ExRORU: A Behavioral Similarity Measure for Process Models based on Extended Refined Ordering Relations with Uncertainty

Shuhao Wang[$], Lijie Wen[$], Jianmin Wang[$], and Jianwen Su[#]

[$]School of Software, Tsinghua University, Beijing 100084, P.R. China
shudiwsh2009@gmail.com,{wenlj,jimwang}@tsinghua.edu.cn
[#]Department of Computer Science, UC Santa Barbara, USA
su@cs.ucsb.edu

Abstract. Tao Jin has proposed new ordering relations with uncertainty between execution of tasks in acyclic process models. He also gave an algorithm to compute the refined ordering relations based on unfolding technology. However, his algorithm cannot work well for cyclic WF-nets and process models with invisible tasks. We extend his work and present a refinement of the relations. To better measure the differentiation between process models, we introduce the notion of sequential direct adjacency. Also, we propose an algorithm to compute relations in arbitrary WF-nets and utilize them to measure the similarity between process models. Experiments on artificial models and real-life models from enterprises show that compared to existing algorithms, our measure is both efficient and effective.

Keywords: Process Model, Refined Ordering Relations with Uncertainty, Sequential Direct Adjacency, Behavioral Similarity

1 Introduction

In a seminal paper [4], Tao Jin has proposed new ordering relations with uncertainty between execution of tasks in acyclic process models. He refines the causal and concurrency relations between two events in a concurrent system with uncertainty according to whether one task is always executed with the other task in the same instance. He then proposes some rules for adjacent tasks and some transitive laws for nonadjacent tasks, based on which he proposes an algorithm to compute the refined ordering relations for acyclic process models. The algorithm is done on the unfolding of a WF-net [6, 3].

Although Tao Jin' work is elegant, it has some drawbacks. First of all, for cyclic Petri nets, his algorithm cannot work correctly. According to his paper, the problem arises from the Global Fairness Assumption [5]. Secondly, invisible tasks have not been taken into consideration so that two models with different behavioral semantics may have the same set of ordering relations. Thirdly, sequential relations do not satisfy transitivity in some mdoels with non-free-choice

constructs. To solve these problems, we extend his ordering relations to more trivial cases which is inspired by the work of DecSerFlow [12], and introduce the concept of sequential direct adjacency to better differentiate the behavioral semantics of two process models. Also, we propose an algorithm to compute the new relations in arbitrary WF-nets.

Furthermore, since our refined relations have a strong ability to represent the behavioral semantics of a process model and can efficiently measure the tiny differentiation between two models, even when they contain invisible tasks, we utilize them as a proper metric to measure the similarity of two process models.

Similarity measure can be widely used in many scenarios, such as model retrieval, process mining and model classification. There have been various proposals to this topic based on textual information, the topology of process models, or their execution semantics [18]. We give a quick introduction to some similarity measure related to our work.

Zha et al. [22] represent a model's behavior by transition adjacency relations (TAR for short). They use the Jaccard coefficient of TAR sets to measure the similarity between models. But TAR cannot tackle the long-distance dependences between transitions, thus it is not able to handle specific structures like non-free-choice constructs. BP algorithm [18] captures the similarity based on behaviroal profiles between transitions, but it cannot deal with invisible tasks and distinguish parallel and loop structures. PTS algorithm by Professor Wang [13] and its improvement CFS by Dong et al. [2] utilize refined trace sets of models to compute their similarity and are the most realistic results so far. But both their methods are very time-consuming. Polyvyanyy et al. [10] propose another definition of relations between transitions named 4C spectrum but the hunderds of types of relations in 4C are too complex for comprehension and hard to compute. It is not a practical method in measuring behavioral similarity.

In this paper, we propose a behavioral similarity measure based on the extended refined ordering relations with uncertainty. It satisfies metric properties, especially the triangle inequality [21]. The algorithm firstly extracts three types of relations and generates the relation sets of a process model. By measuring the similarity between relation sets of process models, a similarity value is finally computed. We conduct experiments using process models from real enterprises and the results show that our similarity measure is both efficient and effective.

The remainder of this paper is structured as follows. The next section gives preliminaries that will be used in this paper. The concept and computation of extended refined ordering relations as well as sequential direct adjacency are introduced in Section 3. We present the method to measure process model similarity in Section 4. Section 5 shows the results of an experimental evaluation, before we conclude the paper and give an outlook on future work in Section 6.

2 Preliminaries

Our work is to measure the similarity between two process models. There are many different notations to describe the business process, such as Business Pro-

cess Model and Notation (BPMN), Event-driven Process Chain (EPC), Yet Another Workflow Language (YAWL) and Petri Net. Since Petri net is suitable to analyze the behavior of process models, we will show our algorithm on Petri net. The concept of the Petri net has its origin in Carl Adam Petri's dissertation [8].

Definition 1 (Petri net). *A Petri net is a triple* (P, T, F)*:*

- 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 (flow relation).*

We use $M : P \to \mathbb{N}$ to denote a marking of Petri net N, where $\mathbb{N}$ is the set of natural numbers. A Petri net system is a pair $\Sigma = (N, M_0)$, where N is a Petri net and M_0 is the initial marking of N. For more details about Petri net, please refer to the work of Murata [7]. A Petri net which models a workflow is called a WorkFlow net (WF-net) [11].

Definition 2 (WF-net). *A Petri net* $PN = (P, T, F)$ *is a WF-net (Workflow net) if and only if:*

- PN *has two special places:* i *and* o*. Place* i *is a source place:* $\bullet i = \emptyset$*. Place* o *is a sink place:* $o\bullet = \emptyset$*;*
- *If we add a transition* t^* *to* PN *which connects place* o *with* i *(i.e.* $\bullet t^* = \{o\}$ *and* $t^*\bullet = \{i\}$*), then the resulting Petri net is strongly connected.*

We extract the relations between activities in a WF-net. An intuitive idea is to generate the reachability graph of a model. We need to consider all possible interleavings of concurrent events, which causes the state explosion problem in a concurrent system [6]. Javier Esparza proposed the notion of complete prefix unfolding to avoid this [3], based on the improvement of McMillan's unfolding algorithm [6]. Therefore, we use his work instead.

Definition 3 (Ordering Relations). *There are three types of ordering relations between nodes of a net,*

- *two nodes* x *and* y *are in causal relation, denoted by* $x < y$*, if the net contains a path with at least one arc leading from* x *to* y*;*
- x *and* y *are in conflict relation, denoted by* $x\#y$*, if the net contains two disjoint paths* $st_1...x_1$ *and* $st_2...x_2$ *starting at the same place* s*;*
- x *and* y *are in concurrency relation, denoted by* x *co* y*, if neither* $x < y$ *nor* $y < x$ *nor* $x\#y$*.*

Definition 4 (Occurrence net). *An occurrence net is a Petri net* $O = (B, E, A)$ *such that* $\forall x, y \in B \cup E : (x, y) \in A^+ \Rightarrow (y, x) \notin A^+$ *and* $\forall b \in B : |\bullet b| \leq 1$*.*

Definition 5 (Branching process). *A branching process of a Petri net system* $\Sigma = (P, T, F, M_0)$ *is a labelled occurrence net* $\beta = (B, E, A, f)$ *where the labelling function* f *satisfies the following properties:*

- *$f(B) \subseteq P$ and $f(E) \subseteq T$ (f preserves the nature of nodes);*
- *for every $e \in E$, the restriction of f to $\bullet e$ is a bijection between $\bullet e$ (in Σ) and $\bullet f(e)$ (in β), and similarly for $e\bullet$ and $f(e)\bullet$ (f preserves the environments of transitions);*
- *the restriction of f to $Min(O)$ is bijection between $Min(O)$ and M_0 (β "starts" at M_0), where $Min(O)$ denotes the set of minimal elements of $B \cup E$ with respect to the causal relation;*
- *for every $e_1, e_2 \in E$, if $\bullet e_1 = \bullet e_2$ and $f(e_1) = f(e_2)$ then $e_1 = e_2$ (β does not duplicate the transitions of Σ).*

There are only sequential and parallel structures but neither loop nor exclusive structures in a branching process. A Petri net system may have infinite branching processes, which differ on "how much it unfolds". The complete finite prefix is the minimal one which contains all the markings contained in the branching process.

Definition 6 (Complete Prefix Unfolding). *Let $O = (B, E, A, f)$ be a occurrence net and $e \in E$ be any event.*

- *A local configuration $\lceil e \rceil$ of an event e in an occurrence net is the set of events that precede e.*
- *The final marking of a local configuration $Mark(\lceil e \rceil)$ is the set of conditions that are marked after all the events in $\lceil e \rceil$ fire.*
- *An adequate order $\prec$ is a strict well-founded partial order on local configurations, so that $\lceil e \rceil \subset \lceil e' \rceil$ implies $\lceil e \rceil \prec \lceil e' \rceil$[1].*
- *An event e of a branching process is a cutoff event if there exists a corresponding event e', such that $Mark(\lceil e \rceil) = Mark(\lceil e' \rceil)$ and $\lceil e' \rceil \prec \lceil e \rceil$.*
- *A complete prefix unfolding is the greatest backward closed subnet of an occurrence net containing no events after cutoff events.*

Definitions about complete prefix unfolding and related concepts come from the work of Esparza [3] and Polyvyanyy [9].

Example 1. Figure 1 shows a sound WF-net and its complete prefix unfolding. In the unfolding, events labelled as B and E are cutoff events.

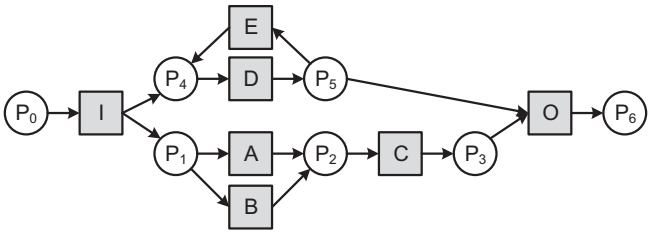
Our extended relations are inspired by the work in [12], which utilizes the relations between activities to model a process.

Definition 7 (Relation Formulas). *Let A and B be two activities of a process, A and B are in the following relation formulas*[2]*:*

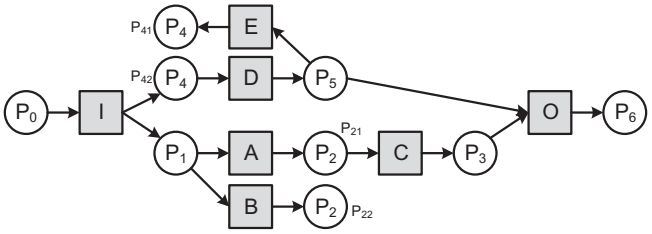
- *Response: every time activity A executes, activity B has to be executed after it.*

[1] Several definitions of *adequate order* exist, and we use the total order for 1-safe systems defined in [3].

[2] Note that the relation formulas are not complete here, since we only need part of them. For more details, please refer to the work of van der Alast [12].



(a) A sound WF-net



(b) The complete prefix unfolding of (a)

Fig. 1. A sound WF-net example and its complete prefix unfolding

- *Precedence: if activity B was executed, it could not have been executed until the activity A was executed.*
- *Alternate Response: after the execution of an activity A, activity B has to be executed and between the execution of each two A at least one activity B has to be executed.*
- *Alternate Precedence: every instance of activity B has to be preceded by an instance of activity A and the next instance of activity B cannot be executed before the next instance of activity A is executed.*

Example 2. In the model of Figure 1(a), D, E in the execution sequence $< I, D, E, A, D, C, E, D, O >$ satisfy the formula *alternate response* and *alternate precedence* while transitions D, O do not satisfy the formula *alternate response* and transitions I, D do not satisfy the formula *alternate precedence*.

3 Extended Refined Ordering Relations with Uncertainty

In this section, we introduce the concept of extended refined ordering relations with uncertainty and its calculation from a WF-net. Let $\Sigma = (P, T, F, M_0)$ be a sound WF-net, and $U = (B, E, A, f)$ be Σ's complete prefix unfolding. Let $x, y \in E$ be two events of the unfolding U.

3.1 Extended Refined Causal and Inverse Causal Relations with Uncertainty

There may be infinite branching processes for a process model, but the complete prefix unfolding use the technique of cutoff event to avoid state explosion prob-

lem. It should be noticed that there would be cutoff events and cutoff conditions (conditions after those cutoff events) in the complete prefix unfolding. We can easily set a mapping between cutoff conditions and the conditions after the corresponding event of a cutoff event. Therefore, the branching process expressed in a complete prefix unfolding never ends on a cutoff condition, but will continue from the mapping of a cutoff condition to the end node instead. As mentioned before, a branching process only contains sequential and parallel structures.

We use expressions like $[ab\{c, d\}e]$ to represent the semantics of a branching process. Successive letters denote the sequential structure while letters in a brace mean a parallel structure. Different branches of a parallel structure are separated by a comma and could be nested. For example, in the complete prefix unfolding of Figure 1(b), $[I\{D, AC\}O]$, $[I\{D, BC\}O]$ and $[I\{DED, AC\}O]$ are some branching processes.

Let Θ be the set containing all the branching process of U, and Θ_x be the set of the branching processes containing event x, i.e, $\Theta_x = \{\beta \in \Theta | x \in \beta\}$. Our extended causal and inverse causal relations are defined using the notion of Θ_x.

Definition 8 (Always Causal). *Events x and y are in always causal relation (denoted as $x \twoheadrightarrow y$) iff: $\forall \beta_x \in \Theta_x$, x and y are in alternate response relation, i.e., among all the branching processes containing event x, after the execution of every x there must be at least one y executed, and before that y there cannot be another event x.*

Definition 9 (Never Causal). *Events x and y are in never causal relation (denoted as $x \nrightarrow y$) iff: $\forall \beta_x \in \Theta_x$, x and y are not in response relation, i.e., among all the branching processes containing event x, after the execution of every x, there mustn't be an instance of y executed.*

Definition 10 (Sometimes Causal). *Events x and y are in sometimes causal relation (denoted as $x \rightharpoonup y$) iff: neither $x \twoheadrightarrow y$ nor $x \nrightarrow y$.*

Example 3. In the model of Figure 1(b), A and C are in *always causal* relation ($A \twoheadrightarrow C$) since they are in *alternate response* relation among all the branching processes containing A. D and O are in *sometimes causal* relation ($D \rightharpoonup O$) since there may be another D between the execution of the first D and O, indicating that D and O are not in *alternate response* relation. C and D are in *never causal* relation ($C \nrightarrow D$).

Definition 11 (Always Inverse Causal). *Events y and x are in always inverse causal relation (denoted as $y \twoheadrightarrow^{-1} x$) iff: $\forall \beta_y \in \Theta_y$, x and y are in alternate precedence relation, i.e., among all the branching processes containing event y, every instance of y has to be preceded by an instance of x and the next instance of event y cannot be executed before the next instance of event x is executed.*

Definition 12 (Never Inverse Causal). *Events y and x are in never inverse causal relation (denoted as $y \nrightarrow^{-1} x$) iff: $\forall \beta_y \in \Theta_y$, x and y are not in precedence relation, i.e., among all the branching processes containing event y, before the execution of every y, there mustn't be an instance of a executed.*

Definition 13 (Sometimes Inverse Causal). *Events y and x are in sometimes inverse causal relation (denoted as $y \backsim^{-1} x$) iff: neither $y \twoheadrightarrow^{-1} x$ nor $y \nrightarrow^{-1} x$.*

Example 4. In the model of Figure 1(a), O and D are in *always inverse causal* relation ($O \twoheadrightarrow^{-1} D$) since they are in *alternate precedence* relation among all the branching processes containing O. C and A are in *sometimes inverse causal* relation ($C \backsim^{-1} A$) since there may be no instance of transition A in some branching processes containing C (such as $[I\{D, BC\}O]$). D and C are in *never inverse causal* relation ($D \nrightarrow^{-1} C$).

The relations above are collectively called causal relation with uncertainty (denoted as $\rightarrow$) and inverse causal relation with uncertainty (denoted as $\rightarrow^{-1}$). We have identified several cases of causal and inverse causal relations and turn them into abstract formulas, as shown in Figure 2. In thses cases, A and B are events of an unfolding. The edge labeled with "Loop X" means that there is a path starting from some condition after X and ending in some condition before X so that X can be executed more than once. The edge labeled with "Skip X" means that there is a path starting from some condition before X and ending in some condition after X so that X may not be executed. In other words, events inside a *loop* structure can be executed more than once while events inside a *skip* structure may not be executed.

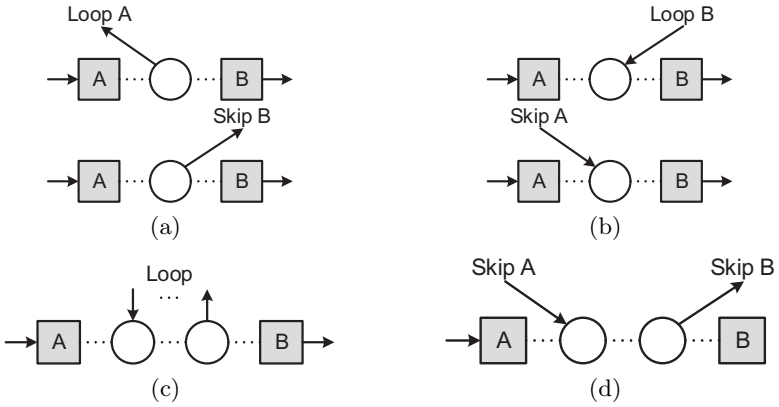


Fig. 2. Abstract formulas of causal and inverse causal relations. (a) $A \backsim B, B \twoheadrightarrow^{-1} A$; (b) $A \twoheadrightarrow B, B \backsim^{-1} A$; (c) $A \twoheadrightarrow B, B \twoheadrightarrow^{-1} A$; (d) $A \backsim B, B \backsim^{-1} A$.

For convenience, we use symbol $*$ to represent any numbers of sequential events excluding A and B. In Figure 2(a), we have branching processes such as $[A*B], [A*A*B], [A*A*A*B]$... for the upper unfolding and $[A*B], [A*]$ for the lower unfolding, both of which indicate that A and B are in *sometimes causal* relation and *always inverse causal* relation, i.e., $A \backsim B, B \twoheadrightarrow^{-1} A$. In Figure

2(b), we have branching processes such as $[A * B], [A * B * B], [A * B * B * B]...$ for the upper unfolding and $[A * B], [*B]$ for the lower unfolding, both of which indicate that A and B are in *always causal* relation and *sometimes inverse causal* relation, i.e., $A \twoheadrightarrow B, B \rightharpoonup^{-1} A$.

In the unfolding of Figure 2(c), a loop structure in the middle part of a branching process will certainly not affect the extended relations between A and B, neither will an exclusive structure, i.e., $A \twoheadrightarrow B, B \twoheadrightarrow^{-1} A$. However, if there are exclusive structures across both A and B, such as the unfolding in Figure 2(d), which has branching processes such as $[A * B], [A*], [*B], [*]$, events A and B are in *sometimes causal* relation and *sometimes inverse causal* relation, i.e., $A \rightharpoonup B, B \rightharpoonup^{-1} A$.

3.2 Extended Refined Concurrent Relations with Uncertainty

A trace, or a full firing sequence is a finite sequence of events $\sigma \in E^*$, leading from the source state to the end state by executing the events in order. Let Ω be the set containing all the traces of β, and Ω_x be the set of the traces containing event x, i.e., $\Omega_x = \{\sigma \in \Omega | x \in \sigma\}$. Our extended concurrent relations are defined using the notion of Ω_x.

We use $\sigma \uparrow X$ to denote the projection of σ onto some event set $X \subseteq E$, i.e., a trace $\sigma' \subseteq \sigma$ which only contains those events in X and remains their order in σ. Let $P(\sigma, x, y) = \sigma \uparrow \{x, y\}$ be the projection of σ onto events x and y.

Definition 14 (Always Concurrent). *x and y are in always concurrent relation (denoted as $x \equiv y$) iff: $\forall \sigma_x \in \Omega_x$,*

- *x and y are in concurrency relation, i.e., x co y;*
- $|P(\sigma_x, x, y)|\%2 = 0 \wedge \forall_{1 \leq k < |P|/2}((P(\sigma_x, x, y)_{2k+1} = x \wedge P(\sigma_x, x, y)_{2k+2} = y) \vee (P(\sigma_x, x, y)_{2k+1} = y \wedge P(\sigma_x, x, y)_{2k+2} = x) \vee (P(\sigma_x, x, y)_{2k+1} = y \wedge P(\sigma_x, x, y)_{2k+2} = y))$.

Definition 15 (Never Concurrent). *x and y are in never concurrent relation (denoted as $x \nparallel y$) iff: $\forall \sigma_x \in \Omega_x$, x and y are not in concurrency relation.*

Definition 16 (Sometimes Concurrent). *x and y are in sometimes concurrent relation (denoted as $x \Uparrow y$) iff: neither $x \equiv y$ nor $x \nparallel y$.*

The relations above are collectively called concurrent relations with uncertainty (denoted as $\|$). We have also identified several cases of concurrent relations and turn them into abstract formulas, as shown in Figure 3. In these formulas, events A and B are all in *concurrency* relation.

In Figure 3(a), we have traces such as $< I*A*B* >, < I*B*A* >, < I*A* >$ for the first unfolding and traces such as $< I * A * B* >, < I * A * A * B* >, < I * A * A * A * B* > ...$ for the second unfolding, which do not satisfy the second condition in Definition 14. Therefore, events A and B of the unfoldings in Figure 3(a) are in *sometimes concurrent* relation, i.e., $A \Uparrow B$. As for the unfolding in Figure 3(b), events A and B may be executed both once or not executed at all,

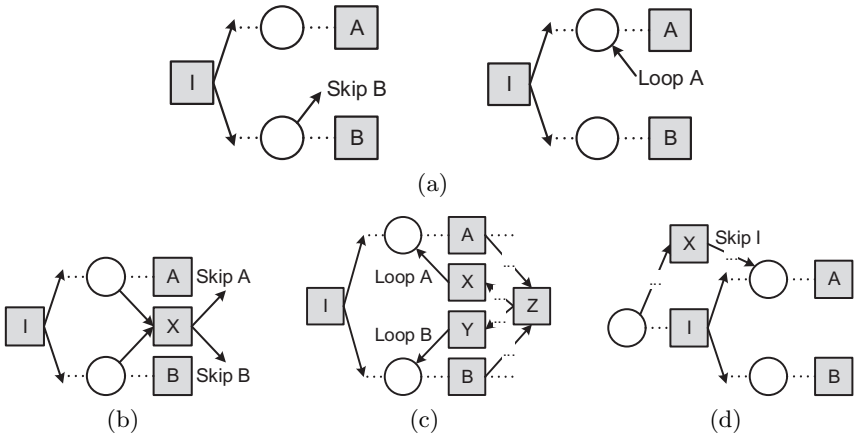


Fig. 3. Abstract formulas of concurrent relations. (a) $A \Uparrow B$; (b) $A \equiv B$; (c) $A \equiv B$; (d) $A \Uparrow B$.

i.e., there are traces such as $< I*A*B* >, < I*B*A* >, < I*X* >$, indicating that $A \equiv B$. It is easily seen that traces in the unfolding of Figure 3(c) satisfy the second rule in Definition 14, so A and B are in *always concurrent* relation, i.e., $A \equiv B$. The unfolding in Figure 3(d) shows a special structure called non-free choice construct, and we will go into details later.

3.3 Computing Extended Refined Ordering Relations Based on Unfolding

As mentioned before, an intuitive idea to derive our relations is based on reachability graphs. Such technique would face a state explosion problem, especially when the WF-net contains a parallel with large scale of transitions. Theoretically, the reachability graph will derive all the reachable states by combining the executions of transitions within a parallel structure. Another problem is that we actually cannot distinguish the causal relation caused by loop structure and parallel structure using the reachability technique. By definitions, causal relations caused by parallel structure are not considered in our extended refined ordering relations. So we adopt the unfolding technology in this paper. The reason why we do not derive the relations based on sound WF-net itself is that WF-net describe the structure of a process rather than behavior, which is just the focus of unfolding technique.

Another intuitive idea on the unfolding technology is to check our relations based on all the branching processes, i.e., Θ of a complete prefix unfolding. However, Θ may be an infinite set, which is not practical to compute and utilize. Therefore, motivated by the abstract formulas in Figure 2 and Figure 3, we will derive our relations on a complete prefix unfolding by tracing the directed path from one event to another and check every condition to determine whether there is a *loop* or *skip* structure.

We may notice the fact that the computations of causal relation and inverse causal relation are similar in the way that we regard inverse causal relation as causal relation in the reversion of the unfolding. In another word, if we reverse the direction of every edge in the unfolding, then the causal relations in the new unfolding (actually the new graph is not an unfolding, but we use the saying to explain our computation here) are the same as the inverse causal relations in the original unfolding. This indicates that we can use the algorithm of causal relations to derive the inverse causal relations by backtracing nodes along the paths of an unfolding.

As for the extended concurrent relations between events a and b, there are actually two relations which should be taken into consideration, i.e., $a \parallel b$ and $b \parallel a$. We firstly find the gate of the parallel structure, and trace the path from the gate to a and b. By checking every condition to find out all the *loop* and *skip* structure, we can derive the extended concurrent relations between two events.

More useful information are the extended relations between transitions in the WF-net, which can be utilized to check the consistency or measure the similarity between process models. We focus on the computation of our relations on a WF-net based on its unfolding.

There may be more than one corresponding event (shadow event in [14]) in the complete prefix unfolding related to a transition in the original WF-net. These corresponding events are different from each other and are treated as unique events when we derive the relations between events in the unfolding. Therefore, when we try to derive the extended ordering relation between transitions in the WF-net, we need to check the relations between each pair of corresponding events of those transitions. Based on this, we give the computation of extended relations between transitions. Let $Corr_E(A)$ be the set containing all the corresponding events of transition A.

Definition 17 (Extended Causal Relations Between Transitions). *Transitions A and B are in*

- *always causal relation (denoted as $A \twoheadrightarrow B$) iff: $\forall a \in Corr_E(A), \exists b \in Corr_E(B)$, s.t. $a \twoheadrightarrow b$;*
- *never causal relation (denoted as $A \nrightarrow B$) iff: $\forall a \in Corr_E(A), b \in Corr_E(B)$, $a \nrightarrow b$;*
- *sometimes causal relation (denoted as $A \rightharpoonup B$) iff: neither $A \twoheadrightarrow B$ nor $A \nrightarrow B$.*

Definition 18 (Extended Inverse Causal Relations Between Transitions). *Transitions B and A are in*

- *always inverse causal relation (denoted as $B \twoheadrightarrow^{-1} A$) iff: $\forall b \in Corr_E(B), \exists a \in Corr_E(A), s.t.\ b \twoheadrightarrow^{-1} a$;*
- *never inverse causal relation (denoted as $B \nrightarrow^{-1} A$) iff: $\forall b \in Corr_E(B), a \in Corr_E(A), b \nrightarrow^{-1} a$;*
- *sometimes inverse causal relation (denoted as $B \rightharpoonup^{-1} A$) iff: neither $B \twoheadrightarrow^{-1} A$ nor $B \nrightarrow^{-1} A$.*

Definition 19 (Extended Concurrent Relations Between Transitions). *Transitions A and B are in*

- *always concurrent relation (denoted as $A \equiv B$) iff: $\forall a \in Corr_E(A), \exists b \in Corr_E(B), s.t.\ a \equiv b$;*
- *never concurrent relation (denoted as $A \nparallel B$) iff: $\forall a \in Corr_E(A), b \in Corr_E(B), a \nparallel b$;*
- *sometimes concurrent relation (denoted as $A \Uparrow B$) iff: neither $A \equiv B$ nor $A \nparallel B$.*

Example 5. Using the notions and computations described in the previous sections, we can derive the extended relations of the model in Figure 1(a). For convenience, we use a matrix to present three types of relations, as shown in Table 1. Transitions in the first column are the former ones in a relation while transitions in the first row are the latter ones in a relation. For example, the second row are the relations between A and the other transitions. Due to limited space, we show three types of relations in one table cell. The rightmost table cell with relation symbols represents the relations between A and O, i.e., $A \twoheadrightarrow O, A \not\twoheadrightarrow^{-1} O, A \nparallel O$.

Table 1. Extended relations of the model in Figure 1(a)

Transitions	A	B	C	D	E	I	O
A	$\not\twoheadrightarrow$ $\not\twoheadrightarrow^{-1}$ $\nparallel$	$\not\twoheadrightarrow$ $\not\twoheadrightarrow^{-1}$ $\nparallel$	$\twoheadrightarrow$ $\not\twoheadrightarrow^{-1}$ $\nparallel$	$\not\twoheadrightarrow$ $\not\twoheadrightarrow^{-1}$ $\equiv$	$\not\twoheadrightarrow$ $\not\twoheadrightarrow^{-1}$ $\Uparrow$	$\not\twoheadrightarrow$ $\twoheadrightarrow^{-1}$ $\nparallel$	$\twoheadrightarrow$ $\not\twoheadrightarrow^{-1}$ $\nparallel$
B	$\not\twoheadrightarrow$ $\not\twoheadrightarrow^{-1}$ $\nparallel$	$\not\twoheadrightarrow$ $\not\twoheadrightarrow^{-1}$ $\nparallel$	$\twoheadrightarrow$ $\not\twoheadrightarrow^{-1}$ $\nparallel$	$\not\twoheadrightarrow$ $\not\twoheadrightarrow^{-1}$ $\equiv$	$\not\twoheadrightarrow$ $\not\twoheadrightarrow^{-1}$ $\Uparrow$	$\not\twoheadrightarrow$ $\twoheadrightarrow^{-1}$ $\nparallel$	$\twoheadrightarrow$ $\not\twoheadrightarrow^{-1}$ $\nparallel$
C	$\not\twoheadrightarrow$ $\not\rightarrow^{-1}$ $\nparallel$	$\not\twoheadrightarrow$ $\not\rightarrow^{-1}$ $\nparallel$	$\not\twoheadrightarrow$ $\not\twoheadrightarrow^{-1}$ $\nparallel$	$\not\twoheadrightarrow$ $\not\twoheadrightarrow^{-1}$ $\equiv$	$\not\twoheadrightarrow$ $\not\twoheadrightarrow^{-1}$ $\Uparrow$	$\not\twoheadrightarrow$ $\twoheadrightarrow^{-1}$ $\nparallel$	$\twoheadrightarrow$ $\not\twoheadrightarrow^{-1}$ $\nparallel$
D	$\not\twoheadrightarrow$ $\not\twoheadrightarrow^{-1}$ $\Uparrow$	$\not\twoheadrightarrow$ $\not\twoheadrightarrow^{-1}$ $\Uparrow$	$\not\twoheadrightarrow$ $\not\twoheadrightarrow^{-1}$ $\Uparrow$	$\not\rightarrow$ $\not\rightarrow^{-1}$ $\nparallel$	$\not\rightarrow$ $\not\rightarrow^{-1}$ $\nparallel$	$\not\twoheadrightarrow$ $\not\rightarrow^{-1}$ $\nparallel$	$\not\rightarrow$ $\not\twoheadrightarrow^{-1}$ $\nparallel$
E	$\not\twoheadrightarrow$ $\not\twoheadrightarrow^{-1}$ $\Uparrow$	$\not\twoheadrightarrow$ $\not\twoheadrightarrow^{-1}$ $\Uparrow$	$\not\twoheadrightarrow$ $\not\twoheadrightarrow^{-1}$ $\Uparrow$	$\not\twoheadrightarrow$ $\not\twoheadrightarrow^{-1}$ $\nparallel$	$\not\rightarrow$ $\not\rightarrow^{-1}$ $\nparallel$	$\not\twoheadrightarrow$ $\not\rightarrow^{-1}$ $\nparallel$	$\not\rightarrow$ $\not\twoheadrightarrow^{-1}$ $\nparallel$
I	$\not\rightarrow$ $\not\twoheadrightarrow^{-1}$ $\nparallel$	$\not\rightarrow$ $\not\twoheadrightarrow^{-1}$ $\nparallel$	$\twoheadrightarrow$ $\not\twoheadrightarrow^{-1}$ $\nparallel$	$\twoheadrightarrow$ $\not\twoheadrightarrow^{-1}$ $\nparallel$	$\not\rightarrow$ $\not\twoheadrightarrow^{-1}$ $\nparallel$	$\not\twoheadrightarrow$ $\not\twoheadrightarrow^{-1}$ $\nparallel$	$\twoheadrightarrow$ $\not\twoheadrightarrow^{-1}$ $\nparallel$
O	$\not\twoheadrightarrow$ $\not\rightarrow^{-1}$ $\nparallel$	$\not\twoheadrightarrow$ $\not\rightarrow^{-1}$ $\nparallel$	$\not\twoheadrightarrow$ $\twoheadrightarrow^{-1}$ $\nparallel$	$\not\twoheadrightarrow$ $\twoheadrightarrow^{-1}$ $\nparallel$	$\not\twoheadrightarrow$ $\not\rightarrow^{-1}$ $\nparallel$	$\not\twoheadrightarrow$ $\twoheadrightarrow^{-1}$ $\nparallel$	$\not\twoheadrightarrow$ $\not\twoheadrightarrow^{-1}$ $\nparallel$

3.4 Sequential Direct Adjacency

We can distinguish most process models from each other by their executing semantics using the extended refined ordering relations with uncertainty. However, there are still models with different semantics but the same extended relations, especially when invisible tasks are introduced. Invisible tasks do not appear in any log trace [1] . For more details about invisible tasks and their mining, please refer to the work of Wen [20].

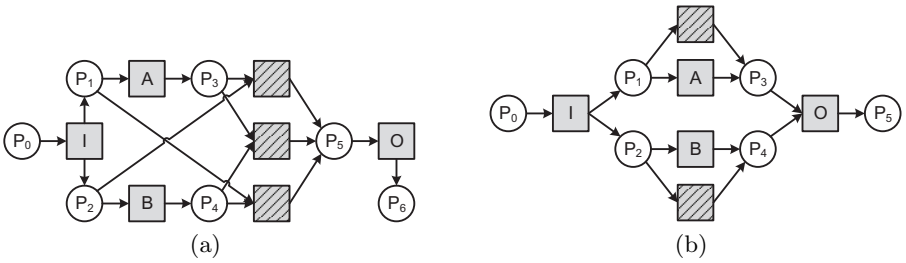


Fig. 4. Two WF-nets with invisible tasks

Example 6. Figure 4 shows an example. The full firing sequences of the model in Figure 4(a) are $< I, A, B, O >$, $< I, B, A, O >$, $< I, A, O >$ and $< I, B, O >$ while the full firing sequences of the model in Figure 4(b) are $< I, A, B, O >$, $< I, B, A, O >$, $< I, A, O >$, $< I, B, O >$ and $< I, O >$. But the extended relations of these two models are the same, as shown in Table 2.

Table 2. Extended relations of the two models in Figure 4

Transitions	A	B	I	O
A	$\nrightarrow$ $\nrightarrow^{-1}$ $\nparallel$	$\nrightarrow$ $\nrightarrow^{-1}$ $\Uparrow$	$\nrightarrow$ $\nRightarrow^{-1}$ $\nparallel$	$\nRightarrow$ $\nrightarrow^{-1}$ $\nparallel$
B	$\nrightarrow$ $\nrightarrow^{-1}$ $\Uparrow$	$\nrightarrow$ $\nrightarrow^{-1}$ $\nparallel$	$\nrightarrow$ $\nRightarrow^{-1}$ $\nparallel$	$\nRightarrow$ $\nrightarrow^{-1}$ $\nparallel$
I	$\nrightarrow$ $\nrightarrow^{-1}$ $\nparallel$	$\nrightarrow$ $\nrightarrow^{-1}$ $\nparallel$	$\nrightarrow$ $\nrightarrow^{-1}$ $\nparallel$	$\nRightarrow$ $\nrightarrow^{-1}$ $\nparallel$
O	$\nrightarrow$ $\nleftarrow^{-1}$ $\nparallel$	$\nrightarrow$ $\nleftarrow^{-1}$ $\nparallel$	$\nrightarrow$ $\nRightarrow^{-1}$ $\nparallel$	$\nrightarrow$ $\nrightarrow^{-1}$ $\nparallel$

By looking into the executing semantics of these models, we find that the introduction of invisible tasks influences the adjacency of transitions, i.e., I can

be directly followed by O in Figure 4(a), but not in Figure 4(b). Inspired by the idea of TAR [22], we propose the concept of sequential direct adjacency to solve the problem of invisible task.

Definition 20 (Sequential Direct Adjacency). *Let $\Sigma = (P, T, F, M_0)$ be a WF-net and $A, B \in T$ are two visible transitions. A and B are in sequential direct adjacency iff:*

- *A and B are not in cocurrency relation;*
- *$\exists \sigma =< t_1, t_2, ..., t_n >\in \Omega, 1 \leq i < j \leq n, t_i = A, t_j = B, s.t.\ \forall k \in (i, j), t_k$ is an invisible transition.*

$A \rightarrow_D B$ ("D" for direct) if they are in sequential direct adjacency while $A \rightarrow_I B$ ("I" for indirect) if not. The set of all the transition pairs which are in sequential direct adjacency is the SDA-set of a model.

We use sequential direct adjacency to distinguish those transition pairs which can be executed adjacently from those which cannot. By applying Definition 20 on the two models in Figure 4, we derive the *SDA-set* as follows:

$$\text{(a)} A \rightarrow_D O, B \rightarrow_D O, I \rightarrow_D A, I \rightarrow_D B$$
$$\text{(b)} A \rightarrow_D O, B \rightarrow_D O, I \rightarrow_D A, I \rightarrow_D B, I \rightarrow_D O$$

However, sequential direct adjacency has the same drawback as TAR that adding new transition to the middle part or the head and tail part of a model has different influence on the scale of SDA-set [16]. To avoid this, we use the method in TAR++ [16]. We add four nodes and four edges to the original WF-net including two places Ps and Pn, two transitions Ts and Tn and four edges $Ps \rightarrow Ts$, $Ts \rightarrow Pi$, $Po \rightarrow Te$, $Te \rightarrow Pe$, where Pi and Po are the source place and sink place of the WF-net. This will make the similarity value between two WF-nets more reasonable. For convenience, we do not show the relations between Ts, Tn and transitions in the original WF-net in our examples.

Using the combination of extended refined ordering relations with uncertainty and sequential direct adjacency, we can distinguish the two models with invisible tasks as well as many model pairs from each other efficiently.

4 Similarity Measure

In this section, we give the definitions about the similarity of two process models based on extended relations and sequential direct adjacency.

4.1 The Importance of Relation

When we measure the similarity of two models, we actually measure the similarity between the relations of them. But our extended relations cannot reflect the different possibilities of transitions that may fire in concurrent and exclusive

structures. Motivated by the work of TAR++ [16], we bring in the concepts about the importance of relation.

[16] has given enough details about the concept about the weight of an edge as well as three rules to determine it. Also, the computation of weights of a WF-net can be found in this paper. We give the definition about the importance of relation based on the weight of an edge.

Definition 21 (The Importance of Relation). *Given a weighted WF-net Σ, for any ordered transition pair A, B in an extended relation, we select one out-edge of A and one in-edge of B, whose coefficients are α, β respectively. We use the minimum number of α and β as the importance value of the extended relation, i.e.,* $\min\{\alpha, \beta\}$.

4.2 Computation of Similarity

By combining extended relations, sequential direct adjacency and the importance of relations together, we give the computation of the similarity between two WF-nets. For any ordered transition pair A, B, we use a triple $R(A, B) = (\gamma, \delta, \theta)$ to represent their extended relation, where γ is the uncertainty ("A" for *always*, "S" for *sometimes* and "N" for *never*), δ is the sequential direct adjacency ("D" for *direct* and "I" for *indirect*), θ is the importance (between 0 and 1). For convenience, we use R_{pair} to denote the transition pair in R.

On the other hand, we think two extended relations with the same δ are more similar than two extended relation with different δ, so we set a weight λ to balance the influence of sequential direct adjacency.

For two extended relations $R1 = (\gamma_1, \delta_1, \theta_1)$ and $R2 = (\gamma_2, \delta_2, \theta_2)$ where $R1_{pair} = R2_{pair}$, we have:

$$
\begin{aligned}
R_1 \cap R_2 &= \begin{cases} 0 \ R1_{pair} & \gamma_1 \neq \gamma_2 \\ \lambda \times \min\{\theta_1, \theta_2\} \ R1_{pair} & \gamma_1 = \gamma_2 \wedge \delta_1 \neq \delta_2 \\ \min\{\theta_1, \theta_2\} \ R1_{pair} & \gamma_1 = \gamma_2 \wedge \delta_1 = \delta_2 \end{cases} \\
R_1 \cup R_2 &= \max\{\theta_1, \theta_2\} \ R1_{pair}
\end{aligned} \tag{1}
$$

We derive three sets $ExRelSet(M)$ of extended relations of a model M, namely $ExRelSet_{\rightarrow}(M)$, $ExRelSet_{\rightarrow^{-1}}(M)$ and $ExRelSet_{\|}(M)$. Now we give the definition of intersection and union operation of these sets.

Definition 22 (Intersection and Union Operation of Relation Sets). *Given two extended relation sets $ExRelSet1$ and $ExRelSet2$, which contain m and n elements respectively. The relations in these sets are marked $R =$*

$(\gamma_R, \delta_R, \theta_R)$, *whose transition pair is denoted as* R_{pair}. *Then we have:*

$$\begin{aligned}
ExRelSet1 \Cap ExRelSet2 =& \{Ri \cap Rj | Ri \in ExRelSet1, \\
& Rj \in ExRelSet2, Ri_{pair} = Rj_{pair}\} \\
ExRelSet1 \Cup ExRelSet2 =& \{Ri \cup Rj | Ri \in ExRelSet1, \\
& Rj \in ExRelSet2, Ri_{pair} = Rj_{pair}\} \\
& \cup \{\theta_{Ri} Ri_{pair} | Ri \in ExRelSet1 \\
& \quad \wedge \nexists R_j \in ExRelSet2, s.t.\ Ri_{pair} = Rj_{pair}\} \\
& \cup \{\theta_{Rj} Rj_{pair} | Rj \in ExRelSet2 \\
& \quad \wedge \nexists R_i \in ExRelSet1, s.t.\ Ri_{pair} = Rj_{pair}\}
\end{aligned} \tag{2}$$

We use Jaccard equation to measure the similarity between two sets:

$$Sim(ExRelSet1, ExRelSet2) = \frac{|ExRelSet1 \Cap ExRelSet2|}{|ExRelSet1 \Cup ExRelSet2|} \tag{3}$$

, where $|ExRelSet1 \Cap ExRelSet2| = \sum \theta_R, \theta_R R_{pair} \in ExRelSet1 \Cap ExRelSet2$ and $|ExRelSet1 \Cup ExRelSet2|$ similarly.

Since there are three sets of extended relations in a model, three similarity values are computed for two models M_1 and M_2, namely $Sim_{\rightarrow}(M_1, M_2)$, $Sim_{\rightarrow^{-1}}(M_1, M_2)$ and $Sim_{\|}(M_1, M_2)$. We assign different weights to these values and derive the final similarity value between M_1 and M_2.

Definition 23 (ExRORU Similarity). *Given two WF-nets* M_1 *and* M_2, *the ExRORU similarity between them is:*

$$Sim(M_1, M_2) = \sum_h \omega_h \cdot Sim_h(M_1, M_2) \tag{4}$$

with $h \in \{\rightarrow, \rightarrow^{-1}, \|\}$ *and weighting factors* $\omega_h \in \mathbb{R}, 0 < \omega_h < 1$ *such that* $\sum_h \omega_h = 1$.

We will show some examples and special cases in the next section.

5 Experimental Evaluation

We have implemented a similarity measure algorithm[3] based on the complete prefix unfolding in jbpt[4]. In this section, we present the experimental evaluation of our algorithm from both performance and quality perspectives. We evaluate the performance by applying our algorithm on real-life dataset and evaluate the quality by comparing it with other algorithms.

[3] https://github.com/shudiwsh2009/ExRORU
[4] https://code.google.com/p/jbpt/

5.1 Special Examples

Before presenting the experimental results of our algorithm, we want to show some special examples such as WF-nets with non-free-choice constructs and multi-relations. Examples in this section together with the models in Figure 4 show the powerful capability of our algorithm to present the behaviors of a process model.

Non-free-choice constructs are difficult to discover, which are the main problems of most existing process mining approaches [1]. Process models with non-free-choice constructs contain a mixture of synchronization and choice, i.e., synchronization and choice are not separated which may create implicit dependencies [19]. Figure 5(a) shows a WF-net with non-free-choice constructs and Figure 5(b) shows a similar WF-net that is free choice.

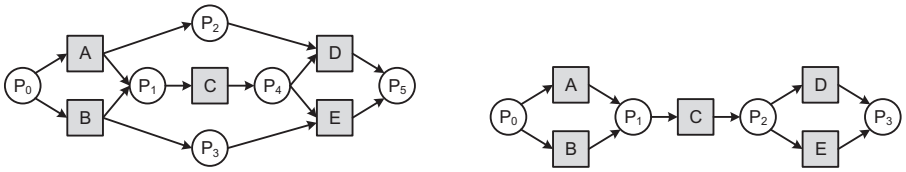
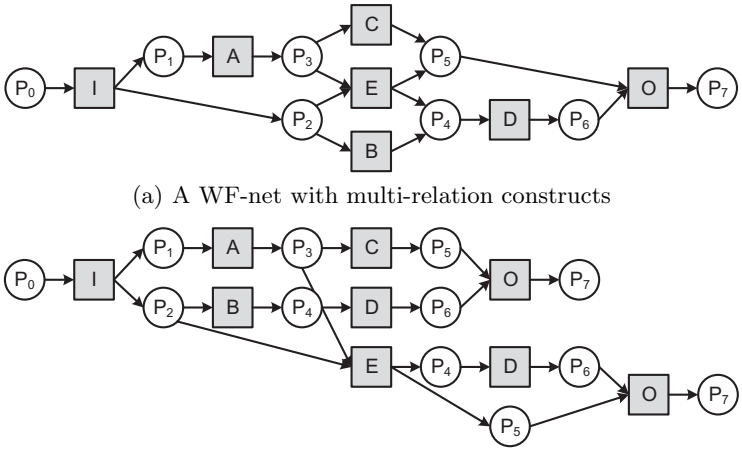


Fig. 5. WF-nets with and without non-free-choice constructs

Most similarity algorithms based on relations between transitions such as TAR cannot distinguish these two models. Take TAR algorithm for example, the TAR-sets of them are both $\{\langle A, C\rangle, \langle C, D\rangle, \langle B, C\rangle, \langle C, E\rangle\}$, which means that the similarity between these two models are 1 using TAR. However, it is obvious that they are not the same, since the trace set Ω of the model in Figure 5(a) is $\{< A, C, D >, < B, C, E >\}$ while the other is $\{< A, C, D >, < A, C, E >, < B, C, D >, < B, C, E >\}$.

Actually, there are only two branching processes in the complete prefix unfolding of the model in Figure 5(a), i.e., $[A\{C\}D]$ and $[B\{C\}E]$. Therefore, we have $A \twoheadrightarrow D$ and $B \twoheadrightarrow E$. On the other hand, there are four branching processes in the complete prefix unfolding of the model in Figure 5(b), i.e., $[ACD], [ACE], [BCD], [BCE]$, indicating that $A \rightharpoonup D$ and $B \rightharpoonup E$. We can see the difference between these two models using our extended relations.

Multi-relation is another special structure we want to discuss. An example WF-net with multi-relation and its complete prefix unfolding are shown in Figure 6. Transitions A and D are in multi-relation since they can both in causal and concurrency relation. From the unfolding in Figure 6(b), we know there are two branching processes, i.e., $\sigma_1 = [I\{AC, BD\}O]$ and $\sigma_2 = [I\{A\}E\{D\}O]$. Using the definitions of our extended relations, we have $A \rightharpoonup D$ and $A \Uparrow D$.



(a) A WF-net with multi-relation constructs

(b) The complete prefix unfolding of (a)

Fig. 6. A WF-net with multi-relation and its complete prefix unfolding

Most similarity measure cannot recognize the multi-relation in a process model since they categorize transition pair into one type of relation, which causes problems and may lead to confusing similarity result for models like Figure 6(a).

5.2 Experimental Setting

Our experimental datasets contain both artificial-generated process models and reference models from several enterprises such as TC, DG and SAP. These collections has 644 models (expressed in Petri nets) in total. The 72 artificial-generated models are mainly used to check the compliance of properties that a good similarity measure should satisfy which models from enterprises are used to check the distance metric requirement and efficiency. The basic features of these models are summarized in Table 3.

Table 3. The features of experimental datasets

Dataset	Size	Average			Minimum			Maximum		
		#transitions	#place	#arcs	#transitions	#place	#arcs	#transitions	#place	#arcs
TC	89	11.472	10.281	22.933	6	5	11	28	29	58
DG	94	8.553	8.883	17.777	1	3	3	34	33	70
SAP	389	4.465	7.504	11.514	1	2	2	21	31	56
Arti-	72	10.457	10.305	21.139	3	4	6	13	13	26

Our experiments are divided into two parts. Firstly, we use reference models to check triangle inequality property, which is an importance aspect of distance metric. We repeatedly extract every three models from the dataset, calculate similarities between each pair of them, transfer these values to distance values

($distance = 1 - similarity$) and test whether they satisfy the triangle inequality property, which means the sum of any two distances values is large than the third one. Secondly, we use artificial-generated models to check seven properties. More details on the properties and experimental methods can be found in [15, 17, 16].

As for comparison, TAR and BP algorithms are based on the relations between transitions as we do while PTS and CFS are based on the abstract trace sets which seems more close to reality. Therefore, we conduct experiments using our algorithm together with the four above and compare them in both efficiency and effectiveness.

5.3 Performance Evaluation on Real-Life Models

The performance of our algorithm on real-life

5.4 Quality Evaluation on Artificial Models

We evaluate our algorithm by checking the satisfaction of several properties that a good similarity algorithm should meet. The following properties come from [15, 17, 16]. We compare our algorithm with TAR, BP, PTS and CFS, which are implemented in BeehiveZ[5].

Property 1 (Sequential structure drift invariance). As shown in Figure 7, given a sequential WF-net, wherever the new transition is inserted, the similarity of the new model and the old one does not change. Namely, $Sim(N1, N2) = Sim(N1, N3) = Sim(N1, N4)$.

Experiment Models In Figure 7, extend the number of transitions to 10 in $N1$ to obtain model $N1'$, then add a new transition to the 11 intervals in $N1'$ including the beginning and the end. Thus we obtain $N2' - N12'$, totally 11 models. The similarity results between $N1'$ and other models are shown in Figure 8.

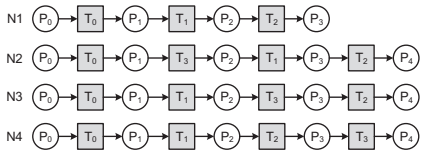


Fig. 7. Sequential structure drift invariance

Fig. 8. Comparison of Algorithms with Property 1

[5] https://github.com/shudiwsh2009/BeehiveZ

Property 2 (Negative correlation by conflict span). As shown in Figure 9, given a WF-net, as the span of new exclusive branch increases, the similarity of the new model and the old ones decreases. Namely, $Sim(N1, N2) > Sim(N1, N3)$.

Experiment Models In Figure 9, extend the number of transitions to 10 in $N1$ to obtain model $N1'$, then add a new exclusive branch which skips over T_0, then T_0, T_1, then T_0, T_1, T_2, and so on until it skips over all the 10 transitions. Thus we obtain $N2' - N11'$, totally 10 models. The similarity results between $N1'$ and other models are shown in Figure 10.

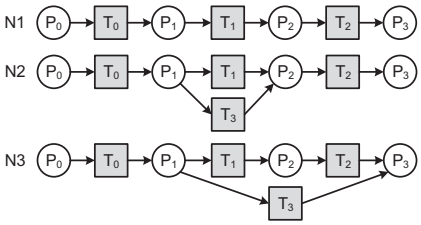


Fig. 9. [Negative correlation by conflict span

Fig. 10. Comparison of Algorithms with Property 2

Property 3 (Unrelated task regression). As shown in Figure 11, given a WF-net, add new branches to the same model in the original model, the more the branches are added, the less the similarity of the new model and the old one is. Namely, $Sim(N1, N2) > Sim(N1, N3)$.

Experiment Models In Figure 11, add new branches to T_2 in $N1'$, a copy of model $N1$. Add one branch as $N2'$, two branches as $N3'$, and so on. Thus we obtain $N2' - N11'$, totally 10 models. The similarity results between $N1'$ and other models are shown in Figure 12.

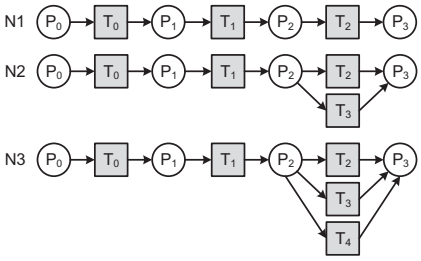


Fig. 11. Unrelated task regression

Fig. 12. Comparison of Algorithms with Property 3

Property 4 (Negative correlation by loop length). As shown in Figure 13, given a WF-net, add a new loop branch to the original model, the more the span of the branch is, the less the similarity of the new model and the old one is. Namely, $Sim(N1, N2) > Sim(N1, N3)$.

Experiment Models In Figure 13, extend the number of transitions to 10 in $N1'$ and obtain model $N1'$, then add a new loop branch which loops over T_1, then T_1, T_2, then T_1, T_2, T_3, and so on until it loops over all the 9 transitions from T_1 to T_9. Thus we obtrain $N2' - N10'$, totally 9 models. The similarity results between $N1'$ and other models are shown in Figure 14.

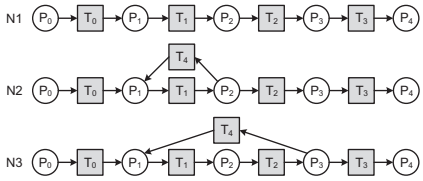


Fig. 13. Negative correlation by loop length

Fig. 14. Comparison of Algorithms with Property 4

Property 5 (Conflict structure drift invariance). As shown in Figure 15, given a WF-net, add a new exclusive branch which skips over only one transition to the original model, wherever the new branch is added to, the similarity of the new model and the old one does not change. Namely, $Sim(N1, N2) = Sim(N1, N3) = Sim(N1, N4)$.

Experiment Models In Figure 15, extend the number of transitions to 10 in $N1$ and obtain model $N1'$, then add new exclusive branch to each of the 10 transitions. Thus we obtain $N2' - N11'$, totally 10 models. The similarity results between $N1'$ and other models are shown in Figure 16.

Table 4 gives an overview of the quality evaluation on existing algorithms. Our algorithm ExRORU is the only one which satisfies all the five properties.

6 Conclusion

References

1. Ana Karla A de Medeiros, Wil MP van der Aalst, and AJMM Weijters. Workflow mining: Current status and future directions. In *On the move to meaningful internet systems 2003: Coopis, doa, and odbase*, pages 389–406. Springer, 2003.

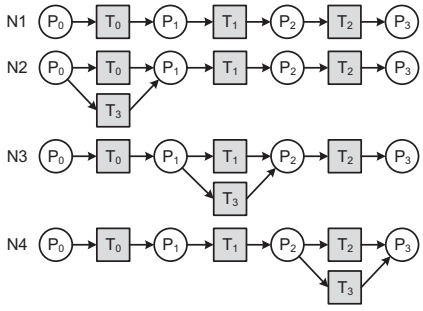


Fig. 15. Conflict structure drift invariance

Fig. 16. Comparison of Algorithms with Property 5

Table 4. Summary of the quality evaluation

Algorithm	Property1	Property2	Property3	Property4	Property5
TAR	×	×	✓	×	×
PTS	✓	✓	✓	×	✓
CFS	✓	✓	✓	✓	✓
BP	✓	×	✓	✓	✓
ExRORU	✓	✓	✓	✓	✓

2. Zihe Dong, Lijie Wen, Haowei Huang, and Jianmin Wang. Cfs: A behavioral similarity algorithm for process models based on complete firing sequences. In *On the Move to Meaningful Internet Systems: OTM 2014 Conferences*, pages 202–219. Springer, 2014.
3. Javier Esparza, Stefan Römer, and Walter Vogler. An improvement of mcmillan's unfolding algorithm. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 87–106. Springer, 1996.
4. Tao Jin, Jianmin Wang, Lijie Wen, and Gen Zou. Computing refined ordering relations with uncertainty for acyclic process models. *Services Computing, IEEE Transactions on*, 7(3):415–426, 2014.
5. Ekkart Kindler and Wil Van Der Aalst. Liveness, fairness, and recurrence in petri nets. *Information Processing Letters*, 70(6):269–274, 1999.
6. Kenneth L. McMillan and David K Probst. A technique of state space search based on unfolding. *Formal Methods in System Design*, 6(1):45–65, 1995.
7. Tadao Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.
8. CA Petri. Kommunikation mit automaten. bonn: Institut für instrumentelle mathematik, schriften des iim, no. 3 (1962); also, english translation: Communication with automata, griffiss air force base. Technical report, New York. Tech. Rep. RADC-TR. 1 (suppl. 1), 1966.
9. Artem Polyvyanyy, Luciano García-Bañuelos, and Marlon Dumas. Structuring acyclic process models. In *Business Process Management*, pages 276–293. Springer, 2010.
10. Artem Polyvyanyy, Matthias Weidlich, Raffaele Conforti, Marcello La Rosa, and Arthur HM ter Hofstede. The 4c spectrum of fundamental behavioral relations for

concurrent systems. In *Application and Theory of Petri Nets and Concurrency*, pages 210–232. Springer, 2014.
11. Wil MP Van der Aalst. The application of petri nets to workflow management. *Journal of circuits, systems, and computers*, 8(01):21–66, 1998.
12. Wil MP Van Der Aalst and Maja Pesic. *DecSerFlow: Towards a truly declarative service flow language*. Springer, 2006.
13. Jianmin Wang, Tengfei He, Lijie Wen, Nianhua Wu, Arthur HM Ter Hofstede, and Jianwen Su. A behavioral similarity measure between labeled petri nets based on principal transition sequences. In *On the Move to Meaningful Internet Systems: OTM 2010*, pages 394–401. Springer, 2010.
14. Jianmin Wang, Shaoxu Song, Xiaochen Zhu, and Xuemin Lin. Efficient recovery of missing events. *Proceedings of the VLDB Endowment*, 6(10):841–852, 2013.
15. S Wang, L Wen, D Wei, J Wang, and ZQ Yan. Ssdt matrix-based behavioral similarity algorithm for process models. *Computer Integrated Manufacturing Systems*, 19(8):1822–1831, 2013.
16. Shuhao Wang, Ming Yin, Zixuan Wang, and Jianmin Wang. Tar++: A new process model similarity algorithm based on the importance of tars. In *Asia Pacific Business Process Management*, pages 98–112. Springer, 2015.
17. Zixuan Wang, Lijie Wen, Jianmin Wang, and Shuhao Wang. Tager: Transition-labeled graph edit distance similarity measure on process models. In *On the Move to Meaningful Internet Systems: OTM 2014 Conferences*, pages 184–201. Springer, 2014.
18. Matthias Weidlich, Jan Mendling, and Mathias Weske. Efficient consistency measurement based on behavioral profiles of process models. *Software Engineering, IEEE Transactions on*, 37(3):410–429, 2011.
19. Lijie Wen, Wil MP van der Aalst, Jianmin Wang, and Jiaguang Sun. Mining process models with non-free-choice constructs. *Data Mining and Knowledge Discovery*, 15(2):145–180, 2007.
20. Lijie Wen, Jianmin Wang, Wil MP van der Aalst, Biqing Huang, and Jiaguang Sun. Mining process models with prime invisible tasks. *Data & Knowledge Engineering*, 69(10):999–1021, 2010.
21. Pavel Zezula, Giuseppe Amato, Vlastislav Dohnal, and Michal Batko. *Similarity search: the metric space approach*, volume 32. Springer Science & Business Media, 2006.
22. Haiping Zha, Jianmin Wang, Lijie Wen, Chaokun Wang, and Jiaguang Sun. A workflow net similarity measure based on transition adjacency relations. *Computers in Industry*, 61(5):463–471, 2010.