# Nested Nets for Adaptive Systems

Kees van Hee[1], Irina A. Lomazova[2,*], Olivia Oanea[1,**],
Alexander Serebrenik[1], Natalia Sidorova[1], and Marc Voorhoeve[1]

[1] Department of Mathematics and Computer Science
Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
{k.m.v.hee, o.i.oanea, a.serebrenik, n.sidorova, m.voorhoeve}@tue.nl
[2] Program Systems Institute of Russian Academy of Science,
Pereslavl-Zalessky, 152020, Russia
irina@lomazova.polnet.botik.ru

**Abstract.** We consider nested nets, i.e. Petri nets in which tokens can be Petri nets themselves. We study the value semantics of nested nets rather than the reference semantics, and apply nested nets to model adaptive workflow, i.e. flexible workflow that can be modified during the execution. A typical domain with a great need for this kind of workflow is health care, from which domain we choose the running example. To achieve the desired flexibility we allow transitions that create new nets out of the existing ones. Therefore, nets with completely new structure can be created at the run time. We show that by careful selection of basic operations on the nets we can obtain a powerful modeling formalism that enforces correctness of models. Moreover, the formalism can be implemented based on existing workflow engines.
**Keywords:** Petri nets; modeling; workflow, adaptivity.

## 1 Introduction

In this paper we consider nested nets, i.e. Petri nets in which tokens can be Petri nets themselves. This means that processes are considered as objects that can be manipulated by other processes. We apply nested nets to construct more flexible workflow management systems. In classical workflow management systems the process structure is determined at design time. During execution no structural changes are possible. This implies that designers need to take into account all possible executions, exceptional situations and combinations of them. In case of so-called *ad hoc workflow* [4, 5, 14, 34] the algorithm for processing cases is not known at design time, so it is impossible to use a classical workflow management system and so-called *case handling systems* are used instead. These systems have no formal process semantics which makes testing and verification difficult.

In this paper we propose a solution with more flexibility for adaptation than classical workflow systems and more structure than ad hoc workflow systems. We call them

*adaptive workflow systems*. By adaptivity we understand an ability to modify processes in a structured way, for instance by replacing a subprocess or extending it. We assume a given library of protocols to be used as basic building blocks for constructing more complex protocols. A natural way to model these complex protocols in an adaptive way is by means of nested Petri nets. Nested Petri nets are Petri nets in which tokens can be Petri nets themselves, called *token nets*. The ability to modify a token net rather than a part of the net itself accounts for the following advantages:

- an ability to update the library of protocols at runtime;
- an ability to modify the ongoing processes at runtime;
- an ability to model decisions taken by different parties (separation of concerns).

Traditionally workflow is modeled by workflow nets [1–3, 6]. We extend this notion by introducing a mechanism for exception handling. Exception handling is recognized as a critical challenge for workflow management systems [22]. In our extended workflow nets we introduce final transitions whose firings reflect exceptional situations. When used as token nets, these firings are normally synchronized with the firings of the higher-level net, which terminates the execution of the lower-level net.

A typical domain with a great need for this kind of adaptive workflow systems is healthcare. Today the medical protocols have the form of guidelines that involve doctors, nurses and paramedical personnel. They are often combinations of lower-level protocols where the way of combining may be determined at runtime depending, e.g., on the patient's state. Therefore, our adaptive workflow nets are well-suited for modeling this kind of systems.

To make the results of this paper applicable in practice, we were looking for an approach that could be implemented by standard workflow engines with only slight modifications. In fact, to execute a transition for which consumed tokens are nets, we have to invoke the workflow engine again for the token nets, however this can be fresh instantiations of the workflow engine.

*Related research*  Net in nets are extensively studied in the Petri net literature: [10, 23–25, 28, 29, 35, 36]. In all these works, the goal was to extend the expressive power or the expressive comfort of Petri nets. So, in object Petri nets, the authors aim at making Petri nets suitable for modeling according to the object-oriented style [36]. Unlike our work, the stress is on mobility and not on constructing new nets from the existing ones. In [10] the authors consider an object-oriented approach in defining synchronization between objects of different classes. For instance, an object might call methods of other objects using the operators: parallel composition, sequential composition and alternative composition. This corresponds to our approach of creating a new net token using these operators.

Nested nets [26] define nets as tokens on top of colored Petri net semantics [21]. Nested nets have the value semantics, i.e., independent copies (multisets) of token nets are considered. The framework we propose is partially inspired by this work, extending it by introducing operations on token nets and including data in the synchronization mechanism.

The idea of controlled modification of token nets is also considered for high-level net and rule (HLNR) systems in [20, 13]. Unlike our approach that easily supports arbi-

trary (but fixed) nesting level and synchronization between different levels of a nested net, the previous results considered nesting of depth one only. Moreover, [20] carries structural modification of P/T token nets by means of rule tokens, whereas our approach uses predefined and well-known operations, such as sequential and parallel composition.

The idea of combining workflow and "nets in nets" is going back to [8], where the authors consider object Petri nets with workflow nets as token nets. Therefore, the differences between our work and [36] are also applicable to this work.

Effective response in presence of exceptions is recognized as a critical challenge for workflow management systems [22, 30]. We believe that exceptions form an essential part of the process, and therefore, they should be included in the model. Alternatively, as suggested in [7, 9, 22] a workflow system can be extended by an exception handling mechanism. However, in such a case no formal reasoning involving exceptions is possible.

Petri nets have been used for modeling of healthcare workflow, also known as careflow [19, 33, 32, 30]. The guideline execution system GUIDE [33] translates formalized guidelines to a hierarchical timed colored Petri net. The resulting net can be run to simulate implementation of the guideline in clinical setting. However, this formalism misses adaptivity and separation of concerns.

The rest of the paper is organized as follows. Section 2 introduces a motivating example from the healthcare world. In Section 3 we present the notion of an extended workflow net and in Section 4 we discuss operations on extended workflow nets. Section 5 describes nested nets and adaptive workflow nets. Finally, in Section 6 we discuss the results presented and possible directions for future work.

## 2   Example: a Medical Protocol

A typical domain with a great need for adaptive workflow systems is *health care*, in particular patient treatment. In this domain the classical way to support the medical process is to use an EPR-system (EPR stands for Electronic Patient Record). An EPR-system is a data-oriented system in which for each patient a data structure is created that is updated at each step of the medical process. Normally the data structure is stored in a relational database system. On top of this database, rule engines are used to support the medical decision making to determine a diagnosis or a treatment. In recent years the *protocol-based medicine* became a standard approach. In this approach for many kinds of medical problems there are protocols defined to derive a diagnosis or a treatment. These *protocols* are in fact process descriptions. Medical experts have to follow the steps of these protocols and to document the decisions they make. This is not only important for the quality of care but also for avoiding claims when things go wrong: medical experts can prove that they treated their patient according to the best practice that is reflected in the protocol. For each patient a specific workflow is created and an EPR-system augmented with workflow system functionality is an important improvement of the process of patient treatment.

Today the medical protocols have the form of *guidelines*. A guideline is not limited to doctors but also cover the workspace of nurses and paramedical personnel. There
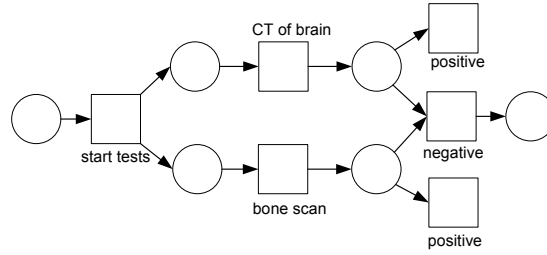
**Fig. 1.** Mandatory test protocol (*MandT⟨positive⟩*)

have been several attempts to formalize guidelines as *flowcharts* and *decision diagrams* and incorporate them into medical decision support systems (MDSS). For example, GLIF [31] and GUIDE [32, 33] support medical processes by enacting guidelines, and integrating them in EPR-systems.

As a motivating example we consider the process of diagnosis and treatment of a small-cell lung cancer (SCLC). The example is inspired by the practice guidelines for SCLC [15] created by the National Comprehensive Cancer Network (SUA) and modified by The University of Texas M.D. Anderson Cancer Center for their patient population.

The guideline is built on the basis of a library of standard protocols. A protocol may describe the decision process for establishing a diagnosis, treatment schemes and tests used in the process of diagnosis or after the treatment. Each protocol has an initial point, a final point and it handles exceptional situations by terminating the process. For instance, the protocol used for mandatory tests in the initial diagnosis stage describes the process of executing two tests (computer tomography of brain and the bone scan) and evaluating their results. An exceptional situation is considered when the result of at least one of the tests turns out positive. We model such a protocol as a workflow net extended with special transitions, which are transitions without output places. These transitions carry an exception label indicating the exceptional situation. Figure 1 shows the extended workflow net modeling the mandatory test protocol.

The interfaces of protocols in the library are given as *Prot⟨ex₁, ..., exₙ⟩*, where *Prot* is the name of the protocol and $ex_1, \ldots, ex_n$ is the list of exception labels. In case the protocol has no exceptions, it is denoted solely by its name.

Figure 2 shows another protocol net (*STests*), modeling the test protocol used in the surveillance stage. The process performs some tests such as CXR test, creatine test and liver function test and registers their results in the variables $x, y, z, u, w$. The evaluation of the tests employs a function that combines the results of the tests. According to constraints specified in predicates, the progress of the cancer and the response to the treatment are evaluated: the cancer can relapse (modeled by the transition labeled *relapse*) or the treatment can have a partial response or good response (modeled by transitions labeled accordingly).

Protocols can make use of other protocols, create new protocols from the existing ones or modify them. For this purpose, we consider nets as colored tokens. The protocols can be built of more primitive ones by using the operations ·, ∥. The operation *init*
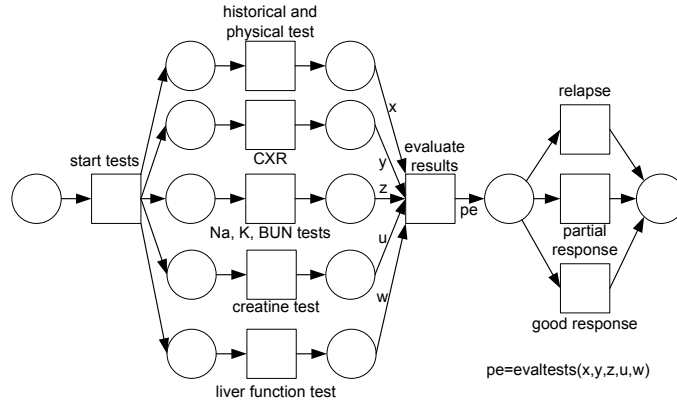
4

**Fig. 2.** Surveillance test protocol (*STests*)

initializes an extended workflow net with its initial marking. Consider the surveillance protocol in Figure 3. The protocol makes use of other therapy and test protocols, namely *Radiotherapy, STests, Cisplatin, Etoposide, ProphT* and *RadCon⟨radiation scarring⟩*. The protocol iterates the surveillance treatment scheme until the results of the surveillance tests show signs of relapse. A regular surveillance treatment is started at runtime by creating a process token (denoted by the constant $str = init(Radiotherapy \cdot STests)$ on the outgoing arc of the transition *start surveillance*). The process of this token consists of the *Radiotherapy* protocol followed by the *STests* protocol. The transitions labeled by *relapse, partial response, good response* in the surveillance test protocol are synchronized with transitions having the same labels in the upper level net (*Surveillance*).

In case of a partial response, a new process token is created, namely $init(((Cisplatin \cdot Etoposide)||Radiotherapy) \cdot STests)$ (radiotherapy performed in parallel with chemotherapy — cisplatin treatment followed by an etoposide treatment — followed by the *STests* protocol).
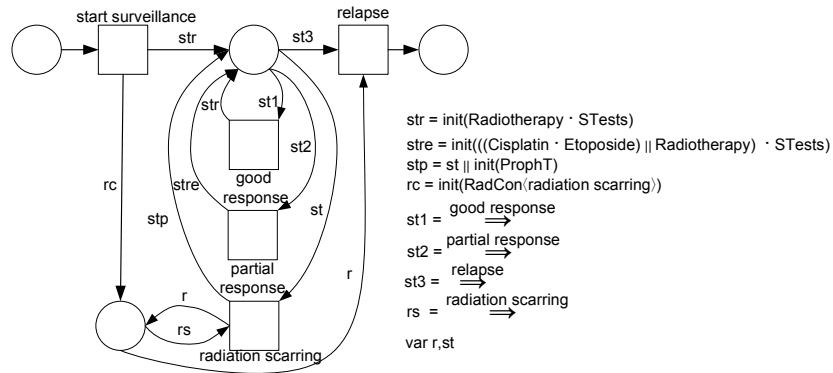


**Fig. 3.** Surveillance protocol (*Surveillance*)

5

$et = init(Mand\langle positive \rangle)$

$lt = init(MandT\langle positive \rangle \parallel FacT\langle positive \rangle)$

$ft = init(Ftests\langle operable, non\text{-}operable \rangle)$

$tr1 = init((Cisplatin \cdot Etoposide \cdot Tests)^6)$

$tr2 = init(Resection \cdot (Cisplatin \cdot Etoposide \cdot Tests)^4)$

$tr3 = init(((Radiotherapy \parallel (Cisplatin \cdot Etoposide)) \cdot Tests)^4)$

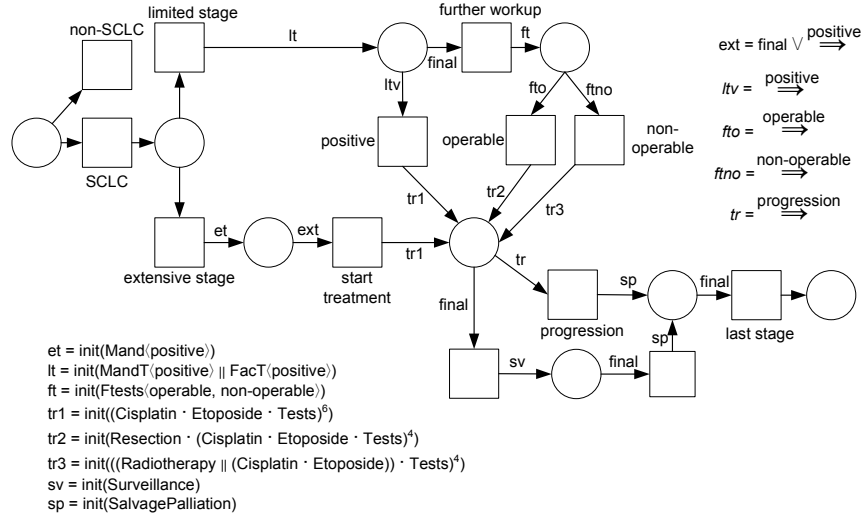$sv = init(Surveillance)$

$sp = init(SalvagePalliation)$

**Fig. 4.** Main SCLC protocol

At the same time with the application of a surveillance treatment, a radiation control protocol (*RadCon⟨radiation scarring⟩*) is used to monitor the radiation effect on the patient. In case the patient shows signs of scars due to the radiation (the exception *radiation scarring* is signaled), a prophylactic treatment (*ProphT*) is conducted in parallel with the actual surveillance protocol (*st||init(ProphT)*).

The protocol from the library describing the main process is depicted in Figure 4. The process starts with the decision on whether the patient has small-cell lung cancer (SCLC) or non-small cell lung cancer (Non-SCLC). The transition *Non-SCLC* models an exception that allows finishing the guideline.

Once the diagnosis has been established, the stage of the illness needs to be assessed. The stage is determined by the extent of spread of the cancer basing on the test results. Here, the protocol makes use of and combines some standard test procedures depending on the preliminary diagnosis of the phase of the cancer. In case the patient shows signs of the extensive stage, a procedure with mandatory tests *MandT⟨positive⟩* (Figure 1) is created. Once the protocol *MandT⟨positive⟩* has terminated or a transition with exception label *positive* indicating that the tests are finished is fired (the arc inscription $ext = final \lor \overset{positive}{\Longrightarrow}$ is true), a specific treatment protocol can be started.

In case the preliminary diagnosis shows signs of the limited stage (*limited stage* transition) more tests are needed than in the extensive case. The protocol executing these tests combines two existing test protocols taken from the library, i.e. *MandT⟨positive⟩* and *FacT⟨positive⟩*, by performing them in parallel. Once one of the tests has a positive outcome (the transition labeled *positive* is fired), it must be synchronized with the respective transition in the main protocol since this is a symptom for the extensive stage cancer. If none of the results turns out positive, the test protocol terminates properly (this is indicated by the arc inscription *final*).

The patient with limited-stage SCLC is further tested to determine whether or not she/he can be operated. For this purpose, the protocol *FTests⟨operable, non-operable⟩* is instantiated, and one of the exceptional outcomes of this test procedure (the patient is non-operable or operable) is synchronized with the respective transition in the SCLC protocol.

For each of the three diagnosis (extensive stage, limited stage operable, and limited stage non-operable), a special treatment scheme is created, which is actually an iteration of chemotherapy, radiotherapy and tests. For example, for the limited stage operable diagnosis, a treatment scheme token considering the resection *Resection*, followed by four iterations of chemotherapy (*Cisplatin* treatment followed by *Etoposide* treatment) and tests (*Tests⟨progression⟩*) is created. Once there is a sign of progression of the cancer after performing the tests at each of the cycles signaled by the occurrence of an exception handled by the transition labeled by *progression*), the treatment is interrupted and the patient goes in the final stage, where only the Palliative/Salvage treatment can be applied. In case the initial treatment has been successfully completed (the transition labeled by *last stage* is fired) a surveillance protocol (*Surveillance*) is created and its completion determines the patient to enter the palliation/salvage stage.

## 3   Extended Workflow Nets

**Preliminaries**  We briefly introduce the notation and the basic concepts that will be used in the remainder of the paper.

$\mathbb{N}$ denotes the set of natural numbers. The set of all subsets of a set $P$ is $2^P$. A relation between sets $P$ and $Q$ is a subset of $P \times Q$. If $R \subseteq P \times Q$ is a relation, then we use the infix notation $p \ R \ q$ for $(p,q) \in R$ and the inverse $R^{-1}$ of $R$ is defined as $\{(q,p) \mid p \ R \ q\}$. Functions are a special kind of relations. Given a function $F$ we write $dom(F)$ to denote the domain of $F$ and $ran(F)$ to denote the range of $F$. If the domain of a function is finite we also write $\{d_1 \mapsto v_1, \ldots, d_n \mapsto v_n\}$. The operator $\circ$ denotes function composition, $(f \circ g)(x)$ is $f(g(x))$. A *bag (multiset) m* over $P$ is a mapping $m : P \to \mathbb{N}$. The set of all bags over $P$ is also denoted by $\mathbb{N}^P$. We use $+$ and $-$ for the sum and the difference of two bags and $=, <, >, \leq, \geq$ for comparisons of bags, which are defined in the standard way. We overload the set notation, writing $\emptyset$ for the empty bag and $\in$ for the element inclusion. We abbreviate bags by writing, for example, $2[p] + [q]$ instead of $\{p \mapsto 2, q \mapsto 1\}$.

Let $F$ be a function, such that $ran(F)$ is a set of sets. Then *the generalized product* of $F$, written as $\Pi(F)$, is the set of all functions $f$ with $dom(f) = dom(F)$ and $f(x) \in F(x)$ holds for all $x \in dom(F)$. Note that $\Pi(F) = \{\emptyset\}$ if $dom(F) = \emptyset$. We further extend this notion and given $S \subseteq dom(F)$ write $\Pi_{x \in S}(F)$ to denote the set of all functions $f$ with $dom(f) = S$ such that for any $x \in S$, $f(x) \in F(x)$.

A *transition system* is a tuple $E = \langle S, \Sigma, T \rangle$ where $S$ is a set of *states*, $\Sigma$ is a set of *actions* and $T \subseteq S \times \Sigma \times S$ is a set of transitions.

**Colored nets.**  In Section 5 we are going to introduce nested nets that are considered as a special kind of colored nets. To this end we define colored Petri nets parameterized with a value universe $\mathcal{U}$. We assume that $\mathcal{U}$ contains the value *black*, which we use as
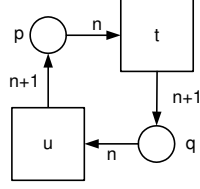
**Fig. 5.** Colored Petri net

an object carrying no information. Our definition does not essentially differ from the existing ones [21] but it has been adapted for our purpose, notably by the addition of transition labels.

In a colored net each place is mapped to a type, which is a subset of $\mathcal{U}$. We also assume a set $L$ of labels for transitions such that $\tau \notin L$. Every label is associated with a unique natural number, called the arity of the label. Then we define the set of transition labels $\Sigma = \{\tau\} \cup \{a(x_1,\ldots,x_n) \mid a \in L, n = arity(a), x_1,\ldots,x_n \in \mathcal{U}\}$. The label $\tau$ is the special "silent" label.

**Definition 1.** *A* colored net *over the universe $\mathcal{U}$ is a 6-tuple $(P,T,F,\upsilon,\rho,\ell)$, where*

- *$P$ and $T$ are disjoint finite sets of* places, *respectively* transitions*;*
- *$F \subseteq (P \times T) \cup (T \times P)$ is a set of* arcs*,*
- *$\upsilon$ is a* place typing *function with $dom(\upsilon) = P$ such that $\upsilon(p) \subseteq \mathcal{U}$ for all $p \in P$;*
- *$\rho$ is a* transition function *with $dom(\rho) = T$ such that $\rho(t) \subseteq \Pi(\varphi_t) \times \Pi(\psi_t)$ for all $t \in T$, where $\varphi_t = \{p \mapsto \upsilon(p) \mid (p,t) \in F\}$ and $\psi_t = \{p \mapsto \upsilon(p) \mid (t,p) \in F\}$;*
- *$\ell$ is a* labeling function *with $dom(\ell) = \{(t,\gamma,\delta) \mid t \in T \wedge (\gamma,\delta) \in \rho(t)\}$ and $ran(\ell) \subseteq \Sigma$.*

The labeling function is used to report information on transition firings. In this way we can distinguish between internal firings that are labeled with $\tau$ and externally observable firings labeled with labels different from $\tau$. Note that the labeling function $\ell$ allows for the association of several different labels to a transition and the same label to different transitions. We illustrate the definition above by means of an example.

*Example 2.* Assume that $\mathcal{U}$ contains the set of natural numbers $\mathbb{N}$. Consider the net in Figure 5. It contains two places, $p$ and $q$, and two transitions, $t$ and $u$, with the following set of arcs $\{(p,t),(t,q),(q,u),(u,p)\}$. Both places are typed by $\mathbb{N}$, i.e., $\upsilon(p) = \upsilon(q) = \mathbb{N}$. The transition relation is given by $\rho(t) = \{((\{p \mapsto n\}, \{q \mapsto n+1\}) \mid n \in \mathbb{N}\}$ and $\rho(u) = \{((\{q \mapsto n\}, \{p \mapsto n+1\}) \mid n \in \mathbb{N}\}$. The labeling function $\ell$ is defined by $\ell(t,\gamma,\delta) = \tau$ and $\ell(u,\gamma,\delta) = a(\gamma(q))$. □

Next we introduce the notion of *marking* of a colored net over a given set of types.

**Definition 3.** *Given a colored net $N = (P,T,F,\upsilon,\rho,\ell)$ over the universe $\mathcal{U}$, a* marking *of $N$ is a function $M : P \times \mathcal{U} \to \mathbb{N}$, such that for any $p \in P$ and any $u \in \mathcal{U}$, $M(p,u) > 0$ implies $u \in \upsilon(p)$. The set of all markings of a colored net $N$ is denoted by $\mu(N)$.*

*A* marked colored net *over $\mathcal{U}$ is a pair $(N,M)$, where $N$ is a colored net over $\mathcal{U}$ and $M \in \mu(N)$ is a colored marking of $N$.*

8

To illustrate this notion we return to our previous example. Some possible markings for the net in Figure 5 are $M_n^p = \{(p,n) \mapsto 1\}$ and $M_n^q = \{(q,n) \mapsto 1\}$ for any $n \in \mathbb{N}$. So $M_n^p$, respectively $M_n^q$, consists of a single token with value $n$ in place $p$, respectively $q$. To simplify the notation when talking about markings we use the bag notation $[(p,n)]$ instead of the functional one $\{(p,n) \mapsto 1\}$. If $n$ coincides with *black* then we write $[p]$.

A colored net defines a transition system which gives the observable behavior of the net. To formalize this notion we denote by $\mathfrak{N}$ ($\mathfrak{M}$) the set of (marked) colored nets over the given universe.

**Definition 4.** *The ternary relation* $\_ \xrightarrow{\ \_\ } \_ \subseteq \mathfrak{M} \times \Sigma \times \mathfrak{M}$ *is defined as the smallest relation such that* $(N, M + \gamma) \xrightarrow{\ell(t,\gamma,\delta)} (N, M + \delta)$ *for all* $(N,M) \in \mathfrak{M}$, $t \in T$ *and* $(\gamma,\delta) \in \rho(t)$. *We also write* $(N,M) \xrightarrow{\sigma}$ *for some* $\sigma \in \Sigma$ *if and only if there exists a marking* $M' \in \mu(N)$ *such that* $(N,M) \xrightarrow{\sigma} (N,M')$. *Finally,* $(N,M) \xrightarrow{*} (N,M')$ *means that there exists a sequence* $(\sigma_1, \ldots, \sigma_n) \in \Sigma^*$ *such that* $(N,M) = (N,M_1) \xrightarrow{\sigma_1} (N,M_2) \xrightarrow{\sigma_2} \ldots \xrightarrow{\sigma_n} (N,M_{n+1}) = (N,M')$. *In this case we also say that* $(N,M')$ *is* reachable *in* $(N,M)$. *The set of all markings reachable in* $(N,M)$ *is denoted* $\mathcal{R}(N,M)$.

Consider net $N_1$ from Example 2. Then $(N_1, M_3^p) \xrightarrow{\tau} (N_1, M_4^q)$.

**Extended workflow nets** Workflow nets (WF nets) are used to model workflow systems (see e.g. [1–3, 6]). Workflow nets have one initial and one final place and every place or transition is on a directed path from the initial to the final place. We extend the notion of a WF net to model *exceptions*, i.e. transitions that should terminate the execution of the current net. As mentioned in the introduction, exception handling is recognized as a critical challenge for workflow management systems.

To ensure that firings of exception transitions are observable and cannot be confused with firings of non-exception transitions, we require the set of transition labels $\Sigma$ to be split into two parts: $\Sigma^e$ for the firings of exception transitions and $\Sigma^n$ for the firings of non-exception transitions. Formally, we require $\Sigma = \Sigma^e \cup \Sigma^n$, $\Sigma^e \cap \Sigma^n = \emptyset$ and $\tau \in \Sigma^n$.

**Definition 5.** *A colored net* $N = (P,T,F,\upsilon,\rho,\ell)$ *over the universe* $\mathcal{U}$ *is an* extended workflow net *(EWF net) with the initial place* $i \in P$, *and the final place* $f \in P$ *and the set of exception transitions* $T' \subseteq T$ *if*

1. $\{t \mid (t,i) \in F\} = \{t \mid (f,t) \in F\} = \emptyset$;
2. $\upsilon(i) = \upsilon(f) = \{black\}$;
3. $t \in T'$ *if and only if* $\ell(t,\gamma,\delta) \in \Sigma^e$ *for all* $(\gamma,\delta) \in \rho(t)$;
4. $\{p \mid (t,p) \in F\} = \emptyset$ *for all* $t \in T'$;
5. *For any node* $n \in (P \cup T)$ *there exists a path from* $i$ *to* $n$;
6. *For any node* $n \in (P \cup T)$ *there exists a path from* $n$ *to a node in* $T' \cup \{f\}$.

Traditional WF nets are EWF nets with the empty set of exception transitions. Since exception transitions produce no output, for any $t \in T'$ whenever $\ell(t,\gamma,\delta)$ is defined, $\delta = \emptyset$.
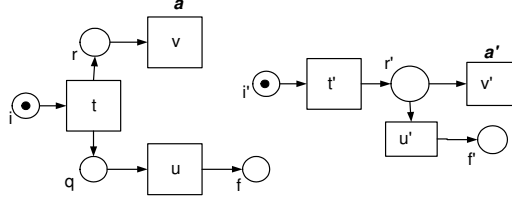
9

**Fig. 6.** EWF nets

*Example 6.* Consider the nets in Figure 6 as colored nets over the universe {*black*}. Both nets are EWF nets: the left-hand side net with the initial place *i* and with the final place *f*, the right-hand side net with the initial place *i'* and with the final place *f'*. The exception transitions are {*v*} and {*v'*}, respectively. All firings of these transitions are labeled by *a* and *a'*, respectively, where $a, a' \in \Sigma^e$. □

EWF nets provide a number of advantages from the modeling point of view. First and foremost they make a clear distinction between normal termination and termination caused by an exception. Unlike traditional WF nets, where special care should be taken to remove all tokens present in the system when an exceptional situation is encountered, no similar overhead is incurred by EWF nets. We define the following concepts regarding initiality and finality of EWF nets.

**Definition 7.** *Let N be an EWF net with initial place i and final place f and let $M \in \mu(N)$. The marked net $(N, M)$ is called* initial, *resp.* final *if and only if $M = [i]$, resp. $M = [f]$. The initialization* init(N) *of N is the marked net $(N, [i])$.*

An important correctness property for EWF nets is *soundness*. Classical WF nets are called sound if one can reach the final marking from any marking reachable from the initial marking [1]. The intuition behind this notion is that no matter what happens, there is always a way to complete the execution and reach the final state. This soundness property is sometimes also called *proper termination* and corresponds to 1-soundness in [18].

**Definition 8.** *An EWF net $N = (P, T, F, \upsilon, \rho, \ell)$ with the initial place i and the final place f over the universe $\mathcal{U}$ is called* sound *if and only if for all $(N, M)$ such that $[i] \xrightarrow{\sigma} M$ for some $\sigma \in (\Sigma^n)^*$ the following two conditions hold:*

1. *either $(N, M) \xrightarrow{\gamma} (N, [f])$ for some $\gamma \in (\Sigma^n)^*$*
   *or there exists $(N, M') \in \mathcal{R}(N, M)$ such that $(N, M') \xrightarrow{a}$ for some $a \in \Sigma^e$;*
2. *$(N, M) \xrightarrow{*} (N, m + [f])$ implies $m = \emptyset$.*

The intuition behind the notion of soundness formalized in Definition 8 is that for any possible behavior there is always a possibility to complete the execution and reach the final state, or to report an exception. The second requirement of soundness stresses that whenever the final state is reached, the execution has been completed. In case of classical WF nets, the second condition of soundness is redundant (cf. Lemma 11 in [18]). The following example shows that this is not the case for EWF nets.

10

*Example 9.* Recall the EWF nets from Figure 6. The left-hand side net is unsound. Indeed, consider $(N_l, [q] + [r])$, which is reachable from $(N_l, [i])$ since it can be obtained by firing $t$. Moreover, $(N_l, [q] + [r]) \xrightarrow{a}$ and $a \in \Sigma^e$. Hence, the first part of the soundness condition is satisfied. However, $(N_l, [q] + [r]) \xrightarrow{*} (N_l, [f] + [r])$, and since $[r] \neq \emptyset$ the second condition of soundness is violated and the net is unsound.

The right-hand side net is sound. Indeed, $\mathcal{R}(N_r, [i']) = \{[i'], [r'], [f'], \emptyset\}$, i.e., the only marking that contains $f'$ is $[f']$ itself. Hence, the second condition is satisfied. The first condition is satisfied since $(N_r, [i']) \xrightarrow{*} (N_r, [r'])$, $(N_r, [r']) \xrightarrow{*} (N_r, [r']) \xrightarrow{a'}$ on the one hand, and $(N_r, [f']) \xrightarrow{*} (N_r, [f'])$ on the other hand. $\qquad\square$

Note that adding an exception transition to a sound EWF net preserves soundness.

**Lemma 10.** *Let $N = (P, T, F, \upsilon, \rho, \ell)$ be a sound EWF net over the universe $\mathcal{U}$. Let $Q \subseteq P \setminus \{f\}$, $t \notin T$ and $h : \Pi_{p \in Q}(\upsilon(p)) \to \Sigma^e$. Let $N'$ be $(P, T \cup \{t\}, F \cup \{(p,t) \mid p \in Q\}, \upsilon, \rho', \ell')$, where*

- $\rho'(u) = \rho(u)$ for $u \in T$ and $\rho'(t) = \Pi(\varphi_t) \times \{\emptyset\}$;
- $\ell'(u, \gamma, \delta) = \ell(u, \gamma, \delta)$ for $u \in T$ and $(\gamma, \delta) \in \rho(u)$, and $\ell'(t, \gamma, \emptyset) = h(\gamma)$ for any $\gamma \in \Pi(\varphi_t)$.

*Then, the net $N'$ is a sound EWF net over $\mathcal{U}$.*

Note that since $t$ is an exception transition we have $\delta = \emptyset$ for any $(\gamma, \delta) \in \rho'(t)$.

*Proof.* The proof is done by checking Definition 8. $\qquad\square$

Lemma 10 allows us to use an incremental approach to modeling by modeling the normal course of events first, and adding exceptions afterward.

We define bisimilarity of EWF nets, by extending the standard notion with preservation of initiality and finality of states, cf. [17].

**Definition 11.** *Let $N_1, N_2$ be EWF nets. A relation $R \subseteq \mu(N_1) \times \mu(N_2)$ is called a* simulation *if and only if for all $M_1, M_1' \in \mu(N_1)$, $M_2 \in \mu(N_2)$ and $\sigma \in \Sigma$ such that $(N_1, M_1) \xrightarrow{\sigma} (N_1, M_1')$ there exists $M_2' \in \mu(N_2)$ such that $(N_2, M_2) \xrightarrow{\sigma} (N_2, M_2')$ and $M_1' \ R \ M_2'$.*

*The marked EWF nets $(N_1, M_1)$ and $(N_2, M_2)$ with initial and final places $i_1$ and $i_2$, $f_1$ and $f_2$ respectively, are called* EWF-bisimilar, *denoted $(N_1, M_1) =_e (N_2, M_2)$ if and only if there exists a relation $R \subseteq \mu(N_1) \times \mu(N_2)$ such that $M_1 \ R \ M_2$, both $R$ and $R^{-1}$ are simulations, and whenever $x \ R \ y$, the following holds:*

- $x = [i_1]$ *if and only if* $y = [i_2]$;
- $x = [f_1]$ *if and only if* $y = [f_2]$;
- $x > [f_1]$ *if and only if* $y > [f_2]$;

*The EWF nets $N_1$ and $N_2$ are EWF-bisimilar, denoted $N_1 =_e N_2$, if and only if their initializations are EWF-bisimilar.*

Note that EWF-bisimilarity is based on the traditional notion of strong bisimilarity. Further notions of bisimilarity, such as branching bisimilarity, can be adapted for EWF nets. EWF-bisimilarity preserves soundness of EWF nets. The requirement $> [f]$ to

be preserved by the simulation relation is required to ensure that the second part of Definition 8 is satisfied. It is not needed for traditional workflow nets. We define the *behavior* of a EWF net as its equivalence class modulo EWF-bisimilarity. The next lemma states that soundness is a behavioral property.

**Lemma 12.** *Let $N_1, N_2$ be EWF-bisimilar nets. Then $N_1$ is sound if and only if $N_2$ is sound.*

*Proof (Sketch).* We prove the "only if" side; the "if" side then follows by symmetry. Assume that $N_1$ is sound. We should prove that $N_2$ is sound. Let $R$ be a bisimulation between $init(N_1)$ and $init(N_2)$. Then, $[i_1] \, R \, [i_2]$. Let $M_2$ be such that $(N_2, [i_2]) \xrightarrow{*} (N_2, M_2)$. Since $R^{-1}$ is a simulation, there exists $M_1$ such that $M_1 \, R \, M_2$ and $(N_1, [i_1]) \xrightarrow{*} (N_1, M_1)$. Since $N_1$ is sound, either $(N_1, M_1) \xrightarrow{*} (N_1, [f_1])$ or $(N_1, M_1) \xrightarrow{*} (N_1, M_1') \xrightarrow{\sigma}$ for some $M_1' \in \mu(N_1)$ and $\sigma \in \Sigma^e$. Since $R$ is a simulation, either $(N_2, M_2) \xrightarrow{*} (N_2, [f_2])$ or $(N_2, M_2) \xrightarrow{*} (N_2, M_2') \xrightarrow{\sigma}$ for some $M_2' \in \mu(N_2)$ .

Finally, we need to prove that $(N_2, [i_2]) \xrightarrow{*} (N_2, [f_2] + m)$ implies $m = \emptyset$. Since $(N_2, [i_2]) \xrightarrow{*} (N_2, [f_2] + m)$, there exists $M_1$ such that $M_1 \, R \, [f_2] + m$ and $(N_1, [i_1]) \xrightarrow{*} (N_1, M_1)$. If $m > \emptyset$, then by the properties of $R$ we have $M_1 > [f_1]$. This contradicts the soundness of $M_1$, so in fact $m = \emptyset$. $\qquad\square$

## 4   Operations on EWF Nets

In this section we discuss predicates and operations that can be applied to EWF nets and marked EWF nets. In Section 5 we will introduce nested nets and we will see how these operations can serve as basic building blocks that can be used in defining the transition function of a higher-level net. We try and keep the set of operations as simple as possible so that they can be performed by existing workflow engines.

From here on in this section, all nets are EWF nets. The set of all (marked) EWF nets is denoted $\mathfrak{N}^w$ ($\mathfrak{M}^w$). We consider a number of predicates and operations on nets and marked nets. Using Definition 7 we convert a net into a marked net by adding to it a corresponding initial marking, and for any marked net we can check whether its marking is the initial one or the final one.

Next, two nets can be combined to produce a new net by means of sequential ($\cdot$) and parallel ($\|$) composition and choice ($+$). Moreover, parallel composition can be applied to marked nets and sequential composition to a marked net and a net. In the past, similar operations have been defined for workflow nets [6]. We adapt these notions for EWF nets.

One of the most natural operations on EWF nets is *sequential composition*. For instance, a doctor can prescribe a resection followed by a medication. Hence, the treatment prescribed by the doctor is the sequential composition of two nets: resection and medication. Other common operations on nets are parallel composition[3] and choice.

---

[3] It should be noted that the parallel composition we define is not exactly the well-known process-algebraic operation. For instance, after the initialization, $x + (y \| z)$ can make an internal step to a state where $x$ cannot be executed anymore.
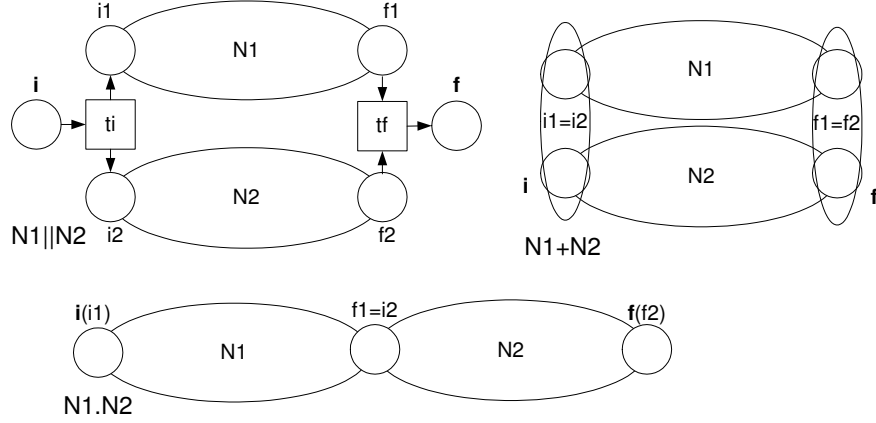
**Fig. 7.** Parallel composition $N1\|N_2$, Choice $N_1 + N_2$, Sequential composition $N1 \cdot N_2$

The three operations are depicted in Figure 7. $N_1$ and $N_2$ are unmarked EWF nets with the initial places $i_1$ and $i_2$, and the final places $f_1$ and $f_2$, respectively. The sign of equality denotes place fusion: for sequential composition the final place of the first net and the initial place of the second net have to be fused, for choice the initial places have to be fused as well as the final ones. Firing the transitions $ti, tf$ is labeled by $\tau$. The operations can be formalized by writing out the net structure, which is trivial but tedious.

**Lemma 13.** *For any $N_1, N_2 \in \mathfrak{N}^w$, $N_1 \cdot N_2, N_1 \| N_2, N_1 + N_2 \in \mathfrak{N}^w$. Moreover, $\cdot$ and $+$ are associative and $\|$ and $+$ are commutative. If $N_1$ and $N_2$ are sound, then $N_1 \cdot N_2, N_1 \| N_2, N_1 + N_2$ are sound as well.*

One of the typical operations one might like to consider is *iteration*. For instance, the doctor can prescribe a certain treatment to be performed three times. Associativity of $\cdot$ allows us to model this by writing $N \cdot N \cdot N$, where $N$ is the unmarked EWF net modeling the treatment. Formally we define $N^k$ as $N$ if $k = 1$, and as $N \cdot N^{k-1}$ if $k > 1$.

Next, consider marked nets. We start with the *sequential composition*. The intention of the sequential composition is to extend a running process by a new functionality. Therefore, we define the sequential composition as an operation on a marked net (a running process) and a net (additional functionality). Formally, we define $(N_1, M) \cdot N_2 = (N_1 \cdot N_2, M)$. Similarly, the *parallel composition* of two marked EWF nets $(N_1, M_1)$ and $(N_2, M_2)$ is defined as $(N_1, M_1) \| (N_2, M_2) = (N_1 \| N_2, M_1 + M_2)$.

A *choice operation* on marked nets does not seem useful for our purpose.

Operations on the marked nets satisfy the following lemma.

**Lemma 14.** *For any $(N_1, M_1), (N_2, M_2) \in \mathfrak{M}^w$ and $N \in \mathfrak{N}^w$, $(N_1, M_1) \cdot N \in \mathfrak{M}^w$, and $(N_1, M_1) \| (N_2, M_2) \in \mathfrak{M}^w$. Moreover, $\cdot$ is associative and $\|$ is commutative.*

Parallel composition and choice are congruences w.r.t. EWF-bisimilarity and sequential composition is a congruence if the first operand is sound. Its proof can be achieved by standard meta-theory of process algebra [16].
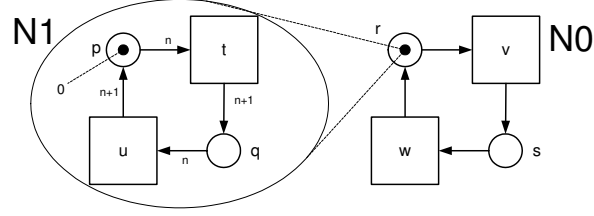
**Fig. 8.** Nested net

**Theorem 15.** *Let $N_1, N_2, N_1', N_2'$ be EWF nets such that $N_1 =_e N_1'$ and $N_2 =_e N_2'$. Then $N_1 \; \theta \; N_2 =_e N_1' \; \theta \; N_2'$, for $\theta \in \{+, \|, \cdot\}$.*

*Let $(N_3, M_3), (N_3', M_3'), (N_4, M_4), (N_4', M_4')$ be marked EWF nets with $(N_3, M_3) =_e (N_3', M_3')$ and $(N_4, M_4) =_e (N_4', M_4')$. Then $(N_3, M_3)\|(N_4, M_4) =_e (N_3', M_3')\|(N_4', M_4')$. and $(N_3, M_3) \cdot N_2 =_e (N_3', M_3') \cdot N_2'$.*

## 5  Nested Nets

In this section we introduce *nested nets*. In Section 2 we showed how nested nets can be used to model adaptive systems. We start by defining general nested nets and then move to nested EWF nets.

Nested nets form an extension of colored nets over a special universe. The only extension is that color of tokens can be changed without firing transitions. We recursively define value universes, nets and marked nets of any depth $n$, for $n = 0 \ldots \omega$, assuming the existence of an initial universe $\mathcal{U}_0$ containing basic values such as integers and composed values like pairs, lists and sets of basic or composed values. Subsequent universes and sets of nets are defined recursively as follows.

**Definition 16.** *The sets $\mathfrak{N}_0, \mathfrak{M}_0$ of nets and marked nets of depth zero are defined as the sets of colored nets and marked colored nets over the universe $\mathcal{U}_0$, respectively. For each $n > 0$ the value universe $\mathcal{U}_n$ and the sets $\mathfrak{N}_n, \mathfrak{M}_n$ of nets and marked nets of depth $n$ are recursively defined by $\mathcal{U}_n = \mathcal{U}_{n-1} \cup \mathfrak{M}_{n-1}$ and $\mathfrak{N}_n$ and $\mathfrak{M}_n$ as the set of colored nets and marked colored nets over $\mathcal{U}_n$. We set $\mathfrak{N}_\omega = \bigcup_{n \geq 0} \mathfrak{N}_n$, $\mathfrak{M}_\omega = \bigcup_{n \geq 0} \mathfrak{M}_n$ and $\mathcal{U}_\omega = \mathcal{U}_0 \cup \mathfrak{M}_\omega$.*

For the sake of uniformity we call elements of $\mathfrak{N}_0$ ($\mathfrak{M}_0$) (marked) nested nets of depth 0. Observe that the recursive definition of a notion of a marked nested net of depth $n$ allows tokens in it to be colored by nested nets of depth $n - 1$.

*Example 17.* To illustrate Definition 16 we first assume that the only elements of $\mathcal{U}_0$ are natural numbers. Then, net $N_1$ we introduced in Example 2 belongs to $\mathfrak{N}_0$. We can use $N_1$ as a token in a higher-level net as illustrated by Figure 8. We call this net $N_0$. It has two places, $r$ and $s$, and two transitions $v$ and $w$. The places $r$ and $s$ are of the "net type", i.e., the place typing function $\upsilon_0$ should map them to subsets of $\mathfrak{M}_\omega$. For instance, we can take $\upsilon : \{r, s\} \to \{(N_1, M) \mid M \in \mu(N_1)\}$. □

The firing relation $\longrightarrow$ for nested nets containing token nets may depend on the firing relation in the tokens themselves. We can thus achieve, for example, vertical synchronization: a transition in a lower-level net and another one in a higher-level net must fire simultaneously. The transition relation $\rho(t)$ may also react to initiality and finality of input tokens.

*Example 18.* Example 17, continued. We define the transition function and allow $v$ and $w$ to fire only if a transition labeled $a(n)$ for some $n \in \mathbb{N}$ fires in $N_1$. Formally, we write this as follows

$$\rho_1(v) = \{(\{r \mapsto (N_1, M)\}, \{s \mapsto (N_1, M')\}) \mid \exists n \in \mathbb{N} :: (N_1, M) \xrightarrow{a(n)} (N_1, M')\},$$

$$\rho_1(w) = \{(\{s \mapsto (N_1, M)\}, \{r \mapsto (N_1, M')\}) \mid \exists n \in \mathbb{N} :: (N_1, M) \xrightarrow{a(n)} (N_1, M')\}.$$

By defining $\rho_1$ as above, we require that firings of both $v$ and $w$ must synchronize with firings of $u$.

To complete the definition of the net we introduce the labeling function $\ell_1$. For any $\gamma$ and $\delta$, $\ell_1(v, \gamma, \delta) = \tau$. For $\ell_1(w, s \mapsto (N_1, M), r \mapsto (N_1, M')) = b(n)$, where $n \in \mathbb{N}$ is such that $(N_1, M) \xrightarrow{a(n)} (N_1, M')$. $\qquad\square$

We have seen that a token net can fire synchronously with the net containing it. However, we also want to allow a token net to proceed on its own, synchronizing only when some specific events occur or special states are reached. For this reason, we allow $\tau$-labeled firing of token nets, thus causing a color change, without firing a transition in the higher-level net. Of course, in $\mathfrak{M}_0$ this cannot occur. We formalize a new firing relation $\Longrightarrow$ in the following definition.

**Definition 19.** *Let the ternary relation $\_ \Longrightarrow \_ \subseteq \mathfrak{M}_\omega \times \Sigma \times \mathfrak{M}_\omega$ be the smallest relation satisfying*

- $(N, M + \gamma) \xrightarrow{\ell(t, \gamma, \delta)} (N, M + \delta)$ *for all $(N, M) \in \mathfrak{M}_\omega$ such that $N = (P, T, F, \upsilon, \rho, \ell)$, $t \in T$ and $(\gamma, \delta) \in \rho(t)$;*
- $x \xRightarrow{\tau} x'$ *implies $(N, M + [(p, x)]) \xRightarrow{\tau} (N, M + [(p, x')])$, for all $(N, M) \in \mathfrak{M}_\omega$ such that $N = (P, T, F, \upsilon, \rho, \ell)$, and all $p \in P$ such that $\upsilon(p) \subseteq \mathfrak{M}_\omega$, and $x \in \upsilon(p)$,*

The classes of (marked) nested nets are $\mathfrak{M}_\omega$ and $\mathfrak{N}_\omega$ respectively with $\Longrightarrow$ as firing relation. This firing relation defines the nested net transition system. Note that $\Longrightarrow$ and $\longrightarrow$ coincide in $\mathfrak{M}_0$. For $\mathfrak{M}_\omega$ we let the $\Longrightarrow$ firing relation replace the $\longrightarrow$ relation and define EWF-bisimilarity and soundness with respect to this relation $\Longrightarrow$.

We illustrate the firing relation introduced by means of Example 2.

*Example 20.* Given the markings of $N_1$ we observe that for any natural number $n$, four kinds of markings are possible for $N_0$: the token in $N_0$ is either in $r$ or in $s$ and the token in $N_1$ has value $n$ and is either in $p$ or in $q$. We denote the four combinations by $M_n^{r(p)}$, $M_n^{r(q)}$, $M_n^{s(p)}$, and $M_n^{s(q)}$. Then, the following sequence of firings is possible starting from $(N_0, M_0^{r(p)})$. First, an autonomous firing can take place in $N_1$, moving the token in it from $p$ to $q$ and increasing its numerical value, i.e., $(N_0, M_0^{r(p)}) \xRightarrow{\tau} (N_0, M_1^{r(q)})$.

Next, $u$ can fire in $N_1$. This transition increments the value of the parameter to 2. The firing of $u$ is labeled by $a(1)$ and by definition of $\rho_1$ $v$ fires as well. It should be noticed that by means of the labeling and the transition function synchronization is achieved. Observe also that since any firing of $v$ is labeled by $\ell_1$ as $\tau$, this synchronized step is also silent, i.e. $(N_0, M_1^{r(q)}) \overset{\tau}{\Longrightarrow} (N_0, M_2^{s(p)})$. Next, $N_1$ can again fire autonomously and again the value of the parameter is incremented: $(N_0, M_2^{s(p)}) \overset{\tau}{\Longrightarrow} (N_0, M_3^{s(q)})$. Finally, a synchronized firing of $u$ and $w$ occurs. The firing of $u$ increments the parameter further. The firing is labeled by $a(3)$. This time the firing of $N_0$ is observable since $\ell_1$ labels all firings of $w$ by $b(n)$, where $n$ is the value of the parameter in the labeling of the firing of $u$, i.e., $(N_0, M_3^{s(q)}) \overset{b(3)}{\Longrightarrow} (N_0, M_4^{r(p)})$. $\qquad\square$

The class of (marked) nested EWF nets are (marked) EWF nets over the universe $\mathcal{U}_\omega$ and the firing relation $\Longrightarrow$, which we represent by $\mathfrak{M}_\omega^w$ and $\mathfrak{N}_\omega^w$ respectively.

The healthcare workflow net discussed in detail in Section 2 is an example of a nested EWF net.

Since nested nets are derived from colored nets, and nested EWF nets are derived from EWF nets, operations and predicates introduced in Section 4 are readily applicable to nested nets. The example in Section 2 uses operators to combine processes modeled as nets. For instance, radiotherapy should be ministered in parallel with a cisplatine treatment followed by an etoposide treatment. After each session of treatment tests are carried out and the entire process is repeated four times. Using the operations we have introduced, this medical scenario can be written as $init(((Radiotherapy \| (Cisplatin \cdot Etoposide)) \cdot Tests)^4)$. This is a sound EWF net, provided the operand nets are sound.

**Theorem 21.** *Let $\mathbb{L} \subseteq \mathfrak{N}_\omega^w$ be a library of sound nested EWF nets. Then any term denoting a (marked) nested EWF net, obtained from the nets in $\mathbb{L}$ by application of $\cdot$, $\|$, $+$ and init is sound.*

In order to apply the theory we need a language to define the transition function. The language should be expressive enough to model protocols such as the one described in Section 2. However, we restrict the set of operations to concentrate on the behavior of the token nets rather than the way they are modeled. We call this class of nested EWF nets *adaptive* workflow nets.

We assume the existence of a library $\mathbb{L}$ of sound adaptive workflow nets. We assume that a countable set of variables $\mathbb{V}$ is given. The set of variables appearing in a term $\mathfrak{t}$ is denoted $Vars(\mathfrak{t})$.

**Definition 22.** *Let $A, B \subseteq \mathbb{V}$, and let $\mathfrak{t}$ be a term constructed from constants from $\mathbb{L}$, and variables from $A \cup B$ by means of application of $\cdot$, $\|$, $+$ and init. Then, the following are called* basic net predicates with respect to $A$ and $B$:

- *initial($\mathfrak{t}$), final($\mathfrak{t}$), $\mathfrak{t} \overset{l}{\longrightarrow}$, where $l \in \Sigma \setminus \{\tau\}$, and for any $v \in Vars(\mathfrak{t})$ it holds that $v \in A$;*
- *$\mathfrak{t} \overset{l}{\longrightarrow} u$, where $u \in B$, $l \in \Sigma \setminus \{\tau\}$, and for any $v \in Vars(\mathfrak{t})$ it holds that $v \in A$;*
- *$u = \mathfrak{t}$, where $u \in B$, and for any $v \in Vars(\mathfrak{t})$ it holds that $v \in A$;*

**Definition 23.** *A nested EWF net* $(P,T,F,\upsilon,\rho,\ell)$ *is called* adaptive *if for each* $t \in T$ *we have* $\rho(t)$ *is defined as follows. Let* $\alpha, \beta : P \to \mathbb{V}$ *be one-to-one mappings with* $\mathrm{ran}(\alpha) \cap \mathrm{ran}(\beta) = \emptyset$, *and* $\phi$ *is a first order logic formula constructed by means of predicates associated with* $\mathcal{U}_0$ *and the basic net predicates with respect to* $\mathrm{ran}(\alpha)$ *and* $\mathrm{ran}(\beta)$ *such that*

- $\exists$ *and* $\forall$ *are applied only to variables of type* $\mathcal{U}_0$;
- *for any free variable* $x$ *of* $\phi(\alpha,\beta)$, *it holds that* $x \in \mathrm{ran}(\alpha) \cup \mathrm{ran}(\beta)$.

*Let* $\theta : \mathbb{V} \to \mathcal{U}_\omega$. *Let* $\alpha^*$ *be the maximal subset of* $\theta \circ \alpha$ *such that for any* $x$, $\alpha^*(x) \in \upsilon(x)$ *and let* $\beta^*$ *be the maximal subset of* $\theta \circ \beta$ *such that for any* $x$, $\beta^*(x) \in \upsilon(x)$. *Then,* $\rho(t)$ *should be* $\{(\gamma,\delta) \mid \gamma \in \alpha^* \wedge \delta \in \beta^* \wedge \phi(\gamma,\delta)\}$.

Note that $\overset{l}{\Longrightarrow}$ coincides with $\overset{l}{\longrightarrow}$ whenever $l \neq \tau$. So the predicates $t \overset{l}{\longrightarrow}$ and $t \overset{l}{\longrightarrow} u$ in Definition 23 coincide with $t \overset{l}{\Longrightarrow}$ and $t \overset{l}{\Longrightarrow} u$ respectively.

The following theorem states a basic fact about adaptive nets. Its proof uses the fact that the unary net predicates only address behavior, i.e., they are true for a net $x$ if and only if they are true for a net $y$ that is EWF-bisimilar to $x$, and binary predicates always contain a variable in $\mathfrak{M}_\omega$. In other words, test predicates such as $x = C$ for $x \in \mathrm{ran}(\gamma)$ and $C \in \mathbb{L}$ are excluded.

**Theorem 24.** *Let* $x \in \mathfrak{M}_\omega^w$ *be EWF-bisimilar to an adaptive workflow net* $y$. *Let* $N = (P,T,F,\upsilon,\rho,\ell)$ *in* $\mathfrak{N}_\omega^w$ *and* $p \in P$ *such that* $\upsilon(p) = \mathfrak{M}_\omega^w$. *Then* $(N, M + [(p,x)]) \in \mathfrak{M}_\omega^w$ *is EWF-bisimilar to* $(N, M + [(p,y)])$.

To illustrate the definition above we formalize a number of transitions from the motivating example discussed in Section 2. We start with the transition labeled *operable* in Figure 4. For the sake of simplicity, assume that this transition is called $t$. It has one input place, say $p$, and one output place, say $q$. Transition $t$ can fire only if a firing labeled *operable* occurs in the token net of the input place. In this case, resection should be performed followed by four iterations of chemotherapy (cisplatin and then etoposide) and testing. Let $\rho(t) = \{(\gamma,\delta) \mid \phi(\gamma,\delta)\}$. Since $p$ is the only input place and $q$ is the only output place, $\gamma$ should have a form $\{p \mapsto x\}$, and $\delta$ should have a form $\{q \mapsto y\}$. To restrict the input we require $x \overset{operable}{\longrightarrow}$. To produce the output we write $y = init(Resection \cdot (Cisplatin \cdot Etoposide \cdot Tests)^4)$. Hence,

$$\rho(t) = \{(\{p \mapsto x\}, \{q \mapsto y\}) \mid x \overset{operable}{\longrightarrow} \wedge$$
$$y = init(Resection \cdot (Cisplatin \cdot Etoposide \cdot Tests)^4)\}.$$

The second example is the transition labeled *radiation scarring* in Figure 3. Whenever radiation scarring is observed by the doctor, prophylactic treatment should be administered in parallel with the ongoing process. We model this part of the guideline as a transition that can fire whenever *radiation scarring* is observed in a lower-level net. This transition produces the parallel composition of one of its inputs and *ProphT*, on the one hand, and returns the radiation control net to its original place. Since *ProphT* is a library net it should be first initialized and then used in the parallel composition.

Assuming that the transition is called *u*, its input and output places are *surv* and *radc*, we write

$$\rho(u) = \{ ( \{ surv \mapsto x_1, radc \mapsto x_2 \}, \{ surv \mapsto y_1, radc \mapsto y_2 \} ) \mid$$
$$x_2 \stackrel{radiation\ scarring}{\longrightarrow} y_2 \ \wedge \ y_1 = x_1 \| init(ProphT) \}.$$

## 6  Conclusion

Inspired by the existing work on the "nets in nets" paradigm [20, 27, 36], we introduced the class of nested nets to model processes with an adjustable structure. We also introduced EWF nets that extend classical workflow nets with an exception handling mechanism. Adaptive workflow nets are nested EWF nets whose behavior depends on the behavior of the token nets rather than on the way the token nets are modeled. Adaptive workflow nets are more flexible than classical workflow nets and have more structure than ad hoc systems.

To manipulate token nets in an adaptive workflow net we have identified a number of operations on EWF nets, including sequential composition, parallel composition and choice.

We have shown that the formalism presented allows to model realistic adaptive systems such as medical guidelines.

*Future work*  Our framework is obviously Turing complete. Hence, properties like soundness are in principle undecidable. Therefore, we would like to investigate subsets of adaptive nets where behavioral properties are decidable. The fact that nets can be built or modified at runtime makes it mandatory to develop patterns of nets for which soundness can be derived (soundness by construction). Another line of research is the implementation of our framework within an existing workflow engine.

## References

1. W. M. P. van der Aalst. Verification of workflow nets. In P. Azéma and G. Balbo, editors, *Application and Theory of Petri Nets 1997, ICATPN'1997*, volume 1248 of *Lecture Notes in Computer Science*. Springer-Verlag, 1997.
2. W. M. P. van der Aalst. The Application of Petri Nets to Workflow Management. *The Journal of Circuits, Systems and Computers*, 8(1):21–66, 1998.
3. W. M. P. van der Aalst. Workflow verification: Finding control-flow errors using Petri-net-based techniques. In W. M. P. van der Aalst, J. Desel, and A. Oberweis, editors, *Business Process Management: Models, Techniques, and Empirical Studies*, volume 1806 of *Lecture Notes in Computer Science*, pages 161–183. Springer-Verlag, 1999.
4. W. M. P. van der Aalst and T. Basten. Inheritance of workflows: an approach to tackling problems related to change. *Theor. Comput. Sci.*, 270(1-2):125–203, 2002.
5. W. M. P. van der Aalst, T. Basten, H. M. W. E. Verbeek, P. A. C. Verkoulen, and M. Voorhoeve. Adaptive workflow-on the interplay between flexibility and support. In *ICEIS*, pages 353–360, 1999.
6. W. M. P. van der Aalst and K. M. van Hee. *Workflow Management: Models, Methods, and Systems*. MIT Press, 2002.

7. W. M. P. van der Aalst, Arthur H. M. ter Hofstede, B. Kiepuszewski, and A. P. Barros. Workflow patterns. *Distributed and Parallel Databases*, 14(1):5–51, 2003.

8. W. M. P. van der Aalst, D. Moldt, R. Valk, and F. Wienberg. Enacting Interorganizational Workflows Using Nets in Nets. In J. Becker, M. Mühlen, and M. Rosemann, editors, *Proceedings of the 1999 Workflow Management Conference Workflow-based Applications, Münster, Nov. 9th 1999*, Working Paper Series of the Department of Information Systems, pages 117–136, University of Münster, Department of Information Systems, Steinfurter Str. 109, 48149 Münster, 1999. Working Paper No. 70.

9. M. Adams, A. H. M. ter Hofstede, D. Edmond, and W. M. P. van der Aalst. Facilitating flexibility and dynamic exception handling in workflows through worklets. In O. Belo, J. Eder, J. Falcão e Cunha, and O. Pastor, editors, *CAiSE Short Paper Proceedings*, volume 161 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2005.

10. O. Biberstein, D. Buchs, and N. Guelfi. Object-oriented nets with algebraic specifications: The CO-OPN/2 formalism. In *Concurrent Object-Oriented Programming and Petri Nets*, volume 2001 of *Lecture Notes in Computer Science*, pages 73–130, 2001.

11. G. Ciardo and P. Darondeau, editors. *Applications and Theory of Petri Nets 2005, 26th International Conference, ICATPN 2005, Miami, USA, June 20-25, 2005, Proceedings*, volume 3536 of *Lecture Notes in Computer Science*. Springer, 2005.

12. J. Desel, W. Reisig, and G. Rozenberg, editors. *Lectures on Concurrency and Petri Nets, Advances in Petri Nets*, volume 3098 of *Lecture Notes in Computer Science*. Springer, 2004.

13. H. Ehrig and J. Padberg. Graph grammars and Petri net transformations. In Desel et al. [12], pages 496–536.

14. A. Fent, H. Reiter, and B. Freitag. Design for change: Evolving workflow specifications in ULTRAflow. In A. B. Pidduck, J. Mylopoulos, C. C. Woo, and M. T. Özsu, editors, *Advanced Information Systems Engineering, 14th International Conference, CAiSE 2002, Toronto, Canada, May 27-31, 2002, Proceedings*, volume 2348 of *Lecture Notes in Computer Science*, pages 516–534. Springer, 2002.

15. F. Fossela, R. Komaki, and G. Walsh. Small-cell lung cancer. Practice Guideline, `http://utm-ext01a.mdacc.tmc.edu/mda/cm/CWTGuide.nsf/LuHTML/SideBar1`, 2000.

16. J. Groote and F. Vaandrager. Structured operational semantics and bisimulation as a congruence. *Information and Computation*, 100(2):202–260, 1992.

17. K. M. van Hee, N. Sidorova, and M. Voorhoeve. Soundness and separability of workflow nets in the stepwise refinement approach. In W. M. P. van der Aalst and E. Best, editors, *ICATPN*, volume 2679 of *Lecture Notes in Computer Science*, pages 337–356. Springer, 2003.

18. K. M. van Hee, N. Sidorova, and M. Voorhoeve. Generalised soundness of workflow nets is decidable. In J. Cortadella and W. Reisig, editors, *Application and Theory of Petri Nets 2004, ICATPN'2004*, volume 3099 of *Lecture Notes in Computer Science*, pages 197–216. Springer-Verlag, 2004.

19. K. Hoffman. Run time modification of algebraic high level nets and algebraic higher order nets using folding and unfolding construction. In G. Hommel, editor, *Proceedings of the 3rd Internation Workshop Communication Based Systems*, pages 55–72. Kluwer Academic Publishers, 2000.

20. K. Hoffmann, H. Ehrig, and T. Mossakowski. High-level nets with nets and rules as tokens. In Ciardo and Darondeau [11], pages 268–288.

21. K. Jensen. *Coloured Petri Nets - Basic Concepts, Analysis Methods and Practical*. Springer-Verlag, 1992.

22. M. Klein and C. Dellarocas. A knowledge-based approach to handling exceptions in workflow systems. *Comput. Supported Coop. Work*, 9(3-4):399–412, 2000.

23. M. Köhler and H. Rölke. Reference and value semantics are equivalent for ordinary object petri nets. In Ciardo and Darondeau [11], pages 309–328.

24. C. Lakos. From coloured Petri nets to object Petri nets. In *ICATPN*, volume 935 of *Lecture Notes in Computer Science*, pages 278–297, 1995.

25. I. A. Lomazova. Nested Petri nets: Multi-level and recursive systems. *Fundam. Inform.*, 47(3-4):283–293, 2001.

26. I. A. Lomazova. Modeling dynamic objects in distributed systems with nested Petri nets. *Fundam. Inform.*, 51(1-2):121–133, 2002.

27. I. A. Lomazova. *Nested Petri nets: modeling and analysis of distributed systems with object structure*. Moscow: Nauchny Mir, 2004. in Russian.

28. I. A. Lomazova and P. Schnoebelen. Some decidability results for nested Petri nets. In D. Bjørner, M. Broy, and A. V. Zamulin, editors, *Ershov Memorial Conference*, volume 1755 of *Lecture Notes in Computer Science*, pages 208–220. Springer, 1999.

29. D. Moldt and F. Wienberg. Multi-agent-systems based on coloured Petri nets. In P. Azéma and G. Balbo, editors, *ICATPN*, volume 1248 of *Lecture Notes in Computer Science*, pages 82–101. Springer, 1997.

30. S. Panzarasa, S. Maddè, S. Quaglini, C. Pistarini, and M. Stefanelli. Evidence-based careflow management systems: the case of post-stroke rehabilitation. *Journal of Biomedical Informatics*, 35(2):123–139, 2002.

31. M. Peleg, A. Boxwala, S. Tu, D. Wang, O. Ogunyemi, and Q. Zengh. Guideline interchange format 3.5 technical specification. InterMed Project, 2004.

32. S. Quaglini, S. Panzarasa, A. Cavallini, G. Micieli, C. Pernice, and M. Stefanelli. Smooth integration of decision support into an existing electronic patient record. In S. Miksch, J. Hunter, and E. T. Keravnou, editors, *AIME*, volume 3581 of *Lecture Notes in Computer Science*, pages 89–93. Springer, 2005.

33. S. Quaglini, M. Stefanelli, A. Cavallini, G. Micieli, C. Fassino, and C. Mossa. Guideline-based careflow systems. *Artificial Intelligence in Medicine*, 20(1):5–22, 2000.

34. S. Rinderle, M. Reichert, and P. Dadam. Correctness criteria for dynamic changes in workflow systems - a survey. *Data Knowl. Eng.*, 50(1):9–34, 2004.

35. R. Valk. Nets in computer organization. In Brauer, W., Reisig, W., and Rozenberg, G., editors, *Lecture Notes in Computer Science: Petri Nets: Applications and Relationships to Other Models of Concurrency, Advances in Petri Nets 1986, Part II, Proceedings of an Advanced Course, Bad Honnef, September 1986*, volume 255, pages 218–233. Springer-Verlag, 1987. NewsletterInfo: 27.

36. R. Valk. Object Petri nets: Using the nets-within-nets paradigm. In Desel et al. [12], pages 819–848.