Interval Soundness of Resource-Constrained Workflow Nets: Decidability and Repair

Elham Ramezani, Natalia Sidorova, Christian Stahl

Interval Soundness of Resource-Constrained Workflow Nets: Decidability and Repair

Elham Ramezani, Natalia Sidorova, and Christian Stahl

Department of Mathematics and Computer Science
Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
{E.Ramezani, N.Sidorova, C.Stahl}@tue.nl

Abstract. Correctness of workflow design cannot be evaluated by checking the execution for one single instance of the workflow, because instances, even when being independent from the data perspective, depend on each other with respect to the resources they rely on for executing tasks. The resources are *shared among the instances* of the same workflow; moreover, other workflows can use the same resources. Therefore, we enrich the workflow model with the model of its *environment* that captures the resource perspective. This allows us to investigate the verification of workflows extended with resources in a more general setting than it was previously done. We focus on the *soundness* property, which means the ability to terminate properly from any reachable state of the system, for every instance of the system. We show the *decision* procedure for soundness and how to *repair* a workflow that is unsound from the resource perspective by synthesizing a controller such that the composition of the workflow and the controller is sound by design.

1 Introduction

A workflow consists of a set of coordinated tasks describing a flow of work for accomplishing some business process within an organization. The occurrence of those tasks may depend on *resources*, such as machines, manpower, and raw material. Often, several *cases* (i.e., instances) of a workflow coexist, and they may all concurrently access the resources. In that sense, the execution of a workflow is similar to executing several threads of a piece of software.

Correctness of classical and resource-constrained workflows has been formalized in terms of the *soundness* property [1,14]. Soundness guarantees that given a finite number of cases and a number of resources of each type, every case has always the possibility to terminate. As we restrict ourselves to *durable* resources in this paper—that is, resources that can neither be created nor destroyed—soundness also ensures that the number of resources initially available remains invariant.

The current notion of soundness for resource-constrained workflows assumes a workflow to be executed in isolation. However, workflows increasingly cross organizational boundaries and are usually intertwined. As a consequence, resources are no longer internal for a workflow but shared among different workflows. This, in fact, requires a different way of modeling workflows. To do so, we propose to enrich the workflow model with an *environment capturing the resource perspective of the workflow*.

The environment is generic in the sense that it can be parameterized, thereby enabling the modeling of relevant instances of practical scenarios. More precisely, the environment allows for borrowing, lending, and permanently adding and removing of resources of each type up to an initially specified number. Moreover, it also creates the cases of the workflow that are to be executed, with the number of cases taken from a specified interval.

We formalize correctness of workflows with shared resources with the notion of *interval soundness*, as it considers intervals of cases and resources. Interval soundness is defined for the composition of the workflow and the corresponding instance of the generic environment. We show that the verification of soundness reduces to check whether for the workflow it is always possible to terminate in the composition with the environment. The state of the environment can thereby be neglected because several invariant properties hold in the environment model which are necessary for interval soundness. To further support the design of interval-sound workflows, we present an approach for *repairing* an unsound workflow by synthesizing a controller (if exists) such that the composition of the workflow and the controller is interval sound.

Our contributions can be summarized as follows:

- A generalization of the model for workflows extended with resources to deal with shared resources;
- A notion of correctness considering intervals of instances and resource vectors and two procedures to decide correctness; and
- An approach to repair an incorrect workflow based on controller synthesis.

We continue by providing the background in Sect. 2. In Sect. 3, we introduce our model of resource-constrained workflow nets, the generic environment for modeling the resource perspective, and define interval soundness. In Sect. 4, we show how interval soundness can be decided, and repairing unsound workflows is studied in Sect. 5. We discuss related work in Sect. 6 and close with a conclusion.

2 Preliminaries

In this section, we provide the basic notations used in this paper, such as Petri nets and workflow nets.

For two sets P and Q, let $P \uplus Q$ denote the disjoint union; writing $P \uplus Q$ expresses the implicit assumption that P and Q are disjoint. A *multiset* or bag m over P is a mapping $m : P \to \mathbb{N}$; for example, $[p_1, 2p_2]$ denotes a multiset m with $m(p_1) = 1$, $m(p_2) = 2$, and $m(p) = 0$ for $p \in P \setminus \{p_1, p_2\}$. We define operations $+, -, =, <, >, \leq, \geq$ on multisets in the standard way. We overload the set notation, writing $\emptyset$ for the empty multiset and $\in$ for the element inclusion. We canonically extend the notion of a multiset over P to supersets $Q \supseteq P$; that is, for a mapping $m : P \to \mathbb{N}$, we extend m to the multiset $m : Q \to \mathbb{N}$ so that for all $p \in Q \setminus P$, $m(p) = 0$. Analogously, a multiset can be restricted to a subset $Q \subseteq P$. For a mapping $m : P \to \mathbb{N}$, the *restriction* of m to the elements in Q is denoted by $m|_Q : Q \to \mathbb{N}$.

Definition 1 (labeled Petri net). A *net* $N = \langle P, T, W \rangle$ consists of

– a finite set P of *places*,
– a finite set T of *transitions* such that P and T are disjoint, and
– a *weight function* $W : (P \times T) \uplus (T \times P) \to \mathbb{N}$.

A *labeled net* $N = \langle P, T, W, l, \Sigma \rangle$ is a net $\langle P, T, W \rangle$ together with an *alphabet* Σ of *actions* and a *labeling function* $l : T \to \Sigma \uplus \{\tau\}$, where τ represents an invisible, internal action. A *(labeled) Petri net* $\langle N, m_N \rangle$ is a (labeled) net N together with an *initial marking* m_N, where a *marking* $m : P \to \mathbb{N}$ is a distribution of tokens over the places. The *incidence matrix* $\mathbf{C}$ of N is defined by $\forall (p,t) \in P \times T : \mathbf{C}(p,t) = W((t,p)) - W((p,t))$.

For a transition $t \in T$, we define the *preset* ${}^\bullet t$ and the *postset* $t^\bullet$ of t as the multisets of places where every $p \in P$ occurs $W((p,t))$ times in ${}^\bullet t$ and $W((t,p))$ times in $t^\bullet$. Analogously, we define for a place $p \in P$ its preset ${}^\bullet p$ and its postset $p^\bullet$. We also lift pre- and postsets to sets of places and of transitions. A place p is a *source* place if ${}^\bullet p = \emptyset$ and a *sink* place if $p^\bullet = \emptyset$.

A transition $t \in T$ is *enabled* at a marking m, denoted by $m \xrightarrow{t}$, if ${}^\bullet t \leq m$. If t is enabled at m, it can *fire*, thereby changing the marking m to a marking $m' = m - {}^\bullet t + t^\bullet$. The firing of t is denoted by $m \xrightarrow{t} m'$; that is, t is enabled at m and firing t results in m'. Depending on the context, we interpret a marking m of N either as a multiset over P or as a vector from $P \to \mathbb{N}$. Firing transitions can be extended to sequences: $m_1 \xrightarrow{t_1} \ldots \xrightarrow{t_{k-1}} m_k$ is a *run* of N if for all $0 < i < k$, $m_i \xrightarrow{t_i} m_{i+1}$. A marking m' is *reachable from* a marking m if there exists a (possibly empty) run $m_1 \xrightarrow{t_1} \ldots \xrightarrow{t_{k-1}} m_k$ with $m = m_1$ and $m' = m_k$; for $v = t_1 \ldots t_{k-1}$, we also write $m \xrightarrow{v} m'$. Marking m' is *reachable* if $m_N = m$. The set $\mathcal{R}(m)$ represents all markings of N that are reachable from m.

A marking m of N is *b-bounded* for a bound $b \in \mathbb{N}$, if $m(p) \leq b$ for all $p \in P$. N is bounded if every reachable marking is b-bounded for some $b \in \mathbb{N}$. A transition $t \in T$ is *live* if from every reachable marking m there is a marking m' such that t is enabled at m'. If all transitions are live, then N is live. A marking m is a *home-marking* if from every reachable marking we can reach m. A set HS of markings of N is a *home-space* if for every reachable marking m, there exists a marking $m' \in HS$ such that m' is reachable from m.

A *place invariant* is a row vector $I : P \to \mathbb{Q}$ such that $I \cdot \mathbf{C} = 0$. When talking about invariants, we consider markings as *vectors*.

In the following, we define two composition operators for labeled Petri nets to model asynchronous composition based on place fusion and synchronous parallel composition based on transition fusion. The composition operator $\oplus$ merges common places of two labeled Petri nets.

Definition 2 (asynchronous composition). Two labeled nets N_1 and N_2 are *a-composable* if $(\Sigma_1 \cup T_1) \cap (\Sigma_2 \cup T_2) = \emptyset$. The *asynchronous composition* of two a-composable labeled nets is the labeled net $N_1 \oplus N_2 = \langle P_1 \cup P_2, T_1 \uplus T_2, W_1 \uplus W_2, l, \Sigma_1 \uplus \Sigma_2 \rangle$ and $l(t) = l_i(t)$ for $t \in T_i$, $i = 1, 2$.

If N_1 and N_2 are labeled Petri nets with initial markings m_{N_1} and m_{N_2}, then the composition is a labeled Petri net with initial marking $m_0 = m_{N_1} + m_{N_2}$.

We define a synchronous composition operator $\|$ where, for each common action a, an a-labeled transition of one labeled Petri net is merged with an a-labeled transition of the other. If there is more than one a-labeled transition in one of the labeled Petri nets, then each of these transitions is merged with a copy of the respective transition of the other labeled Petri net.

Definition 3 (synchronous composition). Two labeled nets N_1 and N_2 are *s-composable* if $(P_1 \uplus T_1) \cap (P_2 \uplus T_2) = (\Sigma_1 \cap \Sigma_2)$. The *synchronous composition* of two s-composable labeled nets is the labeled net $N_1 \| N_2 = \langle P, T, W, l, \Sigma \rangle$, where

- $P = P_1 \uplus P_2$,
- $T = \{t \in T_1 \cup T_2 \mid l_1(t) = \tau \vee l_2(t) = \tau\}$
  $\uplus \{(t_1, t_2) \in T_1 \times T_2 \mid l_1(t_1) = l_2(t_2) \wedge l_1(t_1) \neq \tau\}$,
- $W((p,t)) = \begin{cases} W_i((p,t)), & p \in P_i, t \in T_i, l_i(t) = \tau, i = 1,2, \\ W_i((p,t_i)), & p \in P_i, t = (t_1,t_2), i = 1,2, \\ 0, & \text{otherwise}, \end{cases}$
  $W((t,p)) = \begin{cases} W_i((t,p)), & p \in P_i, t \in T_i, l_i(t) = \tau, i = 1,2, \\ W_i((t_i,p)), & p \in P_i, t = (t_1,t_2), i = 1,2, \\ 0, & \text{otherwise} \end{cases}$
- $l(t) = \begin{cases} l_i(t), & t \in T_i, i = 1,2, \\ l_1(t_1), & t = (t_1,t_2), \end{cases}$
- $\Sigma = \Sigma_1 \cup \Sigma_2$.

If N_1 and N_2 are labeled Petri nets with initial markings m_{N_1} and m_{N_2}, then composition yields a labeled Petri net with initial marking $m_0 = m_{N_1} + m_{N_2}$.

The *labeled transition system* (LTS) $TS_N = \langle Q, \delta, \hat{q}, \Sigma \rangle$ of a labeled Petri net $N = \langle P, T, W, l, \Sigma, m_N \rangle$ consists of a set $Q = \mathcal{R}(m_N)$ of *states*, a set δ of labeled *edges* with $(q, l(t), q') \in \delta$ iff $q \xrightarrow{t} q'$ and $q, q' \in Q$, and an *initial state* $\hat{q} = m_N$.

We define the synchronous product of two labeled transition systems in the standard way: common visible actions are synchronized, all other actions are not. In fact, we have $TS_{N_1 \| N_2}$ and $TS_{N_1} \| TS_{N_2}$ are isomorph.

Definition 4 (synchronous product). The synchronous product of two LTSs TS_1 and TS_2 is the LTS $TS_1 \| TS_2 = \langle Q_1 \times Q_2, \delta, (\hat{q}_1, \hat{q}_2), \Sigma_1 \cup \Sigma_2 \rangle$ with
$\delta = \{((q_1,q_2), x, (q_1',q_2')) \mid (q_1, x, q_1') \in \delta_1, (q_2, x, q_2') \in \delta_2, x \in \Sigma_1 \cup \Sigma_2\}$
$\uplus \{((q_1,q_2), \tau, (q_1',q_2)) \mid (q_1, \tau, q_1') \in \delta_1\}$
$\uplus \{((q_1,q_2), \tau, (q_1,q_2')) \mid (q_2, \tau, q_2') \in \delta_2\}$.

Workflow Nets A workflow refers to the automation of processes by an IT infrastructure, in whole or in part [3]. Workflows are *case*-based; that is, every piece of work is executed for a specific case. The workflow definition specifies which tasks need to be executed for a case and in what order.

We can model a workflow definition as a (labeled) net, thereby modeling tasks by transitions and conditions by places; the state of a case is captured by a marking of

the net. The assumption that a typical workflow has a well-defined starting point and a well-defined ending point imposes syntactic restrictions on Petri nets that result in the following definition of a workflow net [2].

Definition 5 (WF-net). A labeled net $N = \langle P, T, W, l, \Sigma \rangle$ is a *workflow net* (WF-net) if it has a nonempty set of transitions, a single source place i, a single sink place f, and every place and every transition is on a path from i to f.

The *short-circuited net* N_s of N is the labeled net obtained from N by adding a transition t_s with $W((t,i)) = W((f,t)) = 1$ and $l(t_s) = \tau$.

In the first instance, researchers were interested in workflow correctness with respect to a single case. One of the most established correctness properties of WF-nets is *soundness*, as introduced by Van der Aalst [1] in the context of one case. Soundness guarantees that the workflow has always the possibility to terminate. Later on, multi-instance behavior attracted researchers' attention, where WF-nets are considered as parameterized systems modeling the processing of batches of tasks, as introduced in [14]. While in classical workflows cases are considered to be independent and the modeling of multiple cases in one WF-net requires the introduction of id tokens, in batch workflows cases are considered to be undistinguishable and mixable (e.g., it does not matter which employee works on which order) and, as a consequence, cases are modeled with undistinguishable black tokens. Under certain conditions on the workflow structure, called *separability*, the behavior of the WF-net with undistinguishable cases (black tokens) is equivalent (up to trace equivalence) to the behavior of the WF-net with id tokens [14,8,7]. Moreover, every net with id tokens can be transformed into an up-to-bisimulation-equivalent net with black tokens only [14,17].

Capturing the correctness notion for batch workflow nets requires the use of the generalized notion of soundness, as proposed in [14].

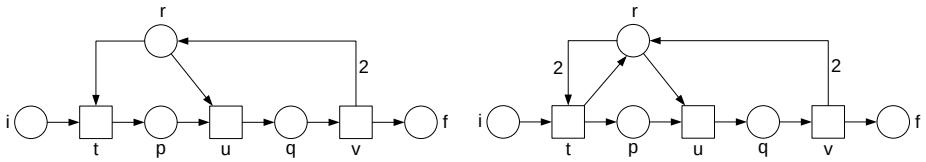
Definition 6 (WF-net soundness). Let $k \in \mathbb{N}$. A WF-net N is *k-sound* if, for every marking m reachable from marking $[k \cdot i]$, we can reach marking $[k \cdot f]$.

The next definition gives a requirement for the correct design of a workflow that can be checked using structural properties of the net [15]. *Nonredundancy* of a place $p \in P$ guarantees that p can potentially be marked with a token in some reachable marking.

Definition 7. Let $N = \langle P, T, W, l, \Sigma \rangle$ be a WF-net. A place $p \in P$ is *nonredundant* if there exist $k \in \mathbb{N}$ and $m \in \mathbb{N}^P$ such that $[k \cdot i] \xrightarrow{*} m \wedge p \in m$.

3 Generalizing Resource-Constrained Workflow Nets

We use the notion of resource-constrained workflow nets (RCWF-nets) [16] to extend the definition of the workflow with resource dependencies of the tasks. The production net of an RCWF-net is a WF-net in its traditional sense, defining the order of task execution, resource places model the resource types used by the workflow, and resource consumption and production are modeled by the arcs from the resource places to the transitions of the production net, and vice versa.



(a) RCWF-net N_1 (b) RCWF-net N_2

Fig. 1. Example of an unsound and a sound RCWF-net.

Definition 8 (RCWF-net). A labeled net $N = \langle P_p \uplus P_r, T, W_p \uplus W_r, l, \Sigma \rangle$ is a *resource-constrained workflow net* (RCWF-net) if

- $N_p = \langle P_p, T, W_p, l, \Sigma \rangle$ is a WF-net, the *production net* of N;
- P_p is the set of *production places*, and P_r is the set of *resource places*; and
- $W_r : (P_r \times T) \cup (T \times P_r) \to \mathbb{N}$ is the *resource weight function*.

The *short-circuited net* N_s of N is the labeled net obtained from N by replacing N_p with its short-circuit net.

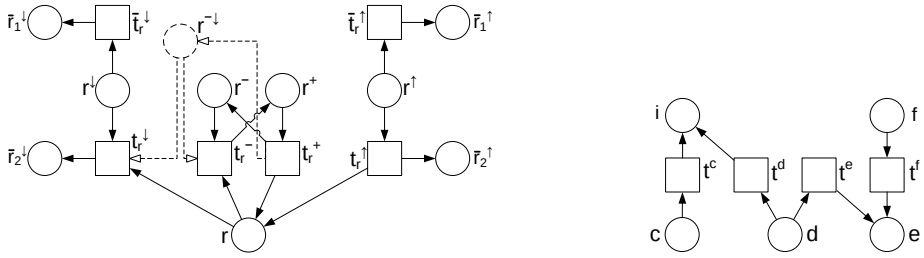
The initial marking $m_N = [k \cdot i] + R$ of an RCWF-net N consists of $k \in \mathbb{N}$ tokens in place i, specifying the number of cases in the workflow that are concurrently executed, and an initial marking for the set P_r of resources places, denoted as a resource vector $R \in \mathbb{N}^{P_r}$.

Example 1. We illustrate the previously introduced concepts using Fig. 1. The nets N_1 and N_2 are RCWF-nets with one resource place r. Arc weights are depicted on the respective arc unless they are equal to 1. Erasing r and its adjacent arcs from N_1 and N_2 yields the (same) production net, a WF-net. This WF-net is k-sound, for any $k > 0$.

3.1 The Generic Environment

To consider RCWF-nets in the setting where the workflow works within some environment that can borrow resources from the workflow or lend more resources to it, we introduce patterns capturing typical behavior of the resource environment. We consider the following actions of the environment: *borrowing* resources (the borrowed resources are then used by other workflows and they can eventually be returned and made available for the workflow again), *lending* resources (i.e., making some additional resources temporarily available, and eventually taking them back, when unused by the workflow), *permanently removing* resources, and *permanently adding* resources. Actual environments allow for a (possibly empty) subset of these actions.

We define the generic environment, built as a union of generic environments defined for every resource type (i.e., place). All the patterns can be obtained by choosing an appropriate initial marking for the generic environment. The generic environment of a resource place r is captured in Fig. 2(a). The transitions $t_r^{\uparrow}$ and $t_r^{\downarrow}$ model permanent addition and removal of resources correspondingly. Their counterparts $\bar{t}_r^{\uparrow}$ and $\bar{t}_r^{\downarrow}$ model the decision of the environment not to lend/borrow a certain number of resources, but



(a) Resource environment (b) Case environment

Fig. 2. Generic resource environment for the resource place r and case environment for a workflow with initial place i and final place f.

removing the tokens from places $r^{\uparrow}$ and $r^{\downarrow}$. The number of tokens in places $r^{\uparrow}$ and $r^{\downarrow}$ in the initial marking gives the bounds for the number of resources that can be added and removed, respectively. Clearly, choosing 0 as initial marking of $r^{\uparrow}$ and $r^{\downarrow}$ makes the corresponding transitions dead, so they might be removed for the corresponding pattern, but for the sake of readability, we prefer to use only the initial marking for configuring the generic environment into a pattern.

Transitions t_r^+ and t_r^- together with places r^+ and r^- model lending and borrowing resources in the following way: The number of tokens in r^+ and r^- in the initial marking corresponds to the number of resources the environment may lend and borrow, respectively. Thus having 0 for the initial marking on r^+ means that the environment cannot lend any resource of type r. When the resources are borrowed, the marking of r^+ is increased by the number of borrowed resources, and the firings of t_r^+ will then correspond to returning those resources by the environment.

The initial marking of the places $r^{\uparrow}$, $r^{\downarrow}$, r^+, and r^- serves thus as the configuration parameter defining the behavior of the environment. In principle, it is possible to elaborate the model further by linking, for example, $r^{\downarrow}$ and r^- by means of choosing the bound for the total number of resources that can be removed permanently or temporarily. We can model this by introducing the place $r^{-\downarrow}$ and the arcs depicted by the dashed lines in Fig. 2(a). The same can be done for other configuration components. Variations on the environment construction are also possible by linking the scheme for adding and removing resources for different resource types. We restrict our attention further to the main structure, without linking the configuration components to each other, although the results hold for environments restricted in this way, too.

Since borrowing is temporary—that is, under the fairness assumption, the environment will eventually return the borrowed resources—the choice of the initial marking for r^- does not change the set of markings reachable in the composition of the workflow and the environment projected on the workflow places: The workflow can always wait until the environment returns the resources borrowed and then proceed. The same applies to lending resources: The workflow can always wait until the environment will lend it the maximal amount of the resources possible, meaning that the behavior of the composition is defined by r^+. The borrowing/lending part of the environment model becomes important when time is taken into consideration, also for transitions t_r^+ and

t_r^-, since borrowing/lending then changes the set of markings reachable in the workflow. Note that also transition $\bar{t}_r^{\downarrow}$, decreasing the amount of resources the environment might remove permanently, only has influence on the set of markings reachable in the workflow when we take time into account.

To create cases of the workflow, we add a generic *case environment* to our generic environment, allowing for an arbitrary number of cases from the interval $[k_1, k_2]$, for some $k_1, k_2 \in \mathbb{N}$, $k_1 \leq k_2$. Figure 2(b) shows the construction. The place c (creation) contains initially k_1 tokens (i.e., the lower bound of cases to be created), the place d (dismissable) contains $k_2 - k_1$ tokens (i.e., cases that can but do not have to be created), and place e (end) is empty. For every case that is not created, a token is produced in the place e by firing t^e. Thus, if all created cases terminate (modeled by a token in f for each case), the place e contains on the termination k_2 tokens.

Definition 9 (generic environment). Let $N = \langle P_p \uplus P_r, T, W_p \uplus W_r, l, \Sigma \rangle$ be an RCWF-net. The *generic environment* of N is a labeled Petri net E such that E and N are a-composable with $((P_p \uplus P_r) \cap P_E) = P_r \uplus \{i, f\}$ and $E = \langle P_E, T_E, W, m_E, l_E, \{\tau\} \rangle$ is defined as

- $P_E = P_r \uplus P_e \uplus \{i, f, c, d, e\}$ with $P_e = \{r^-, r^+, r^{\uparrow}, r^{\downarrow}, \bar{r}_1^{\uparrow}, \bar{r}_2^{\uparrow}, \bar{r}_1^{\downarrow}, \bar{r}_2^{\downarrow} \mid r \in P_r\}$,
- $T_E = \{t_r^-, t_r^+, t^{\uparrow}, t^{\downarrow}, \bar{t}_r^{\uparrow}, \bar{t}_r^{\downarrow} \mid r \in P_r\} \uplus \{t^c, t^d, t^e, t^f\}$,
- $W((r^-, t_r^-)) = W((r, t_r^-)) = W((t_r^-, r^+)) = W((t_r^+, r^-)) = W((t_r^+, r)) = W((r^+, t_r^+)) = W((r^{\uparrow}, t_r^{\uparrow})) = W((t_r^{\uparrow}, \bar{r}_2^{\uparrow})) = W((t_r^{\uparrow}, r)) = W((r^{\uparrow}, \bar{t}_r^{\uparrow})) = W((\bar{t}_r^{\uparrow}, \bar{r}_1^{\uparrow},)) = W((r^{\downarrow}, t_r^{\downarrow})) = W((t_r^{\downarrow}, \bar{r}_2^{\downarrow})) = W((r, t_r^{\downarrow})) = W((r^{\downarrow}, \bar{t}_r^{\downarrow})) = W((\bar{t}_r^{\downarrow}, \bar{r}_1^{\downarrow},)) = 1$ for $r \in P_r$ and
  $W((c, t^c)) = W((t^c, i)) = W((d, t^d)) = W((d, t^e)) = W((t^d, i)) = W((t^e, e)) = W((f, t^f)) = W((t^f, e)) = 1$,
- $m_E(p) = \begin{cases} m_N(p), & p \in P_r \\ m_r^-, & p = r^- \text{ and } m_r^- \text{ is the maximal number of resources } r \text{ the environment can borrow} \\ m_r^+, & p = r^+ \text{ and } m_r^+ \text{ is the maximal number of resources } r \text{ the environment can lend} \\ m_r^{\downarrow}, & p = r^{\downarrow} \text{ and } m_r^{\downarrow} \text{ is the maximal number resources } r \text{ the environment can remove} \\ m_r^{\uparrow}, & p = r^{\uparrow} \text{ and } m_r^{\uparrow} \text{ is the maximal number of resources } r \text{ the environment can add} \\ k_1 \in \mathbb{N}, & p = c \\ k_2 - k_1 \in \mathbb{N}, & p = d \end{cases}$

An *environment* $\langle E, m_E \rangle$ of N consists of E and a concrete initial marking m_E.

3.2 Interval Soundness for RCWF-nets with an Environment

We adapt the definition of soundness for WF-nets to RCWF-nets with an environment. Soundness of an RCWF-net N with an environment $\langle E, m_E \rangle$ guarantees that the underlying production net of N is k-sound, for every k in the interval; that is, also in the

presence of resources, a case has always the possibility to terminate. In addition, we put two conditions on the resources: First, all resources that are initially available in N and E are again available when all cases are terminated. Second, at any reachable marking, the number of available resources does not increase the number of initially available resources. These two criteria are a consequence of our restriction to durable resources, because they ensure that no resources are created or removed.

To guarantee the previous conditions, we define four necessary conditions that are captured in the notion of a *well-defined composition* of N and an arbitrary environment $\langle E, m_E \rangle$. The first condition ensures that the production net of N is k-sound, for every k in the interval. The second condition ensures that no resource tokens can be created by the WF-net; that is, for every firing sequence, the number of resource tokens put by N on the resource places does not exceed the number of tokens taken by N from the resource places (meaning that then every reachable marking has a resource vector $R' \leq R$, unless the environment can add tokens to the resource places). The third condition states that there exists a place invariant for the places c, d and e, guaranteeing that the number of cases remains constant. Likewise, the fourth condition requires that, for every resource place, there exists a place invariant, guaranteeing that the number of resources remains constant.

Definition 10 (well-defined). Let N be an RCWF-net such that the production net of N does not have redundant places. Let $\langle E, m_E \rangle$ be an environment of N. The composition $N \oplus E$ is *well-defined* if the following four properties hold:

1. The production net of N is k-sound, for all $m_E(c) \leq k \leq m_E(c) + m_E(d)$.
2. $\forall x \in \mathbb{Z}^T : (\mathbf{C} \cdot x)|_{P_p \uplus \{e\}} \geq 0$ implies $(\mathbf{C} \cdot x)|_{P_r} \leq 0$.
3. There exists a place invariant I_p such that $I_p(c) = I_p(d) = I_p(e) = 1$ and, for all $p' \in P_E \setminus \{c, d, e\}$, $I_p(p') = 0$.
4. For each $r \in P_r$, there exists a place invariant I_r satisfying $I_r(c) = I_r(d) = I_r(e) = 0, I_r(r) = 1$, and $\forall r' \in P_r \setminus \{r\} : I_r(r') = 0$.

The absence of redundant places is necessary for applying invariant techniques. The next lemma shows that a well-defined composition is bounded.

Lemma 11. *Let N be an RCWF-net and $\langle E, m_E \rangle$ be an environment of N. If $N \oplus E$ is well-defined, then it is bounded.*

Proof. Boundedness of the resource environment follows from Definitions 10(2),(4) and of the case environment from Definition 10(3). The latter argument and Definition 10(1), which implies boundedness of the production net, implies boundedness of N. □

For a well-defined composition $N \oplus E$, we can define interval soundness, which is a more general variant of the soundness notion as defined in [13,5].

Definition 12 (interval soundness). Let N be an RCWF-net and $\langle E, m_E \rangle$ be an environment of N such that $N \oplus E$ is well-defined. Then, N is *sound with* $\langle E, m_E \rangle$ if for all $m \in \mathcal{R}(m_{N \oplus E}) : m \xrightarrow{*} m'$ such that $m'(e) = m_E(c) + m_E(d)$. If m_E is not relevant, we say N is *interval sound*.

Definition 12 captures at least the following relevant instances of interval soundness:

- (k, R)-soundness [13,5] (i.e., we consider a fixed number k of cases and a fixed resource vector R) if $m_E(p) = 0$, for all $p \in P_e \uplus \{d\}$ and $m_E(r) = R(r)$, for all $r \in P_r$;
- up-to $(k, [R, R^+])$-soundness (i.e., (k, R')-soundness for all $R \leq R' \leq R^+$ but the initial resource vector R can be increased up to R^+ at runtime) if $m_E(d) = 0$ and $m_E(r^\downarrow) = 0$, for all $r \in P_r$ and $m_E(r^\uparrow) = R^+(r) - R(r)$, $m_E(r) = R(r)$ for all $r \in P_r$;
- down-to $(k, [R^-, R])$-soundness (i.e., (k, R')-soundness for all $R^- \leq R' \leq R$ but the initial resource vector R can be reduced down to R^- at runtime) if $m_E(d) = 0$ and $m_E(r^\uparrow) = 0$, for all $r \in P_r$ and $m_E(r^\downarrow) = R(r) - R^-(r)$, $m_E(r) = R(r)$ for all $r \in P_r$;
- up-to, down-to $(k, [R^-, R^+])$-soundness (i.e., (k, R)-soundness for all $R^- \leq R \leq R^+$ but the initial resource vector R can be reduced down to R^- or increased up to R^+ at runtime) if $m_E(d) = 0$ and $m_E(r^\uparrow) = R^+(r) - R(r)$, $m_E(r^\downarrow) = R(r) - R^-(r)$, $m_E(r) = R(r)$.

We now relate the previous variants of interval soundness, thereby generalizing them from a fixed number k of cases to an interval $[k_1, k_2]$ of cases.

Lemma 13. *For any RCWF-net N and $k_1, k_2 \in \mathbb{N}$ with $k_1 \leq k_2$, we have*

1. *N is $([k_1, k_2], R')$-sound for all $R \leq R' \leq R^+$ iff N is up-to $([k_1, k_2], [R, R^+])$-sound.*
2. *N is down-to $([k_1, k_2], [R^-, R])$-sound implies N is $([k_1, k_2], R')$-sound for all $R^- \leq R' \leq R$.*
3. *Let $R = R^+$. N is down-to $([k_1, k_2], [R^-, R])$-sound iff N is up-to, down-to $([k_1, k_2], [R^-, R^+])$-sound.*

Proof (Sketch). It suffices to prove the three statements for k, $k_1 \leq k \leq k_2$.

(1) $\Rightarrow$: Let m_0 be the initial marking for (k, R')-soundness with $R' = R^+$ and m'_0 be the initial marking for up-to $(k, [R, R^+])$-soundness. Clearly, we have $m_0|_{P_r} \geq m'_0|_{P_r}$. Let $m'_0 \xrightarrow{\sigma} m'$. Then by the monotonicity of the firing rule and construction of E, we have $m_0 \xrightarrow{\sigma|_{T_N}} m$ and $m|_{P_r} \geq m|_{P_r}$. Thus, if we consider the projection of markings to N, then every marking that is reachable for up-to $(k, [R, R^+])$-soundness is also reachable for (k, R')-soundness.

$\Leftarrow$: Any resource vector R' within the interval can be reached by firing transitions $t_r^\uparrow$, for all $r \in P_r$. Then, every run in the composition for (k, R')-soundness can be replayed in the net for up-to $(k, [R, R^+])$-soundness.

(2) Similar argumentation as in the reverse implication of (1), but this time transitions $t_r^\downarrow$ and $\bar{t}_r^\downarrow$ have to be fired.

(3) $\Rightarrow$: Similar argumentation as for the implication of (1).

$\Leftarrow$: Use argumentation in (2) to decrease the resource vector to R. □

The next lemma gives a necessary condition for interval soundness, thereby justifying the restriction to well-defined compositions of N and E.

Lemma 14 (necessary condition). *Let N be an RCWF-net and $\langle E, m_E\rangle$ be an environment of N. If N is sound with $\langle E, m_E\rangle$, then $N \oplus E$ is well-defined.*

Proof. (1) k-soundness of the production net of N follows from [16, Cor. 4.1].

(2) Follows from [16, Thm. 4.4].

(3) Existence of an invariant I follows from [16, Thm. 4.8] and by the construction of E, we have $I + (c + d + e)$ is also an invariant.

(4) By [16, Thm. 4.8], a resource invariant exists for all $r \in P_r$. Moreover, we have the following invariant: $r + r^- + 2r^+ + 2r^\uparrow + 2\bar{r}_1^\uparrow + \bar{r}_2^\uparrow + r^\downarrow + \bar{r}_1^\downarrow + 2\bar{r}_2^\downarrow$, for $r \in P_r$. So the sum of these two invariants is the resource invariant we are looking for. □

Example 2. Consider the RCWF-net N_1 in Fig. 1(a) and its environment E (see Fig. 4(a) for the entire composition). The production net of N_1 is k-sound, for $k > 0$, and $i + p + q + f + c + d + e$ is a place invariant in the production net. Furthermore, no resource token is created in N_1, and $r + p + 2q + r^- + 2r^+ + 2r^\uparrow + 2\bar{r}_1^\uparrow + \bar{r}_2^\uparrow + r^\downarrow + \bar{r}_1^\downarrow + 2\bar{r}_2^\downarrow$ an invariant for resource place r. Thus, $N_1 \oplus E$ is well-defined. For the same reason, $N_2 \oplus E$ is well-defined. However, N_1 is not (k, R)-sound for all $k = R(r)$. For example, for $k = r = 2$, firing transition t twice yields a deadlock $[2 \cdot p]$. In contrast, N_2 is (k, R)-sound for $R(r) > 1$ and any k. This example exemplifies that well-definedness is only a necessary condition for interval soundness.

4 Deciding Interval Soundness

Definition 12 (interval soundness) gives an immediate decision procedure: An RCWF-net N is sound with $\langle E, m_E\rangle$ if the set $\{m \mid m(e) = m_E(c) + m_E(d)\}$ of markings is a home-space in $N \oplus E$ or respectively $m = [(m_E(c) + m_E(d)) \cdot e]$ is a home-marking in the projection of reachable markings of $N \oplus E$ to the places of N.

Theorem 15 (decision I). *Let N be an RCWF-net and $\langle E, m_E\rangle$ be an environment of N such that $N \oplus E$ is well-defined. Then, N is sound with $\langle E, m_E\rangle$ if the set $\{m \mid m(e) = m_E(c) + m_E(d)\}$ of markings is a home-space in $N \oplus E$.*

Note that we have one decision algorithm, for every instance of interval soundness. As checking a home-space property is decidable [12], we can conclude:

Theorem 16 (decidability). *Let N be an RCWF-net and $\langle E, m_E\rangle$ be an environment of N. Checking whether N is sound with $\langle E, m_E\rangle$ is decidable.*

In the literature, soundness is often reduced to showing that the short-circuited (RC)WF-net is live and bounded. The reduction works if the transition t_s consumes all k cases from the place f and produces k tokens on the place i. Figure 3 illustrates this. The drawback of this construction is that it requires to check a different net for every k. As we consider N with an environment, we propose the following generic reduction to liveness and boundedness:

Theorem 17 (decision II). *Let N be an RCWF-net and $\langle E, m_E\rangle$ be an environment of N such that $N \oplus E$ is well-defined. Let E_s be obtained from E by adding a transition t_s with $W((e, t_s)) = X(c) + X(d)$, $W((t_s, d)) = X(d)$, and $W((t_s, c)) = X(c)$. Then, N is sound with $\langle E, m_E\rangle$ iff all transitions $T_N \uplus \{t_s\}$ of $N \oplus E_s$ are live.*

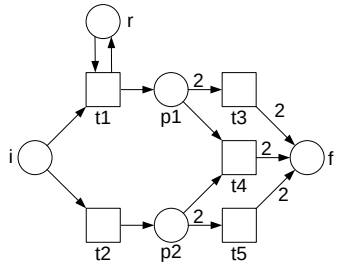


Fig. 3. The RCWF-net N is not $(3,1)$-sound: The firing sequence $t_1t_2t_4t_1$ yields the marking $[2 \cdot f, p_1]$ which is a deadlock. Nevertheless, the short-circuited net of N is bounded and live. However, if the transition t_s consumes all k cases from the place f and produces k token on the place i, then the short-circuited net of N is not live.

Proof. $\Rightarrow$: As N does not have redundant places, it does not have dead transitions in its production net and t_s is not dead either, so we conclude that all transitions in $T_N \uplus \{t_s\}$ are live.

$\Leftarrow$: From liveness of the transition t_s, we conclude that it is always possible to reach a marking m with $m(e) \geq m_E(c) + m_E(d)$. The number of tokens in e at m cannot be greater than $m_E(c) + m_E(d)$ by the invariant in Definition 10(3). Moreover, all places of the production net N_p of N are unmarked in m by the k-soundness of N_p. Firing t_s at m yields m' where the places c and d contain the same number of tokens as in the initial marking m_Z. Because of the invariants covering all places of E (Definition 10(3) and (4)), we conclude that m' is reachable from m_Z. Hence, markings m are a home-space in $N \oplus E_s$, and N is sound with $\langle E, m_E \rangle$. □

Example 3. Using Theorem 17, we can show that the RCWF-net N_1 is not (k, R)-sound whereas the RCWF-net N_2 is.

5 Repairing Interval Unsound RCWF-Nets

In the previous section, we presented an algorithm to decide interval soundness of an RCWF-net N. However, designing an interval-sound workflow or adjusting a workflow if some functionality or the environment has been changed is a nontrivial and error-prone task even for experienced process designers. In order to support process designers, we introduce an approach to *repair* an interval-unsound RCWF-net N if possible so that interval soundness is achieved by design. Clearly, the repaired workflow should be seen as a suggestion to the process designer rather than the ultimate solution.

Requiring the composition $N \oplus E$ to be well-defined reduces the cause of unsoundness to a deadlock or a livelock due to the lack of resources during the production process (see Lemma 11). To repair an RCWF-net N, we therefore propose to automatically construct a *controller* C that controls those transitions of N that produce tokens on or consume tokens from a resource place. This way, we control the order in which certain tasks may occur and prevent the workflow from getting stuck.

Technically, a controller is a labeled Petri net C and will be composed with $N \oplus E$ by merging transitions of N only. These merged transitions of N are then controlled by C in the composition. Another technicality that we leave out in the following is ensuring that the nodes of E and C are pairwise disjoint.

Definition 18 (controller, repairable). Let N be an RCWF-net and $\langle E, m_E \rangle$ be an environment of N. A labeled Petri net C is a *controller* of $N \oplus E$ if C and N are s-composable and replacing Z in Definition 12 with $C \| N \oplus E$ yields soundness of N with $\langle E, m_E \rangle$. If there exists a controller of $N \oplus E$, then $N \oplus E$ is *repairable*.

The following algorithm synthesizes a controller of $N \oplus E$. It takes the state space of N and the environment E as its input, and it outputs an LTS which can, in a next step, be easily transformed into a labeled Petri net.

Definition 19 (controller construction). Let N be an RCWF-net and and $\langle E, m_E \rangle$ be an environment of N such that $Z = N \oplus E$ is well-defined and has a finite state space. Let $\Sigma \subseteq \Sigma_N$ be the set of synchronized actions, and let $TS_Z = \langle Q_Z, \delta_Z, \hat{q}_Z, \Sigma \rangle$ be the LTS of Z after relabeling all actions $x \in \Sigma_N \setminus \Sigma$ to τ. Define a sequence of LTSs TS^i, $i = 0, 1, \ldots$ inductively as follows:

Base : $TS^0 = \langle \mathcal{Q}^0, \delta^0, Q_0, \Sigma \rangle$ with
- $\mathcal{Q}^0 = 2^{Q_Z}$,
- $\delta^0 = \{(Q, x, Q') \in \mathcal{Q}^0 \times \Sigma \times \mathcal{Q}^0 \mid Q' = \{q'_Z \mid \exists q_Z \in Q : q_Z \xrightarrow{\tau^* x \tau^*} q'_Z\}\}$,
- $Q_0 = \{q_Z \mid \hat{q}_Z \xrightarrow{\tau^*} q_Z\}$.

Step : $TS^{i+1} = \langle \mathcal{Q}^{i+1}, \delta^{i+1}, Q_0, \Sigma \rangle$ with
- $\mathcal{Q}^{i+1} = \mathcal{Q}^i \setminus \{Q \in \mathcal{Q}^i \mid \exists (q_Z, Q) \in TS_Z \| TS^i : (q_Z, Q) \not\xrightarrow{*} (q'_Z, Q') \wedge q'_Z|_{\{e\}} = k_1 + k_2\}$[1],
- $\delta^{i+1} = \delta^i \cap (\mathcal{Q}^{i+1} \times \Sigma \times \mathcal{Q}^{i+1})$.

Let j be the smallest number with $TS^j = TS^{j+1}$. If $Q_0 \in \mathcal{Q}^j$, then the corresponding labeled Petri net C of TS^j is a controller of $N \oplus E$.

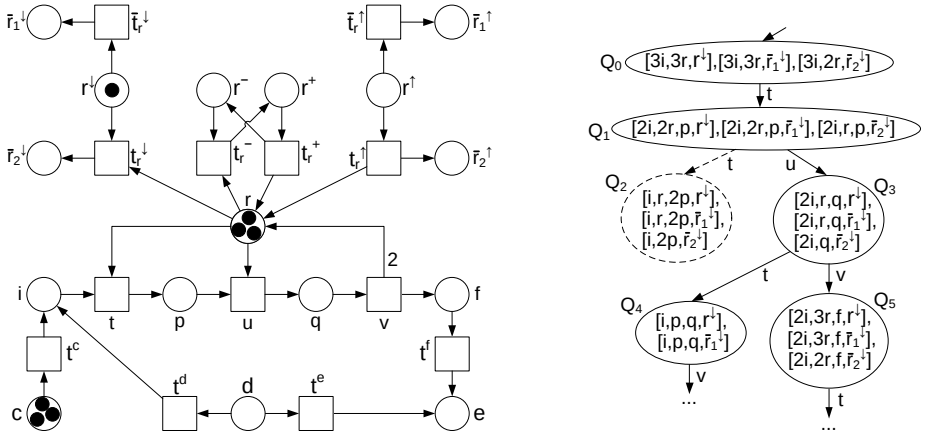
The construction of C allows some level of flexibility: We do not assume all transitions of N to be controllable as we restrict the set of labels of TS^j in the construction to a subset Σ of the alphabet Σ_N. That way, we take into account that not all tasks in a workflow can be controlled. We assume the nodes of C and E to be pairwise disjoint; that is, the controller cannot control actions of the environment. (If one would find it possible to control actions of the environment, we could adapt our construction by labeling certain transitions of E and adding those actions to the alphabet of C.)

Note that C is not necessarily a WF-net, because it may have more than one sink place; thus, the composition $N \| C$ is not an RCWF-net but a labeled Petri net.

Our main result of this section is that a composition $N \oplus E$ is repairable if and only if the algorithm in Definition 19 outputs an LTS with at least one state.

Theorem 20 (justification). *Let N be an RCWF-net, $\langle E, m_E \rangle$ be an environment of N, and C be the labeled Petri net constructed according to Definition 19. Then, C exists iff $N \oplus E$ is repairable.*

[1] For the sake of readability, we see the state q'_Z as its corresponding marking in Z.



(a) $N_1 \oplus E$ (b) Sketch of LTS TS

Fig. 4. Illustration of the controller construction (see Definition 19) for $N_1 \oplus E$, assuming an initial marking $[3 \cdot i, 3 \cdot r, r^{\downarrow}]$ (i.e., the transition t^c has been fired three times) for purposes of simplification. The state Q_2 contains a deadlock and will be removed in one of the iterations.

Proof. $\Rightarrow$: Suppose C is a controller of $N \oplus E$. As TS^0 is the largest structure, there must be a largest i such that there exists a simulation relation of (the state space) of C by TS^i. If there is no simulation relation of C by TS^{i+1}, then $(N \oplus E) \| C$ must violate soundness because this is the reason for removing further states from TS^i.

$\Leftarrow$: The construction of TS^j and thus of C terminates because finiteness of TS_Z ensures that TS^0 is also finite and only removes states and transitions from TS^0 when constructing TS^j. If the resulting TS^j is nonempty, it must be a controller of $N \oplus E$ because all reasons not to be a controller have been erased: We removed all states from which N cannot reach a final state and we iteratively check this. □

So, Definition 19 yields the *most permissive* controller of $N \oplus E$. Other, more specific controllers that have less behavior can also be constructed, for example, by assigning costs to each transition in N and constructing a controller that has the least cost.

Example 4. Figure 4 illustrates the controller construction for $N_1 \oplus E$. For the sake of readability, we keep the size of each controller state (i.e., the number of markings $N_1 \oplus E$ can be in) small by choosing $[3 \cdot i, 3 \cdot r, r^{\downarrow}]$ to be the initial marking of $N_1 \oplus E$— the state space of the controller remains the same. Initially, $N_1 \oplus E$ can be in any of the three markings of the initial state Q_0 of TS. Firing the transition t, yields the three markings depicted in the state Q_1 of TS. The state Q_2 (depicted by a dashed frame) is removed in one of the iterations of the construction, because the marking $[i, 2 \cdot p, \bar{r}_2^{\downarrow}]$ is a deadlock and no final marking. The complete LTS TS, the controller, has 12 states and realizes $tu(tv + vt)u(tv + vt)uv$. Composing the resulting labeled Petri net C of TS with $N_1 \oplus E$ yields a sound net $C \| N_1 \oplus C$. Note that the composition operator $\|$ requires one copy for each occurrence of the transition t, u, and v in C.

6 Related Work

The verification of soundness for RCWF-nets has been investigated by many researchers. On the one hand, interval soundness is a more restrictive instance compared to soundness in [21,13,5], as we assume the number of cases and resources to be fixed within an interval. On the other hand, it is more general because we assume resources to be shared among workflows rather than internal to a workflow. We incorporated this in our model by enriching the model of RCWF-nets as proposed in [6,13] with a generic environment modeling the resource perspective of a WF-net. Moreover, we are neither restricted to one resource type as [13] nor to certain subclasses of WF-nets as [5]. Resource problems with an unbounded number of resource items have been studied in [9].

RCWF-nets can be seen as parameterized (or multi-threaded) systems with two parameters: the number of cases to be executed and the number of available resources. Verification of parameterized systems is a popular topic, but must approaches investigate safety properties with unbounded parameters (e.g., [18]) whereas we assume fixed bounds but consider with soundness a liveness property. A resource interface in [11] defines a safety property over the resources for open system; we defined the generic environment E and hence deal with a closed system. There also exist extensions of the temporal logics CTL and ATL to reason about resources [10,4]. Although the problem instances considered in this paper can be expressed in terms of those logics, verification would require to check the system for all parameters.

Our approach to repair unsound workflows is based on classical controller synthesis [20] and has been defined for Petri nets and soundness in [22]. For an overview of Petri net-based controller synthesis approaches, we refer to [19]. Most of these works focus on the net structure and properties different from soundness; moreover, the main application are manufacturing systems where one tries to construct a scheduler. In contrast, we construct a controller such that the net is robust and thus sound.

7 Conclusion

We investigated the correctness of workflows with shared resources, called interval soundness. To do so, we proposed to enrich the workflow model with a generic environment capturing the resource perspective. An instance of this generic environment models a specific environment that specifies an interval of workflow instances to be created and available resources for each resource type. The generic environment generalizes the existing workflow model extended with resources and captures environments of practical relevance. To decide interval soundness for every instance of the generic environment, we presented two decision procedures, both using invariant properties of the environment. Furthermore, we showed a way to support the design of correct workflows by automatically synthesizing a controller such that the composition of the workflow and the controller is interval sound.

In ongoing work, we are interested in determining a smallest resource vector (based on given requirements) that guarantees soundness. Likewise, we aim at determining the largest resource vector that guarantees soundness or proving that increasing some resource vector does not influence the soundness result. Constructing more specific controllers by assigning costs to transitions is another direction of future work.

References

1. Aalst, W.M.P.v.d.: Verification of workflow nets. In: ICATPN 1997. LNCS, vol. 1248, pp. 407–426. Springer (1997)
2. Aalst, W.M.P.v.d.: The Application of Petri Nets to Workflow Management. The Journal of Circuits, Systems and Computers 8(1), 21–66 (1998)
3. Aalst, W.M.P.v.d., Hee, K.M.v.: Workflow Management: Models, Methods, and Systems. MIT press, Cambridge, MA (2002)
4. Alechina, N., Logan, B., Nga, N.H., Rakib, A.: Resource-bounded alternating-time temporal logic. In: AAMAS 2010. pp. 481–488. IFAAMAS (2010)
5. Barkaoui, K., Benayed, R., Sbai, Z.: Workflow Soundness Verification Based on Structure Theory of Petri Nets. International Journal of Computing & Information Sciences 5(1), 51–62 (2007)
6. Barkaoui, K., Petrucci, L.: Structural Analysis of Workflow Nets with Shared Ressources. In: WFM 1998. pp. 82–95 (1998)
7. Best, E., Darondeau, P.: Separability in persistent petri nets. Fundam. Inform. 113(3-4), 179–203 (2011)
8. Best, E., Esparza, J., Wimmel, H., Wolf, K.: Separability in conflict-free petri nets. In: PSI 2006. LNCS, vol. 4378, pp. 1–18. Springer (2007)
9. Brázdil, T., Jancar, P., Kucera, A.: Reachability games on extended vector addition systems with states. In: ICALP 2010. LNCS, vol. 6199, pp. 478–489. Springer (2010)
10. Bulling, N., Farwer, B.: Expressing properties of resource-bounded systems: The logics rtl* and rtl. In: CLIMA 2009. LNCS, vol. 6214, pp. 22–45. Springer (2010)
11. Chakrabarti, A., de Alfaro, L., Henzinger, T.A., Stoelinga, M.: Resource interfaces. In: EMSOFT 2003. LNCS, vol. 2855, pp. 117–133. Springer (2003)
12. Escrig, D.F., Johnen, C.: Decidability of home space property. LRI report 503, Université Paris-Sud (1989)
13. Hee, K.M.v., Serebrenik, A., Sidorova, N., Voorhoeve, M.: Soundness of resource-constrained workflow nets. In: ICATPN 2005. LNCS, vol. 3536, pp. 250–267. Springer (2005)
14. Hee, K.M.v., Sidorova, N., Voorhoeve, M.: Soundness and separability of workflow nets in the stepwise refinement approach. In: ICATPN 2003. LNCS, vol. 2679, pp. 337–356. Springer (2003)
15. Hee, K.M.v., Sidorova, N., Voorhoeve, M.: Generalised soundness of workflow nets is decidable. In: ICATPN 2004. LNCS, vol. 3099, pp. 197–215. Springer (2004)
16. Hee, K.M.v., Sidorova, N., Voorhoeve, M.: Resource-constrained workflow nets. Fundam. Inform. 71(2-3), 243–257 (2006)
17. Juhás, G., Kazlov, I., Juhásová, A.: Instance deadlock: A mystery behind frozen programs. In: PETRI NETS 2010. LNCS, vol. 6128, pp. 1–17. Springer (2010)
18. Kaiser, A., Kroening, D., Wahl, T.: Dynamic cutoff detection in parameterized concurrent programs. In: CAV 2010. LNCS, vol. 6174, pp. 645–659. Springer (2010)
19. Li, Z.W., Wu, N.Q., Zhou, M.C.: Deadlock control of automated manufacturing systems based on petri nets—a literature review. IEEE Transactions on Systems, Man, and Cybernetics, Part C 42(4), 437–462 (2012)
20. Ramadge, P.J., Wonham, W.M.: Supervisory control of a class of discrete event processes. SIAM J. Control and Optimization 25(1), 206–230 (1987)
21. Sidorova, N., Stahl, C.: Soundness for resource-constrained workflow nets is decidable. IEEE Transactions on Systems, Man and Cybernetics, Part A 43(3), 724–729 (2013)
22. Wolf, K.: Does my service have partners? In: ToPNoC II. pp. 152–171. LNCS 5460, Springer (2009)