Electronic Communications of the EASST
Volume 71 (2015)

Graph Computation Models
Selected Revised Papers from GCM 2014

Analysis of Petri Nets with Context-Free Structure Changes

Nils Erik Flick and Björn Engelmann

20 pages

Guest Editors: Rachid Echahed, Annegret Habel, Mohamed Mosbah
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Analysis of Petri Nets with Context-Free Structure Changes

Nils Erik Flick and Björn Engelmann*

{flick,engelmann}@informatik.uni-oldenburg.de,
https://scare.informatik.uni-oldenburg.de
Carl-von-Ossietzky-Universität, D-26111 Oldenburg

Abstract: Structure-changing Petri nets are Petri nets with transition replacement rules. In this paper, we investigate the restricted class of structure-changing workflow nets and show that two different reachability properties (*concrete* and *abstract* reachability) and word membership in the language of labelled firing sequences are decidable, while a language-based notion of correctness (containment of the language of labelled firing sequences in a regular language) is undecidable.

Keywords: Petri nets, transition replacement, correctness

1 Introduction

Petri nets or place/transition nets [DR98] are system models where resource tokens are moved around on an immutable underlying structure. They originally lack a notion of structure change or reconfiguration, but several structure-changing extensions have been formulated. In this paper, we define Petri nets together with transition replacement rules similar to graph replacement rules [EEPT06].

Petri nets are popular in the context of workflow modelling [van97], where labelled transitions correspond to tasks to be executed. An important property of workflow nets is *soundness*, which is decidable (in the absence of extra features such as reset arcs), and intuitively means the workflow can always terminate correctly and there are no useless transitions. Plain workflow nets, however, lack the ability of representing dynamic evolution [van01, WRR08].

Structure-changing Petri nets offer a potential solution for dealing with dynamic change in workflows by combining some advantages of Petri nets and graph grammars. In a structure-changing Petri net, structure-changing rules interfere with the behaviour of a Petri net. Reconfigurations occur unpredictably, modelling the influence of an uncontrolled environment, such as the dynamic addition of a component, or the unexpected complication or iteration of a task.

In detail, the results are decision procedures for

- the firing word problem for 1-safe structure-changing workflow nets
- reachability in structure-changing Petri nets
- abstract reachability in 1-safe structure-changing workflow nets

* The work of both authors is supported by the German Research Foundation (DFG), grant GRK 1765 (Research Training Group System Correctness under Adverse Conditions)

and undecidability of the containment of a structure-changing workflow net (labelled firing sequence) language in a regular language.

The paper is structured as follows: in Section 2, we introduce structure-changing Petri nets and workflow nets. In Section 3 we state our decidability and undecidability results. Section 4 draws parallels to existing work and Section 5 concludes with an outlook.

2 Structure-Changing Petri nets

In this section, we introduce Petri nets in the sense of [DR98, PW02] and extend them to structure-changing Petri nets as special graph grammars.

Notation. We use $-$ and $+$ to denote set difference and disjoint set union, respectively, and $\mathbb{N}$ for the set of natural numbers including 0. The cardinality of a set X is denoted by $|X|$, the length of a sequence w is also denoted $|w|$, and $|w|_Y$ denotes the number of occurrences in $w \in X^*$ of symbols from $Y \subseteq X$. The symbol ε denotes the empty word.

Assumption 1. There are two disjoint alphabets Σ and R of transition labels and rule names, respectively.

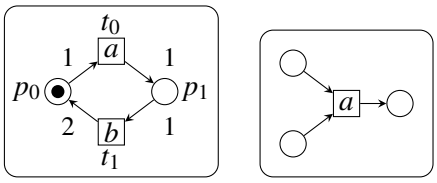
In the following, we introduce structure-changing Petri nets with coloured places. This type of nets will be used in Section 3.

Definition 1 A *net* is a tuple (P,T,F^-,F^+,l,c) where P is a finite set of *places*, T is a finite set of *transitions* with $P \cap T = \emptyset$, $F^-, F^+ : T \times P \to \mathbb{N}$ are functions assigning *preset* and *postset arc multiplicities*, respectively, $l : T \to \Sigma$ is the *labelling* function, $c : P \to \mathbb{N}$ is the *colouring* function. A *marked net* is a pair $\mathcal{N} = (N, M)$, where N is a net and $M : P \to \mathbb{N}$ is called the *marking* of $\mathcal{N}$.

Places and transitions are collectively called *items*. A net is said to be *(strongly) connected*, resp. *acyclic*, if it is (strongly) connected, resp. acyclic, as a graph, regarding arcs as directed edges. *Isomorphism*, denoted $N \cong N'$ (resp. $\mathcal{N} \cong \mathcal{N}'$) of (marked) nets means that there are bijections between the respective place and transition sets that preserve $F^{\pm}$, l and c (and M). The *size* of a net is $|P| + |T|$. A connected net with a single transition is a *1-net*.

Notation. when $F^-(t,p) = k$, one says there is an arc of weight k from p to t. Likewise, when $F^+(t,p) = k$, there is an arc of weight k from t to p. We write ${}^\bullet t$ for $\{p \in P \mid F^-(t,p) > 0\}$ and $t^\bullet$ for $\{p \in P \mid F^+(t,p) > 0\}$. When $M(p) = k$, it means that p is marked with k tokens. When $c(p) = k$, we say that p has colour k. The components of a net named N_x will likewise be named P_x and so on. If there is no subscript, they will be referenced as P and so on. Likewise, the marked net (N_x, M_x) is usually referred to as $\mathcal{N}_x$. The pictorial representation of nets as graph-like diagrams is well known, and a translation to graph grammars has been formalised in [Kre81] and [Cor95].

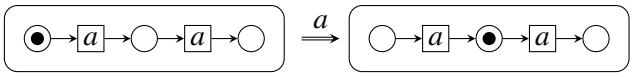
Example 1 (Graphical representation of a marked net; a 1-net:)



Let N be a net. The transition $t \in T$ is *enabled* by the marking M iff for all places p, $F^-(t,p) \leq M(p)$. The successor marking Mt of M via t is then defined by $\forall p \in P$, $(Mt)(p) = M(p) - F^-(t,p) + F^+(t,p)$, and $(N,M) \stackrel{t}{\Rightarrow} (N,Mt)$ is called a *firing step*.

For a sequence $u \in T^*$, the marking Mu is defined recursively as $M\varepsilon := M$ and $Mau := (Ma)u$. The sequence u is said to be *enabled* in M iff Mu is defined.

Example 2 (A firing step:)



Throughout this paper, all replacement rules are of a simple context-free kind that replace a single transition by an unmarked net. The adjective context-free is understood in the sense of hyperedge replacement [Hab92], which is also called context-free replacement. In our case, transitions correspond to hyperedges.

Definition 2 A (context-free) *rule* is a tuple (ρ, N_l, N_r) where $\rho \in R$ is the *rule name*, N_l is a 1-net, and N_r is a net with $P_l \subseteq P_r$, and $\forall p \in P_l, c_l(p) = c_r(p)$ N_l is the *left hand side* and N_r the *right hand side* of the rule.

A *match* of the 1-net N_l in the net N is a mapping $m : P_l \cup \{t_l\} \to P \cup T$ such that m maps the sole transition t_l of N_l to a transition such that $l_l(m(t_l)) = l(t_l)$ and the places $p_l \in P_l$ to places in P such that $\forall p \in P_l, c_l(m(p)) = c(p)$. The notion is extended to matches in marked nets: a *match* of N_l in (N,M) is a match of N_l in N. A *match* of the rule (ρ, N_l, N_r) on a marked net $\mathcal{N}$ is a match of N_l in N. An *abstract application*, with match m, of ρ to a marked net $\mathcal{N}$ is a pair $(\mathcal{N}, \mathcal{N}')$ such that if t_l is the transition in N_l, $T' = T - \{m(t_l)\} + T_r$, $P' = P + (P_r - P_l)$, M' coincides with M on the places from P and has value 0 otherwise. The place colours are as in N and N_r, and

$$F'^{\pm}(t,p) = \begin{cases} F^{\pm}(t,p) & t \in T, p \in P \\ F_r^{\pm}(t,p) & t \in T_r, p \in (P_r - P_l) \\ F_r^{\pm}(t,p') & t \in T_r, p = m(p') \\ 0 & \text{otherwise.} \end{cases}$$

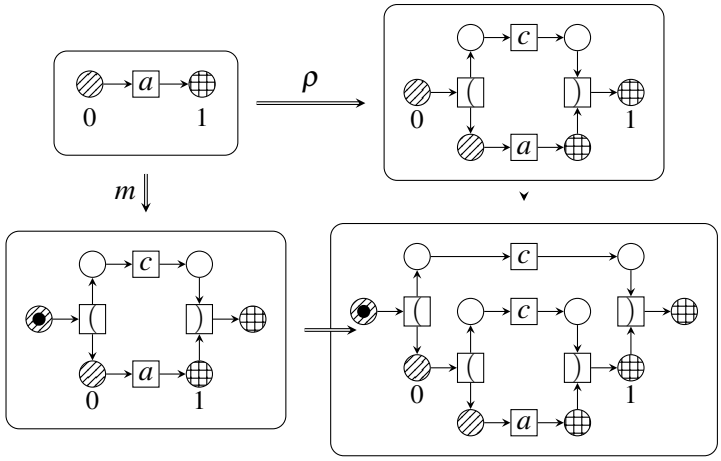
$\mathcal{N} \stackrel{\rho}{\Rightarrow} \mathcal{N}'$ or $\mathcal{N} \stackrel{\rho,m}{\Rightarrow} \mathcal{N}'$ is a *replacement step*.

Note that we always assume $P_r \cup T_r$ to be disjoint from $P \cup T$ in a replacement step (hence the notation "+" instead of "∪" in the above definition). Formally, in a replacement step N_l and N_r are replaced with isomorphic copies whose items do not occur in $P \cup T$, and these copies are

used in the rule application.[1]

Remark 1 *Given $\mathcal{N}$, ρ and a m, the resulting net $\mathcal{N}'$ is uniquely determined.*

Example 3 (A replacement step induced by a rule ρ and a match m:)



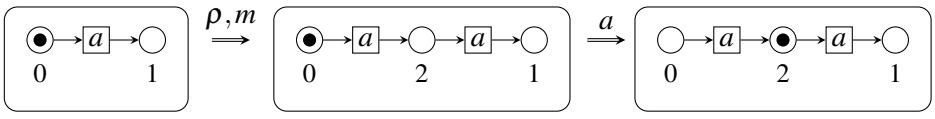
A *step* is either a firing step or a replacement step. We define four functions for extracting the information from a step: for a firing step e, let $\tau(e) := t$, $\lambda(e) := l(t)$, $\mathrm{from}(e) := (N,M)$ and $\mathrm{to}(e) := (N,M')$. For a replacement step, let $\tau(e) := m(t_l)$, $\lambda(e) := \rho$, $\mathrm{from}(e) := \mathcal{N}$ and $\mathrm{to}(e) := \mathcal{N}'$. The functions to, from, τ and λ canonically extend to sequences.

Definition 3 A *structure-changing Petri net* is a tuple $\mathcal{S} = (\mathcal{N},\mathcal{R})$, where $\mathcal{N}$ is a marked net and $\mathcal{R}$ is a finite set of rules.

Remark 2 *Every marked net $\mathcal{N}$ may be seen as a structure-changing Petri net $(\mathcal{N},\emptyset)$ with empty rule set, which will be called a* static *net and by abuse of notation also denoted $\mathcal{N}$. We will not distinguish between $\mathcal{N}$ and $(\mathcal{N},\emptyset)$.*

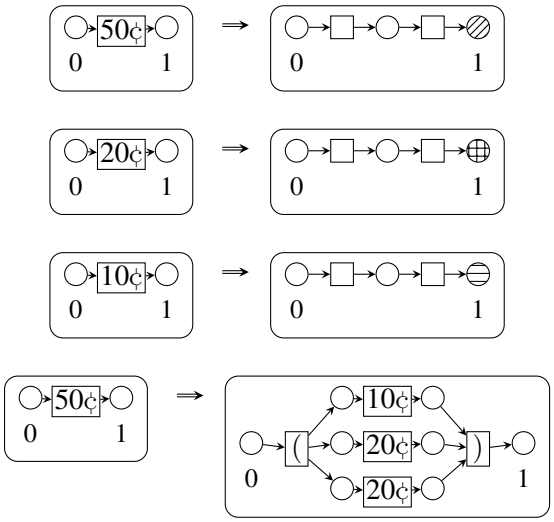
A *derivation* of length $n \in \mathbb{N}$ in a structure-changing Petri net $\mathcal{S} = (\mathcal{N},\mathcal{R})$ is a pair (ξ,σ) of a sequence $\xi_0 ... \xi_{n-1}$ of steps and a sequence $\sigma_0 ... \sigma_n$ of marked nets such that $\mathrm{to}(\xi_i) = \sigma_{i+1}$, $\mathrm{from}(\xi_i) = \sigma_i$ for all $i \in \{1,...,n-1\}$ and $\mathcal{N} = \sigma_0$. We write $\sigma_0 \overset{\xi}{\Rightarrow}_{\mathcal{R}} \sigma_n$ and say that (ξ,σ) *starts in* $\mathcal{N} = \sigma_0$ and *ends in* σ_n.

Example 4 (A structure-changing Petri net derivation. The small numbers serve to track places through the derivation:)



[1] As this causes little confusion in our paper, we will not stress this issue further.

Example 5 (Rules for a vending machine which either returns 50 cent pieces or smaller change. Place colours represent denominations of coins:)



A marked net $\mathcal{N}'$ is said to be reachable in $\mathcal{S}$ from $\mathcal{N}$ iff there is a derivation in $\mathcal{S}$ that starts in $\mathcal{N}$ and ends in $\mathcal{N}'$. The marked nets reachable in $\mathcal{S}$ are also called (reachable) *states* of $\mathcal{S}$. We write $\mathcal{RS}(\mathcal{S})$ for the set of all reachable states of $\mathcal{S}$. In a static net, instead of derivations one considers transition sequences, as they uniquely determine derivations. A net, rule or structure-changing Petri net is *k-coloured* if the highest colour assigned to any place does not exceed $k-1$. A marked net is *k-safe* if any reachable marking has at most k tokens per place.

Remark 3 *Note that the place colouring does not affect the behaviour of the net at all. It will be used in Section 3 to specify abstract markings.*

In the following, we introduce workflow nets [van97], extended by structure-changing rules, as a special case of structure-changing Petri nets.

Definition 4 A *workflow net* is a tuple (N, p_{in}, p_{out}) consisting of a net N and a pair of distinguished places $p_{in}, p_{out} \in P$, the input and output place which have no input respectively no output arcs, subject to the requirement that adding an extra transition from p_{out} to p_{in} would render the net strongly connected.

The data (p_{in}, p_{out}) need not be made explicit, since these places are easily seen to be uniquely determined in a workflow net. Thus we are justified in treating a workflow net as a special net. The *start marking* of a workflow net N, i.e. the marking where only p_{in} is marked with one token and all other places are not marked, is denoted by ${}^\bullet N$. The *end marking* where only p_{out} is marked with exactly one token is denoted by $N^\bullet$. A workflow net N is *sound* iff from any marking reachable from ${}^\bullet N$, $N^\bullet$ is reachable, and for each transition t there is some marking M reachable from ${}^\bullet N$ such that t is enabled in M.

Definition 5 A *structure-changing workflow net* is a structure-changing Petri net $\mathcal{S} = (\mathcal{N}, \mathcal{R})$ such that

(1) $\mathcal{N}$ is a sound workflow net marked with its start marking.

(2) for every rule $(\rho, N_l, N_r) \in \mathcal{R}$, N_l and N_r are sound workflow nets; N_l has two places, one transition, arc weights 1.

(3) $\sum_{t \in T_r} F_r^+(t, p_{out}) = \sum_{t \in T_r} F_r^-(t, p_{in}) = 1$ and the input (resp. output) place of N_r is p_{in} (p_{out}).

Condition (2) implies that only single-input, single-output 1-nets are permitted as left hand sides. The role of condition (3), which in our opinion does not unduly impact modelling capacity, is to avoid certain complications that otherwise arise in the analysis. Although the restrictions rule out markings with multiplicities, it is now possible to create more instances of a subnet by replacing transitions. Structure-changing workflow nets can still capture situations such as workflows that undergo complications as they are executed, as in Example 5.

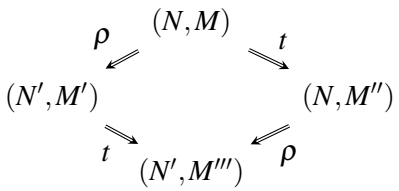
Every reachable state of a *structure-changing workflow net* is a reachable state of some workflow net. A structure-changing workflow net is called *acyclic* if every reachable state is an acyclic net. It is called *1-safe* if every reachable state is marked with at most one token per place.

3 Analysis of Structure-Changing Workflow Nets

In this section, we define and investigate some decision problems for structure-changing workflow nets. We define the derivation language of a structure-changing Petri net and show that while language containment in a regular language is undecidable, two reachability problems are decidable, and the firing word problem is decidable at least for structure-changing workflow nets.

We prove a series of lemmata, corresponding to a local Church-Rosser property for graph grammars [EEPT06], that allow us to equivalently re-arrange derivations, which will be convenient in later proofs.

Lemma 1 (Independence) Given a structure-changing Petri net $(\mathcal{N}, \mathcal{R})$, a firing step $(N, M) \overset{t}{\Rightarrow} (N, M'')$ and a replacement step $(N, M) \overset{\rho, m}{\Rightarrow} (N', M')$ for some $(\rho, N_l, N_r) \in \mathcal{R}$ and match m with $t \neq m(t_l)$, then there is a firing step $(N', M') \overset{t}{\Rightarrow} (N', M''')$ and a replacement step $(N, M'') \overset{\rho}{\Rightarrow} (N', M''')$ where $M'''(p)$ takes the value $M''(p)$ for $p \in P$ and 0 for $p \notin P$.



Proof. Immediate from Definition 2 and the definition of transition firing. □

The result (N', M''') is the same only up to an isomorphism, but this will not really impact any of the results. In the scope of this section, two derivations (ξ, σ) and (ξ', σ') are *equivalent* if they are of the same length n and the marked nets σ_n and σ'_n are isomorphic. Two transition sequences are said to be equivalent if they are equivalent as derivations. Transition sequences

which only differ by a permutation of the steps are easily seen to be equivalent. In a sound workflow net N, a transition sequence u is said to be *terminal* when $({}^\bullet N)u = N^\bullet$.

Lemma 2 (Permuting Steps) If $\mathcal{S}$ is a structure-changing Petri net, for every derivation (ξ,σ) of length n in $\mathcal{S}$, under any of the conditions given below there is a derivation (ξ_π,σ_π) also of length n such that σ_n and $(\sigma_\pi)_n$ are isomorphic, and $\lambda(\xi) = uabv$ and $\lambda(\xi_\pi) = ubav$ for some $u,v \in (R \cup \Sigma)^*$ and $a,b \in (R \cup \Sigma)$. If $\lambda(\xi_i) = a$ and $\lambda(\xi_{i+1}) = b$, $t_a = \tau(\xi_i)$, $t_b = \tau(\xi_{i+1})$, $\mathrm{from}(\xi_i) = \mathcal{N}$, $\mathrm{to}(\xi_i) = \mathrm{from}(\xi_{i+1}) = \mathcal{N}'$ and $\mathrm{to}(\xi_{i+1}) = \mathcal{N}''$,

Situation	Sufficient condition for transposition
$a \in \Sigma, b \in \Sigma$	${}^\bullet t_b \cap t_a{}^\bullet = \emptyset$
$a \in \Sigma, b \in R$	$t_a \notin T - T'$
$a \in R, b \in \Sigma$	$t_b \notin T' - T$
$a \in R, b \in R$	$t_b \notin T' - T$

Proof. $a,b \in \Sigma$: transitions meeting the requirement can be transposed in an enabled sequence: in a net N, if Mt_1t_2 exists and ${}^\bullet t_2 \cap t_1{}^\bullet = \emptyset$, then t_2 is enabled in M: all places in ${}^\bullet t_2$ hold at least as many tokens as in Mt_1, and t_1 is enabled in Mt_2 because $\forall p \in {}^\bullet t_1 \cap {}^\bullet t_2$ have $M(p) \geq F^-(p,t_1) + F^-(p,t_2)$. By commutativity of addition $Mt_1t_2 = Mt_2t_1$ and hence $\sigma_{|u|+2} = (\sigma_\pi)_{|u|+2}$.

$a \in \Sigma, b \in R$: b can be applied to $\sigma_{|u|}$ because rule applicability is independent of the marking, and by Lemma 1 the same state $\sigma_{|u|+2}$ is reached. The condition is necessary to meet the requirements of Lemma 1: otherwise, t_a is not available after the replacement step.

$a \in R, b \in \Sigma$ is similar: since the preset of the transition t_b is unchanged by the application of R, t_b is enabled in the state $\sigma_{|u|}$ and Lemma 1 applies.

$a,b \in R$: since only the matched transition is replaced and all others remain unchanged, the rules can be swapped, yielding an isomorphic result. In this case and the previous one, the condition is necessary because otherwise the transition t_b is not present prior to the event σ_i. □

The homomorphism $t \mapsto (\text{if } t \in A \text{ then } 0 \text{ else } 1)$ defines a pre-order $\precsim_{A,B}$ on sequences of $(A+B)^*$ by comparing the images of sequences under the lexicographic order induced by $0 < 1$ on $\{0,1\}^*$.

Lemma 3 Given a transition sequence u in a marked net with $T = T' + T''$, and $T' \times T''$ contains only pairs (t',t'') where ${}^\bullet t'' \cap t'{}^\bullet = \emptyset$, then any transition sequence that is equivalent to u and minimal with respect to $\precsim_{T',T''}$ is in $T'^*T''^*$.

Proof. A strictly decreasing step along the lexicographic order is possible as long as Lemma 2 can still be applied to a contiguous subsequence $t''t'$, yielding another equivalent transition sequence. □

Definition 6 A step e_i of a derivation (ξ,σ) *causally precedes* another step e_j, $i < j$, if either $\lambda(\xi_i), \lambda(\xi_j) \in \Sigma$ and $\tau(\xi_i)^\bullet \cap {}^\bullet\tau(\xi_j) \neq \emptyset$, or $\lambda(\xi_i) \in R, \lambda(\xi_j) \in \Sigma$ and $\tau(\xi)_j \in \mathrm{codomain}(m_i)$ or $\lambda(\xi_i), \lambda(\xi_j) \in R$ and $\tau(\xi)_j \in \mathrm{from}(e_i)$. The *causal dependency* relation is the transitive closure of the causal precedence relation.

Lemma 4 Any derivation (ξ,σ) equivalent to a given one and minimal with respect to $\precsim_{\Sigma,R}$

has $\lambda(\xi) = r_0 s_0 \ldots s_n r_{n+1}$ where $\forall i, s_i \in \Sigma, r_i \in R^*$ and if $\xi_{i,0} \ldots \xi_{i,n} = r_i$, then $\xi_{i,j}$ causally precedes $\xi_{i,j+1}$, and $\xi_{i,n}$ causally precedes s_{i+1}.

Proof. Analogous to Lemma 3. As long as there is a contiguous subsequence $\xi_i \zeta \xi_j$ where $\lambda(\xi_i) \in R$, $\lambda(\zeta) \in R^*$, $\lambda(\xi_{i+1}) \in \Sigma$ and ξ_i does not causally precede any of $\zeta \xi_j$, by induction over the length of ζ and using Lemma 1, an equivalent but strictly $\precsim_{\Sigma,R}$-smaller derivation with $\ldots \zeta \xi_j \xi_i \ldots$ can be constructed. □

3.1 Firing Word Problem

Definition 7 The *firing language* of a structure-changing Petri net $(\mathcal{N}, \mathcal{R})$ is

$$\mathcal{F}(\mathcal{S}) := \{ f(w) \in (\Sigma \cup R)^* \mid \exists \mathcal{N}' : \mathcal{N} \overset{w}{\Rightarrow}_{\mathcal{R}} \mathcal{N}' \}$$

where f is the homomorphism that deletes all letters of R and leaves transition labels Σ unchanged.

Due to the deleting homomorphism, it is not trivial to see whether a given word w occurs in the firing language of $\mathcal{S}$.

Problem 1 (Firing word problem)**.**

Given:	a structure-changing Petri net $\mathcal{S}$ and a word $w \in \Sigma^*$
Question:	$w \in \mathcal{F}(\mathcal{S})$? Is w contained in the firing language of $\mathcal{S}$?

We leave Problem 1 open and treat the firing word problem for 1-safe structure-changing workflow nets instead.

Problem 2 (Problem 1 for 1-safe structure-changing workflow nets)**.**

Given:	a 1-safe structure-changing workflow net $\mathcal{S}$ and a word $w \in \Sigma^*$
Question:	$w \in \mathcal{F}(\mathcal{S})$? Is w contained in the firing language of $\mathcal{S}$?

The question is decidable for 1-safe structure-changing workflow nets by the following means: while applying rules, one keeps track of the minimum length of any word in which each transition could occur (the *minimum word length* defined below). Rule matches on transitions that can only be used in words longer than w need not be explored, because under the postulated assumptions the minimum word length is non-decreasing under replacement. Rules are only applied to enabled transitions. The creation of redundant transitions is limited by condition (3) of the definition of a structure-changing workflow net.

In a static net $\mathcal{N} = (N, M)$, let $\| \cdot \|_{\mathcal{N}} : T \rightharpoonup \mathbb{N}$ be the partial function assigning to each transition t of the net the minimum length of any derivation using t. It is undefined if there is no such derivation, otherwise $\|t\|_{\mathcal{N}} = min(\{|ut| \mid ut \in T^*, \exists M' = Mut\})$. We call ut with $|ut| = \|t\|_{\mathcal{N}}$ a *minimum-length t-word* of $\mathcal{N}$. It contains t exactly once at the end, or a proper prefix would suffice. Note also that sequences in T^* rather than label words in Σ^*, are considered. Calculating $\|t\|_{\mathcal{N}}$ might in general not be efficient, but it is in principle possible for all nets.

Lemma 5 Let $\mathcal{S} = (\mathcal{N}, \mathcal{R})$ be a 1-safe structure-changing workflow net. If $\mathcal{N} \stackrel{*}{\Rightarrow} \mathcal{N}'$, and $\mathcal{N}' \stackrel{\rho,m}{\Rightarrow} \mathcal{N}''$ is a replacement step using rule (ρ, N_l, N_r), and let $t = m(t_l)$. Then each transition $\underline{t} \in T'' - T'$ has $\|\underline{t}\|_{\mathcal{N}''} = \|t\|_{\mathcal{N}'} - 1 + \|\underline{t}\|_{(N_r, {}^\bullet N_r)} \geq \|t\|_{\mathcal{N}'}$, and each transition $\check{t} \in T - \{t\}$ has $\|\check{t}\|_{\mathcal{N}''} \geq \|\check{t}\|_{\mathcal{N}'}$.

Proof. Let ut be a minimum-length t-word in $\mathcal{N}$ and $v\underline{t}$ a minimum-length $\underline{t}$-word in $(N_r, {}^\bullet N_r)$. Then $uv\underline{t}$ is a minimum-length $\underline{t}$-word in $\mathcal{N}'$.

A transition sequence $v \in T_r^*$ is enabled in some marking in N_r precisely when it is in the corresponding marking (mapping the places via m) in N'' and has the same effect on the places of $P_r - P_l + m(P_l)$ and no effect on all other places. If u is a transition sequence enabled in M', then Lemma 3 guarantees the existence of a transition sequence $\tilde{u}$ that can be decomposed as $\tilde{u} = uv$, where $u \in T'^*$ and $v \in (T'' - T')^*$. For any such uv, because Mu exists and enables t, ut is also a possible transition sequence, and furthermore the shortest one containing t.

As to the second statement, let $u\check{t}$ be a minimum-length $\check{t}$-word of $\mathcal{N}''$. Decomposing u as $u_0 t_0 u_1 \ldots t_n u_{n+1}$ with $u_i \in T'^*$ and $t_i \in T'' - T'$, there is an equivalent $\precsim_{T''-T',T'}$-minimal enabled transition sequence $\tilde{u} = \tilde{u}'_0 s_0 \tilde{u}'_1 s_1 \ldots \tilde{u}'_{m+1}$ with $\tilde{u}'_i \in T'^*$, $s_k \in (T'' - T')^*$ and $s_0 \cdot \ldots \cdot s_m = t_0 \cdot \ldots \cdot t_n$, and $\tilde{u}'_0 \cdot \ldots \cdot \tilde{u}'_{m+1} = u_0 \cdot \ldots \cdot u_{n+1}$, and s_k is terminal in N_r: assuming the contrary would imply that any equivalent transition sequence of the form $\tilde{u}'_0 s_0 \tilde{u}'_1 s_1 \ldots \tilde{u}'_m$ has some s_k such that the condition $({}^\bullet N_r) s_k = N_r{}^\bullet$ is violated. Every transition t_i added in the replacement step falls into one of three sets, which we call *start*, *middle* and *end*. A *start* transition is one enabled in ${}^\bullet N_r$. A *end* transition is one that has the end place of N_r in its postset. A *middle* transition is any other transition of N_r. So if $\tilde{u}'_0 s_0 \tilde{u}'_1 s_1 \ldots \tilde{u}'_m$ is such a sequence where for some $k \in \{0.....m\}$, $({}^\bullet N_r) s_k$ differs from $N_r{}^\bullet$ then the first transition of s_{k+1} (if $k < m-1$) can neither be a start, nor middle, nor end transition. The latter two lead to a contradiction with $\precsim_{T''-T',T'}$-minimality, while the former would contradict 1-safety.

Therefore one can construct a $\check{t}$-word of $\mathcal{N}'$, namely $\tilde{\tilde{u}}'' = \tilde{u}'_0 t \tilde{u}'_1 \ldots \tilde{u}'_m \check{t}$, which is in any case at most as long as $u\check{t}$, thus $\|\check{t}\|_{\mathcal{N}'} \leq \|\check{t}\|_{\mathcal{N}''}$ □

Proposition 1 *It is decidable for any 1-safe structure-changing workflow net $\mathcal{S}$ and word $w \in \Sigma^*$ whether $w \in \mathcal{F}(\mathcal{S})$.*

Proof. The idea of the proof was described in the text following the problem description. The following algorithm decides the question:

```
1:  procedure WORD(𝒩, w)
2:      k := |w|
3:      St := {𝒩}
4:      while w ≠ ε do
5:          St := EXPLORE(St, k − |w|)
6:          aw := w                          ; a was the first letter, w now contains the rest
7:          St := {(N, Ma) | (N, M) ∈ St}    ; try successor by firing, if defined
8:      return St ≠ ∅
9:  procedure EXPLORE(St, j)
10:     repeat
11:         for all 𝒩 ∈ St do                ; gather relevant successors by replacement
```

12: **for all** enabled transitions t of $\mathcal{N}$ **do**
13: **for all** matches m of rules ρ on t **do**
14: $\mathcal{N}' :=$ result of replacement step ; $\mathcal{N} \overset{\rho,m}{\Rightarrow} \mathcal{N}'$
15: **if** $\nexists \mathcal{N}'' \in St : \mathcal{N}' \cong_j \mathcal{N}''$ **then**
16: add $\mathcal{N}'$ to St
17: **until** no *new* states found
18: **return** St

We prove that the algorithm terminates and outputs the correct answer.

The procedure WORD iterates over w. It is correct, by induction on the length of w: the induction basis is witnessed by the trivial case of the empty word. Induction hypothesis: for any $j \leq k \in \mathbb{N}$, the prefix u of w of length j is in the language iff St is not empty. Induction step: for any derivation (ξ, σ) with $f(\lambda(\xi)) = au$, Lemma 4 yields another derivation (ξ', σ'), with $\lambda(\xi') = zau'$, $z \in R^*$ and $f(u') = u$, and each step e_i ($i \leq |z|$) causally preceded by the previous.

Each of the replacement steps e_i whose corresponding rule names form the z prefix causally precedes the next; unless $|z| = 0$, the first transition to be fired, labelled a, is created in the step $e_{|z|-1}$. In any of the first $|z|$ steps, the matched transition is enabled. This makes it unnecessary to ever directly explore the replacement of disabled transitions.

It remains to be seen that the subset of reachable states of $\mathcal{S}$ that actually need to be explored further for the word u is finite: it is always the case that $|z|$, which may well be 0, can be bounded from above by a constant depending only on the rules. The reason why the EXPLORE procedure terminates rests with the comparison that decides whether states are added to St. The comparison is done according to an equivalence relation $\cong_j$ ($j \in \mathbb{N}$): a pair of marked nets is in $\cong_j$ if they have the same subnet induced by considering only the transitions that can contribute to words of length up to j, and the places attached to these. By Lemma 5, minimum word lengths only increase and therefore all transitions with minimum word length exceeding j can be ignored in the exploration.

Only a finite number of equivalence classes of $\cong_j$ occur in the application of replacement steps from $\mathcal{R}$. Let n be the maximum number of items in a right hand side of any rule of $\mathcal{R}$. Regarding each state as a directed graph whose nodes are the places and transitions, and whose arcs are directed edges, let d be the function assigning to each item it the length of the shortest directed path from the place p_{in} to it. Then by induction on the length of the derivation, no rule application leads to a state which, as a graph, has any node with more than n successors. The number of items with d up to $2j$ is bounded by n^{2j}. There is a finite number of 1-safe nets with these properties.

The set of items of $\mathcal{N}$ with d up to $2j$ clearly includes all transitions t with $\|t\|_{\mathcal{N}} \leq j$. Firing may reduce the minimum word length of any transition by at most 1. This makes it safe to compare states by $\cong_{j-1}$ at the next iteration. □

3.2 Reachability Problems

In this subsection, before defining abstract reachability we first show that concrete reachability (without using any abstraction) of a given marked net is decidable. Our construction for deciding concrete reachability is similar to the unfolding construction used in [BCKK04] in a different context. The idea is to unfold all possible rule applications until the size of the smallest net obtained using a new match exceeds the queried $\mathcal{N}$, since the size of the net grows monotonically with each rule application and the set of rules is finite.

In the construction, for each possible match, the net is extended the same way as in a rule application but the matched transitions are preserved. To control the simulation, the rule transitions and control places take care of disabling or enabling net transitions according to the simulated rule applications.

Let $\text{match}_\rho(N) \subseteq T \times P^*$ be the set of all matches of rule ρ on net N, each match being uniquely determined by target transition and place mapping (we assume here that every left hand side is equipped with an arbitrary total order on the place set, inducing a bijection between place mappings and sequences over P).

Construction 1 (k-unfolding)**.** Given a n-coloured structure-changing Petri net $(\mathcal{N},\mathcal{R})$, let N_0 be the net obtained from N by adding new places $\{p_t \mid t \in T\}$ and arcs of weight 1 from each t to its respective p_t and back. The new colour n is chosen for all the p_t places. To each item created in the construction, we assign a *history*, which is a sequence of steps. All transitions t of N_0 have $\text{hist}(t) = \varepsilon$. Items added in a rule application $e_{\mu,\rho}$ matching $t \in T_i$ will have history $hist(t) \cdot e_{\mu,\rho}$.

For $i \leq k$, the set T_{i+1} is computed from T_i by disjointly adding, for every match μ of any rule $(\rho, N_l, N_r) \in \mathcal{R}$, the transitions $T_r - T_l$, and one transition for each match, which we call *match transition*: $\{t_{\text{hist}(t)\cdot e_{\mu,\rho}} \mid \mu \in \text{match}_\rho(N_i)\}$ where $e_{\mu,\rho}$ is the replacement step determined by ρ, μ and N_i.

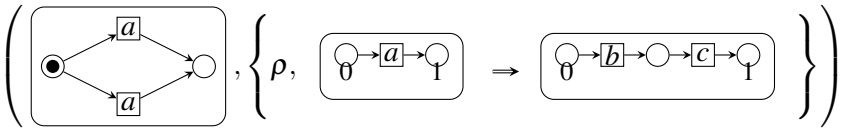
The set P_{i+1} is computed from P_i by disjointly adding, for every match μ of any rule $(\rho, N_l, N_r) \in \mathcal{R}$, the places of $P_r - P_l$, and one *control place* for every right hand side transition: $P_{\text{ctrl}} = \{p_t \mid t \in T_{i+1} - T_i, l(t) \in \Sigma\}$.

The arc weights in N_{i+1} are as follows:

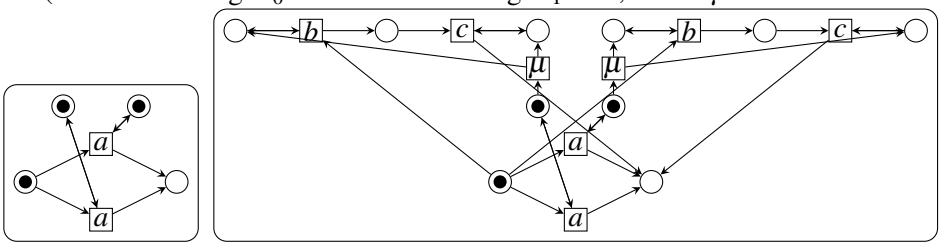
$$F^{\pm}_{i+1}(t,p) = \begin{cases} F^{\pm}_i(t,p) & t \in T_i, p \in P_i \\ F^{\pm}_r(t,p) & t \in T_r, p \in (P_r - P_l) \ \ (*) \\ F^{\pm}_r(t,p') & t \in T_r, p = m(p') \ \ (*) \\ 1 & t \in T_i + T_r, \ p = p_t \\ 1 & t = t_x, \ c(p) = n, \ \text{hist}(p) = xe \ \ (F^+\text{only}) \\ 1 & t = t_{xe}, \ c(p) = n, \ \text{hist}(p) = x \ \ (F^-\text{only}) \\ 0 & \text{otherwise,} \end{cases}$$

where $(*)$ applies to items obtained from the same match of a rule (ρ, N_l, N_r), x is any sequence of steps and e is any step. The p_t places are always assigned colour n and the t_x transitions are labelled with the corresponding rule name from R. The marking M_k agrees with M on the places of N_0, has 1 on each p_t place and 0 elsewhere. $\mathcal{N}_k = (N_k, M_k)$ is the *k-unfolding* of $(\mathcal{N},\mathcal{R})$.

Example 6 (A structure-changing workflow net $\mathcal{S} = (\mathcal{N}, \mathcal{R})$:)



Example 7 (The 0-unfolding $\mathcal{N}_0$ and the 1-unfolding $\mathcal{N}_1$ of $\mathcal{S}$, where μ indicates match transitions:)



Let $\kappa' = \kappa'(\mathcal{S}, \mathcal{N}_Q) \in \mathbb{N}$ be the smallest number such that every marked net $\mathcal{N}$ reachable in any derivation (ξ, σ) without state cycles (i.e. repetition-free σ) and with $|\lambda(\xi)|_R \geq \kappa'$ has no derivation $\mathcal{N} \overset{*}{\Rightarrow}_{\mathcal{R}} (N_Q, M)$ for any M. The number κ' is well-defined because net size increases monotonically along any derivation, and only finitely many nets of any given size are reachable. For the same reason, κ' can be effectively over-approximated by considering all derivations with $|\lambda(\xi)|_\Sigma = 0$ that produce nets of size at most $|P_Q| + |T_Q|$.

Proposition 2 *It is decidable whether a given marked net $\mathcal{N}_Q$ is reachable in a given structure-changing Petri net $\mathcal{S} = (\mathcal{N}, \mathcal{R})$.*

Proof. The idea is to determine the number $\kappa'(\mathcal{S}, \mathcal{N}_Q)$ and reduce the question to reachability in a κ-unfolding $\mathcal{N}_\kappa$ of $\mathcal{S}$ for $\kappa \geq \kappa'(\mathcal{S}, \mathcal{N}_Q)$, the Petri net reachability problem being well-known to be decidable [PW02].

$$\begin{array}{ccc} (\hat{N}, \hat{M}) & \xleftarrow{\text{strip}} & (N_\kappa, M) \\ \rho \text{ or } t \Big\downarrow & & \Big\downarrow t_\mu \text{ or } t \\ (\hat{N}', \hat{M}') & \xleftarrow{\text{strip}} & (N_\kappa, M') \end{array}$$

We prove a mutual simulation of $\mathcal{S}$ and $\mathcal{N}_\kappa$ for any derivation ξ with up to κ rule applications by induction on the length of ξ. As argued in the definition of $\kappa(\mathcal{S}, \mathcal{N}_Q)$, no derivation with $|\lambda(\xi)|_R > \kappa$ yields $\mathcal{N}_Q$. If $|\lambda(\xi)|_R \leq \kappa$, by induction on the length of the derivation, $\mathcal{N}_\kappa$ and $\mathcal{S}$ simulate each other via a mapping strip, defined as follows: the image of (N_κ, M) is the marked net $\hat{\mathcal{N}}$ with transitions $\hat{T} = \{t \in T \mid M(p_t) > 0\}$, places $\hat{P} = (P - P_{\text{ctrl}}) - \{p \in P \mid \forall t \in {}^\bullet p \cup p^\bullet, t \notin \hat{T}\}$ and $\hat{F}^-$, $\hat{F}^+$, $\hat{l}$, $\hat{c}$ are F^-, F^+, l, c with their domains restricted to $\hat{P}$, $\hat{T}$.

If $\hat{\mathcal{N}} \overset{x}{\Rightarrow}_{\mathcal{R}} \hat{\mathcal{N}}'$, if $x \in R$, and by hypothesis, $\hat{\mathcal{N}} = \text{strip}(\mathcal{N}'')$ for $\mathcal{N}'' = (N_\kappa, M)$ for some marking, then $Mt_x = M'$ such that $\text{strip}(N_\kappa, M') = \hat{\mathcal{N}}'$: there is a transition t_x because all rule matches for derivations of up to κ rule applications are represented as transitions in N_κ; the preset of t_x contains exactly the matched transition $\tau(x) = \text{strip}(\hat{t})$ for some $\hat{t} \in \hat{T}$. This transition is not in $\hat{T}'$.

Instead, the items of the right hand side are present in strip(N_κ, M'), according to the replacement step x. If $x \in \Sigma$, then the step is also simulated in strip$(\mathcal{N})$.

If in N_κ, $Mt_x = M'$, then it is easy to see that the images under strip of (N_κ, M) and (N_κ, M') are related by the replacement step with match μ. If $Mt = M'$ for a non-match transition, this corresponds via strip to a firing of that transition.

The simulation faithfully preserves all events in derivations of length up to κ. □

Let us now turn our attention to a different notion of reachability, namely reachability of abstract markings. Since the state space of a structure-changing Petri net can be infinite, to specify interesting state properties it is insufficient to specify the number of tokens on specific concrete places. Instead, we fix a finite set of colours and state constraints generically, for all places of a given colour. The place colours may carry a model-specific meaning, or just encode structural information about the net. We will therefore introduce a notion of *abstract marking*: a multiset that counts the total number of tokens on places of each colour in a structure-changing Petri net.

A multiset over a finite set S, or S-multiset, is a function $S \to \mathbb{N}$. The set of S-multisets is denoted $\mathbb{N}^S$. The singleton multiset mapping a to 1 and everything else to 0 is also written a. The *size* $|m|$ of a multiset m is the sum of the values. Multiset addition is component-wise. A sum $\sum_{x \in m} f(x)$ over a multiset $m \in \mathbb{N}^S$ is defined with multiplicities, $\sum_{x \in S} m(x) f(x)$. Multisets are compared using the product order, i.e. $m \leq m'$ iff $\forall s \in S,\ m(s) \leq m'(s)$. In marked nets, we redefine ${}^\bullet t$ and $t^\bullet$ to mean the P-multisets $p \mapsto F^\pm(t,p)$ for $\pm = +, -$, respectively. The definition of enabledness canonically extends to multisets of transitions. For any marked net $\mathcal{N} = (N, M)$, we define the *colour abstraction* $\alpha : \mathbb{N} \to \mathbb{N}$ to be the function $i \mapsto \sum_{p \in c^{-1}(i)} M(p)$. For k-coloured nets, we restrict the domain of α to $0, \ldots, k-1$. The set of images under α of the reachable states of $\mathcal{S}$ is denoted by $\mathcal{ARS}(\mathcal{S})$, for *abstract reachability set*.

If $\mathcal{N} = (N, M)$ is a marked net, define a set of multisets over $\{0, \ldots, k-1\} + T$, the *extended reachability set* $\mathcal{ER}(\mathcal{N})$, as $\mathcal{ER}(\mathcal{N}) = \mathcal{ARS}(\mathcal{N}) + \{\alpha(m - \sum_{t \in s} {}^\bullet t) + s \mid m \in \mathcal{RS}(\mathcal{N}),\ s \in \mathbb{N}^T \text{ enabled by } m\}$. The set $\mathcal{ER}(\mathcal{N})$ is finite if $\mathcal{N}$ is a sound workflow net with its start marking. Intuitively, it represents all snapshots of the net's state before, after and *during* possibly concurrent firing events.

Problem 3 (Abstract reachability)**.**

Given:	a k-coloured structure-changing Petri net $\mathcal{S}$ for some $k \in \mathbb{N}$ $q : \{0, \ldots, k-1\} \to \mathbb{N}$
Question:	$q \in \mathcal{RS}(\mathcal{S})$? Is the abstract marking q reachable?

Given a structure-changing Petri net $\mathcal{S} = (\mathcal{N}, \mathcal{R})$, we let $\boldsymbol{\mathcal{A}(\mathcal{S})}$ denote the set that comprises $\mathcal{N}$ and the right hand sides of all rules in $\mathcal{R}$. Let $\mathbb{T}(\mathcal{S}) = \biguplus_{N_w \in \mathcal{A}(\mathcal{S})} T_w$ be the set of all transitions occurring in any right hand side or in the start net. ($\biguplus$ stands for a disjoint union of multiple sets). For the intuition behind the following proposition, we would like to refer to Figure 1 on the following page:

Proposition 3 *The abstract reachability problem is decidable for 1-safe structure-changing workflow nets.*

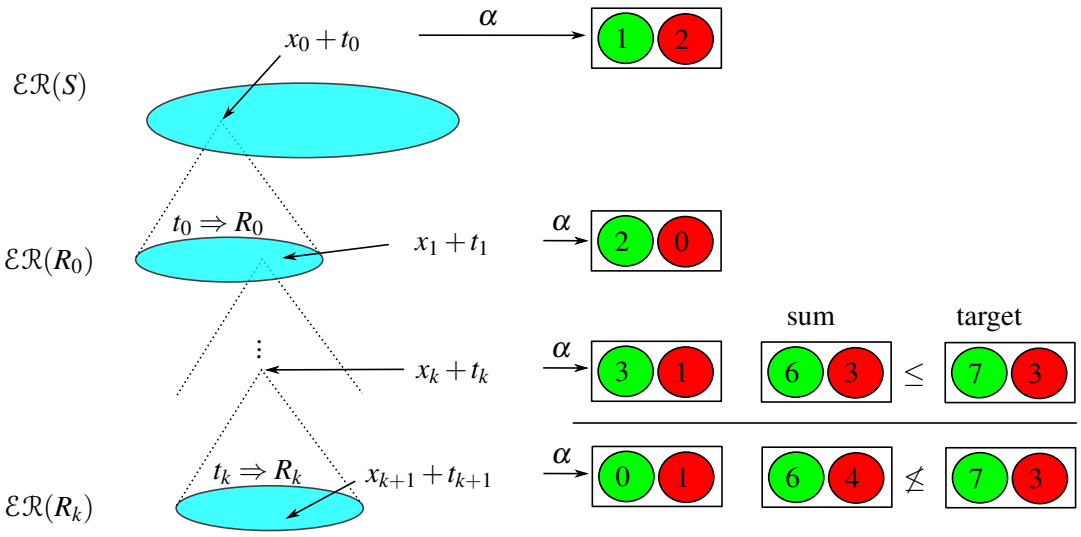


Figure 1: The abstract reachability procedure visualised

Proof. We present an algorithm that, given a structure-changing workflow net $\mathcal{S}$ and $q \in \mathbb{N}^k$, decides whether there is any reachable state $\mathcal{N} \in \mathcal{RS}(\mathcal{S})$ such that $\alpha(\mathcal{N}) = q$. Let $\mathbb{T} = \mathbb{T}(\mathcal{S})$.

1: **procedure** AbstrReach($\mathcal{S}, q$)
2: Queue := $\mathcal{ER}(\mathcal{N})$; Visited := $\emptyset$
3: **while** Queue not empty **do**
4: m := pop(Queue)
5: **if** $m = q$ **then**
6: **return** true
7: **else**
8: **if** not ($|m| > |q|$ or $m|_P \not\leq q$ or $m \in$ Visited) **then**
9: **for all** t, m' such that $m = t + m'$ **do**
10: **for all** $\{\mu \in match_\rho(t) \mid (\rho, N_l, N_r) \in \mathcal{R}\}$ **do**
11: append(Queue, $\{m' + m \mid m \in \mathcal{ER}(N_r, {}^\bullet N_r)\}$)
12: Visited := Visited $\cup \{m\}$
13: **return** false

To explain the algorithm, let us define a multi-rooted directed graph $\mathfrak{W}$, in general infinite. Its nodes are multisets over $\{0, ..., k-1\} + \mathbb{T}$. The elements of $\mathcal{ER}(\mathcal{N})$ are the roots of $\mathfrak{W}$. Each node $t + m$ with $t \in \mathbb{T}$ has successors $x + m$ where $x \in \mathcal{ER}(N_r)$ and N_r is the right hand side of a rule that matches t. Whether a rule can match t can be checked directly from $\mathcal{A}(\mathcal{S})$.

Multiset size increases monotonically along the edges due to the fact that the empty marking cannot be reachable in a sound workflow net. Hence to check for the abstract reachability of q, it suffices to explore a spanning tree of the prefix of $\mathfrak{W}$ induced by the nodes of size not exceeding $|q|$, of which there are finitely many. Branching is finite due to the finite $\mathcal{ER}$ sets of the right hand side nets. By Kőnig's Lemma such a tree is finite, hence the search terminates.

Besides those with at least one transition, $\mathfrak{W}$ contains only colour abstractions of reachable markings: a derivation leading to a suitable marked net can simply be read off a path in $\mathfrak{W}$, because in the case of 1-safe structure-changing workflow net it is immaterial *which* transition is replaced, so the abstraction preserves all information needed to construct a suitable derivation.

Also, $\mathfrak{W}$ contains the colour abstractions of *all* reachable states, by induction on the number of replacement steps in a $\precsim_{\Sigma,R}$-minimal derivation. As long as no rule is applied, the statement clearly holds. Suppose that the property holds for any derivation (ξ,σ) up to a certain length. Any new replacement step $\mathcal{N} \stackrel{\rho}{\Rightarrow} \mathcal{N}'$ using a rule ρ must match an *activated* transition t by the assumption on the derivation. Any state obtained by replacing t and executing a non-terminal sequence of transitions in the right hand side has an abstract marking $m+m''$, where $m+t \in \mathcal{ER}(\mathcal{N})$, and m'' is in the subset of the abstract reachability set of $(\mathcal{N}',\mathcal{R})$ accessible with fewer replacement steps, by induction hypothesis.

It follows that the algorithm is correct since it explores $\mathfrak{W}$ until reaching nodes whose size exceeds $|q|$, and checks whether q is obtained. □

The complexity of the abstract reachability problem is inherited from the corresponding class of static 1-safe nets. For 1-safe nets in general, reachability is PSPACE-complete whereas it is just NP-complete for acyclic 1-safe nets [Esp96].

Proposition 4 *The abstract reachability problem for acyclic 1-safe structure-changing workflow nets is* NP*-complete.*

Proof. It follows from the proof of Proposition 3 that the problem is not only decidable, but in NP: the answer is positive iff there exists a certain easily checked polynomial-length object, namely the path through $\mathfrak{W}$ at most $O(|\mathcal{R}| \cdot |q|)$ nodes of length $O(|q|)$ each. NP-hardness is straightforwardly shown by a reduction from the corresponding problem for acyclic 1-safe nets. The reachability problem of 1-safe nets can be reduced to the reachability problem of 1-safe workflow nets (see Figure 2). This reduction goes as follows: for every place in the 1-safe marked net (N,M_0), a complementary place is added; start end place and $2+3|P|$ extra items are added; the simulation is initialised by filling in the start marking and its complement, and it can be aborted without producing erroneous runs by emptying the net. Reachability of M from M_0 becomes reachability of the corresponding marking from the start marking of the workflow net. The workflow net can be seen to be sound. To preserve acyclicity, one removes the complementary places. The reduction of the reachability problem for 1-safe workflow nets to an abstract reachability problem for the corresponding static 1-safe structure-changing workflow net is trivial: places are coloured bijectively. □

Proposition 5 *The abstract reachability problem for 1-safe structure-changing workflow nets is* PSPACE*-complete.*

Proof. PSPACE-hardness again follows from the corresponding problem for 1-safe nets, the reduction using the same construction presented above. The procedure proposed in the proof of Proposition 3 is also easily seen to be in PSPACE because the upper bound on κ is polynomial in $|q|$ and $|R|$. □

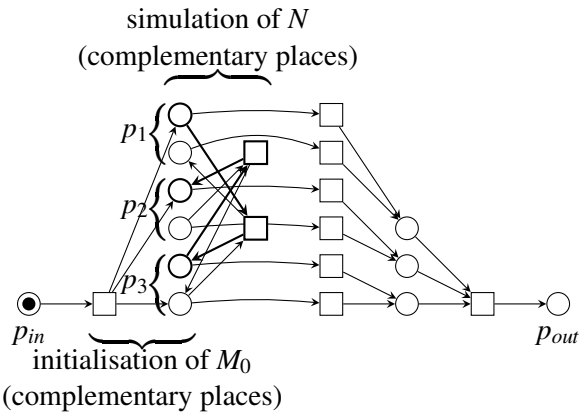


Figure 2: Reduction from 1-safe nets to 1-safe workflow nets.

As a bonus, $\mathfrak{W}$ can be used to decide coverability of an abstract marking q, i.e. the question whether any state $\mathcal{N}$ with $\alpha(\mathcal{N}) \geq q$ can be reached.

Proposition 6 *Coverability of an abstract marking is decidable for 1-safe structure-changing workflow nets.*

Proof. $\mathfrak{W}$ is the reachability graph of a net with place set $\{0,\ldots,k-1\}+\mathbb{T}$ and appropriate transitions that have as preset $t \in \mathbb{T}$ and postset m, for every $m \in \mathcal{ER}(N_r, {}^\bullet N_r)$, where N_r is the right hand side of a rule matching t. Hence coverability is reduced to Petri net coverability, which is decidable [PW02]. □

3.3 Containment Problems

In this subsection, we study the inclusion of the firing language of a structure-changing workflow net in a given regular language. The motivation is that the regular language can specify all desirable net behaviour, and the problem is to check whether any undesirable firing sequences exist or not.

Problem 4 (Containment in Regular Language)**.**

Given:	a regular language $L \subseteq \Sigma^*$ a 1-safe structure-changing Petri net $\mathcal{S}$
Question:	$f(\mathcal{L}(\mathcal{S})) \subseteq L$?

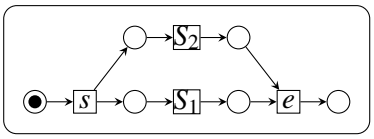
It is well known that the emptiness of the intersection of two context-free languages is undecidable.

Proposition 7 (Undecidability of abstract language compliance) *Containment of a structure-changing workflow net language in a regular language is undecidable even when restricted to acyclic 1-safe structure-changing workflow nets with at most* 2 *tokens in every reachable state.*

Proof. By reduction from the emptiness problem for the intersection of two context-free languages. Let $G_1 = (V_1, T, P_1, S_1)$ and $G_2 = (V_2, T, P_2, S_2)$ be two context-free grammars in Greibach normal form. We assume their non-terminal alphabets disjoint w.l.o.g. ($V_1 \cap V_2 = \emptyset$).

To distinguish the words from the two grammars, we introduce a second terminal alphabet $\hat{T} := \{\hat{a} \mid a \in T\}$ and a function $dup(\varepsilon) := \varepsilon, dup(aw) := a\hat{a} \cdot dup(w)$.

We construct the structure-changing workflow net $\mathcal{S}(G_1, G_2) = (\mathcal{N}, \mathcal{R}_1 \cup \mathcal{R}_2)$. $\mathcal{N}$ is as shown:



For every production $X \to aX_1 \ldots X_n$ in G_1, we add a rule replacing a transition labelled X with a linear sequence of transitions labelled $a, X_1, \ldots, X_n$ to $\mathcal{R}_1$. For productions in G_2, we do the same, but replace the terminal label a with $\hat{a}$ and add the rules to $\mathcal{R}_2$. Disjointness of non-terminal alphabets ensures that the rules in $\mathcal{R}_i$ are only applicable in the subnet resulting from S_i (for $i \in \{1,2\}$). The words accepted in these subnets hence correspond to those generated by the respective grammars. Let L be $s\{a\hat{a} \mid a \in T\}^* e$ with $\{s,e\} \not\subseteq T$. Now,

$$\begin{array}{rcr} \mathcal{L}(G_1) \cap \mathcal{L}(G_2) = \emptyset & \Leftrightarrow & \mathcal{F}(\mathcal{S}(G_1, G_2)) \subseteq \overline{L} \\ \mathcal{L}(G_1) \cap \mathcal{L}(G_2) = \emptyset & \Leftrightarrow & \mathcal{F}(\mathcal{S}(G_1, G_2)) \cap L = \emptyset \\ \exists w \in \mathcal{L}(G_1) \cap \mathcal{L}(G_2) & \Leftrightarrow & \exists w \in \mathcal{F}(\mathcal{S}(G_1, G_2)) \cap L \end{array}$$

$\Rightarrow$: if w is generated from both G_1 and G_2, then a derivation for it exists in both grammars. Since context-free derivations in G_1 and G_2 can be translated into sequences of rule applications in the corresponding subnet, there obviously is a reachable net in $\mathcal{S}(G_1, G_2)$ able to accept $s \cdot dup(w)e \in L$.

$\Leftarrow$: All words in L can be written as $sdup(w)e$ for some $w \in T^*$. To also be accepted by $\mathcal{S}(G_1, G_2)$, there must be a derivation (ξ, σ) with $f(\lambda(\xi)) = s \cdot dup(w)e$. Now, $w \in \mathcal{L}(G_1)$, since $w = f(\lambda(\xi))$ and the subsequence of $\mathcal{R}_1$-steps in ξ directly corresponds to a derivation in G_1. A symmetric argument holds for G_2.

$\mathcal{S}(G_1, G_2)$ is a 1-safe structure-changing workflow net easily seen to have as reachable states only acyclic nets marked with one or two tokens. □

4 Related Work

Petri nets can be extended with structure changes via graph replacement rules, as in [MGH11]. Graph grammars [EEPT06] define replacement steps according to rules that serve to reconfigure the Petri net dynamically and can be mixed with transition firings. Their work, and ours, is closely related to graph grammars and results from graph replacement such as local Church-Rosser and concurrency theorems can be adapted, see [EHP+07, MGH11]. We have concentrated on Petri net specific aspects such as reachability of markings in this paper. [BCKK04] uses similar concepts for model checking graph grammars.

Further noteworthy work includes the box calculus [BDH92], recursive Petri nets [HP99], reconfigurable nets [BLO03], open nets [BCE+07]. These extensions are much more general and

allow structure changes beyond dynamic transition refinement. Safe nets-in-nets [KH10] and Hypernets [Mas11] represent a different kind of dynamic structure. With unbounded nesting of net tokens, it is possible to simulate a 2-counter machine with zero-tests. The notion of activated transitions and transition refinement are also found in [KR07]. Finally, in [vSV03], a notion of refinement for workflow nets was investigated, but with different aims.

5 Conclusion

We have introduced structure-changing Petri nets with context-free transition replacement rules based on graph replacement, and studied a number of decision problems that arose. Suitably translated, our results apply to many formalisms that add structure changes to Petri nets.

An overview of the results follows. The columns stand for structure-changing Petri nets (scpn) and structure-changing workflow nets (general, 1-safe, acyclic 1-safe scwn) respectively, "yes" means decidable, "no" undecidable, "?" unknown):

	scpn	scwn	1-safe	acyclic 1-safe	
concrete reachability	yes	yes	yes	yes	Proposition 2
abstract reachability	?	?	yes	yes	Proposition 3
firing word problem	?	?	yes	yes	Proposition 1
regular specification	no	no	no	no	Proposition 7

The firing word problem turned out to be nontrivial. Proposition 7 places severe limitations on the algorithmic analysis of structure-changing Petri nets. Even for systems with a simple structure and a globally bounded token number, language containment questions are undecidable due to the possibility of imposing synchronisation on concurrent context-free processes.

Further Topics

(1) Structure-changing nets with arbitrary replacement rules hardly seem to offer promising analysis methods. It seems most fruitful to investigate classes of nets that behave sufficiently like the "simple" ones presented here, and on the other hand to apply general analysis methods known from graph replacement.

(2) Decidable problems for relatively simple subclasses of structure-changing Petri nets and case studies to evaluate which features are still lacking in order to model real dynamic workflows. We suspect that not all restrictions are necessary and are working to determine the decidability boundary more accurately.

(3) In the spirit of system correctness under adverse conditions, would be to turn the reachability problems into game problems by considering some nontrivial scheduling between the two kinds of steps, and universal quantification for replacement steps: until now, rule application and firing cooperate.

Acknowledgements: We would like to thank Annegret Habel and several other SCARE members for constructive discussions, and the reviewers for their very constructive remarks.

Bibliography

[BCE+07] P. Baldan, A. Corradini, H. Ehrig, R. Heckel, B. König. Bisimilarity and Behaviour-Preserving Reconfigurations of Open Petri Nets. In *Algebra and Coalgebra in Computer Science*. LNCS 4624, pp. 126–142. 2007.

[BCKK04] P. Baldan, A. Corradini, B. König, B. König. Verifying a Behavioural Logic for Graph Transformation Systems. *ENTCS* 104:5–24, 2004.

[BDH92] E. Best, R. Devillers, J. Hall. The box calculus: A new causal algebra with multi-label communication. In *Advances in Petri Nets*. LNCS 609, pp. 21–69. 1992.

[BLO03] E. Badouel, M. Llorens, J. Oliver. Modeling concurrent systems: Reconfigurable nets. In *Proceedings of Int. Conf. on Parallel and Distributed Processing Techniques and Applications*. CSREA Press, 2003.

[Cor95] A. Corradini. Concurrent Computing: from Petri Nets to Graph Grammars. ENTCS 2, pp. 56–70. Elsevier, 1995.

[DR98] J. Desel, W. Reisig. Place/transition Petri nets. In *Lectures on Petri Nets I: Basic Models*. LNCS 1491, pp. 122–173. 1998.

[EEPT06] H. Ehrig, K. Ehrig, U. Prange, G. Taentzer. *Fundamentals of Algebraic Graph Transformation*. Monographs in Theoretical Computer Science. Springer, 2006.

[EHP+07] H. Ehrig, K. Hoffmann, J. Padberg, U. Prange, C. Ermel. Concurrency in Reconfigurable P/T Systems: Independence of net transformations and token firing in reconfigurable P/T systems. In *Proceedings ICATPN*. Pp. 104–123. 2007.

[Esp96] J. Esparza. Decidability and Complexity of Petri Net Problems - An Introduction. In *Lectures on Petri Nets I: Basic Models*. LNCS 1491, pp. 374–428. 1996.

[Hab92] A. Habel. *Hyperedge Replacement: Grammars and Languages*. LNCS 643. 1992.

[HP99] S. Haddad, D. Poitrenaud. Theoretical Aspects of Recursive Petri Nets. In *Application and Theory of Petri Nets*. LNCS 1639, pp. 228–247. 1999.

[KH10] M. Köhler-Bußmeier, F. Heitmann. Safeness for object nets. *Fundamenta Informaticae* 101(1):29–43, 2010.

[KR07] M. Köhler, H. Rölke. Web Service Orchestration with Super-Dual Object Nets. In *ICATPN*. LNCS 4546, pp. 263–280. 2007.

[Kre81] H.-J. Kreowski. A comparison between Petri-nets and graph grammars. In *Graph-theoretic Concepts in Computer Science*. LNCS 100, pp. 306–317. 1981.

[Mas11] M. Mascheroni. *Hypernets: a class of hierarchical Petri nets*. PhD thesis, Università degli Studi di Milano-Bicocca, 2011.

[MGH11] T. Modica, K. Gabriel, K. Hoffmann. Formalization of Petri Nets with Individual Tokens as Basis for DPO Net Transformations. In *Proceedings PNGT 2010*. Electronic Communications of the EASST 40. 2011.

[PW02] L. Priese, H. Wimmel. *Petri-Netze*. Springer, 2002.

[van97] W. M. P. van der Aalst. Verification of workflow nets. In *Applications and Theory of Petri Nets*. LNCS 1248, pp. 407–426. 1997.

[van01] W. M. P. van der Aalst. Exterminating the Dynamic Change Bug: A Concrete Approach to Support Workflow Change. *Information Sys. Frontiers* 3(3):297–317, 2001.

[vSV03] K. van Hee, N. Sidorova, M. Voorhoeve. Soundness and Separability of Workflow Nets in the Stepwise Refinement Approach. In *Applications and Theory of Petri Nets*. LNCS 2679, pp. 337–356. 2003.

[WRR08] B. Weber, M. Reichert, S. Rinderle-Ma. Change Patterns and Change Support Features - Enhancing Flexibility in Process-Aware Information Systems. *Data and Knowledge Engineering* 66(3):438–466, 2008.