Polynomial Analysis Algorithms for Free Choice Probabilistic Workflow Nets

Javier Esparza[1], Philipp Hoffmann[1(✉)], and Ratul Saha[2]

[1] Technische Universität München, Munich, Germany
esparza@in.tum.de, ph.hoffmann@tum.de
[2] National University of Singapore, Singapore, Singapore
ratul@comp.nus.edu.sg

Abstract. We study Probabilistic Workflow Nets (PWNs), a model extending van der Aalst's workflow nets with probabilities. We give a semantics for PWNs in terms of Markov Decision Processes and introduce a reward model. Using a result by Varacca and Nielsen, we show that the expected reward of a complete execution of the PWN is independent of the scheduler. Extending previous work on reduction of non-probabilistic workflow nets, we present reduction rules that preserve the expected reward. The rules lead to a polynomial-time algorithm in the size of the PWN (not of the Markov decision process) for the computation of the expected reward. In contrast, since the Markov decision process of PWN can be exponentially larger than the PWN itself, all algorithms based on constructing the Markov decision process require exponential time. We report on a sample implementation and its performance on a collection of benchmarks.

1 Introduction

Workflow Petri Nets are a class of Petri nets for the representation and analysis of business processes [1,2,5]. They are a popular formal back-end for different notations like BPMN (Business Process Modeling Notation), EPC (Event-driven Process Chain), or UML Activity Diagrams.

There is recent interest in extending these notations, in particular BPMN, with the concept of cost (see e.g. [16,18,19]). The final goal is the development of tool support for computing the worst-case or the average cost of a business process. A sound foundation for the latter requires to extend Petri nets with probabilities and rewards. Since Petri nets can express complex interplay between nondeterminism and concurrency, the extension is a nontrivial semantic problem which has been studied in detail (see e.g. [3,4,7,21]).

Fortunately, giving a semantics to probabilistic Petri nets is much simpler for *confusion-free* Petri nets [3,21], a class that already captures many control-flow constructs of BPMN. In particular, confusion-free Petri nets strictly contain Workflow Graphs, also called free-choice Workflow Nets [1,9,12,13]. In this

This work was funded by the DFG Project "Negotiations: A Model for Tractable Concurrency".

G. Agha and B. Van Houdt (Eds.): QEST 2016, LNCS 9826, pp. 89–104, 2016.
DOI: 10.1007/978-3-319-43425-4_6

paper we study free choice Workflow Nets extended with rewards and probabilities. Rewards are modeled as real numbers attached to the transitions of the workflow, while, intuitively, probabilities are attached to transitions modeling nondeterministic choices. Our main result is the first polynomial algorithm for computing the expected reward of a workflow.

In order to define expected rewards, we give untimed, probabilistic confusion-free nets a semantics in terms of Markov Decision Processes (MDP), with rewards captured by a reward function. In a nutshell, at each reachable marking the enabled transitions are partitioned into *clusters*. All transitions of a cluster are in conflict, while transitions of different clusters are concurrent. In the MDP semantics, a scheduler selects one of the clusters, while the transition inside this cluster is chosen probabilistically.

In our first contribution, we prove that the expected reward of a confusion-free workflow net is independent of the scheduler resolving the nondeterministic choices, and so we can properly speak of *the* expected reward of a free-choice workflow. The proof relies on a result by Varacca and Nielsen [20] on Mazurkiewicz equivalent schedulers.

Since MDP semantics of concurrent systems captures all possible interleavings of transitions, the MDP of a free-choice workflow can grow exponentially in the size of the net, and so MDP-based algorithms for the expected reward have exponential runtime. In our second contribution we provide a polynomial-time *reduction algorithm* consisting of the repeated application of a set of *reduction rules* that simplify the workflow while preserving its expected reward. Our rules are an extension to the probabilistic case of a set of rules for free-choice Colored Workflow Nets recently presented in [9]. The rules allow one to merge two alternative tasks, summarize or shortcut two consecutive tasks by one, and replace a loop with a probabilistic guard and an exit by a single task. We prove that the rules preserve the expected reward. The proof makes crucial use of the fact that the expected reward is independent of the scheduler.

Finally, as a third contribution we report on a prototype implementation, and on experimental results on a benchmark suite of nearly 1500 free-choice workflows derived from industrial business processes. We compare our algorithm with the different algorithms based on the construction of the MDP implemented in Prism [15].

Due to space limitations, the proofs have been deferred to the extended version [10].

2 Workflow Nets

We recall the definition of a workflow net, and the properties of soundness and 1-safeness.

Definition 1 (Workflow Net [1]). *A* workflow net *is a tuple* $\mathcal{W} = (P, T, F, i, o)$ *where* P *is a finite set of places,* T *is a finite set of transitions* ($P \cap T = \emptyset$), $F \subseteq (P \times T) \cup (T \times P)$ *is a set of arcs,* $i, o \in P$ *are distinguished* initial *and*

final *places such that i has no incoming arcs and o has no outgoing arcsand the graph $(P \cup T, F \cup (o, i))$ is strongly connected.*

We write ${}^\bullet p$ and $p^\bullet$ to denote the input and output transitions of a place p, respectively, and similarly ${}^\bullet t$ and $t^\bullet$ for the input and output places of a transition t. A *marking* M is a function from P to the natural numbers that assigns a number of tokens to each place. A transition t is *enabled* at M if all places of ${}^\bullet t$ contain at least one token in M. An enabled transition may *fire*, removing a token from each place of ${}^\bullet t$ and adding one token to each place of $t^\bullet$. We write $M \xrightarrow{t} M'$ to denote that t is enabled at M and its firing leads to M'. The *initial marking* (*final marking*) of a workflow net, denoted by $\boldsymbol{i}$ ($\boldsymbol{o}$), puts one token on place i (on place o), and no tokens elsewhere. A sequence of transitions $\sigma = t_1\, t_2 \cdots t_n$ is an *occurrence sequence* or *firing sequence* if there are markings $M_1, M_2, \ldots, M_n$ such that $\boldsymbol{i} \xrightarrow{t_1} M_1 \cdots M_{n-1} \xrightarrow{t_n} M_n$. $Fin_{\mathcal{W}}$ is the set of all firing sequences of $\mathcal{W}$ that end in the final marking. A marking is *reachable* if some occurrence sequence ends in that marking.

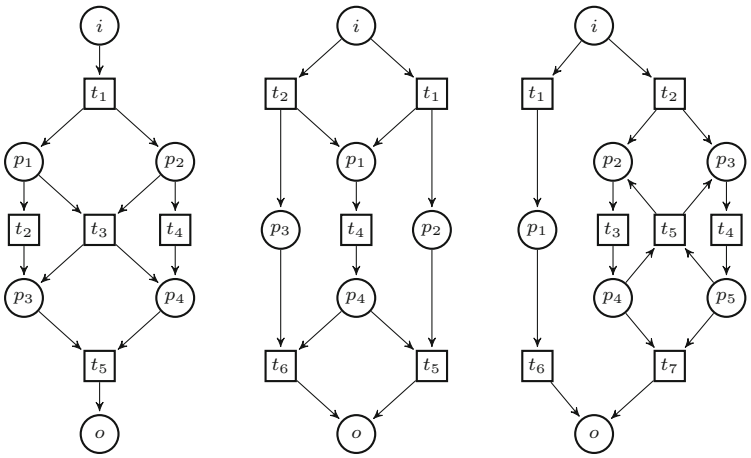


Fig. 1. Three workflow nets

Definition 2 (Soundness and 1-safeness [1]). *A workflow net is* sound *if the final marking is reachable from any reachable marking, and for every transition t there is a reachable marking that enables t. A workflow net is* 1-safe *if $M(p) \leq 1$ for every reachable marking M and for every place p.*

Figure 1 shows three sound and 1-safe workflow nets. In this paper we only consider 1-safe workflow nets, and identify a marking with the set of places that are marked. Markings which only mark a single place are written without brackets and in bold, like the initial marking $\boldsymbol{i}$. In general, deciding if a workflow net is sound and 1-safe is a PSPACE-complete problem. However, for the class of

free-choice workflow nets, introduced below, and for which we obtain our main result, there exists a polynomial algorithm [6].

2.1 Confusion-Free and Free-Choice Workflow Nets

We recall the notions of independent transitions and transitions in conflict.

Definition 3 (Independent Transitions, Conflict). *Two transitions t_1,t_2 of a workflow net are* independent *if ${}^\bullet t_1 \cap {}^\bullet t_2 = \emptyset$. Two transitions are* in conflict *at a marking M if M enables both of them and they are not independent. The set of transitions in conflict with a transition t at a marking M is called the* conflict set *of t at M.*

In Fig. 1 transitions t_2 and t_4 of the left workflow are independent, while t_2 and t_3 are in conflict. The conflict set of t_2 at the marking $\{p_1, p_2\}$ is $\{t_2, t_3\}$, but at the marking $\{p_1, p_4\}$ it is $\{t_2\}$.

It is easy to see that in a 1-safe workflow net two transitions enabled at a marking are either independent or in conflict. Assume that a 1-safe workflow net satisfies that for every reachable marking M, the conflict relation at M is an equivalence relation. Then, at every reachable marking M we can partition the set of enabled transitions into equivalence classes, where transitions in the same class are in conflict and transitions of different classes are independent. Such nets have a simple stochastic semantics: at each reachable marking an equivalence class is selected nondeterministically, and then a transition of the class is selected stochastically with probability proportional to a *weight* attached to the transition. However, not every workflow satisfies this property. For example, the workflow on the left of Figure 1 does not: at the reachable marking marking $\{p_1, p_2\}$ transition t_3 is in conflict with both t_2 and t_4, but t_2 and t_4 are independent. Confusion-free nets, whose probabilistic semantics is studied in [20], are a class of nets in which this kind of situation cannot occur.

Definition 4 (Confusion-Free Workflow Nets). *A marking M of a workflow net is* confused *if there are two independent transitions t_1, t_2 enabled at M such that $M \xrightarrow{t_1} M'$ and the conflict sets of t_2 at M and at M' are different. A 1-safe workflow net is* confusion-free *if no reachable marking is confused.*

The workflows in the middle and on the right of Fig. 1 are confusion-free.

Lemma 1 [20]**.** *Let W be a 1-safe, confusion-free workflow net. For every reachable marking of W the conflict relation on the transitions enabled at M is an equivalence relation.*

Unfortunately, deciding if a 1-safe workflow net is confusion-free is a PSPACE-complete problem (this can be proved by an easy reduction from the reachability problem for 1-safe Petri nets, see [8] for similar proofs). Free-choice workflow nets are a syntactically defined class of confusion-free workflow nets.

Definition 5 (Free-Choice Workflow Nets [1,6]). *A workflow net is* free-choice *if for every two places p_1, p_2 either $p_1^\bullet \cap p_2^\bullet = \emptyset$ or $p_1^\bullet = p_2^\bullet$.*

The workflow in the middle of Fig. 1 is not free-choice, e.g. because of the places p_3 and p_4, but the one on the right is.

It is easy to see that free-choice workflow nets are confusion-free, but even more: in free-choice workflow nets, the conflict set of a transition t is the same at all reachable markings that enable t. To formulate this, we use the notion of a cluster.

Definition 6 (Transition Clusters). *Let $\mathcal{W} = (P, T, F, i, o)$ be a free-choice workflow net. The* cluster *of $t \in T$ is the set of transitions $[t] = \{t' \in T \mid {}^\bullet t \cap {}^\bullet t' \neq \emptyset\}$.*[1]

By the free-choice property, if a marking enables a transition of a cluster, then it enables all of them. We say that the marking enables the cluster; we also say that a cluster fires if one of its transitions fires.

Proposition 1.

- *Let t be a transition of a free-choice workflow net. For every marking that enables t, the conflict set of t at M is the cluster $[t]$.*
- *Free-choice workflow nets are confusion-free.*

3 Probabilistic Workflow Nets

We introduce Probabilistic Workflow Nets, and give them a semantics in terms of Markov Decision Processes. We first recall some basic definitions.

3.1 Markov Decision Processes

For a finite set Q, let $dist(Q)$ denote the set of probability distributions over Q.

Definition 7 (Markov Decision Process). *A* Markov Decision Process *(MDP) is a tuple $\mathcal{M} = (Q, q_0, Steps)$ where Q is a finite set of states, $q_0 \in Q$ is the initial state, and $Steps \colon Q \to 2^{dist(Q)}$ is the probability transition function.*

For a state q, a probabilistic transition corresponds to first nondeterministically choosing a probability distribution $\mu \in Steps(q)$ and then choosing the successor state q' probabilistically according to μ.

A path is a finite or infinite non-empty sequence $\pi = q_0 \xrightarrow{\mu_0} q_1 \xrightarrow{\mu_1} q_2 \ldots$ where $\mu_i \in Steps(q_i)$ for every $i \geq 0$. We denote by $\pi(i)$ the i-th state along π (i.e., the state q_i), and by π^i the prefix of π ending at $\pi(i)$ (if it exists). For a finite path π, we denote by $last(\pi)$ the last state of π. A *scheduler* is a function that maps every finite path π of $\mathcal{M}$ to a distribution of $Steps(last(\pi))$.

For a given scheduler S, let $Paths^S$ denote all infinite paths $\pi = q_0 \xrightarrow{\mu_0} q_1 \xrightarrow{\mu_1} q_2 \ldots$ starting in s_0 and satisfying $\mu_i = S(\pi^i)$ for every $i \geq 0$. We define a probability measure $Prob^S$ on $Paths^S$ in the usual way using cylinder sets [14].

We introduce the notion of rewards for an MDP.

[1] In [6] clusters are defined in a slightly different way.

Definition 8 (Reward). *A* reward function *for an MDP is a function* $rew\colon S \to \mathbb{R}_{\geq 0}$. *For a path* π *and a set of states* F, *the reward* $R(F,\pi)$ *until* F *is reached and the expected reward* $E^S(F)$ *to reach* F *are defined as*

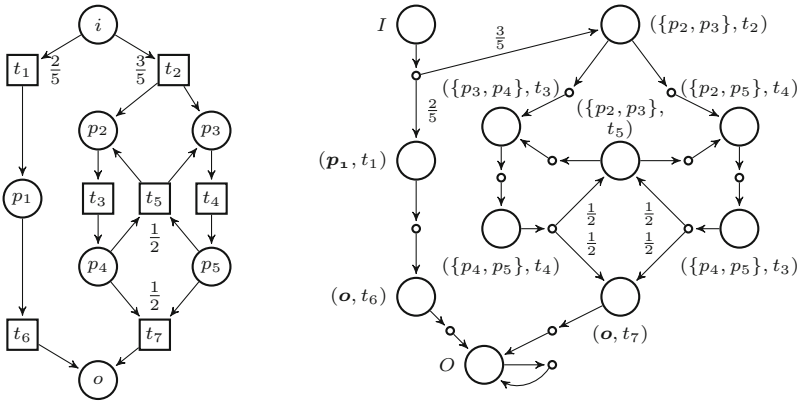
$$R(F,\pi) := \sum_{i=0}^{\min\{j|\pi(j)\in F\}} rew(\pi(i)) \qquad E^S(F) := \int_{\pi\in Paths^S} R(F,\pi)\mathrm{d}Prob^S$$

where $R(F,\pi)$ *is* ∞ *if the minimum does not exist.*

3.2 Syntax and Semantics of Probabilistic Workflow Nets

We introduce Probabilistic Workflow Nets with Rewards, just called Probabilistic Workflow Nets or PWNs in the rest of the paper.

Definition 9 (Probabilistic Workflow Net with Rewards). *A* Probabilistic Workflow Net with Rewards*(PWN) is a tuple* (P,T,F,i,o,w,r) *where* (P,T,F,i,o) *is a 1-safe confusion-free workflow net, and* $w,r\colon T \to \mathbb{R}^+$ *are a* weight function *and a* reward function*, respectively.*



(a) PN with weights

(b) MDP for the example

Fig. 2. Running example

Figure 2a shows a free-choice PWN. All transitions have reward 1, and so only the weights are represented. Unlabeled transitions have weight 1.

The semantics of a PWN is an MDP with a reward function. Intuitively, the states of the MDP are pairs (M,t), where M is a marking, and t is the transition that was fired to reach M (since the same marking can be reached by firing different transitions, the MDP can have states (M,t_1), (M,t_2) for $t_1 \neq t_2$). Additionally there is a distinguished initial and final states I,O. The transition relation *Steps* is independent of the transition t, i.e., $Steps((M,t_1)) = Steps((M,t_2))$ for any two transitions t_1,t_2, and the reward of a state (M,t) is the reward of the transition t. Figure 2b shows the MDP of the PWN of Fig. 2a, representing only the states reachable from the initial state.

Definition 10 (Probability Distribution). *Let $\mathcal{W} = (P, T, F, i, o, w, r)$ be a PWN, let M be a 1-safe marking of $\mathcal{W}$ enabling at least one transition, and let C be a conflict set enabled at M. The* probability distribution $P_{M,C}$ over *T is obtained by normalizing the weights of the transitions in C, and assigning probability 0 to all other transitions.*

Definition 11 (MDP and Reward Function of a PWN). *Let $\mathcal{W} = (P, T, F, i, o, w, r)$ be a PWN. The MDP $M_{\mathcal{W}} = (Q, q_0, Steps)$ of $\mathcal{W}$ is defined as follows:*

- *$Q = (\mathcal{M} \times T) \cup \{I, O\}$ where $\mathcal{M}$ are the 1-safe markings of $\mathcal{W}$, and $q_0 = I$.*
- *For every transition t:*
  - *$Steps((\boldsymbol{o}, t))$ contains exactly one distribution, which assigns probability 1 to state o, and probability 0 to all other states.*
  - *For every marking $M \neq \boldsymbol{o}$ enabling no transitions, $Steps((M, t))$ contains exactly one distribution, which assigns probability 1 to (M, t), and probability 0 to all other states.*
  - *For every marking M enabling at least one transition, $Steps((M, t))$ contains a distribution μ_C for each conflict set C of transitions enabled at M. The distribution μ_C is defined as follows. For the states I, O: $\mu_C(I) = 0 = \mu_C(O)$. For each state (M', t') such that $t' \in C$ and $M \xrightarrow{t'} M'$: $\mu_C((M', t')) = P_{M,C}(t')$. For all other states (M', t'): $\mu_C((M', t')) = 0$.*
  - *$Steps(I) = Steps((\boldsymbol{i}, t))$ for any transition t.*
  - *$Steps(O) = Steps((\boldsymbol{o}, t))$ for any transition t.*

The reward function *$rew_{\mathcal{W}}$ of $\mathcal{W}$ is defined by: $rew_{\mathcal{W}}(I) = 0 = rew_{\mathcal{W}}(O)$, and $rew_{\mathcal{W}}((M, t)) = r(t)$.*

In Fig. 2a, $Steps(i)$ is a singleton set that contains the probability distribution which assigns probability $\frac{2}{5}$ to the state $(\boldsymbol{p_1}, t_1)$ and probability $\frac{3}{5}$ to the state $(\{p_2, p_3\}, t_2)$. $Steps((\{p_2, p_3\}, t_2))$ contains two probability distributions, that assign probability 1 to $(\{p_5, p_3\}, t_4)$ and $(\{p_2, p_6\}, t_4)$, respectively.

We define a correspondence between firing sequences and MDP paths.

Definition 12. *Let $\mathcal{W}$ be a PWN, and let $M_{\mathcal{W}}$ be its associated MDP. Let $\sigma = t_1 t_2 \dots t_n$ be a firing sequence of $\mathcal{W}$. The path $\Pi(\sigma)$ of $M_{\mathcal{W}}$ corresponding to σ is $\pi_\sigma = I \xrightarrow{\mu_0} (M_1, t_1) \xrightarrow{\mu_1} (M_2, t_2) \xrightarrow{\mu_2} \dots$, where $M_0 = \boldsymbol{i}$ and for every $1 \leq k$:*

- *M_k is the marking reached by firing $t_1 \dots t_k$ from $\boldsymbol{i}$, and*
- *μ_k is the unique distribution of $Steps(M_{k-1}, t_{k-1})$ such that $\mu(t_k) > 0$.*

Let $\pi = I \xrightarrow{\mu_0} (M_1, t_1) \cdots (M_n, t_n)$ be a path of $M_{\mathcal{W}}$. The sequence $\Sigma(\pi)$ corresponding to π is $\sigma_\pi = t_1 \dots t_n$.

It follows immediately from the definition of $M_{\mathcal{W}}$ that the functions Π and Σ are inverses of each other. For a path π of the MDP that ends in state $last(\pi)$, the distributions in $Steps(last(\pi))$ are obtained from the conflict sets enabled

after $\Sigma(\pi)$ has fired, if any. If no conflict set is enabled the choice is always trivial by construction. Therefore, a scheduler of the MDP $\mathcal{M}_W$ can be equivalently defined as a function that assigns to each firing sequence $\sigma \in T^*$ one of the conflict sets enabled after σ has fired. In our example, after t_2 fires, the conflict sets $\{t_3\}$ and $\{t_4\}$ are concurrently enabled. A scheduler chooses either $\{t_3\}$ or $\{t_4\}$. A possible scheduler always chooses $\{t_3\}$ every time the marking $\{p_2, p_3\}$ is reached, and produces sequences in which t_3 always occurs before t_4, while others may behave differently.

Convention: In the rest of the paper we define schedulers as functions from firing sequences to conflict sets.

In particular, this definition allows us to define the *probabilistic language* of a scheduler as the function that assigns to each finite firing sequence σ the probability of the cylinder of all paths that "follow" σ. Formally:

Definition 13 (Probabilistic Language of a Scheduler [20]). *The* probabilistic language ν_S *of a scheduler* S *is the function* $\nu_S \colon T^* \to \mathbb{R}^+$ *defined by* $\nu_S(\sigma) = Prob^S(cyl^S(\Pi(\sigma)))$. *A transition sequence* σ *is* produced *by* S *if* $\nu_S(\sigma) > 0$.

The reward function extends to transition sequences in the natural way by taking the sum of all rewards. In pictures, we label transitions with pairs (w, c), where w is a weight and c a reward. See for example Fig. 3a.

We now introduce the expected reward of a PWN under a scheduler.

Definition 14 (Expected Reward of a PWN Under a Scheduler). *Let* $\mathcal{W}$ *be a PWN, and let* S *be a scheduler of its MDP* $M_{\mathcal{W}}$. *The expected reward* $V^S(\mathcal{W})$ *of* $\mathcal{W}$ *under* S *is the expected reward* $E^S(O)$ *to reach the final state* O *of* $M_{\mathcal{W}}$.

Given a firing sequence σ, we have $r(\sigma) = R(O, \Pi(\sigma))$ by the definition of the reward function and the fact that O can only occur at the very end of π_σ.

Lemma 2. *Let* $\mathcal{W}$ *be a sound PWN, and let* S *be a scheduler. Then* $V^S(\mathcal{W})$ *is finite and* $V^S(\mathcal{W}) = \sum_{\pi \in \Pi} R(O, \pi) \cdot Prob^S(cyl^S(\pi)) = \sum_{\sigma \in Fin_{\mathcal{W}}} r(\sigma) \cdot \nu_S(\sigma)$, *where* Π_O *are the paths of the MDP* $M_{\mathcal{W}}$ *leading from the initial state* I *to the state* O *(without looping in* O*).*

3.3 Expected Reward of a PWN

Using a result by Varacca and Nielsen [20], we prove that the expected reward of a PWN is the same for all schedulers, which allows us to speak of "the" expected reward of a PWN. We first define partial schedulers.

Definition 15 (Partial Schedulers). *A* partial scheduler *of length* n *is the restriction of a scheduler to firing sequences of length less than* n. *Given two partial schedulers* S_1, S_2 *of lengths* n_{S_1}, n_{S_2}, *we say that* S_1 extends S_2 *if* $n_{S_1} \geq$

n_{S_2} and S_2 is the restriction of S_1 to firing sequences of length less than n_{S_2}. The probabilistic language *ν_S of a partial scheduler S of length n is the function $\nu_S \colon T^{\leq n} \to \mathbb{R}^+$ defined by $\nu_S(\sigma) = Prob^S(cyl^S(\Pi(\sigma)))$. A transition sequence σ is* produced *by S if $\nu_S(\sigma) > 0$.*

Observe that if σ is not a firing sequence, then $\nu_S(\sigma) = 0$ for every scheduler S. In our running example there are exactly two partial schedulers S_1, S_2 of length 2; after t_2 they choose t_3 or t_4, respectively: $S_1 \colon \epsilon \mapsto \{t_1, t_2\}$ $t_1 \mapsto \{t_6\}$ $t_2 \mapsto \{t_3\}$ and $S_2 \colon \epsilon \mapsto \{t_1, t_2\}$ $t_1 \mapsto \{t_6\}$ $t_2 \mapsto \{t_4\}$. We have $\nu_{S_1}(t_2t_3) = 3/5$, and $\nu_{S_2}(t_2t_3) = 0$.

For finite transition sequences, Mazurkiewicz equivalence, denoted by $\equiv$, is the smallest congruence such that $\sigma t_1 t_2 \sigma' \equiv \sigma t_2 t_1 \sigma'$ for every $\sigma, \sigma' \in T^*$ and for any two *independent* transitions t_1, t_2 [17]. We extend Mazurkiewicz equivalence to partial schedulers.

Definition 16 (Mazurkiewicz Equivalence of Partial Schedulers). *Given a partial scheduler S of length n, we denote by F_S the set of firing sequences σ of $\mathcal{W}$ produced by S such that either $|\sigma| = n$ or σ leads to a marking that enables no transitions.*

Two partial schedulers S_1, S_2 with probabilistic languages ν_{S_1} and ν_{S_2} are Mazurkiewicz equivalent*, denoted $S_1 \equiv S_2$, if they have the same length and there is a bijection $\phi \colon F_{S_1} \to F_{S_2}$ such that $\sigma \equiv \phi(\sigma)$ and $\nu_{S_1}(\sigma) = \nu_{S_2}(\phi(\sigma))$ for every $\sigma \in F_n$.*

The two partial schedulers of our running example are not Mazurkiewicz equivalent. Indeed, we have $F_{S_1} = \{t_1t_6, t_2t_3\}$ and $F_{S_2} = \{t_1t_6, t_2t_4\}$, and no bijection satisfies $\sigma \equiv \phi(\sigma)$ for every $\sigma \in F_{S_1}$. We can now present the main result of [20], in our terminology and for PWNs.[2]

Theorem 1 (Equivalent Extension of Schedulers [20][3]). *Let S_1, S_2 be two partial schedulers. There exist two partial schedulers S_1', S_2' such that S_1' extends S_1, S_2' extends S_2 and $S_1' \equiv S_2'$.*

In our example, S_1 can be extended to S_1' by adding $t_1t_6 \mapsto \emptyset$ and $t_2t_3 \mapsto t_4$, and S_2 to S_2' by adding $t_1t_6 \mapsto \emptyset$ and $t_2t_4 \mapsto t_3$. Now we have $F_{S_1'} = \{t_1t_6, t_2t_3t_4\}$ and $F_{S_2'} = \{t_1t_6, t_2t_4t_3\}$. The obvious bijection shows $S_1' \equiv S_2'$, because we have $t_2t_3t_4 \equiv t_2t_4t_3$ and $\nu_{S_1'}(t_2t_3t_4) = 3/5 = \nu_{S_2}(t_2t_4t_3)$.

Using Theorem 1, we are able to prove one of our central theorems.

Theorem 2. *Let $\mathcal{W}$ be a PWN. There exists a value v such that for every scheduler S of $M_{\mathcal{W}}$, the expected reward $V^S(\mathcal{W})$ is equal to v.*

[2] In [20], enabled conflict sets are called actions, and markings are called cases.

[3] Stated as Theorem 2, the original paper gives this theorem with S_1' and S_2' being (non-partial) schedulers. However, in the paper equivalence is only defined for partial schedulers and the schedulers constructed in the proof are also partial.

Proof Sketch. Given two schedulers, we construct a bijection between the transition sequences they produce that end in the final marking. This bijection maps each transition sequence to a Mazurkiewicz equivalent one. To do this, for each $k \geq 0$ we reduce the schedulers to partial schedulers of length k, extend them to equivalent schedulers using Theorem 1, and map every sequence of length k one of them produces to a Mazurkiewicz equivalent one of the other. Since equivalent transition sequences have the same reward, applying Lemma 2 yields that the values of the two schedulers are equal.

Free-choice PWNs. By Proposition 1, in free-choice PWNs the conflict set of a given transition is its cluster, and so its probability is the same at any reachable marking enabling it. We label a transition directly with this probability.

Convention: We assume that the weights of the transitions of a cluste are normalized, i.e. the weights are already a probability distribution.

In the next section we present a reduction algorithm that decides if a given free-choice PWN is sound or not, and if sound computes its expected reward. If the PWN is unsound, then we just apply the following lemma:

Lemma 3. *The expected reward of an unsound free-choice PWN is infinite.*

4 Reduction Rules

We transform the reduction rules of [9] for non-probabilistic (colored) workflow nets into rules for probabilistic workflow nets.

Definition 17 (Rules, Correctness, and Completeness). *A* rule *R is a binary relation on the set of PWNs. We write $\mathcal{W}_1 \xrightarrow{R} \mathcal{W}_2$ for $(\mathcal{W}_1, \mathcal{W}_2) \in R$.*

A rule R is correct *if $\mathcal{W}_1 \xrightarrow{R} \mathcal{W}_2$ implies that $\mathcal{W}_1$ and $\mathcal{W}_2$ are either both sound or both unsound, and have the same expected reward.*

A set $\mathcal{R}$ of rules is complete *for a class of PWNs if for every sound PWN $\mathcal{W}$ in that class there exists a sequence $\mathcal{W} \xrightarrow{R_1} \mathcal{W}_1 \cdots \xrightarrow{R_n} \mathcal{W}'$ such that $\mathcal{W}'$ is a PWN consisting of a single transition t between the two only places i and o.*

As in [9], we describe rules as pairs of a *guard* and an *action*. $\mathcal{W}_1 \xrightarrow{R} \mathcal{W}_2$ holds if $\mathcal{W}_1$ satisfies the guard, and $\mathcal{W}_2$ is a possible result of applying the action to $\mathcal{W}_1$.

Merge Rule. The *merge rule* merges two transitions with the same input and output places into one single transition. The weight of the new transition is the sum of the old weights, and the reward is the weighted average of the reward of the two merged transitions.

Guard: $\mathcal{W}$ contains two transitions $t_1 \neq t_2$ such that ${}^\bullet t_1 = {}^\bullet t_2$ and $t_1^\bullet = t_2^\bullet$.

Action: (1) $T := (T \setminus \{t_1, t_2\}) \cup \{t_m\}$, where t_m is a fresh name.
(2) $t_m^\bullet := t_1^\bullet$ and ${}^\bullet t_m := {}^\bullet t_1$.
(3) $r(t_m) := w(t_1) \cdot r(t_1) + w(t_2) \cdot r(t_2)$.
(4) $w(t_m) = w(t_1) + w(t_2)$.

Iteration Rule. Loosely speaking, the iteration rule removes arbitrary iterations of a transition by adjusting the weights of the possible successor transitions. The probabilities are normalized again and the reward of each successor transition increases by a geometric series dependent on the reward and weight of the removed transition.

Guard: $\mathcal{W}$ contains a cluster c with a transition $t \in c$ such that $t^\bullet = {}^\bullet t$.

Action: (1) $T := (T \setminus \{t\})$.
(2) For all $t' \in c \setminus \{t\}$: $r(t') := \frac{w(t)}{1-w(t)} \cdot r(t) + r(t')$
(3) For all $t' \in c \setminus \{t\}$: $w(t') := \frac{w(t')}{1-w(t)}$

Observe that $\frac{w(t)}{1-w(t)} \cdot r(t) = (1 - w(t)) \cdot \sum_{i=0}^{\infty} w(t)^i \cdot i \cdot r(t)$ captures the fact that t can be executed arbitrarily often, each execution yields the reward $r(t)$, and eventually some other transition occurs. For an example of an application of the iteration rule, consult Fig. 3b and c. Transition t_9 has been removed and as a result the label of transition t_7 changed.

Shortcut rule. The shortcut rule merges transitions of two clusters into one single transition with the same effect. The reward of the new transition is the sum of the rewards of the old transitions, and its weight the product of the old weights.

A transition t *unconditionally enables* a cluster c if ${}^\bullet t' \subseteq t^\bullet$ for some transition $t' \in c$. Observe that if t unconditionally enables c then any marking reached by firing t enables every transition in c.

Guard: $\mathcal{W}$ contains a transition t and a cluster $c \neq [t]$ such that t unconditionally enables c.

Action: (1) $T := (T \setminus \{t\}) \cup \{t'_s \mid t' \in c\}$, where t'_s are fresh names.
(2) For all $t' \in c$: ${}^\bullet t'_s := {}^\bullet t$ and $t'^\bullet_s := (t^\bullet \setminus {}^\bullet t') \cup t'^\bullet$.
(3) For all $t' \in c$: $r(t'_s) := r(t) + r(t')$.
(4) For all $t' \in c$: $w(t'_s) = w(t) \cdot w(t')$.
(5) If ${}^\bullet p = \emptyset$ for all $p \in c$, then remove c from $\mathcal{W}$.

For an example shortcut rule application, compare the example of Fig. 2a with the net in Fig. 3a. The transition t_1 which unconditionally enabled the cluster $[t_6]$ has been shortcut, a new transition t_8 has been created, and t_1, p_1 and t_6 have been removed.

Theorem 3. *The merge, shortcut and iteration rules are correct for PWNs.*

Proof. That the rules preserve soundness is shown in [9]. To show that the rules preserve the expected reward we use Theorem 2: For each rule, we carefully choose schedulers for the PWNs before and after the application of the rule, and show that their expected rewards are equal. We sketch the idea for the shortcut rule. Let $\mathcal{W}_1$, $\mathcal{W}_2$ be such that $\mathcal{W}_1 \xrightarrow{\text{shortcut}} \mathcal{W}_2$. Let c, t be as in Definition 4. Let S_1 be a scheduler for W_1 such that $S_1(\sigma_1) = c$ if σ_1 ends with t. We define a bijection ϕ that maps firing sequences in $\mathcal{W}_2$ to firing sequences in $\mathcal{W}_1$ by replacing every occurrence of t'_s by $t\,t'$. We define a scheduler S_2 for $\mathcal{W}_2$ by $S_2(\sigma_2) = S_1(\phi(\sigma_2))$. Let now σ_2 be a firing sequence in $\mathcal{W}_2$ and let $\sigma_1 = \phi(\sigma_2)$.

We prove that σ_1 and σ_2 have the same reward and also $\nu_{S_1}(\sigma_1) = \nu_{S_2}(\sigma_2)$. Indeed, since the only difference is that every occurrence of t'_s is replaced by $t\,t'$ and $r(t'_S) = r(t) + r(t')$ and $w(t'_s) = w(t)w(t')$ by the definition of the shortcut rule, the reward must be equal and $\nu_{S_1}(\sigma_1) = \nu_{S_2}(\sigma_2)$. We now use these equalities, the fact that there is a bijection between firing sequences that end with the final marking, and Lemma 2:

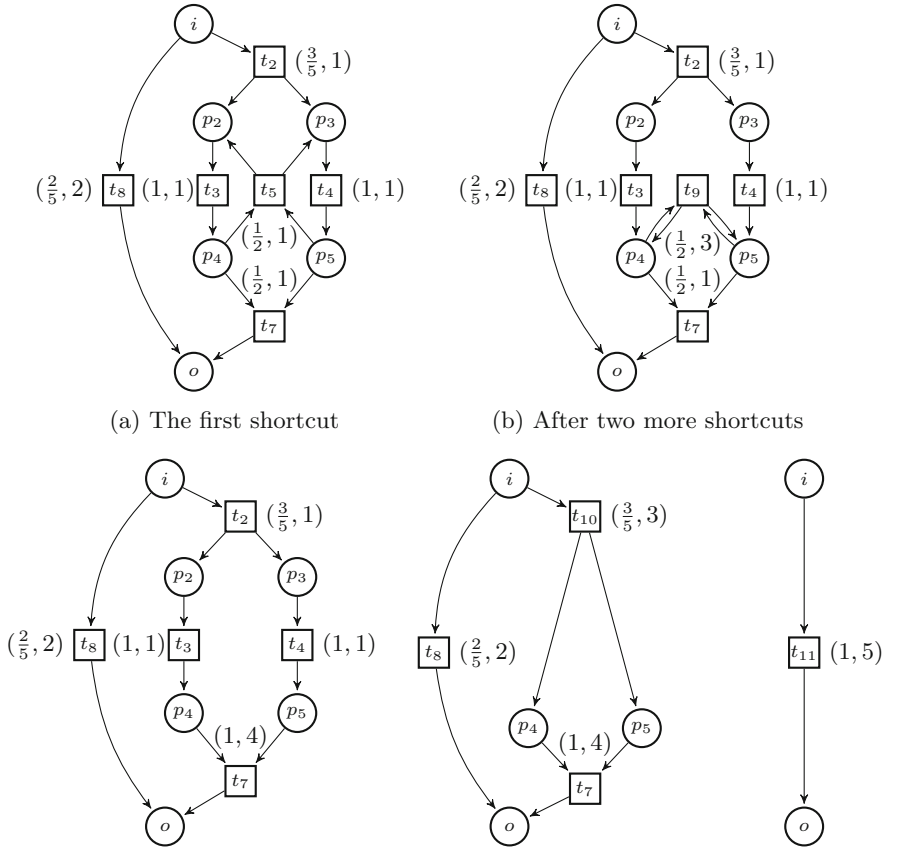
$$V(\mathcal{W}_2) = \sum_{\sigma_2 \in Fin_{\mathcal{W}_2}} r(\sigma_2) \cdot \nu_{S_2}(\sigma) = \sum_{\sigma_2 \in Fin_{\mathcal{W}_2}} r(\phi(\sigma_2)) \cdot \nu_{S_1}(\phi(\sigma_2))$$
$$= \sum_{\sigma_1 \in Fin_{\mathcal{W}_1}} r(\sigma_1) \cdot \nu_{S_1}(\sigma_1) = V(\mathcal{W}_1). \qquad \square$$

In [9] we provide a reduction algorithm for non-probabilistic free-choice workflow, and prove the following result.

Theorem 4 (Completeness [9]). *The reduction algorithm summarizes every sound free choice workflow net in at most $\mathcal{O}(|C|^4 \cdot |T|)$ applications of the shortcut rule and $\mathcal{O}(|C|^4 + |C|^2 \cdot |T|)$ applications of the merge and iteration rules, where C is the set of clusters of the net. Any unsound free-choice workflow nets can be recognized as unsound in the same number of rule applications.*

We illustrate a complete reduction by reducing the example of Fig. 2a. We set the reward for each transition to 1, so the expected reward of the net is the expected number of transition firings until the final marking is reached. Initially, t_1 unconditionally enables $[t_6]$ and we apply the shortcut rule. Since $[t_6] = \{t_6\}$, exactly one new transition t_8 is created. Furthermore t_1, p_1 and t_6 are removed (Fig. 3a). Now, t_5 unconditionally enables $[t_3]$ and $[t_4]$. We apply the shortcut rule twice and call the result t_9 (Fig. 3b). Transition t_9 now satisfies the guard of the iteration rule and can be removed, changing the label of t_7 (Fig. 3c). Since t_2 unconditionally enables $[t_3]$ and $[t_4]$, we apply the shortcut rule twice and call the result t_{10} (Fig. 3d). After short-cutting t_{10}, we apply the merge rule to the two remaining transitions, which yields a net with one single transition labeled by $(1, 5)$ (Fig. 3e). So the net terminates with probability 1 after firing 5 transitions in average.

Fixing a Scheduler. Since the expected reward of a PWN $\mathcal{W}$ is independent of the scheduler, we can fix a scheduler S and compute the expected reward $V^S(\mathcal{W})$. This requires to compute only the Markov chain induced by S, which can be much smaller than the MDP. However, it is easy to see that this idea does not lead to a polynomial algorithm. Consider the free-choice PWN of Fig. 4, and the scheduler that always chooses the largest enabled cluster according to the order $\{t_{11}, t_{12}\} > \cdots > \{t_{n1}, t_{n2}\} > \{u_{11}\} > \{u_{12}\} > \cdots > \{u_{n1}\} > \{u_{n2}\}$. Then for every subset $K \subset \{1, \ldots, n\}$ the Markov chain contains a state enabling $\{u_{i1} \mid i \in K\} \cup \{u_{i2} \mid i \notin K\}$, and has therefore exponential size. There might be a procedure to find a suitable scheduler for a given PWN such that the Markov chain has polynomial size, but we do not know of such a procedure.



(a) The first shortcut (b) After two more shortcuts

(c) After iteration rule (d) After two more shortcuts (e) Final net

Fig. 3. Example of reduction

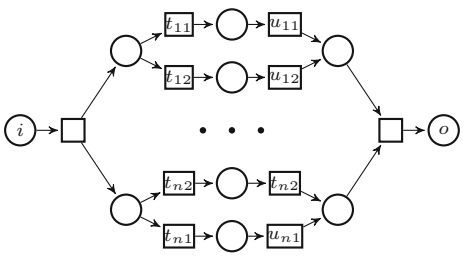


Fig. 4. Example

5 Experimental Evaluation

We have implemented our reduction algorithm as an extension of the algorithm described in [9]. In this section we report on its performance and on a comparison with PRISM [15].

Industrial benchmarks. The benchmark suite consists of 1385 free-choice workflow nets, previously studied in [11], of which 470 nets are sound. The workflows correspond to business models designed at IBM. Since they do not contain probabilistic information, we assigned to each transition t the probability $1 / |[t]|$ (i.e., the probability is distributed uniformly among the transitions of a cluster). We study the following questions, which can be answered by both our algorithm and PRISM: Is the probability to reach the final marking equal to one (equivalent to "is the net sound?"). And if so, how many transitions must be fired in average to reach the final marking? (This corresponds to a reward function assigning reward 1 to each transition.)

All experiments were carried out on an i7-3820 CPU using 1 GB of memory.

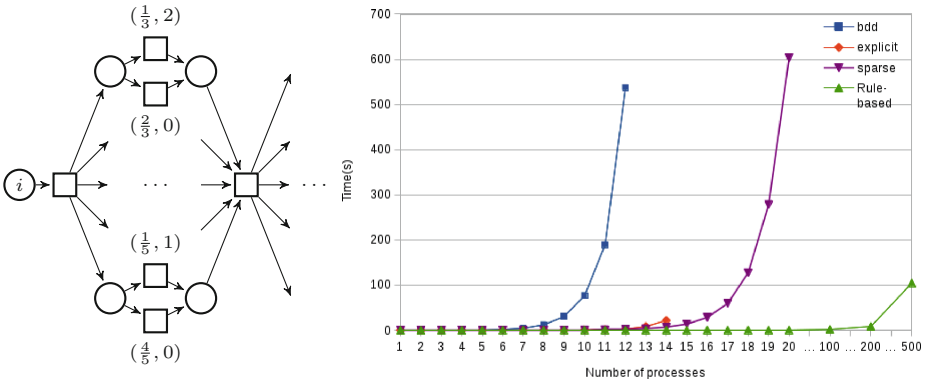
PRISM has three different analysis engines able to compute expected rewards: explicit, sparse and symbolic (bdd). In a preliminary experiment with a timeout of 30 s, we observed that the explicit engine clearly outperforms the other two: It solved 1309 cases, while the bdd and sparse engines only solved 636 and 638 cases, respectively. Moreover, 418 and 423 of the unsolved cases were due to memory overflow, so even with a larger timeout the explicit engine is still leading. For this reason, in the comparison we only used the explicit engine.

After increasing the timeout to 10 min, the explicit engine did not solve any further case, leaving 76 cases unsolved. This was due to the large state space of the nets: 69 out of the 76 have over 10^6 reachable states.

The 1309 cases were solved by the explicit engine in 353 s, with about 10 s for the larger nets. Our implementation solved all 1385 cases in 5 s combined. It never needs more than 20 ms for a single net, even for those with more than 10^7 states.

In the unsound case, our implementation still reduces the reachable state space, which makes it easier to apply state exploration tools. After reduction, the 69 nets with at least 10^6 states had an average of 5950 states, with the largest at 313443 reachable states.

An Academic Benchmark. Many workflows in our suite have a large state space because of fragments modeling the following situation. Multiple processes do a computation step in parallel, after which they synchronize. Process i may execute its step normally with probability p_i, or a failure may occur with probability $1-p_i$, which requires to take a recovery action and therefore has a higher cost. Such a scenario is modeled by the free-choice PWNs net of Fig. 5a, where the probabilities and costs are chosen at random. The scenario can also be easily modeled in PRISM. Figure 5b shows the time needed by the three PRISM engines and by our implementation for computing the expected reward using a time limit of 10 min. The number of reachable states grows exponentially in the number processes, and the explicit engine runs out of memory for 15 processes, the symbolic engine times



(a) PWN (b) Runtimes for the academic benchmark

Fig. 5. Academic benchmark

out for 13 processes, and the sparse engine reaches the time limit at 20 processes. However, since the rule-based approach does not need to construct the state space, we can easily solve the problem with up to 500 processes.

6 Conclusion

We have presented a set of reduction rules for PWNs with rewards that preserve soundness and the expected reward of the net, and are complete for free-choice PWNs. While the semantics and the expected reward are defined via an associated MDP, our rules work directly on the workflow net. The rules lead to the first polynomial-time algorithm to compute the expected reward.

In future work we want to generalize our algorithm to compute the probability of non-termination and the conditional expected reward under termination, which is of interest in the unsound case, and also to compute the expected time to termination for timed workflow nets.

Acknowledgments. We thank the anonymous referees for their comments, and especially the one who helped us correct a mistake in Lemmas 2 and 3.

References

1. Van der Aalst, W.: The application of Petri nets to workflow management. J. Circuits Syst. Comput. **8**(1), 21–66 (1998)
2. Van der Aalst, W., Van Hee, K.M.: Workflow Management: Models, Methods, and Systems. MIT press, Cambridge (2004)
3. Abbes, S., Benveniste, A.: True-concurrency probabilistic models: Markov nets and a law of large numbers. Theoret. Comput. Sci. **390**(2–3), 129–170 (2008)
4. Abbes, S., Benveniste, A.: Concurrency, σ-algebras, and probabilistic fairness. In: de Alfaro, L. (ed.) FOSSACS 2009. LNCS, vol. 5504, pp. 380–394. Springer, Heidelberg (2009)

5. Desel, J., Erwin, T.: Modeling, simulation and analysis of business processes. In: Aalst, W.M.P., Desel, J., Oberweis, A. (eds.) Business Process Management: Models, Techniques, and Empirical Studies. LNCS, vol. 1806, pp. 129–141. Springer, Heidelberg (2000)
6. Desel, J., Esparza, J.: Free Choice Petri Nets, vol. 40. Cambridge University Press, Cambridge (2005)
7. Eisentraut, C., Hermanns, H., Katoen, J.-P., Zhang, L.: A semantics for every GSPN. In: Colom, J.-M., Desel, J. (eds.) PETRI NETS 2013. LNCS, vol. 7927, pp. 90–109. Springer, Heidelberg (2013)
8. Esparza, J.: Decidability and complexity of Petri net problems – an introduction. In: Reisig, W., Rozenberg, G. (eds.) APN 1998. LNCS, vol. 1491, pp. 374–428. Springer, Heidelberg (1998)
9. Esparza, J., Hoffmann, P.: Reduction rules for colored workflow nets. In: Stevens, P., et al. (eds.) FASE 2016. LNCS, vol. 9633, pp. 342–358. Springer, Heidelberg (2016). doi:10.1007/978-3-662-49665-7_20
10. Eszarza, J., Hoffmann, P., Saha, R.: Polynomial analysis algorithms for free choice probabilistic workflow nets (2016). arXiv:1606.00175 [cs.LO]
11. Fahland, D., Favre, C., Jobstmann, B., Koehler, J., Lohmann, N., Völzer, H., Wolf, K.: Instantaneous soundness checking of industrial business process models. In: Dayal, U., Eder, J., Koehler, J., Reijers, H.A. (eds.) BPM 2009. LNCS, vol. 5701, pp. 278–293. Springer, Heidelberg (2009)
12. Favre, C., Fahland, D., Völzer, H.: The relationship between workflow graphs and free-choice workflow nets. Inf. Syst. **47**, 197–219 (2015)
13. Favre, C., Völzer, H., Müller, P.: Diagnostic information for control-flow analysis of workflow graphs (a.k.a. Free-Choice workflow nets). In: Chechik, M., Raskin, J.-F. (eds.) TACAS 2016. LNCS, vol. 9636, pp. 463–479. Springer, Heidelberg (2016). doi:10.1007/978-3-662-49674-9_27
14. Kemeny, J.G., Snell, J.L., Knapp, A.W.: Denumerable Markov Chains: With a Chapter of Markov Random Fields by David Griffeath. GTM, vol. 40. Springer Science & Business Media, New York (2012)
15. Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: verification of probabilistic real-time systems. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 585–591. Springer, Heidelberg (2011)
16. Magnani, M., Cucci, F.: BPMN: how much does it cost? An incremental approach. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 80–87. Springer, Heidelberg (2007)
17. Mazurkiewicz, A.: Trace theory. In: Brauer, W., Reisig, W., Rozenberg, G. (eds.) APN 1986. LNCS, vol. 255, pp. 278–324. Springer, Heidelberg (1987)
18. Saeedi, K., Zhao, L., Sampaio, P.R.F.: Extending BPMN for supporting customer-facing service quality requirements. In: ICWS 2010, pp. 616–623. IEEE Computer Society (2010)
19. Sampath, P., Wirsing, M.: Evaluation of cost based best practices in business processes. In: Halpin, T., Nurcan, S., Krogstie, J., Soffer, P., Proper, E., Schmidt, R., Bider, I. (eds.) BPMDS 2011 and EMMSAD 2011. LNBIP, vol. 81, pp. 61–74. Springer, Heidelberg (2011)
20. Varacca, D., Nielsen, M.: Probabilistic Petri nets and Mazurkiewicz equivalence. Unpublished Manuscript (2003). http://www.lacl.fr/~dvaracca/works.html. Accessed 27 May 2016
21. Varacca, D., Völzer, H., Winskel, G.: Probabilistic event structures and domains. Theoret. Comput. Sci. **358**(2–3), 173–199 (2006)