Controlling Petri Net Behavior using Priorities for Transitions*

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

[1] National Research University Higher School of Economics (HSE),
Moscow, 101000, Russia
ilomazova@hse.ru
[2] Humboldt-Universität zu Berlin,
Unter den Linden 6, 10099 Berlin, Germany
popova@informatik.hu-berlin.de

Abstract. In this paper we consider controlling Petri net behavior with the help of priorities. By controlling we understand forcing a process to behave in a desirable way, by adding constraints ascribing priorities to transitions, and hence transforming a classic Petri net into a Priority Petri net. Liveness and boundedness are crucial properties in many application areas, e.g. workflow modeling and bioinformatics. The main correctness property for workflow models is soundness, which can be reduced to liveness and boundedness for a modified net. In biological models, liveness and boundedness are important for system stability. The problem of making a given live, but unbounded Petri net live and bounded by adding priorities constraints is studied in this paper. We specify necessary conditions for solvability of this problem and present a method for ascribing priorities to net transitions so that the net becomes bounded, staying live.

1 Introduction

Petri nets are widely used for modeling and analysis of distributed systems. These can range from technical systems to systems of business processes, or biological systems. Although such systems are very different in their substance 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. The first two properties concern the liveness of the model, the second two are covered by boundedness studies. In most of the practical systems the finiteness of all possible situations is a highly desired property, i.e., the model should be bounded.

A typical example is a business process model, represented by a workflow net — a special kind of a Petri net. The crucial correctness property for workflow nets is soundness, also called proper termination [1]. The soundness property and its variations are intencively studied in the literature [1, 2, 5–7, 9].

* This work is supported by the Basic Research Program of the National Research University Higher School of Economics

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 [10, 11].

In practice it could be 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 scheduling. In [3] 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 transition priorities can ensure boundedness of a given live Petri net, retaining its liveness. In contrast to brute-force approach, priorities allow local and more flexible control — not just forcing one 'good' execution.

Priority scheduling is an appropriate solution for workflow systems. For biological system it is not so appropriate, since there is no mechanism to give priority to this or that action in biological systems. Scheduling biological systems with the help of time constraints would provide a much more natural solution [12]. This will be a theme of our further research.

In this paper we study adding priorities in order to transform a live and unbounded model, represented by a Petri nets, into a live and bounded model. Thereby we want to fully preserve the structure of the net.

The paper is organized as follows. In Section 2 we recall some basic definitions in the theory of Petri nets and subsequently in Section 3 we give a motivating example. In Section 4 we represent a sufficient condition for transforming a live and unbounded Petri net into a live and bounded one by adding priorities and finally Section 5 contains some conclusions.

2 Preliminaries

Let *Nat* denote the set of natural numbers (including zero).

Let P and T be disjoint sets of *places* and *transitions* and let $F \subseteq (P \times T) \cup (T \times P) \to Nat$ be a *flow relation*. Then $\mathcal{N} = (P, T, F)$ is a *unmarked Petri net*. A *marking* in a Petri net is a function $m : P \to Nat$, mapping each place to some natural number (possibly zero). A *Petri net* $(\mathcal{N}, m_0)$ is an unmarked Petri net $\mathcal{N}$ with its initial marking m_0. We use vector notation for marking, 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 $m_1 \xrightarrow{t_1} m_2 \xrightarrow{t_2} \ldots$. 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' iff there is a run $m' = m_1 \to m_2 \to \cdots \to m_n = m$; m is *reachable in* (N, m_0) iff m is reachable from the initial marking. For an unmarked Petri net $\mathcal{N}$ by $\mathcal{R}(\mathcal{N}, m)$ we denote the set of all markings reachable in $\mathcal{N}$ from the marking m, by $\mathcal{R}(\mathcal{N})$ – the set of all markings reachable in $\mathcal{N}$ from its initial marking. 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 $t_1, \ldots, t_n$ 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 leave 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 leave 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 leaves.

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 $\kappa \in Nat$. 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 all places p from 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 $m'(p) \geq m(p)$, then this sequence of transitions is also enabled in m'. A marking m' *strictly covers* a marking m (denoted $m' > m$) iff $m'(p) \geq m(p)$ and $m \neq m$.

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 on a place. Formally,

a generalized marking is a mapping $m : P \to Nat \cup \{\omega\}$. 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 leave 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' gets 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. A *priority relation* for $\mathcal{N}$ is a partial order $(T, \ll)$, i.e., the relation $\ll$ is reflexive, antisymmetric and transitive. For a subset $S \subseteq T$ a minimal element in S (w.r.t. $\ll$) is a transition $t \in S$, such that for all $t \in S$, $t' \neq t$ implies $t' \not\ll t$. Obviously, there can be several minimal elements in S.

A (marked or unmarked) *Petri net with priorities* is a (marked or unmarked) Petri net together with a priority relation. For $\mathcal{P} = (\mathcal{N}, \ll)$ being a Petri net with priorities, we call the Petri net $\mathcal{N}$ the skeleton of $\mathcal{P}$ and denote it with $S(\mathcal{P})$. A priority relation $\ll$ can be specified by assigning a priority label (natural number) $\pi(t) \in Nat$ to each transition t. Then we set $t \ll t'$ iff $\pi(t) < \pi(t')$ and represent priority graphically by priority labels, given in curly brackets.

Fig. 2 gives an example of a Petri net with priorities. Here $\pi(a) = 1, \pi(b) = 3, \pi(c) = 2, \pi(d) = 2$, and hence $a \ll c, a \ll d, c \ll b, d \ll b$.

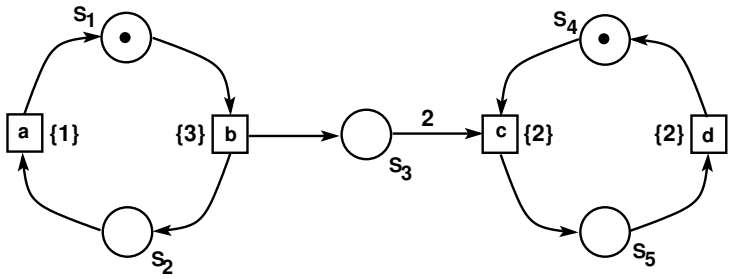

Fig. 1. An example of a marked Petri net $\mathcal{P}_1 = (\mathcal{N}_1, \ll, m_0)$ with priorities , where $m_0 = (1, 0, 0, 1, 0)$, priority labels are given in curly brackets.

The firing rule for a Petri net $\mathcal{P} = (N, \ll)$ with priorities is defined as follows. Let m be a marking in $\mathcal{N}$, and Q be a set of transitions enabled in m (according to usual rules for Petri nets). Then only minimal w.r.t. $\ll$ transition may fire in m, i.e. transitions with higher priorities are always preferred over transitions with lower priorities.

For the Petri net $\mathcal{P}_1 = (\mathcal{N}_1, \ll, m_0)$ in Fig. 2 the firing sequence *baba* is feasible and leads to the marking $m_1 = (0, 1, 2, 1, 0)$. Both transitions c and b are enabled in m_1, but only c can fire in m_1, since $b \ll c$.

Liveness can be defined in several ways for Petri nets [8]. 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* iff for every reachable marking m there exists a sequence of firings starting from m, which includes t, i.e. in a Petri net N $\forall m \in \mathcal{R}(N, m_0) : \exists \sigma \in T^* : m \xrightarrow{\sigma} m' \xrightarrow{t}$.

3 Motivating Examples

In [11] and [10] 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 transition invariants (i.e. each transition enters into at least one transition invariant with a non-zero component), transition invariants can be used for computing time durations for transitions, which make the net bounded. In other words, this method allows to transform a live and unbounded Petri net, covered by transition invariants, into a live and bounded Timed Petri net [12] with the same structure. Unfortunately this method does not always work, as it was shown in [11]. It was even not known, whether the condition of covering all transitions in a Petri net with some transition invariant is necessary for transforming a live and unbounded net into a bounded one.

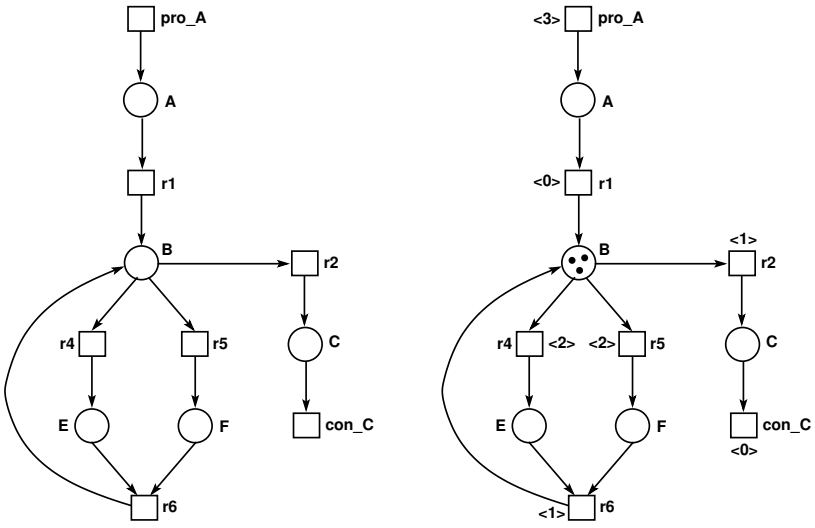

Fig. 2. Live and unbounded Petri nets

Consider now two Petri nets in Fig.3. These two Petri nets differ only in their initial markings. The left Petri net has the empty initial marking, and the right one has initially three tokens in the place B. Both nets are live and unbounded. Actually, all their places are unbounded. Both nets are covered by (the same) two minimal transition invariants, since these nets have the same net structure.

However, there is a great difference between these nets. The right net can be transformed into a live and bounded net by ascribing time durations to its transitions, which are graphically represented in angle brackets here. This turns the right net into a Timed Petri net. The reader can find firing rules for Timed Petri nets in [12], and make certain that with time durations the net becomes bounded. Quite the contrary, the left net cannot be transformed into a bounded net by adding time durations.

These examples show, that a possibility to transform a live and unbounded net into a bounded one can depend not only on transition invariants, but on initial markings as well.

Live and unbounded Petri nets were considered also in [4]. 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 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.

4 Priorities for Making a Petri Net Bounded

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 Petri net with priorities (with the same skeleton), i.e. by adding priorities to its transitions. To solve this problem we shall try for transition priorities, which would exclude runs leading to unboundedness.

We start by studying some properties of live and bounded Petri nets.

Proposition 1. *Let $(\mathcal{N}, m_0)$ be a live and bounded Petri net. Then there exists a feasible cyclic run, including all transitions in $\mathcal{N}$.*

Proof. Since $(\mathcal{N}, m_0)$ is live, no dead marking is reachable in it, and all maximal runs are infinite. The net $(\mathcal{N}, m_0)$ is bounded, hence its reachability set is finite, and each infinite run includes a cycle. Moreover, such a run has a form $\tau\sigma^\star$, where τ is a (prefix) finite initial run, and σ is a feasible cyclic run.

Liveness means that each transition may potentially fire from any reachable marking. Then there exists a run, which includes all net transitions infinitely often. The cyclic part of this run contains all net transitions.

Proposition 2. *Let $(\mathcal{N}, m_0)$ be a Petri net, and let $(\mathcal{N}, \ll, m_0)$ be a Petri net with priorities, obtained from $\mathcal{N}$ by adding a transition priority relation $\ll$. Then the reachability tree for $(\mathcal{N}, \ll, m_0)$ is a subgraph of the reachability tree for $(\mathcal{N}, m_0)$.*

Proof. Straightforward from the definitions.

The next Theorem defines necessary conditions for a possibility to recover boundedness for a live Petri net with the help of transition priorities.

Theorem 1. *Let $(\mathcal{N}, m_0)$ be a Petri net. If for some priority relation $\ll$ the Petri net $(\mathcal{N}, \ll, m_0)$ with priorities is live and bounded, then there exists a feasible cyclic run in $(\mathcal{N}, m_0)$, which includes all transitions in $\mathcal{N}$.*

Proof. Follows from Propositions 1 and 2.

Corollary 1. *Let $(\mathcal{N}, m_0)$ be a live and unbounded Petri net. If there exists a priority relation $\ll$ on the set T of transitions in $\mathcal{N}$, s.t. the Petri net $(\mathcal{N}, \ll, m_0)$ with priorities is live and bounded, then there exists a transition invariant without zero components for $\mathcal{N}$, i.e. all transitions in $\mathcal{N}$ are covered by some T-invariant.*

So, given a live and unbounded Petri net $(\mathcal{N}, m_0)$, before looking for constraints, 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 transition invariants for the net $\mathcal{N}$. If there is no a transition invariant, covering all transitions in $\mathcal{N}$, then the net cannot be recovered. Note, that transition invariants can be computed in polynomial time on the size of the net.

If there is such a transition 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 transition invariants with non-zero components. To do this check the algorithm, proposed in [3] 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 necessary conditions hold for it. We would like to find transition priorities, that will make the net bounded, keeping its liveness. The procedure will be illustrated on the net $\mathcal{N}_1$ in Fig. 2.

We do this in four stages.

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 [3]. 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 due to the Theorem 1 the problem does not have a solution. So, let

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

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

Thus, for example, the net $(\mathcal{N}_1, m_0)$ 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 $\mathcal{C}(\mathcal{N}, m_0)$.

The spine tree for Petri net $(\mathcal{N}_1, m_0)$ 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 leave 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 leave and gets the green color.
   (d) Otherwise, leave v' uncolored.
2. Color the node v in yellow.

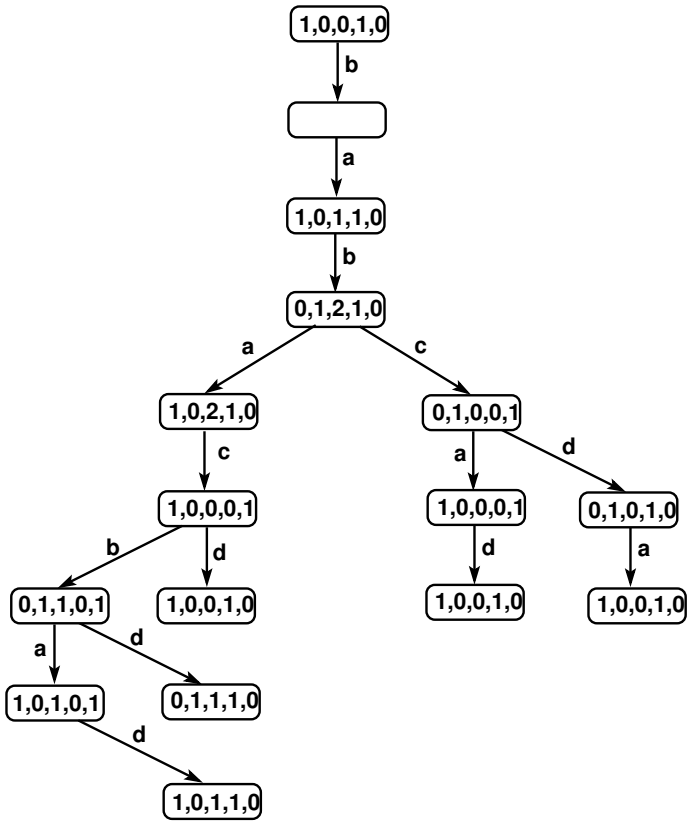

Fig. 3. The spine tree for the net $(\mathcal{N}_1, m_0)$.

The spine-based coverability tree for our example net $\mathcal{N}_1$ is shown in Fig. 4. Here node colors are used to illustrate the tree construction. A leave and some inner node have the same color, if they have the same markings, or the leave 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.

Stage 4. Compute a priority relation. Let $\mathcal{T}$ be a spine-based coverability tree. By the construction of $\mathcal{T}$, all its leaves are colored either in green, or red. In this stage we construct a priority relation $\mathcal{R} \subseteq T \times T$. Initially $\mathcal{R}$ is empty.

Let v be one of the leaves in with the maximal distance from the root, let u be the parent of v. Two cases are possible.

(1) All children of u (including v) have the same color (green, or red). Then delete all children of u from the tree, make u a leave and color it with the color of v.

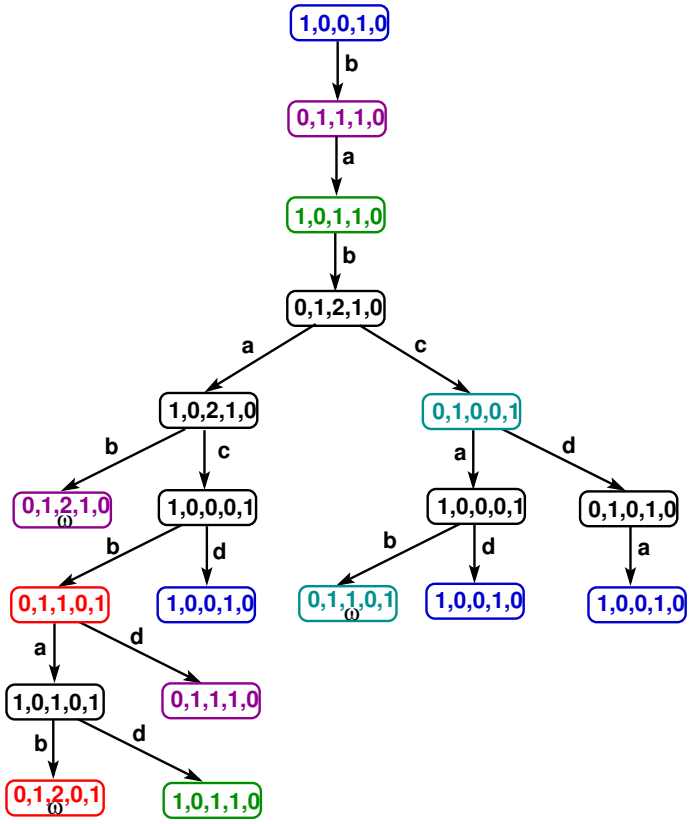

Fig. 4. The spine-based coverability tree for $(\mathcal{N}_1, m_0)$.

(2) Some children nodes are colored in red, and some – in green.

- If u is not in the spine of $\mathcal{T}$, then delete all children of u from the tree, make u a leave and color it in red.
- If not, let T_r be the set of all transitions, corresponding to arcs going from u to nodes colored in red, and similarly, T_g – the set of all transitions labeling arcs going from u to nodes colored in green. Note, that $T_r \cap T_g = \emptyset$. Then define $\mathcal{R}_u = \{(t_1, t_2) | t_1 \in T_r, t_2 \in T_g\}$, and add all pairs from $\mathcal{R}_u$ to the priority relation $\mathcal{R}$, i.e. $\mathcal{R} := \mathcal{R} \cup \mathcal{R}_u$. After that delete all children nodes of u, and make u a leave colored in green.

Repeat this procedure until there are no more red leaves in $\mathcal{T}$. Finally, define the relation $\ll$ for $(\mathcal{N}, m_0)$ as the transitive and reflexive closure of $\mathcal{R}$.

Applying the procedure of the stage 4 to the spine-based coverability tree for our example net $(\mathcal{N}_1, m_0)$ (cf. Fig. 4) we sequentially obtain: $d \ll b$, $c \ll b$. That means, that we could choose even weaker priorities, than shown in Fig. 2, i.e.

transitions a, c and d could have the same priorities to guarantee liveness and boundedness.

Theorem 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 $\ll$ be the relation constructed for $(\mathcal{N}, m_0)$ according to the algorithm described above. If $\ll$ is a partial order (i.e. $\ll$ is antisymmetric), then $\ll$ is a priority relation for $\mathcal{N}$, and the Petri net $(\mathcal{N}, \ll, m_0)$ with priorities is live and bounded.*

Proof. (Sketch) Note, that if a node in a spine-based coverability tree has a descendant leave, colored in red, then the marking of this node can enable an unbounded run, leaving to a ω-marking. If it has a descendant leave, 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 relation $\mathcal{R}$ is extended in such a way, that cyclic runs prioritize unbounded runs.

Each step at the stage 4 reduces the number of nodes in a spine-based coverability tree, so the process on the stage 4 will terminate. Initially, the spine tree for $(\mathcal{N}, m_0)$ is not empty, and the root has descendant nodes colored in green, so the process will terminate correctly, i.e. with only green leaves (possibly one).

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

The net $(\mathcal{N}, \ll, m_0)$ is bounded, because all unbounded branches in $(\mathcal{N}, m_0)$ are cut by priorities.

The above algorithm allows to compute priorities needed for the net boundedness, only in the case, when such priorities exist and the algorithm computes a partial order. We have presented some necessary conditions for existence of needed priorities, but we do not know whether these conditions are sufficient.

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. It is an open problem, whether a failure of the modified algorithm, i.e. the computed relation is contradictory (not a partial order), means that required priorities do not exist.

5 Conclusion

In this paper we have investigated the possibility for obtaining a live and bounded Petri net from a live and unbounded one by adding priorities. We have presented necessary conditions for existence of such priorities. This conditions are not sufficient, but help to exclude unsolvable cases. Then an algorithm for computing transition priorities for transforming a live and unbounded Petri net into live and bounded net by ascribing priorities to transition was developed. When terminates successfully, this algorithm guarantees that computed priorities solve the problem, i.e. the net with priorities is live and bounded. It's an open problem, whether a solution exists, when the algorithm fails to compute consistent

priorities. The further research will concern this open problem and the study of the existence of more effective methods.

Another challenging question – to translate the priorities into time intervals or time durations for transitions. This problem is rather important for biological systems, because priorities do not go with a primary eligibility criterion.

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*, 23(3):333–363, Springer, 2011.
2. V.A. Bashkin, I.A. Lomazova: Soundness of Workflow Nets with an Unbounded Resource is Decidable. In: Joint Proc. of PNSE'13 and ModBE'13. CEUR Workshop Proceedings, vol. 989, pages 61–75. CEUR-WS.org, 2013.
3. J. Desel: On Cyclic Behaviour of Unbounded Petri Nets. In: Application of Concurrency to System Design (ACSD) , pages 110–119. 13th International Conference on Application of Concurrency to System Design, IEEE, 2013.
4. J. Desel: Schwach beschränkte Petrinetze. 12ter Workshop Algorithmen und Werkzeuge fr Petrinetze, 29. - 30. September 2005.
5. 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.
6. I. A. Lomazova. Interacting Workflow Nets for Workflow Process Re-Engineering. *Fundamenta Informaticae*, Vol. 101, No 1-2, pages 59-70, 2010,
7. I.A. Lomazova, I.V. Romanov: Analyzing Compatibility of Services via Resource Conformance. *Fundamenta Informaticae*, Vol. 128, No. 1, pages 129–141, 2013.
8. 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.
9. 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.
10. M. Heiner: Biochemically Interpreted Petri Nets - Two Open Problems. Talk, Seminaire MeFoSyLoMa (Formal Methods for Hardware and Software Systems), Universit Paris 13, June 2007.
11. M Heiner: Time Petri nets for modelling and analysis of biochemical networks - on the influence of time. Talk, MaReBio, Marseille, November 2008.
12. L. Popova-Zeugmann: Time and Petri Nets. Springer-Verlag, Springer Heidelberg New York Dordrecht London, 2013.