A Behavioral Similarity Measure between Labeled Petri Nets Based on Principal Transition Sequences

Jianmin Wang[1,2,3], Tengfei He[1,2], Lijie Wen[1,2,3], Nianhua Wu[1,2], Arthur H.M. ter Hofstede[4,5⋆], and Jianwen Su[6]

[1] School of Software, Tsinghua University, 100084, Beijing, China
[2] Key Laboratory for Information System Security, Ministry of Education
[3] Tsinghua National Laboratory for Information Science and Technology, China
jimwang@tsinghua.edu.cn,{htf08,wenlj00,chp04}@mails.thu.edu.cn
[4] Queensland University of Technology, Brisbane, Australia
a.terhofstede@qut.edu.au
[5] Eindhoven University of Technology, Eindhoven, The Netherlands
[6] University of California, Santa Barbara, CA 93106-5110, U.S.A.
su@cs.ucsb.edu

Abstract. Being able to determine the degree of similarity between process models is important for management, reuse, and analysis of business process models. While various approaches to measuring process model similarity have been proposed in the literature, these approaches tend to use structural aspects of a process model rather than their behavior. In this paper we propose a novel method to determine the degree of similarity between process models, which exploits their semantics. Our approach is designed for labeled Petri nets as these can be seen as a foundational theory for process modeling. As the set of traces of a labeled Petri net may be infinite, the challenge is to find a way to represent behavioral characteristics of a net in a finite manner. Therefore, the proposed similarity measure is based on the notion of so-called "principal transition sequences", which aim to provide an approximation of the essence of a process model. This paper defines a novel similarity measure, proposes a method to compute it, and demonstrates that it offers certain benefits with respect to the state-of-the-art in this field.

1 Introduction

Business process models form an essential part of many of today's enterprises. They represent valuable intellectual property and they may e.g. be used for communicating business practices (both for internal and external purposes), for business process improvement, and as a basis for workflow automation. Therefore it is not surprising that business process model collections can become quite large. In fact, in [19] it is pointed out that for multinational corporations such

⋆ Senior Visiting Scholar of Tsinghua University.

collections may contain thousands of process models. These models are created in areas such as enterprise architecture, product lifecycle management, HR capacity planning, project management, knowledge management, web service composition etc. Models are often not created from scratch, they need to comply with existing policies, and duplication of models, or parts of models, is to be avoided. Therefore being able to find similar process models effectively is useful, if not essential. The potential size of business process model repositories mandates automated support for such similarity searches. The effectiveness of this support hinges on the chosen *similarity measure* for business process model comparisons.

The concept of business process model similarity has attracted attention in the recent BPM literature but, more broadly, similarity measures have applications in other areas as well, for example (see [7]) in the area of web services for the purposes of matchmaking, mining and clustering in web service directories. At the same time, this problem also holds interest for the data management community due to their increased interest in process data [21].

Business process similarity measures can exploit the tasks in a process model (through their labels), the dependencies between these tasks (by considering the control-flow relationships specified) as well as the semantics of a process model (by taking the dynamic behavior of these control-flow dependencies into account). Recent efforts in the BPM community have tended to focus on similarity measures based on task labels or on relationships between tasks as specified in the process model (or both). This has meant that algorithms from the fields of information retrieval and graph theory could be exploited. However, such approaches may not adequately take the behavior of a process model into account, as it is pointed out in [2] that two processes may look quite similar, considering the tasks labels and the process structure, but may behave quite differently.

In this paper focus is therefore on a similarity measure that is based on the behavior of process models. As will be explained, our approach differs from other approaches reported in the literature [2, 4, 22, 10], that also take process behavior into account. In order to concretise our approach we have chosen labeled Petri nets as the process modeling formalism. These provide an accepted foundation for business process modeling and analysis [1, 15]. From a fundamental perspective, behavioral similarity of labeled Petri nets can be measured through the use of the concept of bisimilarity, by considering the languages generated by the nets, or by looking at their reachability graphs. According to the information-theoretic definition of similarity [14], similarity (or distance) between processes can be directly determined through the use of the complete set of firing sequences. For a Petri net with loops this set may not be finite however, and an approach based on the use of all firing sequences therefore does not provide a good basis for determining behavioral similarity. Consequently, in our approach we introduce the notion of "principal transition sequence", to provide a finite approximation of the essence of the behavior of a process model. This notion is then used to define a measure for business process model similarity. Therefore the key contributions of this paper are the definition of a similarity measure based on process behavior, an approach to computing this measure, and an exploration of its significance.

The remainder of this paper is organized as follows. Section 2 defines the basic concepts used throughout this paper. Section 3 introduces the notion of principal transition sequence, and Section 4 presents the proposed similarity measure that is based on such sequences, and Section 5 is concerned with its evaluation. Related work is discussed in Section 6, while Section 7 concludes the paper and discusses potential future work.

2 Preliminaries

This section defines fundamental, and well-known, notions of Petri nets used throughout the paper (see e.g. [17, 23]).

Definition 1 (Petri net). *A Petri net is a quadruple $\Sigma = (S, T, F, M_0)$ where S is a finite set of places, T is a finite set of transitions such that $S \cap T = \varnothing$, $F \subseteq (S \times T) \cup (T \times S)$ is a set of arcs, and $M_0 : S \to \mathbb{N}$ (the set of natural numbers) is the initial marking.*

We may refer to the structure of a Petri net $\Sigma = (S, T, F, M_0)$ without the initial marking as $N = (S, T, F)$, and we may also refer to a Petri net as a pair (N, M_0) where N is the net structure and M_0 the initial marking.

Example 1. For Petri net Σ_1 in Fig. 1, $S = \{p_0, p_1, p_2, p_3, p_4, p_5\}$, $T = \{t_0, t_1, t_2, t_3, t_4, t_5\}$, $F = \{(p_0, t_0), (t_0, p_1), (p_1, t_1), (p_1, t_2), (t_1, p_2), (t_2, p_3), (p_2, t_3), (p_3, t_4), (t_3, p_4), (t_4, p_4), (p_4, t_5), (t_5, p_5)\}$, and $M_0 = \{(p_0, 1), (p_1, 0), (p_2, 0), (p_3, 0), (p_4, 0), (p_5, 0)\}$. In order to shorten the denotation of markings and to facilitate their use in algebraic operations we will write markings as vectors, e.g. M_0 can be written as $(1, 0, 0, 0, 0, 0)$.

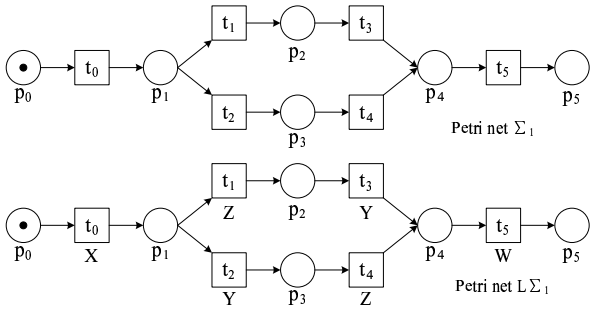


Fig. 1. Petri net and labeled Petri net

For each $x \in S \cup T$, the pre-set of x, denoted as $\bullet x$, is the set of elements which are input of x, i.e. $\bullet x = \{y | (y, x) \in F\}$, while the post-set of x, denoted as $x\bullet$, is the set of elements which are output of x, i.e. $x\bullet = \{y | (x, y) \in F\}$.

Example 2. For Petri net Σ_1 in Fig. 1, $\bullet t_1 = \{p_1\}$ and $t_1\bullet = \{p_2\}$.

Definition 2 (Labeled Petri Net). *A labeled Petri net is a six-tuple $L\Sigma = (S, T, F, A, L, M_0)$, where (S, T, F, M_0) is a Petri net, A is a finite set of action names, and L is a mapping from T to $A \cup \{\varepsilon\}$ (ε represents an action not visible to the outside world).*

Having assigned names, including ε, to transitions, we can capture duplicate and hidden tasks naturally. We sometimes refer to a labeled Petri net as a Petri net when no confusion can occur.

Example 3. For the labeled Petri net $L\Sigma_1$ in Fig. 1, where S, T, F and M_0 are same as the Petri net Σ_1 of Fig. 1. For this net $A = \{X, Y, Z, W\}$ and $L = \{(t_0, X), (t_1, Y), (t_2, Z), (t_3, Z), (t_4, Y), (t_5, W)\}$.

In some cases we do not want to distinguish between transitions and their labels. When the labels are unique for each transition we may use a label to refer to a transition.

Definition 3 (Firing Rule). *Suppose that $L\Sigma = (S, T, F, A, L, M_0)$ is a labeled Petri net. A transition $t \in T$ is said to be enabled in marking M, denoted as $M[t\rangle$, iff $M(s) \geq 1$ for each input place $s \in {\bullet}t$. If transition t is enabled in M_1, firing t results in a marking M_2 such that the following holds:*

$$M_2(s) = \begin{cases} M_1(s) - 1 & \text{if} s \in {\bullet}t - t{\bullet}; \\ M_1(s) + 1 & \text{if} s \in t\bullet - \bullet t; \\ M_1(s) & \text{otherwise.} \end{cases}$$

The notation $M_1[t\rangle M_2$ captures that M_2 is the result of firing t in M_1.

Example 4. For the Petri net $L\Sigma_1$ of Fig. 1, t_0 is enabled at $M_0 = (1, 0, 0, 0, 0, 0)$ while the other transitions are not. Firing t_0 results in the marking $(0, 1, 0, 0, 0, 0)$.

Definition 4 (Incidence Matrix). *Let $L\Sigma = (S, T, F, A, L, M_0)$ be a labeled Petri net, $C_{L\Sigma} = [c_{ij}]_{|S|\times|T|}$ is the incidence matrix of $L\Sigma$ defined by*

$$c_{ij} = \begin{cases} 1 & \text{if} (t_j, s_i) \in F \wedge (s_i, t_j) \notin F; \\ -1 & \text{if} (t_j, s_i) \notin F \wedge (s_i, t_j) \in F; \\ 0 & \text{otherwise.} \end{cases}$$

When no confusion can occur we write C instead of $C_{L\Sigma}$.

Example 5. The incidence matrix of both Petri nets shown in Fig. 1 is the same and it is shown in Fig. 2.

$$\begin{pmatrix} -1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Fig. 2. The incidence matrix of the Petri nets shown in Fig. 1

Definition 5 (Occurrence Vector of Transition Sequence). *Let $L\Sigma = (S, T, F, A, L, M_0)$ be a labeled Petri net and σ be a sequence of transitions. The*

associated occurrence vector X_σ of σ is defined as $(\#(t_1,\sigma),\#(t_2,\sigma),\ldots,\#(t_{|T|},\sigma))$ where for $1 \leq i \leq |T|$, $\#(t_i,\sigma)$ denotes the number of occurrences of t_i in σ. The marking that results from firing transition sequence σ in marking M can be computed as $M^{\mathrm{Tr}} + C_{L\Sigma} \cdot X_\sigma^{\mathrm{Tr}}$ (where e.g. X_σ^{Tr} is the transpose of occurrence vector X_σ). A transition sequence σ of length n is referred to as effective *for a marking M iff the first transition is enabled in M and firing the first i transitions in order always means that the $i+1$th transition is enabled (for any $1 \leq i \leq n-1$). If M' is the marking resulting from firing the transitions in σ in order starting from marking M, we can write $M[\sigma\rangle M'$ to denote this fact.*

Example 6. For the labeled Petri net $L\Sigma_1$ in Fig. 1, $\sigma = t_0t_1t_3$ is an effective transition sequence for $M_0 = (1,0,0,0,0,0)$ and firing this sequence takes us to marking $(0,0,0,0,1,0)$. The occurrence vector X_σ is $(1,1,0,1,0,0)$, hence this marking can be computed as the transpose of $M_0^{\mathrm{Tr}} + C_{L\Sigma_1} \cdot X_\sigma^{\mathrm{Tr}}$.

Definition 6 (Coverability). *Let M_1 and M_2 be reachable markings of a labeled Petri net $L\Sigma = (S,T,F,A,L,M_0)$, that is, there exist firing sequences σ_1 and σ_2, such that $M_0[\sigma_1\rangle M_1$ and $M_0[\sigma_2\rangle M_2$. We say that M_1 is covered by M_2 iff $M_2(p) \geq M_1(p)$ for each place $p \in S$.*

Given a Petri net, starting from its initial marking M_0 we can start to explore the state space of reachable markings. This state space is potentially infinite and then the associated reachability tree is infinite. To deal with this problem one can consider *coverability trees*. The notion of *coverability tree* is one of the fundamental methods for behavioral analysis of Petri nets [17]. In coverability trees a special symbol ω is used, which has the properties that for any integer n, $\omega > n$ and $\omega \pm n = \omega$. Through the use of this special symbol we can deal with unbounded places (a place p is unbounded iff for any number k a marking M can be reached such that $|M(p)| > k$) which are the cause of a state space being infinite. In [17] an algorithm is presented for the construction of a coverability tree and in [23] it is shown that a coverability tree is always finite.

Example 7. The coverability tree of both Petri nets in Fig. 1 is the same and is shown in Fig. 3.

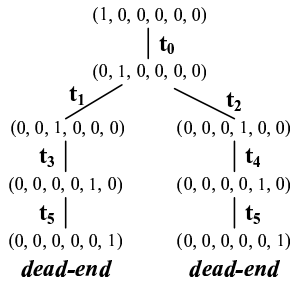


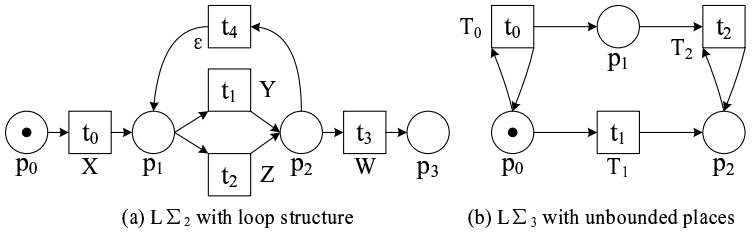
Fig. 3. The coverability tree of the Petri nets of Fig. 1

3 Principal Transition Sequences

This section first provides more information about coverability trees before giving the definition of principal transition sequences. These transition sequences can be seen as a characterisation of transition sequences that lead from the initial marking to a marking that is final (i.e. no transition is enabled). Hence they are an approximation of the behavior of a Petri net. The section concludes with the provision of an algorithm to compute principal transition sequences.

3.1 More about Coverability Trees

A Petri net may have loops or unbounded places. When it has loops we will find markings during the exploration of the state space that we have encountered before. Consider for example the Petri net of Fig.4(a). When the transition sequence $t_0t_1t_4$ is fired in the initial marking, the resulting marking $(0, 1, 0, 0)$ also results from just firing t_0 in the initial marking. Hence in the coverability tree, shown in Fig. 5, when we encounter $(0, 1, 0, 0)$ again, we can mark it as "old" (cf. [17]). For a node v_o in a coverability tree that we have marked as "old" we refer to the node with the identical marking that is closest to the root and on the path from the root to v_o as its anchor and we write anchor(v_o). Hence the anchor of node v_5 in the coverability tree for $L\Sigma_2$ is v_1.



(a) L Σ 2 with loop structure (b) L Σ 3 with unbounded places

Fig. 4. Labeled Petri nets: a) with a loop b) with unbounded places

When a Petri net has unbounded places its state space is infinite. Consider the Petri net of Fig. 4(b). In this net the place p_1 is unbounded which can be seen by repeatedly firing transition t_0 in the initial marking. When we explore the state space and fire t_0 and then immediately t_0 again we discover a marking $(1, 2, 0)$ which, when compared to the previous marking $(1, 1, 0)$, marks every place with at least as many tokens and one place with more. As we know that we can keep firing t_0 and can thus obtain an arbitrary number of tokens in p_1 we collapse all markings of the form $(1, i, 0)$ into one marking $(1, \omega, 0)$. This is illustrated in the coverability tree for $L\Sigma_3$ shown in Fig. 5(b).

Transition sequences on the path from an anchor node to one of its leaf nodes can occur repeatedly, sometimes an infinite number of times, sometimes only a finite number of times. For example, consider the transition sequence t_1t_4 in the coverability tree of Fig. 5(a), in the corresponding Petri net this sequence can occur an infinite number of times. The transition sequence t_2 in the coverability tree of Fig. 5(b) can only occur a finite number of times as it is bounded by the number of previous occurrences of t_0.

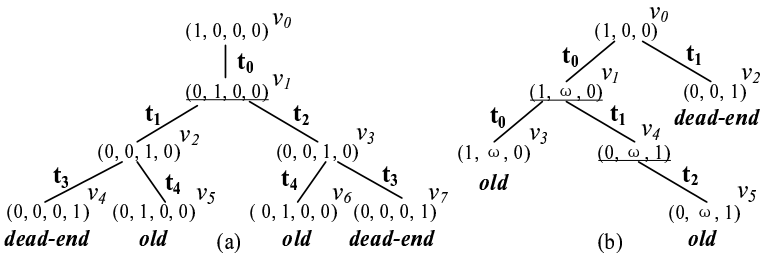


Fig. 5. Anchor nodes in coverability trees of $L\Sigma_2$ and $L\Sigma_3$

3.2 Definition of Principal Transition Sequence

In the definition of principal transition sequences we use the auxiliary function $\mathsf{ts}(v_1, v_2, CT_{L\Sigma})$ which yields the transition path from node v_1 to node v_2 in coverability tree $CT_{L\Sigma}$.

Definition 7 (Principal Transition Sequence). *Let $L\Sigma = (S, T, F, A, L, M_0)$ be a labeled Petri net, $C_{L\Sigma}$ its incidence matrix, $CT_{L\Sigma}$ its coverability tree with root v_r, V_d its set of dead-end nodes and V_o its set of old nodes. The set of principal transition sequences (PTSs) of $L\Sigma$, $\mathsf{pts}(L\Sigma)$, is the minimal set defined by:*

(1) If $v_d \in V_d$ then $\mathsf{ts}(v_r, v_d, CT_{L\Sigma}) \in \mathsf{pts}(L\Sigma)$
(2) If $v_o \in V_o$ then $\mathsf{ts}(v_r, \mathrm{anchor}(v_o), CT_{L\Sigma}) \in \mathsf{pts}(L\Sigma)$
(3) If $v_o \in V_o$ then $\mathsf{ts}(\mathrm{anchor}(v_o), v_o, CT_{L\Sigma}) \in \mathsf{pts}(L\Sigma)$

Loosely speaking, principal transition sequences can be used to compose firing sequences of a labeled Petri net $L\Sigma$. When there are neither loops nor unbounded places, the principal transition sequences of step (1) correspond to all firing sequences of $L\Sigma$. When there is a loop or an unbounded place, then step (2) captures the prefixes of firing sequences and step (3) their repeatable parts.

Example 8. The principal transition sequences of $L\Sigma_1$ are $t_0t_1t_3t_5$ and $t_0t_2t_4t_5$, and these are all the firing sequences. The principal firing sequences of $L\Sigma_2$ on the other hand are $t_0t_1t_3$, $t_0t_2t_3$, t_0, t_1t_4 and t_2t_4.

As the different kinds of principal transition sequences correspond to different behavior we classify them as follows. The *primary* PTSs of a Petri net $L\Sigma$, denoted as $\mathcal{P}_1(L\Sigma)$, are those PTSs that fall into category (1) or (2). These are the PTSs that correspond to nonrepeatable transition sequences. A principal transition sequence σ is an element of the set of *finitely repeatable* PTSs of $L\Sigma$, denoted as $\mathcal{P}_2(L\Sigma)$, iff it falls into category (3) and $C_{L\Sigma} \cdot X_\sigma^{\mathrm{Tr}}$ contains a negative number. It σ falls into category (3), but $C_{L\Sigma} \cdot X_\sigma^{\mathrm{Tr}}$ does not contain a negative number, it is an element of the set of *infinitely repeatable* PTSs, denoted as $\mathcal{P}_3(L\Sigma)$.

3.3 Computations of PTSs

In this section we present the algorithm for computing the various types of principal transition sequences as identified in Section 3.2. In this algorithm apart from auxiliary function $\mathsf{ts}(v_1, v_2, CT_{L\Sigma})$, we also use the auxiliary predicate $\mathsf{neg}(V)$ which determines whether there are any negative values in vector V.

Algorithm 1:Generation of the various PTS sets
Input: A coverability tree $CT_{L\Sigma}$ for a labeled Petri net $L\Sigma$
In this tree V_d is the set of dead-end nodes, V_o is the set of old nodes, and v_r is the root node
Output: $\mathcal{P}_1(L\Sigma)$, $\mathcal{P}_2(L\Sigma)$, $\mathcal{P}_3(L\Sigma)$
$\mathcal{P}_1(L\Sigma) := \varnothing$; $\mathcal{P}_2(L\Sigma) := \varnothing$; $\mathcal{P}_3(L\Sigma) := \varnothing$;
for each $v_f \in V_d \cup V_o$
begin
if $v_f \in V_d$ **then** $\mathcal{P}_1(L\Sigma) := \mathcal{P}_1(L\Sigma) \cup \mathsf{ts}(v_r, v_f, CT_{L\Sigma})$;
if $v_f \in V_o$ **then** $\mathcal{P}_1(L\Sigma) := \mathcal{P}_1(L\Sigma) \cup \mathsf{ts}(v_r, \text{anchor}(v_f), CT_{L\Sigma})$;
if $v_f \in V_o$ **then**
begin
$\sigma := \mathsf{ts}(\text{anchor}(v_f), v_f, CT_{L\Sigma})$;
if $\mathsf{neg}(C_{L\Sigma} \cdot X_\sigma^{\text{Tr}})$
then $\mathcal{P}_2(L\Sigma) := \mathcal{P}_2(L\Sigma) \cup \sigma$
else $\mathcal{P}_3(L\Sigma) := \mathcal{P}_3(L\Sigma) \cup \sigma$
end
end

Example 9. According to Algorithm 1, $\mathcal{P}_1(L\Sigma_1) = \{t_0t_1t_3t_5, t_0t_2t_4t_5\}$, $\mathcal{P}_2(L\Sigma_1) = \varnothing$, $\mathcal{P}_3(L\Sigma_1) = \varnothing$; $\mathcal{P}_1(L\Sigma_2) = \{t_0, t_0t_1t_3, t_0t_2t_3\}$, $\mathcal{P}_2(L\Sigma_2) = \varnothing$, $\mathcal{P}_3(L\Sigma_2) = \{t_1t_4, t_2t_4\}$; $\mathcal{P}_1(L\Sigma_3) = \{t_1, t_0, t_0t_1\}$, $\mathcal{P}_2(L\Sigma_3) = \{t_2\}$, $\mathcal{P}_3(L\Sigma_3) = \{t_0\}$.

The set of principal transition sequences is finite as it is a set of paths constructed from a finite tree. In fact, for any Petri net $L\Sigma$, $|\mathsf{pts}(L\Sigma)| \leq |V_d| + 2 \times |V_o|$. It is also not difficult to see that any transition sequence in $L\Sigma$ that leads from the initial marking to a dead marking can be composed from the principal transition sequences of $L\Sigma$ (note that the reverse does not hold). From a process modeling perspective these sequences are of special interest as a process model is expected to terminate (hence the behaviour of a sound process model can be reasonably approximated by its set of principal transition sequences, although it should be remarked that e.g. branching time is lost).

4 PTS-based Similarity

In this section a similarity measure between labeled Petri nets is defined which is based on PTSs. As the set of PTSs of a labeled Petri net provides an approximation of its behaviour, this measure takes the semantics of the nets involved into account.

Transition labels describe the intended actions of the various transitions in a labeled Petri net. For a transition t, its label is given by $L(t)$ and this notation can be naturally extended to transition sequences. Hence $L(\sigma)$ captures the trace associated with transition sequence σ. This trace can be seen as a string in the language defined by alphabet A of the net. Labels of transitions may differ, but still be considered similar to various degrees. Measurements of label similarity can be based on measures introduced for string similarity (several string similarity notions based on edit distance can be found in [8]). The degree of similarity of transitions can then be measured in terms of the degree of similarity between their labels. In this paper we simplify matters by defining the distance between the same labels as 1 and the distance between different labels as 0.

Definition 8 (Similarity of Two Principal Transition Sequences). *Let $L\Sigma$ and $L\Sigma'$ be labeled Petri nets and σ and σ' be transition sequences of $L\Sigma$ and $L\Sigma'$ respectively. The similarity of σ and σ',* $\mathrm{Sim}(\sigma, \sigma')$*, is based on the their longest common subsequence given by* $\mathrm{lcs}(\sigma, \sigma')$*, and is defined as:*

$$\mathrm{Sim}(\sigma, \sigma') = \frac{\mathrm{length}(\mathrm{lcs}(L(\sigma), L(\sigma')))}{\max(\mathrm{length}(L(\sigma)), \mathrm{length}(L(\sigma')))}$$

Example 10. Let $\sigma = t_0t_1t_3$ and $\sigma' = t_0t_1t_4t_2t_3$ be transition sequences in Petri net $L\Sigma_2$. Then $L(\sigma) = XYW$, $L(\sigma') = XYZW$, and $\mathrm{lcs}(L(\sigma), L(\sigma')) = XYW$. Therefore $\mathrm{Sim}(\sigma, \sigma') = 0.75$.

Now that we have defined a method for determining the degree of similarity between two transition sequences we need to extend the method to deal with sets of transition sequences in order to determine the degree of similarity between two sets of PTSs. In [10] similarity between label sets S_1 and S_2 is determined by taking, for each element from S_1, the element with the highest degree of similarity from S_2, these maximum degrees of similarity are then added for all elements of S_1 and the total sum is divided by the total number of elements of S_1. This unidirectional approach can be transformed into a bidirectional approach, where similarity is a symmetric operation, by using the similarity measure introduced in [11]. In the following definition we use this similarity measure and adapt it to sets of transition sequences.

Definition 9 (Similarity of Two Sets of Transition Sequences based on [11]). *Let P and Q be sets of transition sequences, then if neither P nor Q is the empty set:*

$$\mathrm{Sim}(P, Q) = \frac{\sum\limits_{\sigma \in P} \max\limits_{\sigma' \in Q} \mathrm{Sim}(\sigma, \sigma') + \sum\limits_{\sigma' \in Q} \max\limits_{\sigma \in P} \mathrm{Sim}(\sigma', \sigma)}{|P| + |Q|}$$

If either P or Q equals the empty set, but not both, then $\mathrm{Sim}(P, Q) = 0$*, while if both P and Q are the empty set* $\mathrm{Sim}(P, Q) = 1$*.*

Having defined a notion of similarity between sets of transition sequences we can apply this notion to sets of PTSs to define a notion of similarity between labeled Petri nets.

Definition 10 (Similarity of Two Labeled Petri Nets). *The degree of similarity between labeled Petri nets $L\Sigma$ and $L\Sigma'$,* $\mathrm{Sim}(L\Sigma, L\Sigma')$*, is defined as:*

$$\mathrm{Sim}(L\Sigma, L\Sigma') = \sum_{i=1}^{3} \lambda_i \times \mathrm{Sim}(\mathcal{P}_i(L\Sigma), \mathcal{P}_i(L\Sigma')), \lambda_i = \frac{|\mathcal{P}_i(L\Sigma)| + |\mathcal{P}_i(L\Sigma')|}{|\mathsf{pts}(L\Sigma)| + |\mathsf{pts}(L\Sigma')|}$$

Hence, they capture the ratios between the numbers of the various types of PTSs and the total number of PTSs.

5 Experimental Evaluation

The algorithms for the PTS-based similarity measure (abbreviated as PTS), TAR similarity measure (abbreviated as TAR) [24] and Bae's similarity measure (abbreviated as Bae) [7] were implemented in the BeehiveZ system[7], which aims to provide a general framework for process data management. The code of the algorithm for the causal footprint similarity measure (abbreviated as CFP) [10] was directly taken from the open source process mining environment ProM 5.2[8] and integrated into the BeehiveZ system.

In this section, two experiments are discussed in order to provide an initial indication of the efficiency and effectiveness of the proposed approach. In the first experiment the effectiveness of the proposed similarity measure is investigated through the use of six artificially created process models. The results are contrasted with the similarity measure proposed in [10]. In the second experiment a set of real-life process models are used for the purpose of evaluating both the efficiency and the effectiveness of the proposed similarity measure. The results are contrasted with the TAR similarity measure [24], Bae's similarity measure [7], and the causal footprint similarity measure [10]. Finally, the time and space complexity of the algorithm for the computation of PTS-based similarity as well as its limitations are discussed.

5.1 Experiment I

In Fig. 6 six simple and interrelated models are shown. Models (a)-(c) do not have loops, while models (a')-(c') do. Model (a') equals Model (a) with the addition of a loop, similarly Model (b') equals Model (b) with the addition of a loop, and finally Model (c') equals Model (c) with the addition of two loops.

For this collection of six artificially created models the PTS-based similarity measure (Sim) and the similarity measure of [10] (Sim') is computed for three different groupings. In the first grouping we consider models (a)-(c) and pairwise compute their degree of similarity. In the second grouping the degrees of similarity are determined for the combinations (a)-(a'), (b)-(b'), and (c)-(c'). In the third grouping the degree of similarity is computed among the models (a')-(c'). The PTSs of the six models are shown in Table 1. The degrees of similarity

[7] http://sourceforge.net/projects/beehivez/files/
[8] www.processmining.org

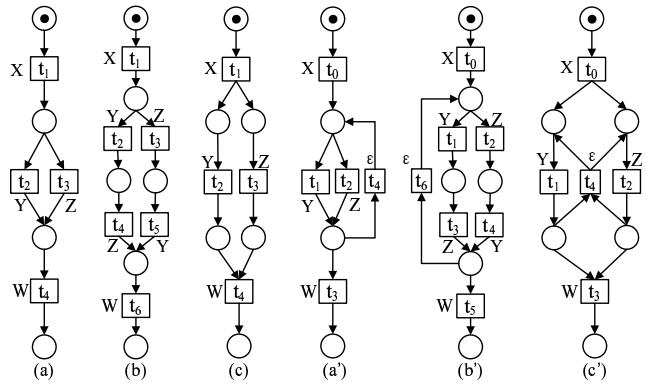


Fig. 6. Six interrelated models

between the various pairs of process models in the three groupings are shown in Table 2.

Table 1. The PTSs of the models in Fig. 6

	$\mathcal{P}_1(L\Sigma)$	$\mathcal{P}_2(L\Sigma)$	$\mathcal{P}_3(L\Sigma)$
(a)	$\{t_1t_2t_4, t_1t_3t_4\}$	$\varnothing$	$\varnothing$
(b)	$\{t_1t_2t_4t_6, t_1t_3t_5t_6\}$	$\varnothing$	$\varnothing$
(c)	$\{t_1t_2t_3t_4, t_1t_3t_2t_4\}$	$\varnothing$	$\varnothing$
(a')	$\{t_0, t_0t_1t_3, t_0t_2t_3\}$	$\varnothing$	$\{t_1t_4, t_2t_4\}$
(b')	$\{t_0, t_0t_1t_3t_5, t_0t_2t_4t_5\}$	$\varnothing$	$\{t_1t_3t_6, t_2t_4t_6\}$
(c')	$\{t_0, t_0t_1t_2t_3, t_0t_2t_1t_3\}$	$\varnothing$	$\{t_1t_2t_4, t_2t_1t_4\}$

As can be seen from Table 2 the causal footprint similarity measure considers models (a) and (c) as quite similar (more so than the PTS-based measure) even though one involves a choice between Y and Z and in the other both Y and Z need to be executed. On the other hand, when only a loop is added the degree of similarity is much lower than the degree of similarity as measured by the PTS-based approach. As the set of traces is the same for models (b) and (c) as well as for models (b') and (c') the PTS-based approach rates their degree of similarity as 100%. Especially for the latter combination this constitutes a real difference with the degree of similarity as measured through the causal footprint approach. As the degree of similarity between two models is to some extent subjective, we leave it up to the reader to draw their own conclusions from Table 2.

5.2 Experiment II

In this section the PTS-based similarity measure is evaluated using real-life process models. We obtained the real life business process models from an industrial boiler manufacturer in the south-east of China. These models were created by an external consulting company using EPCs [20] for the following six SAP ERP

Table 2. Degrees of similarity between various combinations of the six process models

	λ_1	λ_2	λ_3	$Sim(L\Sigma, L\Sigma')$	$Sim'(L\Sigma, L\Sigma')$
(a,b)	1	0	0	75%	66.74%
(a,c)	1	0	0	75%	92.85%
(b,c)	1	0	0	100%	92.85%
(a,a')	5/7	0	2/7	61.9%	67.02%
(b,b')	5/7	0	2/7	60.7%	41.70%
(c,c')	5/7	0	2/7	60.7%	85.86%
(a',b')	6/10	0	4/10	70%	62.22%
(a',c')	6/10	0	4/10	70%	72.47%
(b',c')	6/10	0	4/10	100%	72.47%

modules: Production Planning, Project System, Material Management, Financial Accounting, Cost Controlling, and Sales and Distribution. Table 3 shows the distribution of these business process models over these modules.

Table 3. The business process models from an industrial boiler manufacturer: P# - Number of Process Models E# - Number of Events, F# - Number of Functions, LP# - Number of Proces Models with Loops, L# - Number of Loops, PP# - Number of Process Models with Parallelism, PS# - Number of Parallel Structures, PF# - Number of Parallel Functions

Module	P#	E#	F#	LP#	L#	PP#	PS#	PF#
Production Planning(PP)	12	72	41	0	0	1	1	2
Project System(PS)	3	8	5	0	0	0	0	0
Material Management(MM)	38	282	209	2	2	2	2	5
Financial Accounting(FA)	37	288	178	10	16	6	7	17
Cost Controlling(CO)	18	148	90	9	13	3	3	8
Sales and Distribution(SD)	6	41	27	2	3	1	2	5

From Table 3, it can be seen that there are 114 business process models in the PP, PS, MM, FI, CO and SD modules combined. In terms of number of process models, the biggest module is MM, which contains 38 EPC models, while the smallest module is PS, which only contains 3 EPC models. In these 114 EPCs, the total number of events is 839 and the total number of functions is 550 indicating a relatively high use of control-flow constructs. There are 23 EPCs that contain loops, 20% of the total number of models.

The EPCs were received in MS Word documents and manually modeled in ARIS. The ARIS' AML files were provided as input for ProM in order to generate the corresponding Petri nets. Here it should be remarked that OR-joins in EPCs are not always correctly preserved in the mapping to Petri nets by ProM (it can also be remarked that there are inherent problems with the semantics of OR-joins in EPCs, see [3]), therefore those models that contained OR-joins were

modified by hand (and as a result sometimes problems in the original models were fixed). Roughly speaking the events of EPCs correspond to places in Petri nets, the functions in EPCs to transitions in Petri nets, and the names of the functions correspond to the labels of the corresponding transitions.

The similarity between each pair of the 114 converted LPNs is computed and then the values of three metrics are determined. The first metric is the average time (AvgT) it takes to compute the degree of similarity between two LPNs (only for those computations which can be done within the time limit, which for this experiment was set at 1 hour). The second metric is the percentage of pairs that satisfies the triangular inequality (TriR). A pair of LPNs x and y satisfies the triangular inequality iff for every LPN z: $\mathrm{Sim}(x, z) + \mathrm{Sim}(z, y) \geq \mathrm{Sim}(x, y)$. The third and final metric is the computability fraction (ComF), which is the percentage of LPNs for which the similarity measure could be computed (as mentioned, the limit is set at 1 hour). Note that focus is on a single LPN and whether for this LPN the preparation for the computation of the degree of similarity with other LPNs can be completed in time. The results of the comparison of the four algorithms identified earlier along these metrics can be found in Table 4.

Table 4. Results of the comparison of the four algorithms applied to 114 real-life LPNs

	PTS	CFP	Bae	TAR
AvgT (ms)	25.6	14800	4.4	0.9
TriR (%)	99.98	98.96	99.69	98.33
ComF (%)	100	96.49	100	100

As far as ComF is concerned, the CFP algorithm cannot compute the forward links and backward links of four LPNs (i.e., FA.002, MM.211, MM.212, MM.242) in less than one hour. The common characteristic of these four LPNs is that they contain long sequences of transitions (i.e., more than 20 transitions in a path). It can thus be concluded that the CFP algorithm, contrary to the other three algorithms, has difficulties in dealing with long transitions paths.

The PTS algorithm fares best as far as the TriR metric is concerned. The CFP algorithm performs the worst in terms of the AvgT metric. This is because the calculation of the closure of forward and backward links is time consuming. Performance worsens in case there are long transitions paths in the process model, as observed before.

Now we make two observations that do not relate to the above real-life SAP EPC models, but that provide more insight into the effectiveness of the algorithm by Bae et al. and the TAR algorithm.

For the LPNs in Figure 6(a) and Figure 6(b) the algorithm by Bae et al. computes their degree of similarity as 1. The reason for this is that this algorithm is based on the structure of LPNs and does not take the semantics of splits (AND vs XOR) into account. The other three algorithms return more reasonable similarity measures for these LPNs, which are behaviorally not equivalent.

The application of the TAR algorithm to the LPNs in Figure 7(a) and Figure 7(b) yields a degree of similarity of 1 although these LPNs are not behaviorally equivalent. The full firing sequences of these two LPNs are $\{ACD, ACE, BCD, BCE\}$ and $\{ACD, BCE\}$ respectively. The TAR sets of these nets are the same however, i.e., $\{AC, CD, BC, CE\}$, and as a result, the TAR algorithm cannot make a distinction between them. The application of the PTS algorithm to these nets does not yield a degree of similarity of 1 and therefore, for this case, the PTS algorithm is more effective than the TAR algorithm.

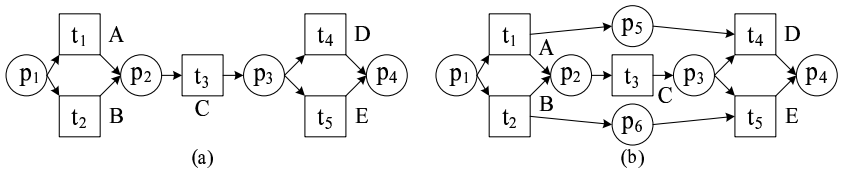


Fig. 7. Two LPNs which are not behaviorally equivalent

Furthermore, we evaluate the four algorithms with another process model set, which contains 591 LPNs converted from 604 SAP reference models by ProM. The comparison results are shown in Table 5. For ComF, the time limit, which takes to compute the degree of similarity between two LPNs, is set to one minute for this data set.

Comparing with Table 4, the ComF values of PTS and TAR decreased slightly. The reason is that there are several LPNs with large parallel structures in this data set, we will discuss it in section 5.3. Other conclusions can be drawn similarly here as those from Table 4.

Table 5. Results of the comparison of the four algorithms applied to 591 LPNs converted from SAP reference models

	PTS	CFP	Bae	TAR
AvgT (ms)	17.8	2706	10.7	18.9
TriR (%)	99.79	93.04	99.54	97.86
ComF (%)	98.48	91.54	100	99.83

5.3 More discussion about the PTS algorithm

For the PTS algorithm, time and space are mainly spent on the calculation of a given LPN's coverability tree and the extraction of all PTSs from this tree. In our implementation, the coverability graph is used instead of the coverability tree to reduce the use of space.

For the time and space complexity of the PTS algorithm, in case there is no parallelism, it can be observed that the length of transition paths as well as the length of loops in an LPN both correspond to paths of similar length in the coverability tree (their length is linear in terms of the length of the original paths). The most problematic situations arise in the presence of large parallel structures. To examine the worst case complexity of the PTS algorithm in an

intuitive manner, we have constructed an LPN with a large parallel structure and drawn it using PIPE 2.5[9]. This LPN is shown in Figure 8 and, as can be seen, apart from the only fork transition (t_0) and the only join transition (t_{n+1}), both having the empty label ε, all transitions are labeled and can be executed in parallel. This type of LPN constitutes the worst case for the PTS algorithm (and incidentally for the TAR algorithm as well).

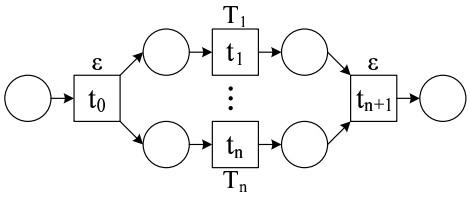


Fig. 8. A special kind of LPNs that cannot be handled easily by the PTS algorithm

The time complexity for calculating the coverability graph of an LPN such as shown in Figure 8 is $n!$ according to Murata [17], while the space complexity of it is 2^n. Both the time and the space complexity are too high if n is more than 20, which constitutes the boundary of the PTS algorithm from a theoretical perspective. However, as shown in Section 5.2, this limitation may not be too significant in practice as LPNs such as shown in Figure 8 with more than 15 tasks in a parallel structure may not occur that often in reality.

6 Related Work

In recent years more and more process models have become available due to their uptake in a wide range of industrial applications [18], this has triggered an increased interest in academia in the notion of process model similarity. In this section we will briefly explore some related work.

The majority of existing work in the area of business process similarity bases itself on structural elements and/or task labels. Various notions of edit distance are used to this end. For example, in [13] high-level change operations are used as a basis for measuring the distance between process models, while in [9] the problem of measuring process model distance is transformed into a graph matching problem so that the notion of edit distance in graphs can be used. In [16] a method is provided for measuring process variant similarity. In [7, 5, 6] workflow dependency graphs are converted into normalized process network matrices and similarity of processes is measured by calculating the metric space distances between these normalized matrices. All these approaches essentially abstract from behavior and as pointed out in [2], process models may be quite similar in terms of their structure and the labels used, but their behavior may be quite different.

In order to take behavior into account when measuring process model similarity, a process language needs to be used that has a formal semantics. As pointed out in e.g. [1], Petri nets provide a solid foundation for workflow modeling and

[9] http://pipe2.sourceforge.net/

analysis. However in order to develop a behavior-based similarity measure, caution should be exercised as it has been shown [12] that trace equivalence of Petri nets and equality of reachability trees are undecidable problems. In practice, process models may exist that have an infinite number of associated traces and/or states [2]. In order to overcome the problem of dealing with infinite trace sets and to address the moment-of-choice problem, in [2, 4] a behavior-based similarity measure of two processes is proposed that uses event logs containing typical behavior. Through this approach even models with infinite trace sets can be compared, however it is dependent on these event logs containing "typical behavior". Such logs may not always be available in practice and their quality may be difficult to ascertain.

Causal footprints are used in [10] as a basis for measuring process model similarity. Causal footprints consist of two kinds of links, look-forward and look-backward. To determine the degree of similarity between two process models (EPCs in this case) index vectors are constructed using these causal footprints and the functions in the models. The similarity of the two models is then computed as the "cosine of the angle between their index vectors" ([10], p. 8). The approach takes the degree of label similarity into account. An advantage of the approach in [10] is the fact that their structural analysis can handle process models with many parallel branches without suffering from the problem of state space explosion as our approach does. On the other hand our approach holds more potential when it comes to recognizing lack of similarity between process models that are structurally similar, but behaviorally different.

In [24] a behavior-based similarity measure is proposed for workflow nets based on transition adjacency relations. Transitions t_1 and t_2 are adjacent if t_2 immediately follows t_1 in every firing sequence in which either of them occur. The computation of all transition adjacency relations may also be affected by state space explosion. In addition, adjacency relations essentially correspond to sequences of size two while in our approach principal transition sequences may be of arbitrary length. It may thus be expected that, generally speaking, in our approach behavior is captured more accurately.

7 Conclusions

In this paper a new similarity measure for labeled Petri nets was proposed that takes behavioral aspects into account. The method for determining the degree of similarity between two nets circumvents the problem of having to deal with infinite trace sets or infinite reachability graphs by computing the sets of principal transition sequences. Experiments, both with artificial and real-life models, were conducted to explore characteristics of the proposed similarity measure.

There are some limitations to the work presented. For example, while the proposed similarity measure takes behavioral aspects into account, the set of principal transition sequences of a net do not fully characterize its behavior. It remains an open question to what extent the resulting loss of information affects the quality of the similarity measure. If this loss is significant, refinement

of the proposed similarity measure is required. In addition, the computational complexity of the proposed approach relies on the computation of coverability trees. More work is required to explore to what extent this poses limitations in practice and what can be done to manage this complexity.

Acknowledgements

The work is partially supported by the National Basic Research Program of China (No. 2009CB320700 and No. 2007CB310802) and the National High-Tech Development Program of China (No. 2008AA042301 and No. 2007AA040607).

References

1. W.M.P. van der Aalst. The application of petri nets to workflow management. *Journal of Circuits, Systems, and Computers*, 8(1):21–66, 1998.
2. W.M.P. van der Aalst, A.K. Alves de Medeiros, and A.J.M.M. Weijters. Process equivalence: Comparing two process models based on observed behavior. In *Business Process Management*, pages 129–144, 2006.
3. W.M.P. van der Aalst, J. Desel, and E. Kindler. On the Semantics of EPCs: A Vicious Circle. In M. Rump and F.J. Nüttgens, editors, *Proceedings of the EPK 2002: Business Process Management using EPCs*, pages 71–80, Trier, Germany, 2002. Gesellschaft für Informatik.
4. A.K. Alves de Medeiros, W.M.P. van der Aalst, and A.J.M.M. Weijters. Quantifying process equivalence based on observed behavior. *DKE*, 64(1):55–74, 2008.
5. J. Bae, J. Caverlee, L. Liu, and H. Yan. Process mining by measuring process block similarity. In J. Eder and S. Dustdar, editors, *Proceedings of BPM 2006 Workshops, Vienna, Austria, September, 2006*, volume 4103 of *Lecture Notes in Computer Science*, pages 141–152. Springer, 2006.
6. J. Bae, L. Liu, J. Caverlee, and W.B. Rouse. Process mining, discovery, and integration using distance measures. In *2006 IEEE Int. Conf. on Web Services (ICWS 2006), September 2006, Chicago, Illinois, USA*, pages 479–488. IEEE Computer Society, 2006.
7. J. Bae, L. Liu, J. Caverlee, L. Zhang, and H. Bae. Development of distance measures for process mining, discovery and integration. *Int. J. Web Service Res.*, 4(4):1–17, 2007.
8. W.W. Cohen, P.D. Ravikumar, and S.E. Fienberg. A comparison of string distance metrics for name-matching tasks. In S. Kambhampati and C.A. Knoblock, editors, *Proceedings of IJCAI-03 Workshop on Information Integration on the Web (IIWeb-03), August 9-10, 2003, Acapulco, Mexico*, pages 73–78, 2003.
9. J.C. Corrales, D. Grigori, and M. Bouzeghoub. Bpel processes matchmaking for service discovery. In R. Meersman and Z. Tari, editors, *Part I of the Proceedings of the OTM Confederated International Conferences, Montpellier, France, 2006*, volume 4275 of *Lecture Notes in Computer Science*, pages 237–254. Springer, 2006.
10. R.M. Dijkman, M. Dumas, B.F. van Dongen, R. Käärik, and J. Mendling. Similarity of business process models: Metrics and evaluation. BETA Working Paper Series, WP 269, Eindhoven University of Technology, The Netherlands, 2009.

11. K. Huang, Z. Zhou, Y. Han, G. Li, and J. Wang. An algorithm for calculating process similarity to cluster open-source process designs. In H. Jin, Y. Pan, and N. Xiao, editors, *Proceedings of Grid and Cooperative Computing - GCC 2004 Workshops: Wuhan, China, October 21-24, 2004*, volume 3252 of *Lecture Notes in Computer Science*, pages 107–114. Springer, 2004.
12. P. Jancar. Undecidability of bisimilarity for petri nets and some related problems. *Theor. Comput. Sci.*, 148(2):281–301, 1995.
13. C. Li, M. Reichert, and A. Wombacher. On measuring process model similarity based on high-level change operations. In Q. Li, S. Spaccapietra, E.S.K. Yu, and A. Olivé, editors, *Proceedings of the 27th International Conference on Conceptual Modeling, Barcelona, Spain, October 20-24, 2008*, volume 5231 of *Lecture Notes in Computer Science*, pages 248–264. Springer, 2008.
14. D. Lin. An information-theoretic definition of similarity. In J.W. Shavlik, editor, *Proceedings of the Fifteenth Int. Conf. on Machine Learning (ICML 1998), Madison, WI, USA, July, 1998*, pages 296–304. Morgan Kaufmann, 1998.
15. D. Liu, J. Wang, S.C.F. Chan, J. Sun, and L. Zhang. Modeling workflow processes with colored petri nets. *Computers in Industry*, 49(3):267–281, 2002.
16. N.M. Mahmod, S.W. Sadiq, and R. Lu. Similarity matching of business process variants. In J. Cordeiro and J. Filipe, editors, *Proceedings of the Tenth Int. Conf. on Enterprise Information Systems, Volume ISAS-2, Barcelona, Spain, June, 2008*, pages 234–239, 2008.
17. T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.
18. M. La Rosa, H.A. Reijers, W.M.P. van der Aalst, R.M. Dijkman, J. Mendling, M. Dumas, and L. García-Bañuelos. Apromore : An advanced process model repository (in press). http://eprints.qut.edu.au/27448/, Queensland University of Technology, Brisbane, Australia, 2009.
19. M. Rosemann. Potential pitfalls of process modeling: part b. *Business Process Management Journal*, 12(3):377–384, 2006.
20. A.-W. Scheer, O. Thomas, and O. Adam. *Process Aware Information Systems: Bridging People and Software Through Process Technology*, chapter Process Modeling Using Event-Driven Process Chains, pages 119–146. John Wiley & Sons, Hoboken, New Jersey, 2005.
21. J. Su. Letter from the special issue editor. *IEEE Data Eng. Bull.*, 32(3):2, 2009.
22. A. Wombacher, P. Fankhauser, B. Mahleko, and E.J. Neuhold. Matchmaking for business processes. In *2003 IEEE Int. Conf. on Electronic Commerce (CEC 2003), June 2003, Newport Beach, CA, USA*, pages 7–11. IEEE Computer Society, 2003.
23. C. Yuan. *Principles of Petri Nets (Chinese version)*. Publishing House of Electronics Industry, Beijing, China, 1998.
24. H. Zha, J. Wang, L. Wen, C. Wang, and J. Sun. A workflow net similarity measure based on transition adjacency relations. *Computers in Industry*, 61:463–471, 2010.