforming it into a Time Petri net (with the same skeleton), i.e. by adding intervals to its transitions. To solve this problem we will associate transition intervals (if possible), which would exclude runs leading to unboundedness.

We start by recalling some properties of live and bounded Petri nets considered in our article [12]. Then, instead to assign priorities to the transitions we add intervals. We will show that this time solution is more precise than the solution with priorities. In this connection "more precise" means that the set of reachable markings in the time-dependent net (sometimes properly) covers the set of reachable markings in the net with priorities.

It is clear that if a live PN is bounded then there exists a feasible cyclic run, including all transitions of the PN. Furthermore, a TPN is obtained from a PN by a adding time interval to each transition. Then the reachability tree is a subgraph of the reachability tree of the PN. Hence, it is obvious that if for some interval function I the TPN $\mathcal{Z} = (\mathcal{N}, m_0, I)$ is live and bounded, then there exists a feasible cyclic run in $(\mathcal{N}, m_0)$ which includes all transitions of $\mathcal{N}$.

Proposition 1. *Let $(\mathcal{N}, m_0)$ be a live and unbounded Petri net. If there exists an interval function I such that the TPN $(\mathcal{N}, I, m_0)$ is live and bounded, then there exists a T-invariant without zero components for $\mathcal{N}$, i.e. all transitions in $\mathcal{N}$ are covered by some T-invariant.*

As we already set in [12], given a live and unbounded Petri net $(\mathcal{N}, m_0)$, before looking for times, which would transform the net into a bounded (and still live) Petri net, it makes sense first to check necessary conditions. First one could compute T-invariants for the net $\mathcal{N}$. If there is no T-invariant, covering all transitions in $\mathcal{N}$, then the net cannot be recovered, i.e., the net cannot become live due to adding time, priorities etc.

If there is such a T-invariant, then a more strong necessary condition can be checked: whether there exists a feasible cyclic run in $(\mathcal{N}, m_0)$, which includes all transitions in $\mathcal{N}$, i.e. a cyclic run realizing one of T-invariants with non-zero components. To do this check the algorithm, proposed in [5] by J. Desel, can be used. This algorithm is based on constructing a coverability net — a special extension of a coverability graph, and can take an exponential time. However, if a net does not have too much concurrency and a small number of unbounded places, this method can be acceptable.

Now let $(\mathcal{N}, m_0)$ be a live and unbounded Petri net, and let the above necessary conditions are satisfied. We would like to find time intervals for the transition that will make the net bounded, keeping its liveness. The procedure will be illustrated by the net $(\mathcal{N}^*, m_0^*)$ in Fig. 2.

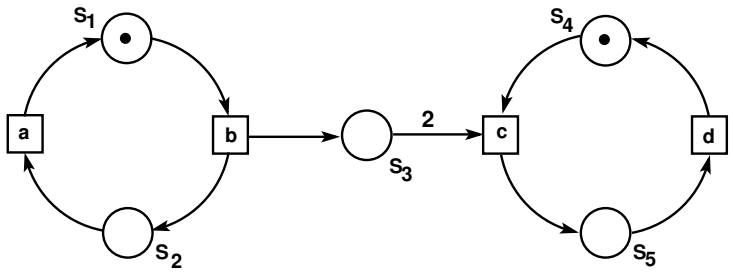

Fig. 2. An example of a live and unbounded marked Petri net $(\mathcal{N}^*, m_0^*)$.

The following algorithm is a modification of the algorithm given in [12]. Here reason we use the stages 1,2 and 3 of the 'old' algorithm and add new stages 4 and 5.

Stage 1. Find all minimal feasible cycles, which include all transitions. As already mentioned, this can be done by the technique described by J. Desel in [5]. Moreover, following this technique for each minimal feasible cyclic run σ we can simultaneously find a finite initial run τ, such that $\tau\sigma^\star$ is an initial run in $(\mathcal{N}, m_0)$.

If (N, m_0) does not have such cycles, then the problem does not have a solution. So, let

$$\begin{aligned}\mathcal{C}(\mathcal{N}, m_0) := \{\, \tau\sigma \,|\, & \tau\sigma^\star \text{ is an initial run in } (\mathcal{N}, m_0),\\ & \tau \text{ does not include } \sigma \text{ and}\\ & \sigma \text{ includes all transitions in } \mathcal{N} \,\}\end{aligned}$$

be a set of all minimal feasible cyclic runs together with prefixes leading to the cycles.

Thus, for example, the net $(\mathcal{N}^*, m_0^*)$ in Fig. 2 has five minimal cyclic runs with all transitions. Three of them have empty prefixes, and two have prefixes $\tau_1 = b$ and $\tau_2 = ba$, respectively:

$$\begin{array}{l} babcda \\ babcad \\ babacd \end{array} \quad \text{and} \quad \begin{array}{l} \overbrace{b}^{\tau_1=}abacbd \\ \underbrace{ba}_{\tau_2=}bacbad \end{array}.$$

Stage 2. Construct a spine tree. A *spine tree* is a subgraph of a reachability tree, containing exactly all runs from $\mathcal{C}(\mathcal{N}, m_0)$.

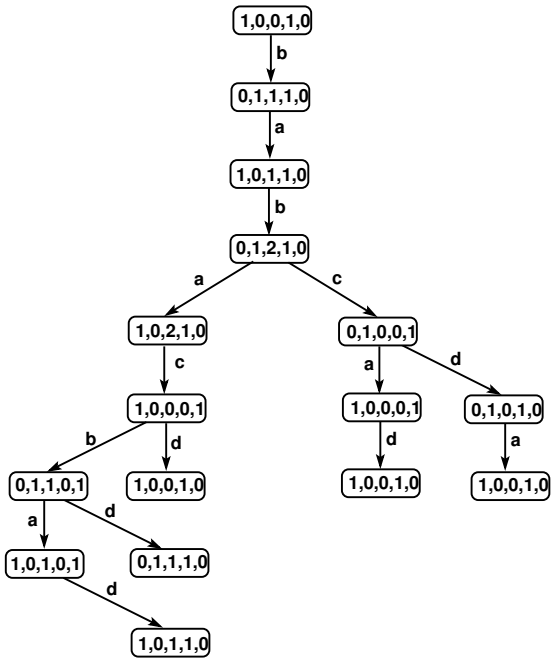

Fig. 3. The spine tree for the net $(\mathcal{N}^*, m_0^*)$.

The spine tree for Petri net $(\mathcal{N}, m_0)$ from our example is shown in Fig. 3.

Note, that a spine tree contains the behavior that should be saved to keep a Petri net live.

Stage 3. Construct a spine-based coverability tree. A *spine-based coverability tree* is a special kind of a coverability tree, that includes a spine tree as a backbone. Leaves in a spine-based coverability tree will be additionally colored with green or red. This coloring will be used then for computing transition priorities.

The spine-based coverability tree for a Petri net $(\mathcal{N}, m_0)$ is defined constructively by the following algorithm:

Step 1. Start with the spine tree for $(\mathcal{N}, m_0)$. Color all leaves in the spine tree in green.

Step 2. Repeat until all nodes are colored:

For each uncolored node v labeled with a marking m:

1. check whether there is a marking m', directly reachable from m and not included in the current tree. For each such marking m', where $m \xrightarrow{t} m'$:
   (a) Add a node v' labeled with m' as well as the corresponding arc from v to v' labeled with t.
   (b) If the marking m' strictly covers a marking in some node on the path from the root to v', then v' becomes a leaf and gets the red color.
   (c) Otherwise, if the marking m' coincides with a marking labeling some node on the path from the root to v', then v' becomes a leaf and gets the green color.

(d) Otherwise, leave v' uncolored.
2. Color the node v in yellow.

The spine-based coverability tree for our example net $(\mathcal{N}^*, m_0^*)$ is shown in Fig. 4. Here node colors are used to illustrate the tree construction. A leaf and some inner node have the same color, if they have the same markings, or the leaf marking strictly covers the marking of its ancestor. Strictly covering leaves are marked with the ω-symbol, they are 'red' leaves. All other leaves are 'green' leaves.

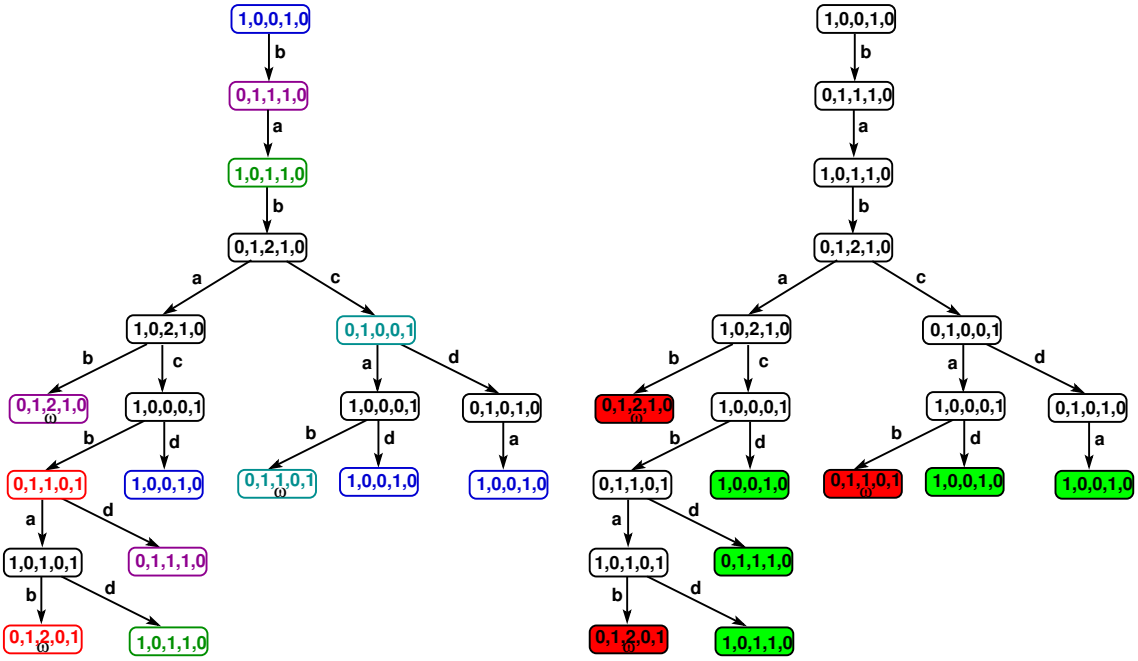

Fig. 4. The spine-based coverability tree for $(\mathcal{N}^*, m_0^*)$: **Left:** A leaf and some inner node have the same color, if they have the same markings, or the leaf marking strictly covers the marking of its ancestor. Strictly covering leaves are marked with the ω-symbol. **Right:** The spine-based coverability tree for $(\mathcal{N}^*, m_0^*)$ after finishing stage 3.

Stage 4. Compute a parametric state space. Let $\mathcal{T}$ be a spine-based coverability tree. Consider the TPN $(\mathcal{N}, m_0, I)$ with time interval $[a_t, b_t]$ for each transition $t \in T$. All a_t, b_t are unknown and have to be calculated in stage 5. By the construction of $\mathcal{T}$, all its leaves are colored either in green, or red. In this stage we construct an interval function $I : T \to \mathbb{Q}_0^+ \times \mathbb{Q}_0^+$. For this we consider every path from the root to a green leaf as a parametric run. Additionally, we forbid a branching to a red leaf using strict inequality.

(1) Let v_g be a green leaf and let σ be the path from the root to this leaf. Consider the parametric run $(\sigma(x), B_\sigma)$.

(2) Let v_r be a red leaf. Consider the path σ from the root to this leaf. Let v^* be the youngest ancestor of v_r such that at least one run goes from v^* to a green leaf v_g. The initial node v_0 is labeled with the marking m_0. Let the node v^* be labeled with m^*, v_r with m_r and v_g with m_g. Finally, let σ^* be the path from the root to the node v^*, σ_r the path from the node v^* to v_g and $t_r\sigma_r$ the path from the node v^* to v_r. That means, we have the situation

$$m_0 \xrightarrow{\sigma^*} m^* \xrightarrow{\sigma_g} v_g, \quad m_0 \xrightarrow{\sigma^*} m^* \xrightarrow{t_r} \xrightarrow{\sigma_r} v, \quad \sigma = \sigma^* t_r \sigma_r$$

and there is not a path from a node in σ_r to a green one. Hence, using time, we will forbid the firing of the transition t_r in m^*. For this reason we add to B_{σ^*} the constrain (strong inequality) $h_{\sigma^*}(t_r) < a_{t_r}$.

Stage 5. Compute a parametric state space. Let

$$B := \bigcup \{B_\sigma \mid \sigma \text{ is an initial run to a green node}\} \cup \{0 \le a_t \le b_t \mid t \in T\}.$$

B is the set of all constrains which have to be fulfilled in order to keep the live behavior of the net and to forbid all transition sequences leading to unboundedness. Clearly, B is a system of linear inequalities and it can be solved in $\mathbb{Q}_0^+$. Actually, we are interested in finding solutions for all a_t's and b_t's such that the resulting system of inequalities is solvable. Of course, when we can find rational values for solutions for all a_t's and b_t's then we can find also integer values for them.

At this point we would like to notice that instead of the full parametric space of the spine-based coverability tree we can use only a part of them, consisting of paths including together all transitions of the net. In this case it is possible that some markings which does not lead to unboundedness in the PN will be not reachable in the TPN.

Applying the procedure of the stages 4 and 5 to the spine-based coverability tree for our example net $(\mathcal{N}^*, m_0^*)$ (cf. Fig. 4) we sequentially obtain the following parametric state space and eventually the following inequality system B:

$K_\varepsilon = \{\big(\ (1,0,0,1,0),(\sharp,x_0,\sharp,\sharp)\ \big) \mid 0 \le x_0 \le b_b\}$,
$K_b = \{\big(\ (0,1,1,1,0),(x_1,\sharp,\sharp,\sharp)\ \big) \mid a_b \le x_0 \le b_b,\ 0 \le x_1 \le b_a\}$,
$K_{ba} = \{\big(\ (1,0,1,1,0),(\sharp,x_2,\sharp,\sharp)\ \big) \mid a_b \le x_0 \le b_b,\ a_a \le x_1 \le b_a\ 0 \le x_2 \le b_b\}$,

$$K_{bab} = \left\{ \big(\ (0,1,2,1,0),(x_3,\sharp,x_3,\sharp)\ \big) \,\middle|\, \begin{array}{l} a_b \le x_0 \le b_b,\ a_a \le x_1 \le b_a,\ a_b \le x_2 \le b_b, \\ 0 \le x_3 \le b_a,\ x_3 \le b_c \end{array} \right\},$$

$$K_{babc} = \left\{ \big(\ (0,1,0,0,1),(x_3+x_4,\sharp,\sharp,x_4)\ \big) \,\middle|\, \begin{array}{l} a_b \le x_0 \le b_b,\ a_a \le x_1 \le b_a,\ a_b \le x_2 \le b_b, \\ a_c \le x_3 \le b_c,\ x_3+x_4 \le b_a, 0 \le x_4 \le b_d \end{array} \right\},$$

$$K_{babcd} = \left\{ \big(\ (0,1,0,1,0),(x_3+x_4+x_5,\sharp,\sharp,\sharp)\ \big) \,\middle|\, \begin{array}{l} a_b \le x_0 \le b_b,\ a_a \le x_1 \le b_a,\ a_b \le x_2 \le b_b, \\ a_c \le x_3 \le b_c,\ a_d \le x_4 \le b_d,\ 0 \le x_5 \\ x_3+x_4+x_5 \le b_a, \end{array} \right\},$$

$$K_{babcda} = \left\{ \big(\ (1,0,0,1,0),(\sharp,x_6,\sharp,\sharp)\ \big) \,\middle|\, \begin{array}{l} a_b \le x_0 \le b_b,\ a_a \le x_1 \le b_a,\ a_b \le x_2 \le b_b, \\ a_c \le x_3 \le b_c,\ a_d \le x_4 \le b_d,\ 0 \le x_5 \\ a_a \le x_3+x_4+x_5 \le b_a,\ 0 \le x_6 \le b_b \end{array} \right\},$$

$$K_{babca} = \left\{ \big(\ (1,0,0,0,1),(\sharp,x_7,\sharp,x_7)\ \big) \,\middle|\, \begin{array}{l} a_b \le x_0 \le b_b,\ a_a \le x_1 \le b_a,\ a_b \le x_2 \le b_b, \\ a_c \le x_3 \le b_c,\ a_a \le x_3+x_4 \le b_a, 0 \le x_4 \le b_d \\ 0 \le x_7 \le b_d,\ x_7 < a_b \end{array} \right\},$$

Here we have add the strong inequality $x_7 < a_b$ in order to forbid the firing of the transition b in the marking $(1,0,0,0,1)$.

$$K_{babcad} = \left\{ \big(\ (1,0,0,1,0),(\sharp,x_7+x_8,\sharp,\sharp)\ \big) \,\middle|\, \begin{array}{l} a_b \le x_0 \le b_b,\ a_a \le x_1 \le b_a,\ a_b \le x_2 \le b_b, \\ a_c \le x_3 \le b_c,\ a_a \le x_3+x_4 \le b_a,\ 0 \le x_4 \le b_d \\ a_d \le x_7 \le b_d,\ x_7 < a_b,\ 0 \le x_8,\ x7+x_8 \le b_b \end{array} \right\},$$

$$K_{baba} = \left\{ \big(\ (1,0,2,1,0),(\sharp,x_9,x_3+x_9,\sharp)\ \big) \,\middle|\, \begin{array}{l} a_b \le x_0 \le b_b,\ a_a \le x_1 \le b_a,\ a_b \le x_2 \le b_b, \\ a_a \le x_3 \le b_a,\ x_3+x_9 \le b_c,\ 0 \le x_9 \le b_b, \\ x_9 < a_b \end{array} \right\},$$

$$K_{babac} = \left\{ \big(\ (1,0,0,0,1),(\sharp,x_9+x_{10},\sharp,x_{10})\ \big) \,\middle|\, \begin{array}{l} a_b \le x_0 \le b_b,\ a_a \le x_1 \le b_a,\ a_b \le x_2 \le b_b, \\ a_a \le x_3 \le b_a,\ a_c \le x_3+x_9 \le b_c,\ 0 \le x_9 \le b_b, \\ x_9 < a_b,\ 0 \le x_{10} \le b_d,\ x_9+x_{10} \le b_b \end{array} \right\},$$

$$K_{babacd} = \left\{ \big(\ (1,0,0,1,0),(\sharp,x_9+x_{10}+x_{11},\sharp,\sharp)\ \big) \,\middle|\, \begin{array}{l} a_b \le x_0 \le b_b,\ a_a \le x_1 \le b_a,\ a_b \le x_2 \le b_b, \\ a_a \le x_3 \le b_a,\ a_c \le x_3+x_9 \le b_c, \\ 0 \le x_9 \le b_b, x_9 < a_b,\ a_d \le x_{10} \le b_d, \\ x_9+x_{10}+x_{11} \le b_b,\ 0 \le x_{11} \end{array} \right\},$$

$$K_{babacb} = \left\{ \big(\ (0,1,1,0,1),(x_{12},\sharp,\sharp,x_{10}+x_{12})\ \big) \,\middle|\, \begin{array}{l} a_b \le x_0 \le b_b,\ a_a \le x_1 \le b_a,\ a_b \le x_2 \le b_b, \\ a_a \le x_3 \le b_a,\ a_c \le x_3+x_9 \le b_c,\ 0 \le x_9 \le b_b, \\ x_9 < a_b,\ 0 \le x_{10} \le b_d,\ x_9+x_{10}+x_{11} \le b_b \\ 0 \le x_{12} \le b_a,\ x_{10}+x_{12} \le b_d \end{array} \right\},$$

$$K_{babacba} = \left\{ \left(\ (1,0,1,0,1), (\sharp, x_{13}, \sharp, x_{10}+x_{12}+x_{13}) \ \right) \middle| \begin{array}{l} a_b \le x_0 \le b_b,\ a_a \le x_1 \le b_a, \\ a_b \le x_2 \le b_b,\ a_a \le x_3 \le b_a, \\ a_c \le x_3 + x_9 \le b_c,\ 0 \le x_9 \le b_b, \\ x_9 < a_b,\ 0 \le x_{10} \le b_d, \\ a_b \le x_9 + x_{10} \le b_b,\ a_a \le x_{12} \le b_a, \\ x_{10} + x_{12} + x_{13} \le b_d,\ x_{13} < a_b \end{array} \right\},$$

$$K_{babacbad} = \left\{ \left((1,0,1,1,0), (\sharp, x_{13}+x_{14}, \sharp, \sharp) \right) \middle| \begin{array}{l} a_b \le x_0 \le b_b,\ a_a \le x_1 \le b_a, \\ a_b \le x_2 \le b_b,\ a_a \le x_3 \le b_a, \\ a_c \le x_3 + x_9 \le b_c,\ 0 \le x_9 \le b_b, \\ x_9 < a_b,\ 0 \le x_{10} \le b_d, \\ a_b \le x_9 + x_{10} \le b_b,\ a_a \le x_{12} \le b_a, \\ a_d \le x_{10} + x_{12} + x_{13} \le b_d,\ x_{13} < a_b \\ 0 \le x_{14},\ x_{13} + x_{14} \le b_b \end{array} \right\},$$

$$K_{babacbd} = \left\{ \left((0,1,1,1,0), (x_{12}+x_{15}, \sharp, \sharp, \sharp) \right) \middle| \begin{array}{l} a_b \le x_0 \le b_b,\ a_a \le x_1 \le b_a,\ a_b \le x_2 \le b_b, \\ a_a \le x_3 \le b_a,\ a_c \le x_3 + x_9 \le b_c,\ 0 \le x_9 \le b_b, \\ x_9 < a_b,\ 0 \le x_{10} \le b_d,\ a_b \le x_9 + x_{10} \le b_b \\ 0 \le x_{12} \le b_a,\ a_d \le x_{10} + x_{12} \le b_d \\ 0 \le x_{15},\ x_{12} + x_{15} \le b_a \end{array} \right\}.$$

Hence, we obtain for B the following inequality system, excluding redundant inequalities:

$$B = \left\{ \begin{array}{lllll} a_b \le x_0 \le b_b & a_d \le x_4 \le b_d & 0 \le x_8 & 0 \le x_{13} < a_b & a_a \le x_3 + x_4 + x_5 \le b_a \\ a_a \le x_1 \le b_a & 0 \le x_5 & x_9 < a_b & 0 \le x_{14} & a_c \le x_3 + x_9 \le b_c \\ a_b \le x_2 \le b_b & 0 \le x_6 \le b_b & a_d \le x_{10} & 0 \le x_{15} & x_9 + x_{10} + x_{11} \le b_b \\ a_a \le x_3 \le b_a & 0 \le x_7 \le b_d & 0 \le x_{11} & a_b \le x_9 + x_{10} & a_d \le x_{10} + x_{12} + x_{13} \le b_d \\ a_c \le x_3 & x_7 < a_b & a_a \le x_{12} \le b_a & x_{13} + x_{14} \le b_b & x7 + x_8 \le b_b \\ 0 \le a_a & 0 \le a_b & 0 \le a_c & a_d \le x_{10} + x_{12} & x_{12} + x_{15} \le b_a \\ 0 \le a_d & & & & \end{array} \right.$$

Subsequently, with respect to the properties of a interval function (cf. [18]) it has to be true:

$$0 \le a_t \le b_t \ \text{ for all } t \in \{a,b,c,d\},\ a_b \ge 1,\ b_d \ge 1.$$

Thus, a solution (with minimal values) for the interval function I^* is, e.g., $a_a = b_a = 0, a_b = b_b = 1, a_c = b_c = 0, a_d = 0$ and $b_b = 1$. All possible integer solutions for x_1, ... , x_{15} are the base for constructing the reachability graph and the reachability tree of the TPN. Actually, having all interval bounds we construct the reachability graph/tree successively and without calculating any inequality systems, cf. [18]. The reachability graph of the TPN has 11 states and is was calculated with INA or tina (cf. [20], [21]). An analyzing protocol is included as an appendix.

Proposition 2. *Let $(\mathcal{N}, m_0)$ be a live and unbounded Petri net, for which there exists a feasible cyclic run, which includes all transitions in $\mathcal{N}$. Let then I be an interval function calculated for $(\mathcal{N}, m_0)$ according to the algorithm described above. Then the TPN $(\mathcal{N}, I, m_0)$ is live and bounded.*

Proof. (Idea) Note, that if a node in a spine-based coverability tree has a descendant leaf, colored in red, then the marking of this node can enable an unbounded run, leading to an ω-marking. If it has a descendant leaf, colored in green, then the marking of this node can enable a cycle, which includes all transitions in $\mathcal{N}$.

At each step at the stage 4 the parametric run forces initial runs to the markings labeling green leaves in the spine-based coverability tree of the Petri net and forbid the branching of each initial run ending into a marking which labels read leaf.

The set B of inequalities defined in stage 5 obtains all constains successively calculated in stage 4 and the conditions for interval bounds.

The resulting Time Petri net $(\mathcal{N}, I, m_0)$ is live, because it retains all initial cycles with all transitions.

The net $(\mathcal{N}, I, m_0)$ is bounded, because all unbounded branches in $(\mathcal{N}, m_0)$ are cut by time constrains. □

The above algorithm allows to calculate an interval function needed for the net boundedness, only in the case, when such function exist.

In order to calculate minimal values for the interval bounds a_t and b_t we can consider the Linear Programming

$$\min \{ \sum_{t \in T} (a_t + b_t) \mid B \}.$$

Note also, that in the above algorithm we keep all feasible cycles, containing all transitions. The algorithm can be modified to keep at least one such cycle.

Let us compare both algorithms for controlling the behavior of a Petri net: By using priorities, cf. [12], and by using time constrains as represented here. Both algorithms can "repair" a live and unbounded Petri net into a live and bounded. But, the second algorithm works also in some cases when the first one does not work and it works always when the first one does it. The reason for that is the statical use of the priority relation. The time intervals introduce also a kind of priority relation between the transitions, but this priority is a dynamic one. For example, for the net $(\mathcal{N}^*, m_0^*)$ it is not possible to find a priority relation such that the firing of the transition d is prioritized to transition b but once, at the marking $(1, 0, 0, 0, 1)$ — where both transitions b and d may fire — b should fire. This is possible assigning time intervals, as we have seen. Thus, when the spine-based coverability tree would consist only of the left main branch, cf. Fig. 4, then there would not be a priority solution but a solution with time constrains.

4 Conclusion

In this paper we have studied the possibility for obtaining a live and bounded Petri net from a live and unbounded one by adding time constrains as time intervals associated to each transition. We have presented necessary conditions for existence of such priorities. These conditions are not sufficient, but help to exclude unsolvable cases. Furthermore, we have represented an algorithm for computing (minimal) time intervals for transforming a live and unbounded Petri net into live and bounded and live net. The resulting net is a Time Petri net with the prior skeleton (underlying timeless net).

This algorithm converts a live and unbounded Petri net into a live and bounded Time Petri net even in some cases, when our previous algorithm for finding a priority relation for retaining the liveness does not work. Thus, the algorithm represented here is more powerful than the priority algorithm.

References

1. W.M.P. van der Aalst, K.M. van Hee, A.H.M. Hofstede, N. Sidorova, H.M.W. Verbeek, M. Voorhoeve, M.T. Wynn: Soundness of Workflow Nets: Classification, Decidability, and Analysis. *Formal Aspects of Computing*, Vol. 23, No 3, pages 333–363, Springer, 2011.
2. V.A. Bashkin, I.A. Lomazova: Decidability of k-Soundness for Workflow Nets with an Unbounded Resource. Transactions on Petri Nets and Other Models of Concurrency IX, Lecture Notes in Computer Science, Vol. 8910, pages 1–78. Springer, 2014.
3. E. Best, R. Devillers: Characterisation of the State Spaces of Live and Bounded Marked Graph Petri Nets. In: Language and Automata Theory and Applications, Lecture Notes in Computer Science, Vol. 8370, pages 161-172, 2014.
4. Th. H. Cormen, Ch. E. Leiserson, R. L. Rivest, C. Stein: Introduction to Algorithms. MIT Press, 2001.
5. J. Desel: On Cyclic Behaviour of Unbounded Petri Nets. In: Application of Concurrency to System Design (ACSD) . 13th International Conference on Application of Concurrency to System Design, pages 110–119. IEEE, 2013.

6. J. Desel: Schwach beschränkte Petrinetze. 12ter Workshop "Algorithmen und Werkzeuge für Petrinetze", 29. - 30. September 2005.
7. K. van Hee, O. Oanea, A. Serebrenik, N. Sidorova, M. Voorhoeve, I.A. Lomazova: Checking Properties of Adaptive Workflow Nets. *Fundamenta Informaticae*, Vol. 79, No. 3, pages 347–362, 2007.
8. M. Heiner: Biochemically Interpreted Petri Nets - Two Open Problems. Talk, Seminaire MeFoSyLoMa (Formal Methods for Hardware and Software Systems), Université Paris 13, June 2007.
9. M Heiner: Time Petri nets for modelling and analysis of biochemical networks - on the influence of time. Talk, MaReBio, Marseille, November 2008.
10. C. Liu, Cong, A. Kondratyev, Y. Watanabe, J. Desel, A. Sangiovanni-Vincentelli: Schedulability analysis of Petri nets based on structural properties. *Fundamenta Informaticae*, Vol. 86, No 3, pages 325–341, 2008.
11. I. A. Lomazova: Interacting Workflow Nets for Workflow Process Re-Engineering. *Fundamenta Informaticae*, Vol. 101, No 1-2, pages 59–70, 2010.
12. I. A. Lomazova, L. Popova-Zeugmann: Controlling Petri Net Behavior using Priorities for Transitions. Proceedings of the Workshop Concurrency, Specification & Programming'2014, Sept. 29 - Oct. 1, 2014, pp. 126-137, 2014.
13. I.A. Lomazova, I.V. Romanov: Analyzing Compatibility of Services via Resource Conformance. *Fundamenta Informaticae*, Vol. 128, No. 1, pages 129–141, 2013.
14. P. Merlin: A Study of the Recoverability of Communication Protocols. PhD Theses, Irvine, 1974.
15. T. Murata: Petri Nets: Properties, Analysis and Applications. An invited survey paper. Proceedings of the IEEE, Vol.77, No.4 pp.541–580, April, 1989.
16. N. Sidorova, C. Stahl: Soundness for Resource-Contrained Workflow Nets is Decidable. *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 43, No. 3, pages 724–72, 2013.
17. L. Popova: On Time Petri Nets. J. Inform. Process. Cybern. EIK 27(1991)4, pages 227–244, 1991.
18. L. Popova-Zeugmann: Time and Petri Nets. Springer-Verlag, Springer Heidelberg New York Dordrecht London, 2013.
19. N. Ridder, K. Lautenbach: Liveness in bounded Petri nets which are covered by T-invariants. In: R. Valette (Ed.), Application and Theory of Petri Nets, Lecture Notes in Computer Science, Vol. 815, pages 358–378, Springer, 1994.
20. P.H. Starke INA - Integrated Net Analyzer. Manual, Berlin, 1997.
21. B. Berthomieu TIme petri Net Analyzer. url = http://www.laas.fr/bernard/tina/. LAAS / CNRS, Toulouse, France, 2.9.8 released, 2009, http://www.laas.fr/bernard/tina/,

5 Appendix

```
Integrated Net Analyzer [v2.2p6-Mar 23 2001-linux] session report:
Current net options are:
    token  type: black        (for Place/Transition nets)
    time option: intervals
    elements   : transitions
    firing rule: normal
    priorities : not to be used
    strategy   : single transitions
    line length: 80

Net read from irina_mit_int.pnt
Information on elementary structural properties:
Current name options are:
     transition names to be written
     place names to be written
The net is statically conflict-free.
The net is dynamically conflict-free.
The net is pure.
The net is not ordinary.
The net is not conservative.
The net is not subconservative.
The net is not a state machine.
The net is free choice.
The net is extended free choice.
The net is extended simple.
The net is marked.
The net is not marked with exactly one token.
The net is a marked graph.
The net is homogenous.
The net has not a non-blocking multiplicity.
The net has no nonempty clean trap.
The net has no transitions without pre-place.
The net has no transitions without post-place.
The net has no places without pre-transition.
The net has no places without post-transition.
Maximal in/out-degree:  2
The net is connected.
The net is not strongly connected.
ORD HOM NBM PUR CSV SCF CON SC  Ft0 tF0 Fp0 pF0 MG  SM  FC  EFC ES
 N   Y   N   Y   N   Y   Y   N   N   N   N   N   Y   N   Y   Y   Y
CPI CTI  B  SB  REV DSt BSt DTr DCF  L   LV L&S
 ?   ?   ?   ?   ?   ?   ?   ?   Y   ?   ?   ?
Current analysis options are:
     no symmetrical reduction
     no depth restriction
     do not use a 'bad' predicate
Computation of the reachability graph
States generated: 11
Markings: 10     Clock positions: 9
Arcs generated: 16

Capacities needed:
Place 1 2 3 4 5
 Cap: 1 1 2 1 1
```

```
The net has no dead transitions at the initial marking.
The net has no dead reachable states.
The net is bounded.
The net is not safe.
The net is not live and safe.
Livenesstest:
Computing the strongly connected components

The net is live.
The net is live, if dead transitions are ignored.
The net is covered by semipositive T-invariants.
The computed graph is strongly connected.
The net is reversible (resetable).

State nr.    1     Marking nr.    1     Clocks nr.    1
P.nr: 1 2 3 4 5
toks: 1 0 0 1 0
T.nr: 1 2 3 4
time: - 0 - -
==[1,t2]=> s2
State nr.    2     Marking nr.    2     Clocks nr.    2
P.nr: 1 2 3 4 5
toks: 0 1 1 1 0
T.nr: 1 2 3 4
time: 0 - - -
==[0,t1]=> s3
State nr.    3     Marking nr.    3     Clocks nr.    1
P.nr: 1 2 3 4 5
toks: 1 0 1 1 0
T.nr: 1 2 3 4
time: - 0 - -
==[1,t2]=> s4
State nr.    4     Marking nr.    4     Clocks nr.    3
P.nr: 1 2 3 4 5
toks: 0 1 2 1 0
T.nr: 1 2 3 4
time: 0 - 0 -
==[0,t1]=> s5
==[0,t3]=> s10
State nr.    5     Marking nr.    5     Clocks nr.    4
P.nr: 1 2 3 4 5
toks: 1 0 2 1 0
T.nr: 1 2 3 4
time: - 0 0 -
==[0,t3]=> s6
State nr.    6     Marking nr.    6     Clocks nr.    5
P.nr: 1 2 3 4 5
toks: 1 0 0 0 1
T.nr: 1 2 3 4
time: - 0 - 0
==[1,t2]=> s7
==[0,t4]=> s1
==[1,t4]=> s9
State nr.    7     Marking nr.    7     Clocks nr.    6
```

```
P.nr: 1 2 3 4 5
toks: 0 1 1 0 1
T.nr: 1 2 3 4
time: 0 - - 1
==[0,t1]=> s8
==[0,t4]=> s2
State nr.    8     Marking nr.    8     Clocks nr.    7
P.nr: 1 2 3 4 5
toks: 1 0 1 0 1
T.nr: 1 2 3 4
time: - 0 - 1
==[0,t4]=> s3
State nr.    9     Marking nr.    1     Clocks nr.    8
P.nr: 1 2 3 4 5
toks: 1 0 0 1 0
T.nr: 1 2 3 4
time: - 1 - -
==[0,t2]=> s2
State nr.   10     Marking nr.    9     Clocks nr.    9
P.nr: 1 2 3 4 5
toks: 0 1 0 0 1
T.nr: 1 2 3 4
time: 0 - - 0
==[0,t1]=> s6
==[0,t4]=> s11
State nr.   11     Marking nr.   10     Clocks nr.    2
P.nr: 1 2 3 4 5
toks: 0 1 0 1 0
T.nr: 1 2 3 4
time: 0 - - -
==[0,t1]=> s1
ORD HOM NBM PUR CSV SCF CON SC  Ft0 tF0 Fp0 pF0 MG  SM  FC  EFC ES
 N   Y   N   Y   N   Y   Y   N   N   N   N   N   Y   N   Y   Y   Y
CPI CTI  B  SB  REV DSt BSt DTr DCF  L   LV L&S
 ?   Y   Y   ?   Y   N   ?   N   Y   Y   Y   N
Current options written to options.ina

End of Analyzer session.
```