Deadlocks and livelocks in resource constrained workflow nets

Gabriel Juhás[1], Ana Juhásová[2] and Tomáš Kováčik[1]

[1]Faculty of Electrical Engineering and Information Technology, Slovak University of Technology in Bratislava, Ilkovičova 3, 812 19 Bratislava, Slovakia

[2]BIREGAL s.r.o. Klincová 37/B, 821 08 Bratislava, Slovakia

Abstract

The paper is presenting a method for detection of locks (both deadlocks and livelocks) in discrete event systems with shared resources and multiple instances modeled by resource constrained workflow nets. We consider that multiple instances with determined initial state and a correct final state are running in workflow nets. We consider workflow processes, in which instances can share several types of resources, with instances neither creating nor destroying shared resources. It means, that instances can use resources but the used resources are returned at the latest by the correct finish of the instance. Such resources are said to be durable. Examples of durable resources include resources of information systems, such as memory and processors or employees in roles in an organizational structure. We consider processes, which have enough resources to execute a single instance, such processes are said to be sound. A lock is a state of the process, where several instances are running but because of the lack of shared resources not all running instances can finish properly. The main result of the paper is the theorem stating that in sound workflow processes with several types of shared durable resources for given initial number of resources and an arbitrary unbounded number of running instances it is enough to test locks for a finite bounded number of instances, with the upper bound indicated.

Keywords

Petri net, Deadlock, Livelock, Resource constrained workflow net

1. Introduction

In [1] authors work with workflow processes with instances, in which instances share common resources. The processes are modeled by resource constrained workflow nets, where shared resources are modeled by so called static places. Except shared resources, instances are independent. Shared resources considered in [1] are durable, i.e. the resources are neither created nor destroyed by instances. Resources in information systems, such as memory, processors, i/o ports are typical examples of durable resources. Similarly, employees in particular roles considered by workflow processes can represent an example of durable resources.

The main problem solved in [1] is the problem of correct finish of all instances of a workflow process with shared durable resources. The problem is solved for processes with one type of durable resources. The method from [1] decides whether there is a number of shared resources

The Workshop Algorithms & Theories for the Analysis of Event Data (ATAED 2022): A Satellite Event of the 43rd International Conference on Application and Theory of Petri Nets and Concurrency, June 19 - 24, 2022, Bergen, Norway

gabriel.juhas@stuba.sk (G. Juhás); ana.juhasova@biregal.sk (A. Juhásová); tomas_kovacik@stuba.sk (T. Kováčik)

0000-0001-8302-5112 (G. Juhás); 0000-0003-3157-8609 (T. Kováčik)

such that for this number of resources and any greater number of resources any number of instances can be correctly finished. If the answer is true, then the workflow process is according to [1] called sound.

In [2] authors solve the soundness problem for several subclasses of resource constrained workflow nets. In [3] four necessary conditions of soundness of resource constrained workflow nets based on structural analysis are presented.

In [4, 5] we defined a technique based on a constructor and a runtime net to detect instance deadlocks of resource constrained workflow nets (called workflow nets with static places) for a fixed number of durable resources and an arbitrary number of instances. However, method from [4] did not work for livelocks.

In [6, 7] authors using a technique based on constructor and using a transformed net from [4] (called production net in [6, 7]) proved that the soundness problem for resource constrained workflow nets which are sound for one instance, is decidable even for the case where resources are not durable. The proof is based on the reduction to the home space problem of Petri nets, which was proved to be decidable in [8]. At the same time, the papers [6, 7] prove that in general, i.e. if the soundness for one instance is not required and the resources does not necessarily need to be durable, the problem of soundness in resource constrained workflow nets is undecidable. In [9] authors using the constructor from [4] proved, that if the nets are sound for one case, then the soundness problem is decidable even if the instances are not independent (the investigated net corresponding to our runtime net is not serializable). For more about serializability see e.g. [10, 11]. Paper [9] extends an unsuccessful attempt to prove decidability of soundness for resource constrained workflow nets presented in [12]. In the proof of results from [9] authors again used reduction to home space problem of Petri nets. As it is written at the end of the paper [9], soundness of resource constrained workflow nets is decidable, but up to now there is no effective algorithm, because the algorithm proposed in [9] requires the test of general reachability in possibly unbounded Petri nets [13, 14, 15, 16, 17]. The similar argument can be found at the end of original paper solving home space problem [8].

There are workflow processes, which for some fixed initial number of shared resources can correctly finish any number of instances, but for some greater number of shared resources the same process cannot finish for some number of instances running in parallel. Thus, such processes are not sound according to the definition of [1, 9]. An example of such a not sound process is on Figure 1, which can finish any number of instances for the given number of shared resources in static places *free key, free memory, free processor*. However, adding any number of resources to the static place *free key* will cause that the process may deadlock for the number of instances greater than 3. In general, once the number of resources in the place *free key* is smaller than the sum of free resources in places *free memory* and *free processor*, then the process will have no deadlock, otherwise it will contain a deadlock for any number of instances greater of equal the sum of shared resources in places *free memory* and *free processor*.

In this work we will prove, that in order to decide, whether the workflow process with arbitrary finite number of resource types and arbitrary unbounded number of instances with fixed initial number of durable shared resources will contain a lock (i.e. a deadlock or a liveclock), it suffices to check the existence of locks for bounded number of instances. We also will show how to compute such upper bounds.

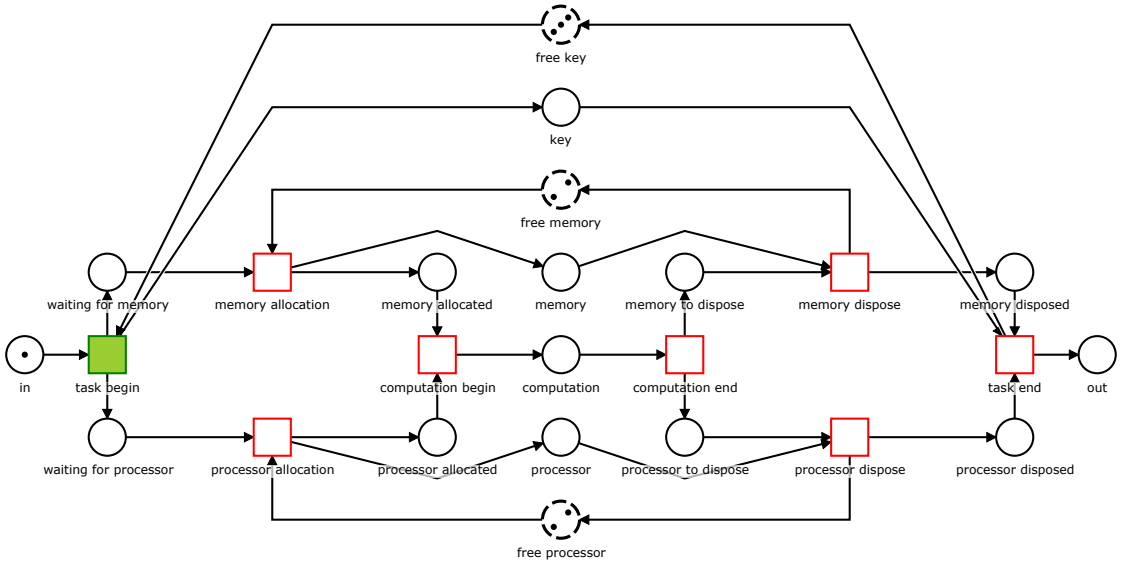

Figure 1: A marked remembering workflow net with static places, modeling allocation of memory units and processors to computing tasks with possibility to constrain the number of tasks, which can be executed in parallel.

2. Labelled transition systems

As a basic model of process behaviour we will use labelled transition systems [18, 19].

Definition 1 (Labelled transition system).
A labelled transition system is an ordered triple $(S, E, \longrightarrow)$*, where*

- S *is a set of states,*
- E *is a set of events,*
- $\longrightarrow \subseteq S \times E \times S$ *is a transition relation.*

The fact, that (s, e, s') *belongs to* $\longrightarrow$ *is referred as* $s \xrightarrow{e} s'$*.*

We will use labelled transition systems that have determined an initial state from which any other state is reachable.

To define reachability of states, we first define the notion of a sequence and the notion of an index set.

Definition 2 (Index set).
Index set $\mathbb{I}$ *is a subset of positive integers satisfying: if a positive integer* i *belongs to the index set* $\mathbb{I}$*, then any positive integer smaller than* i *belongs to the index set* $\mathbb{I}$*. If* $\mathbb{I}$ *is a finite nonempty set with the maximum* n*, then the number* n *is referred as* $max_{\mathbb{I}}$*. If* $\mathbb{I}$ *is the empty set, then* $max_{\mathbb{I}} = 0$*.*

Definition 3 (Sequence).
Let $\mathbb{I}$ be an index set and S be a set. Then a function $\alpha : \mathbb{I} \to S$ associating an element $\alpha(i)$ from the set S to each index i from the index set $\mathbb{I}$, is called a sequence of elements from the set S. If $\mathbb{I}$ is a finite set, then we say that α is finite sequence with length $max_{\mathbb{I}}$. Especially, if $\mathbb{I}$ is the empty set, then we say that α is the empty sequence. If $\mathbb{I}$ is equal to the set of all positive integers, then we say that α is an infinite sequence.

For illustration, consider a sequence of characters $\alpha = abba$ from the set of characters $S = \{a, b\}$. Intuitively, we understand that it is a finite sequence with length 4. In accordance with Definition 2 we use the index set $\mathbb{I} = \{1, 2, 3, 4\}$. The sequences α is formalized as follows: $\alpha(1) = a$, $\alpha(2) = b$, $\alpha(3) = b$ a $\alpha(4) = a$, i.e. the first element of the sequence α is character a, the second element is b, the third element is again b a the fourth element is character a.

Definition 4 (Occurrence sequence, reachability).
Let $(S, E, \longrightarrow)$ be a labelled transition systems. Let $s \in S$ be a state and let $\epsilon : \mathbb{I} \to E$ be a finite sequence of events. Sequence ϵ can occur in state s iff there exists a function $\sigma : \mathbb{I} \cup \{0\} \to S$ such that $\sigma(0) = s$ and for each positive integer $i \in \mathbb{I}$: $\sigma(i-1) \xrightarrow{\epsilon(i)} \sigma(i)$. The occurrence of the sequence ϵ in the state s leads to the state $\sigma(max_{\mathbb{I}})$.

A state $s' \in S$ is reachable from a state s iff there is an occurrence sequence ϵ, which leads from s to s'.

The fact that a finite sequence ϵ can occur in s and its occurrence leads to s' is referred as $s \xrightarrow{\epsilon} s'$.

Remember, that according to Definition 4 for any state s we have: the empty sequence can occur in s and its occurrence leads to s, i.e. s is self-reachable by occurrence of the empty sequence.

Definition 5 (Pointed labelled transition system).
A pointed labelled transitions system is an ordered quadruple $(S, E, \longrightarrow, q)$, where $(S, E, \longrightarrow)$ is a labelled transition system and $q \in S$ is its initial state, while for each state $s \in S$ there holds that s is reachable from q.

3. Petri nets

In this section we briefly introduce the basic definition of Petri nets [20, 21, 22, 23, 24, 25].

Definition 6 (Petri net).
A Petri net is an ordered quadruple (P, T, I, O), where:

- *P is a set of places*
- *T is a set of transitions*
- *$P \cap T = \emptyset$*
- *$I : P \times T \to \mathbb{N}$ is an input function, where $\mathbb{N}$ stands for the set of non-negative integers.*
- *$O : P \times T \to \mathbb{N}$ is an output function.*

A state of a Petri net is given by a marking.

Definition 7 (Marking).
Let $PN = (P, T, I, O)$ be a Petri net. A function $m : P \to \mathbb{N}$ attaching a non-negative integer to each place is called a marking of Petri net PN. The value $m(p)$ defines the number of tokens in a place $p \in P$. A marking will be written in form of a sum of marking of places, i.e. $\sum_{p \in P} m(p)p$. The set of places, for which $m(p)$ is greater than zero, is called the support of marking m and is denoted by $sup(m)$, formally for each $p \in P$ there holds that p belongs to $sup(m)$ iff $m(p) > 0$.

Places can be understood as types of tokens. As an example, if a set of places is given by $P = \{a, b, c\}$, we have three types of tokens, tokens of type a, tokens of type b and tokens of type c. Consider a marking $m : P \to \mathbb{N}$ such that $m(a) = 2$, $m(b) = 0$ and finally $m(c) = 1$. It means, that the marking expresses a state with two tokens of type a, no tokens of type b and one token of type c, which will be expressed by expression $2a + 0b + 1c$, or more simply by expression $2a + c$ (i.e. two tokens of type a and a token of type c), generally by expression $\sum_{p \in P} m(p)p$.

The fact, that for markings m and m' there holds $m(p) \leq m'(p)$ for each $p \in P$, is denoted by $m \leq m'$. Further, $m < m'$, respectively $m' > m$ denotes the fact that $m \leq m'$ and there exists $p \in P$ such that $m(p) < m'(p)$. If $m < m'$, we say that m is smaller than m', and m' is greater than m, respectively. The sum of two markings m a m' is marking $m + m'$ such that $(m + m')(p) = m(p) + m'(p)$ for each $p \in P$. If $m \leq m'$, then the difference of markings m' a m is given by marking $m' - m$ such that $(m' - m)(p) = m'(p) - m(p)$ for each $p \in P$.

Dynamics of a Petri net is given by firing of transitions.

Definition 8 (Transition firing).
Let $PN = (P, T, I, O)$ be a Petri net. Let $m : P \to \mathbb{N}$ be a marking and $t \in T$ be a transition of the net PN. Transition t is enabled to fire in marking m iff for each $p \in P$ there holds: $m(p) \geq I(p, t)$.

If transition t is enabled to fire in marking m, then its firing in m leads to marking m' such that for each $p \in P$ there holds: $m'(p) = m(p) - I(p, t) + O(p, t)$.

We will consider that a Petri net has initial marking. A Petri net together with an initial marking will be referred as marked Petri net.

Definition 9 (Marked Petri net).
A marked Petri net is an ordered quintuple $MPN = (P, T, I, O, m_0)$, where $PN = (P, T, I, O)$ is a Petri net and m_0 is a marking of PN called initial marking.

Graphically places are depicted as circles, markings of places by number of tokens inside of places, transitions are depicted as squares. Enabled transitions are filled (by green color). Non-zero value of input function $I(p, t)$ is expressed by an arrow from place p to transition t, while the value greater than one is written by the arc. Non-zero value of output function $O(p, t)$, is expressed by an arrow from transition t to place p, while the value greater than one is written by the arc. These non-zero values will be referred as weights of arcs. All Petri nets used in this work where modelled in an online Petri net editor, accessible at www.petriflow.com.

Petri nets defines a labelled transition system in a natural way by firing enabled transitions.

Definition 10 (Reachability graph of a Petri net).
Let $PN = (P, T, I, O)$ be a Petri net. A labelled transition system $(S, E, \longrightarrow)$, where

- *S is the set of all markings, i.e. the set of all functions from P to non-negative integers $\mathbb{N}$,*
- *$E = T$,*
- *$m \xrightarrow{t} m'$ iff transition t is enabled to fire in marking m and firing of t in m leads to marking m',*

is called reachability graph of Petri net PN.

We also define reachability graphs of marked Petri nets.

Definition 11. (Reachability graph of a marked Petri Net)
Let $MPN = (P, T, I, O, m_0)$ be a marked Petri net. Let $(S, E, \longrightarrow)$ be reachability graph of Petri net $PN = (P, T, I, O)$. Let $[m_0\rangle$ denote the set of all markings reachable from m_0 in reachability graph of Petri net PN. Then pointed labelled transition system $([m_0\rangle, E, \longrightarrow \cap([m_0\rangle \times E \times [m_0\rangle), m_0)$ is called reachability graph of marked Petri net MPN. If sequence ϵ can occur in m and its occurrence in m leads to m' in the reachability graph of MPN, then we say that sequence ϵ is enabled to fire in marking m in net MPN and its firing in m leads to the marking m' in net MPN. We also say that marking m' is reachable from marking m in the marked Petri net MPN.

Definition 12 (Boundedness).
A marked Petri net $MPN = (P, T, I, O, m_0)$ is bounded iff there exists a function $b : P \to \mathbb{N}$, such that for each marking m reachable from m_0 in MPN there holds: $m \leq b$. Function b is called the bound of net MPN.

In the case that a marked Petri net has finitely many places is the boundedness equivalent with the finiteness of the number of reachable markings. For more results on boundedness see e.g. [26, 27, 28].

Corollary 1. *A marked Petri net $MPN = (P, T, I, O, m_0)$ with finite set of places is bounded iff the number of markings reachable from m_0 in MPN is finite.*

4. Workflow nets

We focus on processes, where instances has a unique start and unique correct finish. In literature, such systems are modeled by workflow nets [29, 30, 31, 32].

Definition 13 (Workflow net).
A workflow net is a Petri net $PN = (P, T, I, O)$ with finite number of places and transitions in which there exists a unique place $in \in P$ and a unique place $out \in P$ such that

- *for each $t \in T$ there holds $O(in, t) = 0$ and there exists such $t \in T$ that $I(in, t) \neq 0$,*
- *for each $t \in T$ there holds $I(out, t) = 0$ and there exists such $t \in T$ that $O(out, t) \neq 0$.*

Place in is called input place of workflow net PN and place out is called output place of workflow net PN.

A marked workflow net is a workflow net with a special initial marking, in which only the input place is marked.

Definition 14 (Marked workflow net).
A marked workflow net is a marked Petri net $MPN = (P, T, I, O, m_0)$, where $PN = (P, T, I, O)$ is a workflow net and $m_0 = in$, i.e. $m_0(in) = 1$ and $m_0(p) = 0$ for each $p \in P$ different from in.

5. Petri nets and workflow nets with static places

Processes with instances and shared resources will be modeled by Petri nets with static places [4, 5]. Static places inspired by static variables in Java will model shared resources. Petri nets with static places were inspired by workflow nets with static places originally defined in papers [33, 1, 3] under the name resource constrained workflow nets.

Definition 15 (Petri net with static places).
Petri net with static places is a quintuple $PNS = (D, S, T, I, O)$, where

- *D is a set of dynamic places*
- *S is a set of static places*
- *$D \cap S = \emptyset$*
- *$PN = (P = D \cup S, T, I, O)$ is a Petri net.*

Static places will be depicted by dashed circles.

Definition 16. (Marked Petri net with static places)
Marked Petri net with static places is a sextuple $MPNS = (D, S, T, I, O, m_0)$, where

- *$PNS = (D, S, T, I, O)$ is a Petri net with static places*
- *$MPN = (P = D \cup S, T, I, O, m_0)$ is marked Petri net.*

We say that a transition is enabled to fire in a marked Petri net with static places $MPNS$ if it is enabled to fire in marked Petri net MPN. A reachability graph of a marked Petri net with static places $MPNS$ is the reachability graph of marked Petri net MPN. All notion defined for marked Petri net MPN will analogously used for marked Petri net with static places $MPNS$.

Definition 17. (Workflow net with static places/resource constrained workflow net) *A workflow net with static places, also called resource constrained workflow net, is a Petri net with static places $W = (D, S, T, I, O)$, where $PN = (P = D \cup S, T, I, O)$ is a workflow net such that $in \in D$ and $out \in D$.*

Initial marking of a marked workflow net with static places contains one token in the input place, no tokens in dynamic places different from the input place and any number of tokens representing shared resources in static places.

Definition 18. (Marked workflow net with static places)
A marked workflow net with static places is a sextuple $MW = (D, S, T, I, O, m_0)$, where

- *$W = (D, S, T, I, O)$ is a workflow net with static places*
- *m_0 is initial marking such that $m(in) = 1$ and $m(d) = 0$ for each dynamic place $d \in D$, which is different from in.*

In comparison with [33, 1, 3] we formalize durable resources via (weak) complementary places [34] of static places.

Definition 19. (Remembering workflow net with static places)
Let $W = (D, S, T, I, O)$ be such workflow net with static places, that for each places $s \in S$ there exists a unique (weak)complementary place $d_s \in D$ different from in and out, which for each $t \in T$ satisfies $I(d_s, t) - O(d_s, t) = O(s, t) - I(s, t)$. Workflow net W is called remembering workflow net with static places. The set of all complementary places of static places is denoted by D_S.

The equality $I(d_s, t) - O(d_s, t) = O(s, t) - I(s, t)$ in a marked remembering workflow net with static places implies that in case that $O(s, t) - I(s, t) = 0$, we also have $I(d_s, t) - O(d_s, t) = 0$. It means, that firing any transition may create a token in a complementary place d_s of a static place s iff it consumes a token from the static place s. Because in any initial marking the complementary place d_s is empty, we get that the sum of tokens in a static place s and its complementary place d_s equals $m_0(s)$.

Corollary 2. *Let $MW = (D, S, T, I, O, m_0)$ be a marked remembering workflow net with static places. Let $s \in S$ be a static place and let $d_s \in D_S$ be its complementary place. Then for each marking m reachable from m_0 there holds: $m(s) + m(d_s) = m_0(s)$ and therefore $m(s) \leq m_0(s)$.*

1-soundness of a marked workflow net with static places is defined analogously to soundness of workflow nets in [32] as the ability to finish correctly a single instance.

Definition 20. (Final marking of marked workflow net with static places)
Let $MW = (D, S, T, I, O, m_0)$ be a marked workflow net with static places. A marking m_f of net MW reachable from m_0 is called a final marking of MW iff there holds: $m_f(out) = 1$, $m_f(d) = 0$ for each $d \in D$ different from out.

Definition 21. (1-soundness of marked workflow net with static places)
Let $MW = (D, S, T, I, O, m_0)$ be a marked workflow net with static places and let $out \in P$ denote the output place. Marked workflow net with static places MW is 1-sound iff for each marking m reachable from m_0 there holds:

- *there exists a final marking m_f of net MW, which is reachable from marking m,*
- *if $m(out) \geq 1$ then m is a final marking of net MW.*

For marked remembering workflow nets with static places we have:

Corollary 3. *Let $MW = (D, S, T, I, O, m_0)$ be a marked remembering workflow net with static places and let m_f be a final marking of MW. Then $m_f(s) = m_0(s)$ for each $s \in S$ and therefore m_f is a unique final marking of MW.*

For 1-sound marked remembering workflow nets with static places we get:

Lemma 1. *If a marked remembering workflow net with static places is 1-sound then the number of its reachable markings is finite, i.e. the net is bounded.*

Proof. Let the number of markings reachable from m_0 is not finite. According to Dickson's lemma [35], let m and m' are markings reachable from m_0 such that $m < m'$. From definition of 1-soundness we get that $m'(out) = 1$ or $m'(out) = 0$ and $m(out) = 1$ or $m(out) = 0$. Because $m < m'$, we also get $m(s) \leq m'(s)$ for each $s \in S$.

- The combination $m'(out) = 1$ and $m(out) = 1$ means that $m' = m_f = m$, what contradicts that $m < m'$.
- The combination $m'(out) = 0$ and $m(out) = 1$ is in contradiction with $m < m'$.
- The combination $m'(out) = 1$ and $m(out) = 0$ means that $m' = m_f$. Assuming that $m_f > m$ we get $m(d) = 0$ for each $d \in D$). From first item of 1-soundness we further get that m_f is reachable from m by firing of a sequence of transitions. From definition of transition firing it is clear that whenever a sequence is enabled to fire from a marking, it is enabled to fire from any greater marking, and therefore also from m_f. Because firing of that sequence leads from m to m_f, its firing from m_f leads to the marking m'' such that $m''(out) = m_f(out) + (m_f(out) - m(out))$, i.e. $m''(out) = 1 + (1 - 0)) = 2$, what contradicts with the second item of 1-soundness implying that for each marking m'' reachable from m_0 there holds $m''(out) \leq 1$.
- The combination $m'(out) = 0$ a $m(out) = 0$ means by assumption $m < m'$, that there exists $p \in P$ different from out such that $m'(p) > m(p)$. At the same time from the first item of 1-soundness we get that m_f is reachable from m by firing a sequence of transitions. From definition of transition firing it is clear that whenever a sequence is enabled to fire from a marking, it is enabled to fire from any greater marking, and therefore also from m'. Because firing of that sequence leads from m to m_f, its firing from m' leads to the marking $m'' = m' + (m_f - m)$. Because $m'(out) = 0$ and $m(out) = 0$, we get $m''(out) = 1$. At the same time $m'(p) > m(p)$. If $p \in D$, we get $m''(p) > 0$, what contradicts the second item of 1-soundness. If $p \in S$, we get $m''(p) > m_f(p)$, what contradicts Corollaries 2 a 3.

It means that if a marked remembering workflow net with static places is 1-sound, then the number of its reachable markings is finite and therefore the net is bounded. □

6. Reachability nets

In order to investigate the ability to finish properly arbitrary number of instances running in parallel, we need to find a net, which will simulate the runtime environment with copies of a

dynamic part of a workflow net for each instance sharing just static places. Such a net should prohibit the mixing of tokens from different copies. In other words, the net simulating the runtime environment should separate reachable markings of dynamic places of instances. One possible way to reach this goal is to use the reachability graph of the original workflow net as an inspiration.

First we will define some basic notions which will be further used to define such so called reachability net.

Definition 22 (Notation).
Let P be a set, let A be a set and let D be a set such that $D \subseteq P$. Let $m : P \to A$ be a function. Let $m|D$ denote restriction of m to D, i.e. $m|D : D \to A$ such that $m|D(d) = m(d)$ for each $d \in D$. Let S be a set such that $S \subseteq P$, and let $D \cap S = \emptyset$ (D a S are disjoint), then we denote $m|D \cup m|S = m|(D \cup S)$. Let us denote by $P \setminus D$ the set difference P and D, as a set satisfying $D \cap (P \setminus D) = \emptyset$ and $(P \setminus D) \cup D = P$ (we define the set difference only for the case that D is a subset of P). Let $[P \to A]$ denote the set of all functions from P to A. For A being a finite set, let $|A|$ denote the number of elements of A.

Definition 23 (Set of reachable D-markings).
Let $MPN = (P, T, I, O, m_0)$ be a marked Petri net. Let D be a subset of places, i.e. $D \subseteq P$. Let m be a marking of MPN. Then function $m|D$ is called a D-marking. By symbol $[m_0\rangle|D$ we denote the set of all D-markings w satisfying: $w \in [m_0\rangle|D$ iff there exists such $m \in [m_0\rangle$ that $w = m|D$. It means that symbol $[m_0\rangle|D$ denotes the set of all D-markings $m|D$ such that m is reachable from m_0. We also say that $[m_0\rangle|D$ denote the set of all D-markings reachable from initial D-marking $m_0|D$ in MPN.

Each reachable D-marking $m|D$ from $[m_0\rangle|D$ of the original workflow net will become to be a place in reachability net. In addition, static places of the original workflow net will be static places of reachability net. Elements of transition relation (m, t, m') z $\longrightarrow$ of the reachability graph of original workflow net will be used to create transitions of the reachability net. A triple $(m|D, t, m'|D) \in ([m_0\rangle|D) \times T \times ([m_0\rangle|D)$ will be a transition of the reachability net iff $m \xrightarrow{t} m'$. A transition $(m|D, t, m'|D)$ of the reachability net will consume exactly one token from the place $m|D$ of the reachability net and it will produce exactly one token in place $m'|D$ of the reachability net. Arcs and their weights between static places and transition $(m|D, t, m'|D)$ of the reachability net will be identical with arcs between static places and transition t of the original net. Finally, we add a constructor consisting of two transitions $\{new, stop\}$ and one place {source}.

In graphical expression of reachability nets, we will label transition $(m|D, t, m'|D)$ of the reachability net only by the name of transition t of the original net.

Definition 24 (Reachability net).
Let $MW = (D, S, T, I, O, m_0)$ be a marked workflow net with static places and let new, $stop$ and $source$ denote elements satisfying $\{new, stop, source\} \cap (D \cup S \cup T) = \emptyset$. Reachability net of the net MW is the marked Petri net $MPN = (P^r, T^r, I^r, O^r, m_0^r)$, where

- $P^r = ([m_0\rangle|D) \cup S \cup \{source\}$.

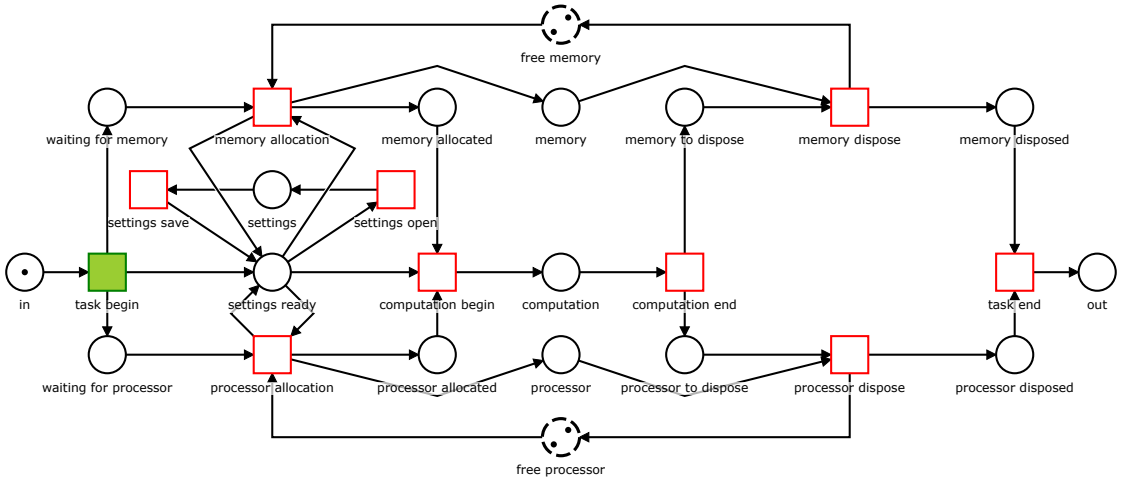

Figure 2: A 1-sound marked remembering workflow net with static places, modeling allocation of memory units and processors to computing tasks with possibility to change the setting for the computation.

- $T^r = T^a \cup \{new, stop\}$, *where* T^a *is the set of all triples* $(x,t,y) \in ([m_0\rangle|D) \times T \times ([m_0\rangle|D)$, *for which there exists a triple* $(m,t,m') \in [m_0\rangle \times T \times [m_0\rangle$, *such that* $x = m|D$, $y = m'|D$ *and* $m \xrightarrow{t} m'$, *i.e. the transitions of the reachability net are* new *a* $stop$ *and such triples* (x,t,y), *where* x *a* y *are D-markings and* t *is enabled to fire in marking* m *in* MW *and its firing leads to* m' *in* MW, *while* $x = m|D$ *and* $y = m'|D$.
- $I^r(x,(x,t,y)) = 1$ *for each* $(x,t,y) \in T^a$
- $I^r(source, new) = 1$ *and* $I^r(source, stop) = 1$, *i.e. constructor* new *is enabled to fire only in a marking with a token in place* $source$, *similarly* $stop$
- $I^r(s,(x,t,y)) = I(s,t)$ *for each* $s \in S$, $(x,t,y) \in T^a$, *i.e. arcs from static places to copies of transitions are copied*
- $O^r((x,t,y),y) = 1$ *for each* $(x,t,y) \in T^a$
- $O^r(in, new) = 1$ *and* $O^r(source, new) = 1$, *i.e. firing of constructor* new *creates a token in place* in *and returns a token to place* $source$ *(remember that from Definition 18 we get* $in = m_0|D$*)*
- $O^r(s,(x,t,y)) = O(s,t)$ *for each* $s \in S$, $(x,t,y) \in T^a$, *i.e. arcs from copies of transitions to static places are copied*
- $I^r(p,t) = 0$ *a* $O^r(p,t) = 0$ *for each other pairs* $(p,t) \in (P^r \times T^r)$
- $m_0^r(source) = 1$, $m_0^r|S = m_0|S$ *a* $m_0^r(x) = 0$ *for each* $x \in [m_0\rangle|D$, *i.e. in the initial marking is one token in place* $source$, *tokens in static places are copied and places of the reachability net equal to reachable D-markings of the net* MW *are empty.*

To illustrate a reachability net, we will consider the net in Figure 2

Existence of the complementary places of static places in remembering workflow net implies following result:

Corollary 4. *Let $MW = (D, S, T, I, O, m_0)$ be a marked remembering workflow net with static places. Then for each two markings m and m' from $[m_0\rangle$ there holds: if $m|D = m'|D$ then $m = m'$.*

Corollary 5. *Let $MW = (D, S, T, I, O, m_0)$ be a 1-sound marked remembering workflow net with static places. Then its reachability net $MPN = (P^r, T^r, I^r, O^r, m_0^r)$ has a finite number of places P^r and a finite number of transitions T^r.*

Places in the reachability net of remembering net represent states of the instance, including information how many tokens from static places are used by the instance. Each token in a place of the reachability net represent an instance in the corresponding state. The number of tokens in a place of the reachability net determine how many instances are in that state. Thus, marking of the reachability net determines how many instances are in states represented by single places of the reachability net.

Definition 25 (Final marking of a reachability net).
Let $MW = (D, S, T, I, O, m_0)$ be a marked workflow net with static places and let a marked Petri net $MPN = (P^r, T^r, I^r, O^r, m_0^r)$ be the reachability net of the net MW. A marking m_f^r of the reachability net MPN reachable from m_0^r, is called a final marking of MPN, if $m_f^r(x) = 0$ for each $x \in P^r \setminus (S \cup \{out\})$ (where out denotes in accordance with the introduced notation the reachable D-marking given by function $1 \cdot out$ from D to $\mathbb{N}$).

A lock of the reachability net is defined as a marking, from which no final marking is reachable.

Definition 26. (Lock, deadlock and livelock of a reachability net)
Let $MW = (D, S, T, I, O, m_0)$ be a marked workflow net with static places and let a marked Petri net $MPN = (P^r, T^r, I^r, O^r, m_0^r)$ be the reachability net of the net MW. A marking m^r of the reachability net MPN reachable from the initial marking m_0^r, is called a lock, if from m^r no final marking of the reachability net MPN is reachable. If no transition of the reachability net MPN is enabled to fire in a lock m^r, then it is called a deadlock of MPN, otherwise it is called a livelock of MPN.

7. Basic locks of a reachability net

Following result, which is implied directly by the construction of the reachability net is important for detection of locks of reachability nets.

Lemma 2. *Let $MW = (D, S, T, I, O, m_0)$ be a 1-sound marked remembering workflow net with static places and let a marked Petri net $MPN = (P^r, T^r, I^r, O^r, m_0^r)$ be the reachability net of the net MW. Let marking m_1^r be reachable from m_0^r in the reachability net MPN. Then for arbitrary marking $w : (P^r \setminus S) \to \mathbb{N}$ satisfying $m_1^r|(P^r \setminus S) > w$ there exists such marking m_2^r reachable from m_0^r in MPN, that $m_2^r|(P^r \setminus S) = w$.*

In the following definition we will divide the places of the reachability net according to the fact, whether they represent states, in which resources from static places are used.

Definition 27 (Places with resources).
Let $MW = (D, S, T, I, O, m_0)$ be a 1-sound marked remembering workflow net with static places and let a marked Petri net $MPN = (P^r, T^r, I^r, O^r, m_0^r)$ be the reachability net of the net MW.

- *A place $x \in P^r \setminus S$ of the reachability net is called a place without resource s for $s \in S$ if either $x = source$ or for the complementary places $d_s \in D_S$ of place s there holds $x(d_s) = 0$.*
- *The set of all places without resource s is denoted by B_s^r.*
- *If place $x \in P^r \setminus S$ of the reachability net is a place without resource for each $s \in S$, we call it simply a place without resources.*
- *The set of all places without resources is denoted by B^r.*
- *A place $x \in P^r \setminus S$ of the reachability net is called place with resource s for $s \in S$ if for the complementary place $d_s \in D_S$ of s there holds $x(d_s) > 0$. The set of all places with resource s is denoted by Z_s^r.*
- *If for a place $x \in P^r \setminus S$ of the reachability net there is $s \in S$ such that x is a place with resource s, then we call it simply a place with resources.*
- *The set of all places with resources is denoted by Z^r.*
- *For a place $x \in Z^r$ of the reachability net we denote by symbol $Z(x)$ the set of such $s \in S$, for which $x \in Z_s^r$.*

Similarly as states, we divide the transitions of the reachability net for those, that need tokens from static places to be enabled to fire and those, that do not need tokens from static places. We will divide the places according to the fact, whether there is a transition, which moves a token from a place with resources and at the same time it requires resources.

Definition 28. (Transitions requiring resources, critical places)
Let $MW = (D, S, T, I, O, m_0)$ be a 1-sound marked remembering workflow net with static places and let a marked Petri net $MPN = (P^r, T^r, I^r, O^r, m_0^r)$ be the reachability net of the net MW.

- *A transition $(x, t, y) \in T^a$ is called a transition requiring resources if $I(s, t) > 0$ for some $s \in S$.*
- *If for a place with resources $x \in Z^r$ there holds that there is a transition $(x, t, y) \in T^a$ requiring resources, then $x$is called a critical place of the reachability net MPN.*
- *The set of all critical places is denoted by K^r.*

On Figure 3 there is the reachability net of the 1-sound marked remembering workflow net with static places from Figure 2.

The set of places without resources B^r in the reachability net on Figure 3 is given by six places: by place *source*, by place *in*, by place *waiting for memory + waiting for processor + settings ready*, by place *waiting for memory + waiting for processor + settings*, by place *memory disposed + processor disposed* and by the place *out*.

The set of places with resources Z^r is given by ten places, two of which are critical.

First critical place of the reachability net is its place *waiting for memory + processor allocated + settings ready + processor*. Second critical place of the reachability net is its place *memory allocated + waiting for processor + settings ready + memory*.

On Figure 4 there is a livelock of the process for ten instances.

Corollary 6. *Let $MW = (D, S, T, I, O, m_0)$ be a 1-sound marked remembering workflow net with static places and let a marked Petri net $MPN = (P^r, T^r, I^r, O^r, m_0^r)$ be the reachability net of the net MW. Let $s \in S$ be a static place and let $d_s \in D_S$ be its complementary place. Then for each marking m^r reachable from m_0^r there holds $m^r(s) + \sum_{x \in Z_s^r} m^r(x) \cdot x(d_s) = m_0(s)$, and therefore $m^r(s) \leq m_0^r(s) = m_0(s)$.*

Another important result is given as follows:

Lemma 3. *Let $MW = (D, S, T, I, O, m_0)$ be a 1-sound marked remembering workflow net with static places and let a marked Petri net $MPN = (P^r, T^r, I^r, O^r, m_0^r)$ be the reachability net of the net MW. Then for arbitrary marking m_1^r reachable from m_0^r in the reachability net there exists a marking m_2^r reachable from m_1^r such that $m_2^r(x) = 0$ for each $x \in Z^r \setminus K^r$.*

Proof. We show, that if a token is in a place $x \in Z^r \setminus K^r$, we can move that token to a places without resources from B^r, or to a critical place from K^r. Repeating the procedure we can set to zero all places $x \in Z^r \setminus K^r$. Because the original remembering workflow net is 1-sound, from each $x \in Z^r \subseteq [m_0\rangle|D$ is in original workflow net reachable a final marking $out \cup m_0|S$. It means, that for each place $x \in Z^r \setminus K^r$ there exists a finite sequence $\epsilon : \mathbb{I} \to T$ a $\sigma : \mathbb{I} \cup \{0\} \to [m_0\rangle|D$ such that $\sigma(0) = x$, $(\sigma(i-1), \epsilon(i), \sigma(i)) \in T^a$ for each positive integer $i \in \mathbb{I}$ and $\sigma(max_{\mathbb{I}}) = out$. Let $\alpha : \mathbb{I} \to T^a$ be a sequence of transitions from T^a such that $\alpha(i) = (\sigma(i-1), \epsilon(i), \sigma(i))$ for $i \in \mathbb{I}$.

- Let no transition $\alpha(i)$, where $i \in \mathbb{I}$, is a transition requiring resources. Then sequence α is enabled to fire in any marking of the reachability net $m_1^r \in [m_0^r\rangle$ satisfying $m_1(x) > 0$ and its firing leads to marking m_2^r, such that $m_2^r(x) = m_1^r(x) - 1$ and $m_2^r(out) = m_1^r(out) + 1$, i.e. firing of sequence α moves a token from a place with resources x of the reachability net to the place without resources out of the rechability net.
- Let there is an $i \in \mathbb{I}$ such that transition $\alpha(i)$ is a transition requiring resources. Let i be the smallest index, for which $\alpha(i)$ is a transitions requiring resources. Because x is not a critical place, $i \geq 2$.
  - Let $\sigma(i-1) \in B^r$, i.e. $\sigma(i-1)$ is a place without resources. Then sequence $\alpha|\{1, \ldots, i-1\}$ is enabled to fire in each marking of the reachability net $m_1 \in [m_0^r\rangle$ satisfying $m_1^r(x) > 0$ and its firing leads to marking m_2^r, such that $m_2^r(x) = m_1^r(x) - 1$ and $m_2^r(\sigma(i-1)) = m_1^r(\sigma(i-1)) + 1$, i.e. firing of sequence α moves a token from place x of the reachability net to a place without resources $\sigma(i-1)$ of the reachability net.
  - Let $\sigma(i-1) \in Z^r$, i.e. $\sigma(i-1)$ is a place with resources. Because $\alpha(i) = (\sigma(i-1), \epsilon(i), \sigma(i))$ is a transition requiring resources, $\sigma(i-1) \in K^r$, i.e. $\sigma(i-1)$ is a critical place. Then sequence $\alpha|\{1, \ldots, i-1\}$ is enabled to fire in each marking of the rechability net $m_1^r \in [m_0^r\rangle$ satisfying $m_1(x) > 0$ and its firing leads to marking m_2^r, such that $m_2^r(x) = m_1^r(x) - 1$ and $m_2^r(\sigma(i-1)) = m_1^r(\sigma(i-1)) + 1$, i.e. firing of sequence α moves a token from place x of the reachability net to a critical place $\sigma(i-1)$ of the reachability net.

□

If we apply the result of Lemma 3 to locks, we get that from each lock of the reachability net we can reach a lock, in which from all places with resources only critical places are marked. These locks are called critical locks.

Definition 29 (Critical locks of a reachability net).
Let $MW = (D, S, T, I, O, m_0)$ be a 1-sound marked remembering workflow net with static places and let a marked Petri net $MPN = (P^r, T^r, I^r, O^r, m_0^r)$ be the reachability net of the net MW. Then a lock m^r, such that $m^r(x) = 0$ for each $x \in Z^r \setminus K^r$, is called a critical lock of the reachability net.

Corollary 7. *Let $MW = (D, S, T, I, O, m_0)$ be a 1-sound marked remembering workflow net with static places and let a marked Petri net $MPN = (P^r, T^r, I^r, O^r, m_0^r)$ be the reachability net of the net MW. Then there holds: if an arbitrary marking m_1^r reachable from m_0^r in the reachability net is a lock, then there exists a critical lock m_2^r reachable from m_1^r.*

The lock in Figure 4 is not a critical lock, because the place with resources *waiting for memory + processor allocated + settings + processor* and the place with resources *memory allocated + waiting for processor + settings + memory* are not empty. These places with resources are not critical places of the reachability net.

From marking in Figure 4 the marking in Figure 5 is reachable. The marking in Figure 5 is a critical lock.

A special set of critical lock are those locks, for which no other places except critical places and static places are marked. Such lock are called basic locks.

Definition 30 (Basic locks of a reachability net).
Let $MW = (D, S, T, I, O, m_0)$ be a 1-sound marked remembering workflow net with static places and let a marked Petri net $MPN = (P^r, T^r, I^r, O^r, m_0^r)$ be the reachability net of the net MW. Then a critical lock m^r, such that $m^r(x) = 0$ for each place without resources $x \in B^r$, is called a basic lock of the reachability net.

The following results states, that if m_1^r is a critical lock, then marking m_2^r created from m_1^r by removing all tokens from all places without resources, is reachable from z m_0^r and therefore it is a basic lock of the reachability net.

Lemma 4. *Let $MW = (D, S, T, I, O, m_0)$ be a 1-sound marked remembering workflow net with static places and let a marked Petri net $MPN = (P^r, T^r, I^r, O^r, m_0^r)$ be the reachability net of the net MW. Then there holds: if an arbitrary marking m_1^r reachable from m_0^r in the reachability net is a critical lock but not a basic lock, then m_2^r, such that $m_2^r(x) = m_1^r(x)$ for each place with resources $x \in Z^r$ and $m_2^r(x) = 0$ for each place without resources $x \in B^r$, is a basic lock of the reachability net.*

Proof. According to Lemma 2, m_2^r is reachable from m_0^r. Assume, that m_2^r is not a basic lock. Because m_1^r is a critical lock, $m_2^r(x) = 0$ for each $x \in Z^r \setminus K^r$, i.e. m_2^r satisfies the condition of critical locks. Because $m_2^r(x) = 0$ for each place without resources $x \in B^r$, m_2^r satisfies also the condition of basic locks. It means, that m_2^r is not a lock. Then there is a sequence of transitions

β enabled to fire in m_2^r, such that its firing leads to a final marking m_f^r, where $m_f^r|S = m_0^r|S$. Because $m_2^r(x) = m_1^r(x)$ for each place with resources $x \in Z^r$, i.e. in non-static places of the reachability net differ m_2^r and m_1^r only in marking of places without resources, there also holds that $m_2^r|S = m_1^r|S$, i.e. the markings of static places of m_2^r and m_1^r are equal. Then the sequence β is enabled to fire also in m_1^r and its firing leads to m_3^r such that $m_3^r|Z^r = 0$ and $m_3^r|S = m_0^r|S$.

Because original net MW is 1-sound, (similarly as in the proof of Lemma 3) from each $x \in B^r \subseteq [m_0\rangle|D$ is in MW reachable the final marking $out \cup m_0|S$ of MW. It means, that for each place $x \in B^r$ there is a finite sequence $\epsilon : \mathbb{I} \to T$ a $\sigma : \mathbb{I} \cup \{0\} \to [m_0\rangle|D$ such that $\sigma(0) = x$, $(\sigma(i-1), \epsilon(i), \sigma(i)) \in T^a$ for each finite positive integer $i \in \mathbb{I}$ and $\sigma(max_{\mathbb{I}}) = out$. Let $\alpha : \mathbb{I} \to T^a$ be the sequence of transitions from T^a such that $\alpha(i) = (\sigma(i-1), \epsilon(i), \sigma(i))$ for $i \in \mathbb{I}$. Sequence α is enabled to fire in each marking of the reachability net $m_1^r \in [m_0^r\rangle$ satisfying $m_1(x) > 0$ and $m_1^r|S = m_0^r|S$ and its firing leads to marking m_4^r, where $m_4^r(x) = m_1^r(x) - 1$ and $m_4^r(out) = m_1^r(out) + 1$, i.e. firing of sequence α moves a token from a place without resources x of the reachability net to a place without resources out of the reachability net, and also $m_4^r|Z^r = 0$, $m_4^r|B^r \setminus \{x, out\} = m_1^r|B^r \setminus \{x, out\}$ a $m_4^r|S = m_1^r|S = m_0^r|S$. By repeating the firing of sequence α we get marking m_5^r, where $m_5^r(x) = 0$ and $m_5^r(out) = m_1^r(out) + m_1^r(x)$, i.e. $m_1^r(x)$-times repeated firing of sequence α moves all $m_1^r(x)$ tokens from place without resources x of the reachability net to the place without resources out of the reachability net, and also $m_5^r|Z^r = 0$, $m_5^r|B^r \setminus \{x, out\} = m_1^r|B^r \setminus \{x, out\}$ and $m_5^r|S = m_1^r|S = m_0^r|S$. By repeating the procedure for such $x \in B^r$ that marking of x is not zero, we get a marking m_f^r such that $m_f^r|((Br \setminus \{out\}) \cup Z^r) = 0$ and $m_f^r|S = m_0^r|S$, i.e. we get a final marking of the reachability net. This contradicts with the assumption that m_1 is a critical lock. □

By application of Lemma 4 on critical lock from Figure 5 we get the reachable basic lock in Figure 6.

Altogether, we get the following main result.

Theorem 1. *Let $MW = (D, S, T, I, O, m_0)$ be a 1-sound marked remembering workflow net with static places and let a marked Petri net $MPN = (P^r, T^r, I^r, O^r, m_0^r)$ be the reachability net of the net MW. Then there holds: if there exists a lock m_1^r, then there exists a basic lock m_2^r.*

We also get:

Corollary 8. *Let $MW = (D, S, T, I, O, m_0)$ be a 1-sound marked remembering workflow net with static places and let a marked Petri net $MPN = (P^r, T^r, I^r, O^r, m_0^r)$ be the reachability net of the net MW. If the reachability net has no critical places, then it has no locks.*

Critical places are bounded in the reachability net, and therefore basic locks are bounded, i.e. there are finitely many basic locks.

Definition 31 (Simple bound of critical places).
Let $MW = (D, S, T, I, O, m_0)$ be a 1-sound marked remembering workflow net with static places and let a marked Petri net $MPN = (P^r, T^r, I^r, O^r, m_0^r)$ be the reachability net of the net MW. Let $k \in K^r$ be a critical place. Let $s \in Z(k)$ and let $d_s \in D_S$ be the complementary

place of s. Denote $pb(k,s) = m_0(s) \div k(d_s)$, where symbol $\div$ denotes integer division. Denote $sbound(k) = \min_{s \in Z(k)} pb(k,s)$ the minimum of values $pb(k,s)$ for $s \in Z(k)$. The values $sbound(k)$ is called the simple bound of a critical place k. The sum of values

$$sbound(K^r) = \sum_{k \in K^r} sbound(k)$$

is called the simple bound of critical places K^r.

Corollary 9. *Let $MW = (D, S, T, I, O, m_0)$ be a 1-sound marked remembering workflow net with static places and let a marked Petri net $MPN = (P^r, T^r, I^r, O^r, m_0^r)$ be the reachability net of the net MW. Let m^r be an arbitrary marking of the reachability net reachable from m_0^r. For each critical place $k \in K^r$ there holds $m^r(k) \leq sbound(k)$, i.e. the number of tokens in any reachable marking m^r does not exceed the simple bound of a critical place. There holds that $\sum_{k \in K^r} m^r(k) \leq sbound(K^r)$. In addition, $sbound(K^r) \geq 1$.*

More exact upper bound of the number of tokens in critical places can be computed as a solution of the following integer linear programming problem.

Definition 32 (Bound of critical places).
Let $MW = (D, S, T, I, O, m_0)$ be a 1-sound marked remembering workflow net with static places and let a marked Petri net $MPN = (P^r, T^r, I^r, O^r, m_0^r)$ be the reachability net of the net MW. Let $y : K^r \to \mathbb{N}$ be an integer solution of the linear inequality system

$$\sum_{k \in K^r} k(d_s) \cdot y(k) \leq m_0(s) \; for \; s \in S$$

$$y(k) \geq 0 \; for \; k \in K^r$$

which maximizes the objective function

$$\sum_{k \in K^r} k(d_s) \cdot y(k)$$

This maximal value

$$bound(K^r) = \sum_{k \in K^r} k(d_s) \cdot y(k)$$

is called the bound of critical places K^r.

From the formulation of the integer linear programming problem we get the following result.

Corollary 10. *Let $MW = (D, S, T, I, O, m_0)$ be a 1-sound marked remembering workflow net with static places and let a marked Petri net $MPN = (P^r, T^r, I^r, O^r, m_0^r)$ be the reachability net of the net MW. Let m^r be an arbitrary marking of the reachability net reachable from m_0^r. Then there holds $\sum_{k \in K^r} m^r(k) \leq bound(K^r)$. Moreover, there holds that $sbound(K^r) \geq bound(K^r)$.*

Methods for solving the integer linear programming problems can be found e.g. in [36].

For the reachability net in Figure 4 of the 1-sound marked remembering workflow net with static places from Figure 2 we get $sbound(K^r) = bound(K^r) = 4$.

From definition of the reachability net we get the following result.

Lemma 5. *Let $MW = (D, S, T, I, O, m_0)$ be a 1-sound marked remembering workflow net with static places and let a marked Petri net $MPN = (P^r, T^r, I^r, O^r, m_0^r)$ be the reachability net of the net MW. Let m_1^r be an arbitrary marking of the reachability net reachable from m_0^r. Then for each marking m_2^r reachable from m_1^r there holds:*

$$\sum_{x \in [m_0\rangle|D} m_1^r(x) \leq \sum_{x \in [m_0\rangle|D} m_2^r(x)$$

If $m_1^r(source) = 0$ then there holds:

$$\sum_{x \in [m_0\rangle|D} m_1^r(x) = \sum_{x \in [m_0\rangle|D} m_2^r(x)$$

Consider a constrained reachability net for a given positive integer $n \in \mathbb{Z}$ by adding a place, let us call it a *buffer*, from which firing of transition new consumes just one token, while in the initial marking this place get n number of tokens, i.e. $m_0^r(\textit{buffer}) = n$.

Such constrained reachability net preserves all the reachable markings for which the sum of tokens in places from $[m_0\rangle|D$ does not exceed n. It also preserves the firing of all transitions from T^a in these markings. The transition new is enabled to fire in such a net exactly n-times. Such net is bounded and represent the running of at most n instances in parallel. Basic locks in such a constrained net can be defined analogously as for the reachability net without constraints, just allowing non zero marking of $buffer$.

If one set the initial value $m_0^r(\textit{buffer}) = sbound(K^r)$ or $m_0^r(\textit{buffer}) = bound(K^r)$ one can guarantee that whenever the reachability net has a basic lock then the constrained net has the basic lock differing at most in the marking of the added place *buffer*. The constrained reachability net with $m_0^r(\textit{buffer}) = sbound(K^r) = bound(K^r) = 4$ for the reachability net from Figure 3 is in Figure 7.

8. Conclusion

We have investigated workflow processes with independent instances, which share durable resources. We have considered the fixed number of resources and arbitrary number of instances. We modeled such nets by so called 1-sound marked remembering workflow nets with static places, which are in fact resource constrained workflow nets sound for one instance. We have shown that detection of deadlocks and livelocks of such processes can be reduced to detection of basic locks of the constrained bounded reachability nets of 1-sound marked remembering workflow nets with static places.

One of the main aims of the further research is to find out how to synthesize a minimally restrictive completion of a net which has deadlocks or livelock in order to avoid them. Another aim of the further research is to determine the set of initial markings of static places, i.e. to determine the number of resources, needed to avoid the deadlock and livelocks.

References

[1] K. Hee, A. Serebrenik, N. Sidorova, M. Voorhoeve, Soundness of resource-constrained workflow nets, in: Applications and Theory of Petri nets, LNCS, volume 3536, Springer, Berlin, Heidelberg, 2005, pp. 250–267. doi:10.1007/11494744_15.

[2] K. Barkaoui, R. Ben Ayed, Z. Sbai, Workflow soundness verification based on structure theory of Petri nets, International Journal of Computing and Information Sciences 5 (2007) 51–61.

[3] K. Hee, N. Sidorova, M. Voorhoeve, Resource-constrained workflow nets., Fundamenta Informaticae 71 (2006) 243–257.

[4] G. Juhás, I. Kazlov, A. Juhásová, Instance deadlock: A mystery behind frozen programs, in: Applications and Theory of Petri Nets, LNCS, volume 6128, Springer, Berlin, Heidelberg, 2010, pp. 1–17. doi:10.1007/978-3-642-13675-7_1.

[5] A. Juhásová, G. Juhás, Soundess of resource constrained workflow nets is decidable., Petri Net Newsletter. (2009) 3–6.

[6] M. Martos-Salgado, F. Rosa-Velardo, Dynamic soundness in resource-constrained workflow nets, in: Formal Techniques for Distributed Systems, LNCS, volume 6722, Springer, Berlin, Heidelberg, 2011, pp. 259–273. doi:10.1007/978-3-642-21461-5_17.

[7] M. R. Martos Salgado, Verification of priced and timed extensions of Petri Nets with multiple instances., Ph.D. thesis, 2016.

[8] D. F. Escrig, C. Johnen, Decidability of home space property, in: LRI Report, volume 503, Université Paris-Sud, 1989, pp. 1–25.

[9] N. Sidorova, C. Stahl, Soundness for resource-constrained workflow nets is decidable, IEEE Transactions on Systems, Man, and Cybernetics: Systems 43 (2013) 724–729. doi:10.1109/TSMCA.2012.2210415.

[10] K. Hee, N. Sidorova, M. Voorhoeve, Soundness and separability of workflow nets in the stepwise refinement approach, in: Applications and Theory of Petri nets, LNCS, volume 2679, Springer, Berlin, Heidelberg, 2003, pp. 337–356. doi:10.1007/3-540-44919-1_22.

[11] E. Best, P. Darondeau, Separability in persistent petri nets, Fundamenta Informaticae 113 (2011) 179–203.

[12] F. L. Tiplea, C. Bocaneala, Decidability results for soundness criteria of resource-constrained workflow nets, IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans 42 (2012) 238–249. doi:10.1109/TSMCA.2011.2147310.

[13] E. Mayr, Persistence of vector replacement systems is decidable, Acta Informatica 15 (1981) 309–318. URL: https://doi.org/10.1007/BF00289268. doi:10.1007/BF00289268.

[14] S. R. Kosaraju, Decidability of reachability in vector addition systems (preliminary version), in: Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing, STOC '82, Association for Computing Machinery, New York, NY, USA, 1982, p. 267–281. doi:10.1145/800070.802201.

[15] E. W. Mayr, An algorithm for the general Petri net reachability problem, SIAM Journal on computing 13 (1984) 441–460.

[16] J. Lambert, A structure to decide reachability in Petri nets, Theoretical Computer Science 99 (1992) 79–104. doi:https://doi.org/10.1016/0304-3975(92)90173-D.

[17] J. Leroux, Vector addition system reachability problem: A short self-contained proof, ACM

SIGPLAN Notices 46 (2011) 307–316. doi:10.1145/1925844.1926421.
[18] R. Keller, Formal verification of parallel programs, Communications of the ACM 19 (1976) 371–384. doi:10.1145/360248.360251.
[19] G. Winskel, M. Nielsen, Models for Concurrency, Oxford University Press, Inc., USA, 1995, p. 1–148.
[20] R. M. Karp, R. E. Miller, Parallel program schemata, Journal of Computer and System Sciences 3 (1969).
[21] T. Murata, Petri nets: Properties, analysis and applications, Proceedings of the IEEE 77 (1989) 541–580. doi:10.1109/5.24143.
[22] J. L. Peterson, Petri nets, ACM Computing Surveys 9 (1977) 223–252. doi:10.1145/356698.356702.
[23] W. Reisig, A Primer in Petri Net Design, Springer, Berlin, Heidelberg, 1992.
[24] J. Desel, W. Reisig, Place/transition Petri nets, in: Lectures on Petri Nets I: Basic Models: Advances in Petri Nets, LNCS, volume 1491, Springer, Berlin, Heidelberg, 1998, pp. 122–173. doi:10.1007/3-540-65306-6_15.
[25] J. Desel, G. Juhás, What is a Petri net? An informal answer for an informed reader, in: Unifying Petri Nets, Advances in Petri Nets, LNCS, volume 2128, Springer, Berlin, Heidelberg, 2001, pp. 1–25. doi:10.1007/3-540-45541-8_1.
[26] R. Lipton, The reachability problem is exponential-space hard, in: Technical Report, volume 62, Department of Computer Science, Yale University, 1976.
[27] C. Rackoff, The covering and boundedness problems for vector addition systems, Theoretical Compututer Science 6 (1978) 223–231. doi:10.1016/0304-3975(78)90036-1.
[28] L. E. Rosier, H.-C. Yen, A multiparameter analysis of the boundedness problem for vector addition systems, Journal of Computer and System Sciences 32 (1986) 105–135. doi:https://doi.org/10.1016/0022-0000(86)90006-1.
[29] W. M. P. van der Aalst, Three good reasons for using a Petri-net-based workflow management system, in: Information and Process Integration in Enterprises, Springer US, Boston, MA, 1998, pp. 161–182. doi:10.1007/978-1-4615-5499-8_10.
[30] W. M. P. van der Aalst, Verification of workflow nets, in: Application and Theory of Petri Nets 1997, Springer Berlin Heidelberg, Berlin, Heidelberg, 1997, pp. 407–426. doi:10.1007/3-540-63139-9_48.
[31] W. M. P. van der Aalst, The application of Petri nets to workflow management, Journal of Circuits, Systems, and Computers 8 (1998) 21–66. doi:10.1142/S0218126698000043.
[32] W. Van Der Aalst, K. M. Van Hee, K. van Hee, Workflow management: models, methods, and systems, The MIT Press, Cambridge, Massachusetts, 2002.
[33] K. Barkaoui, L. Petrucci, Structural analysis of workflow nets with shared ressources, in: Workflow Management: Net-based Concepts, Models, Techniques and Tools (WFM'98). Computing science reports, volume 9807, TU Eindhoven, 1998, pp. 82–95.
[34] R. Devillers, The semantics of capacities in P/T nets, in: Advances in Petri Nets 1989, Springer, Berlin, Heidelberg, 1990, pp. 128–150. doi:10.1007/3-540-52494-0_28.
[35] D. Figueira, S. Figueira, S. Schmitz, P. Schnoebelen, Ackermannian and primitive-recursive bounds with Dickson's lemma, in: 2011 IEEE 26th Annual Symposium on Logic in Computer Science, 2011, pp. 269–278. doi:10.1109/LICS.2011.39.
[36] A. Schrijver, Theory of Linear and Integer Programming, John Wiley, 1986.

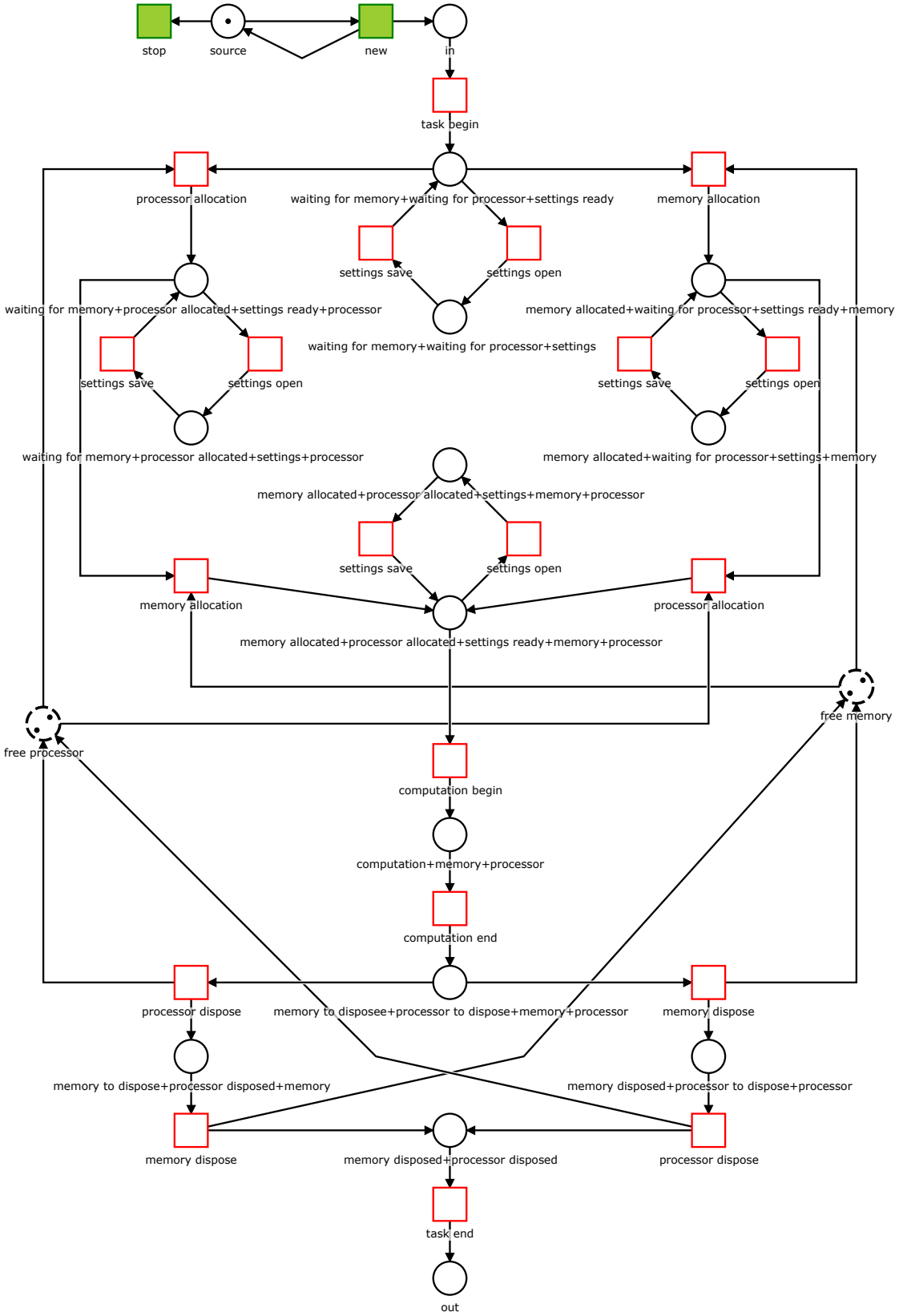

Figure 3: The reachability net of the 1-sound marked remembering workflow net with static places from Figure 2

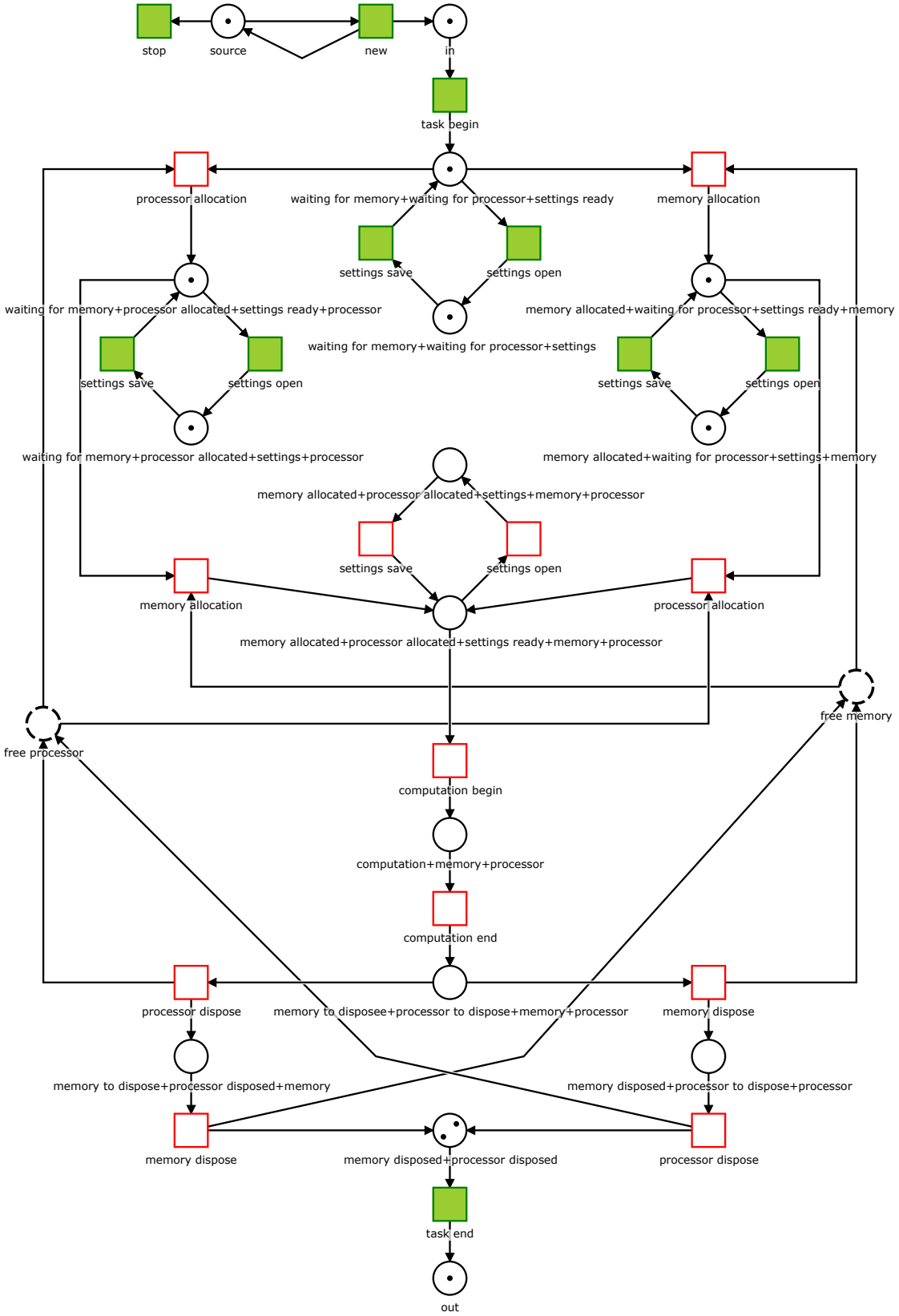

Figure 4: The reachability net of the 1-sound marked remembering workflow net with static places from Figure 2 in a livelock for ten instances

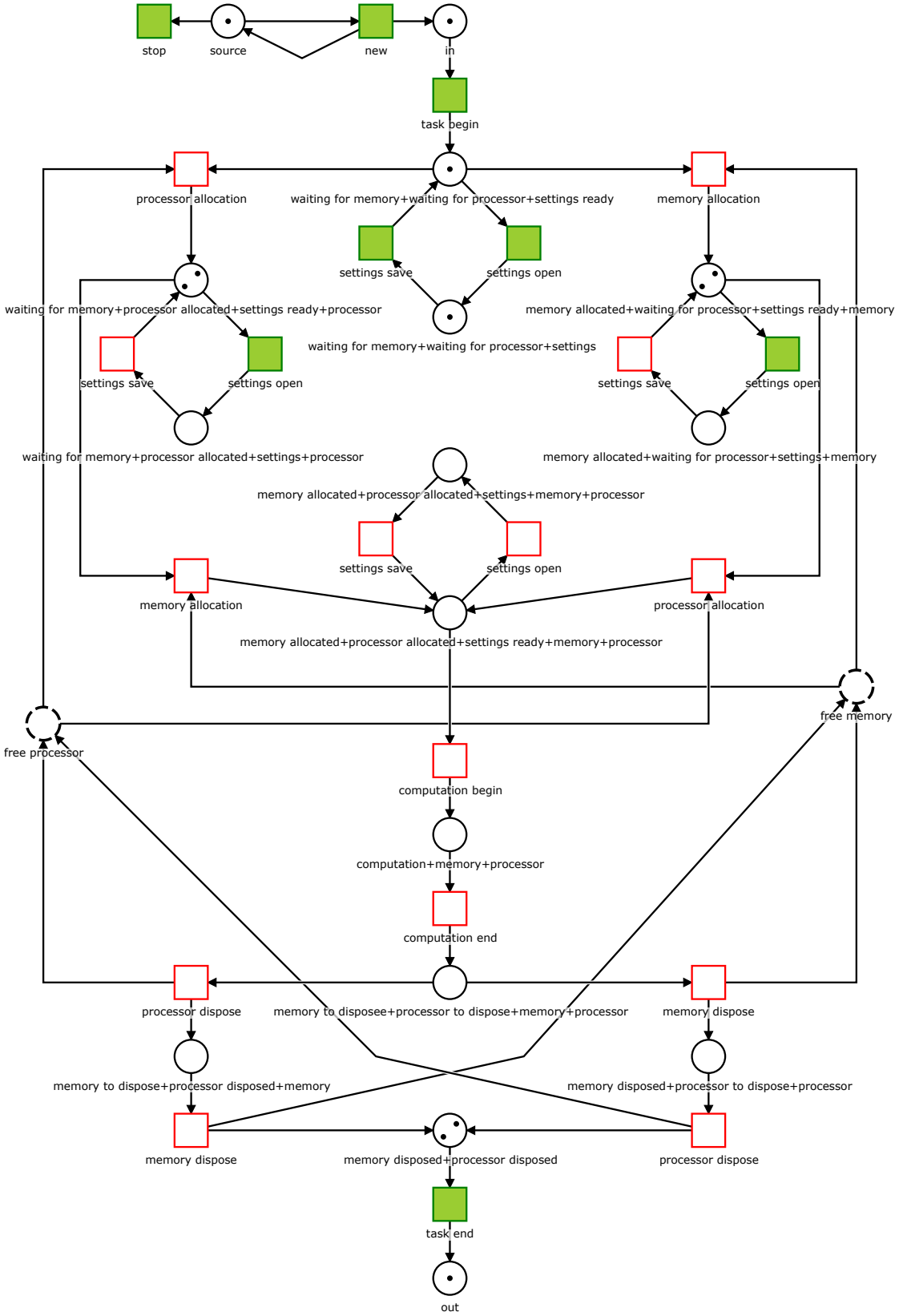

Figure 5: The reachability net of the 1-sound marked remembering workflow net with static places from Figure 2 in a critical livelock for ten instances

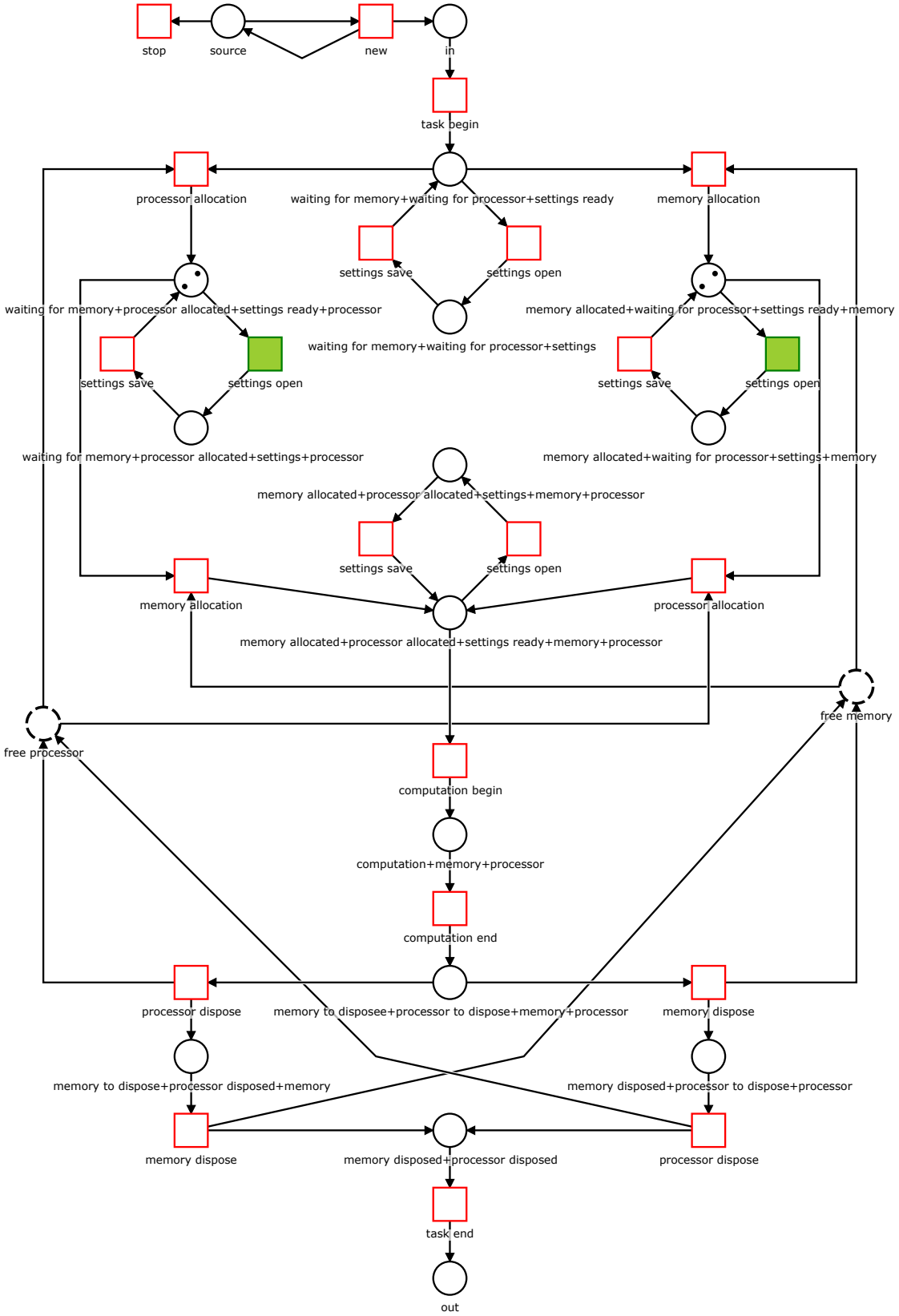

Figure 6: The reachability net of the 1-sound marked remembering workflow net with static places from Figure 2 in a basic livelock for four instances

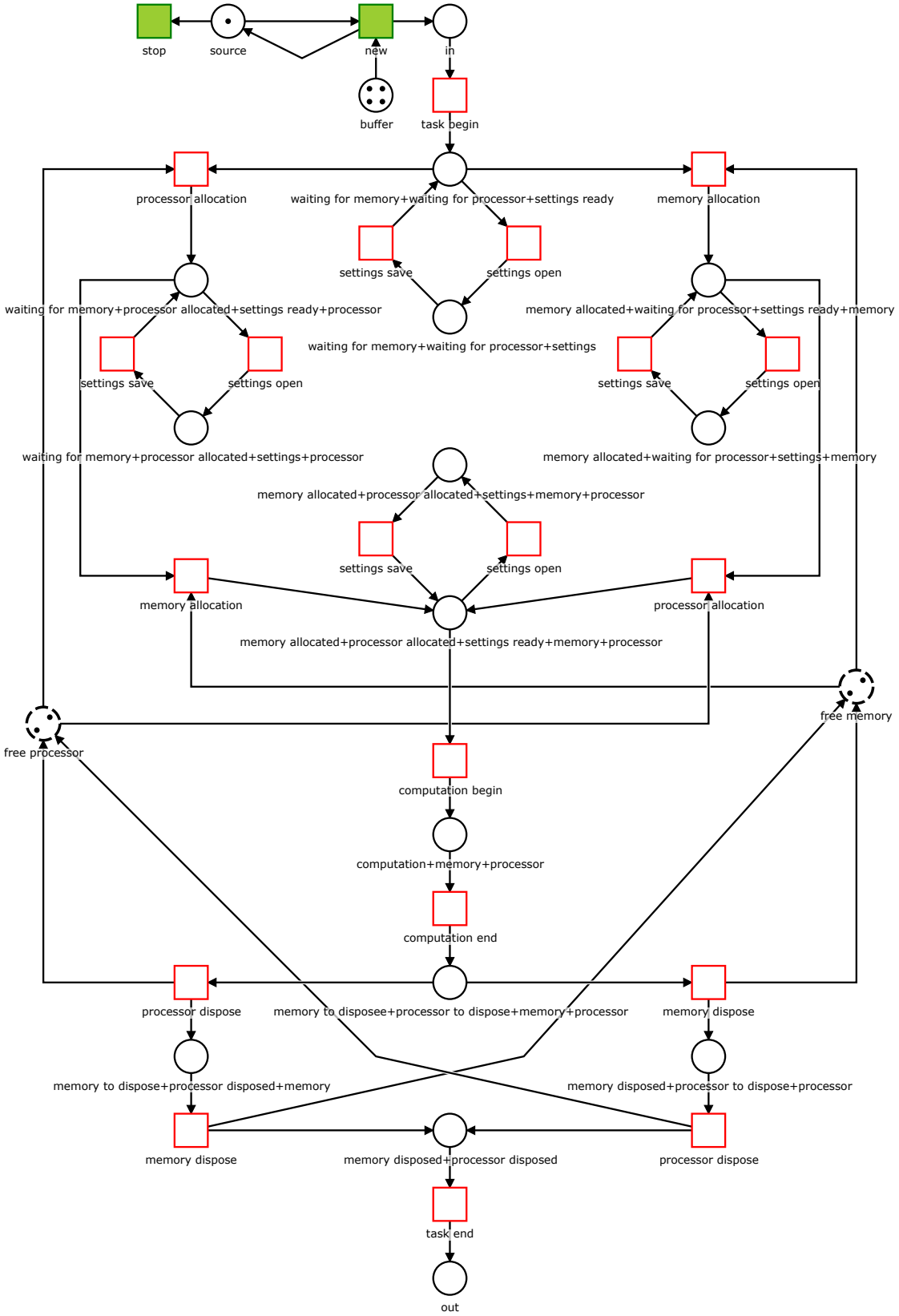

Figure 7: n-constrained bounded reachability net of the reachability net from Figure 3 for $n = sbound(K^r) = bound(K^r) = 4$