2022 19th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE).
Mexico City, Mexico. November 9-11, 2022

Discovering a Class of Workflow Nets with Reduced Exceeding Language

Marisol Vera-Arellano
CINVESTAV, Unidad Guadalajara
Av. Del Bosque 1145. Col. El Bajio
45019 Zapopan Jal. Mexico
marisol.vera@cinvestav.mx

Ernesto López-Mellado
CINVESTAV, Unidad Guadalajara
Av. Del Bosque 1145. Col. El Bajio
45019 Zapopan Jal. Mexico
e.lopez@cinvestav.mx

Abstract— **This paper deals with the automated discovery of workflow nets (WFN) from logs of event traces that represent the observed behaviour of a discrete event process. The aim is to obtain a WFN *N* that represents closely the language of the log λ, whose surplus language of the model $\mathcal{L}(N)\backslash\lambda$ is low. The approach pursues to build a subclass of block-structured WFN composed of block subnets with cycles discovered from subsets of λ. This paper focuses on the discovery of blocks from subsets of traces that have the same event alphabet. In the proposed technique, a block is formed by two subnets using the same event labels: a) $N_<$, a WFN state-machine derived from the causal relations between events, and b) N_C, a WFN marked-graph that allows firing once all the different transitions. Then, N is obtained by the synchronous composition of $N_<$ and N_C ($N_<||N_C$). The algorithms derived from the method, which have polynomial time complexity, build highly precise models.**

Keywords— Process discovery, Workflow nets, Surplus language, High precision.

I. Introduction

Process Mining (PM) is a recent research area that deals with "discovering, monitoring, and improving real processes by extracting knowledge from the event logs available in the current system" [1, 2]. An event log λ stores the behaviour exhibited by the process as sequences or traces of event names. In general, the event log is a multiset of traces.

The main branch of PM is Process discovery (PD); the primary aim of PD methods is to determine a model *N* that represents the behaviour recorded in λ. Discovery methods have been proposed for dealing with two classes of discrete event processes: a) business processes [2, 3] where *N* is expressed as Workflow Nets (WFN), and b) industrial manufacturing processes [4, 5], where *N* is given as cyclic Petri nets (PN).

In general, the obtained model *N* by most of the discovery methods replay the log λ, which is the aim of such methods, but it may execute other traces not included in the log, i.e., the surplus language $\mathcal{L}(N)\backslash\lambda$. The quality measure that evaluates the exceeding language is called *Precision* [2]; $|\mathcal{L}(N)\backslash\lambda|$ is lower, and the precision of *N* is higher. To illustrate this concept consider the log {xabcy, xcbay}; a WFN that replays only the two traces is shown in Figure 1.a; it has a precision of 1.0. Conversely, the WFN shown if Figure 1.b replays six traces including those of the log; it has a precision of 0.333.

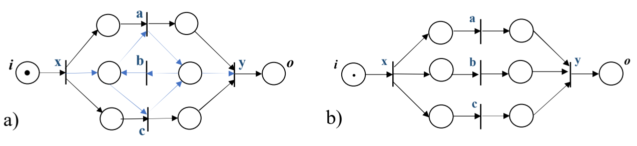

Fig. 1. Two WFNs that replay the log {xabcy, xcbay}.

In this research, we are interested in obtaining workflow nets (WFN) with high precision. The proposed method can produce a class of block-structured WFN [6], where the net is the alternative composition of WFN (blocks) that execute once the transitions, and that eventually, may include cycles. Figure 2 shows a WFN exhibiting this feature.

In this approach, first, the event log λ is partitioned into sub-logs λ_m having a high number of common events. Then, every λ_m is treated for detecting iterative sub-sequences in the traces (cycles), which are removed from the traces. Afterwards, a WFN *N* with reduced surplus behaviour is created; then, the removed cycles are added. Finally, the subnets are gathered by applying the alternative composition (Or-split from place *i*, and Or-join to place *o*).

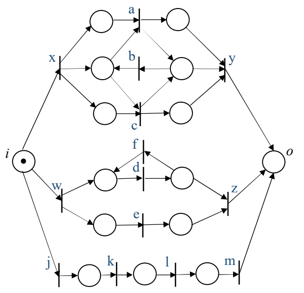

Fig. 2. Example of the addressed WFN subclass.

In this paper we focus on the synthesis of WFNs from sub-logs of traces including the same events; such nets execute every event only once. This is the main feature of the complete method.

The main idea of this discovery technique involves several steps: i) First, a WFN $N_<$ that preserves the event precedence in the traces is determined. ii) Then, a WFN N_C is built; it can execute once all the events in any order. iii) The parallel composition of $N_<$ and N_C ($N_<||N_C$) is performed. iv) Finally, several structural simplifications may be applied.

The paper is organized as follows. Section II includes basic definitions of Petri nets and formulates the discovery problem. Section III describes the method for determining the block WFN. Finally, Section IV discusses the structural complexity of the discovered models through several case studies with diverse precision.

II. Preliminaries

A. Petri nets

Basic concepts and notation of ordinary PN used in this paper are recalled [1].

Definition 1. An ordinary Petri Net structure G is a bipartite digraph represented by the 3-tuple $G = (P, T, F)$; where:

- $P = \{p_1, p_2, ..., p_{|P|}\}$ and $T = \{t_1, t_2, ..., t_{|T|}\}$ are finite sets of nodes named places and transitions respectively;
- $F \subseteq P \times T \cup T \times P$ is a relation representing the arcs between the nodes. For any node $x \in P \cup T$, ${}^\bullet x = \{y | (y, x) \in F\}$ and $x^\bullet = \{y | (x, y) \in F\}$.
- The incidence matrix of G is $C = [c_{ij}]$; where $c_{ij} = -1$ if $(p_i, t_j) \in F$ and $(t_j, p_i) \notin F$; $c_{ij} = 1$ if $(t_j, p_i) \in F$ and $(p_i, t_j) \notin F$; $c_{ij} = 0$ otherwise.

Definition 2. A *marking* $M: P \rightarrow \mathbb{N}^{\geq 0}$ determines the number of tokens within the places; where $\mathbb{N}^{\geq 0}$ is the set of non-negative integers. A marking M, usually denoted by a vector $(\mathbb{N})^{|P|}$, describes the current state of the modelled system.

A *Petri Net system* or Petri Net (PN) is the pair $N = (G, M_0)$, where G is a PN structure and M_0 is an initial marking. $R(G,M_0)$ denotes the set of all *reachable markings* from M_0.

Definition 3. A PN system is *1-bounded* or *safe* iff, for any $M_i \in R(G,M_0)$ and any $p \in P$, $M_i(p) \leq 1$. A PN system is *live* iff, for every reachable marking $M_i \in R(G,M_0)$ and $t \in T$ there is a $M_k \in R(G,M_i)$ such that t is enabled in M_k.

Definition 4. *A PN state machine* is a subclass of *PN* whose structure fulfils $|{}^\bullet t_j| = |t_j^\bullet| = 1$, $\forall t_j \in T$. *A marked graph* is a subclass of *PN* whose structure fulfils $|{}^\bullet p_i| = |p_i^\bullet| = 1$, $\forall p_i \in P$.

Definition 5. A *t-invariant Y_i of a PN* is a non-negative integer solution to the equation $CY_i = 0$. The *support* of Y_i (*t-support*) denoted as $<Y_i>$ is the set of transitions whose corresponding elements in Y_i are positive. Y is *minimal* if its support is not included in the support of other t-invariant. A *t-component* $G(Y_i)$ is a subnet of *PN* induced by a $<Y_i>$: $G(Y_i) = (P_i, T_i, F_i)$, where $P_i = {}^\bullet<Y_i> \cup <Y_i>^\bullet$, $T_i = <Y_i>$, $F_i = (P_i \times T_i \cup P_i \times T_i) \cap F$. In a t-invariant Y_i, if we have initial marking (M_0) that enables a $t_i \in <Y_i>$, when t_i is fired, then M_0 can be reached again by firing only transitions in $<Y_i>$.

B. Workflow nets

Definition 6. A *WorkFlow net* (WFN) N is a subclass of *PN* owning the following properties [1]: (i) it has two special places: i and o. Place i is a source place: ${}^\bullet i = \emptyset$, and place o is a sink place: $o^\bullet = \emptyset$. (ii) If a transition t_e is added to *PN* connecting place o to the place i, then the resulting *PN* (called *extended* WFN) is strongly connected.

Definition 7. A WFN (N, M_0) is said to be *sound* iff any marking $M_i \in R(N, M_0)$, $o \in M_i \rightarrow M_i = [o]$ and $[o] \in R(N, M_i)$ and (N, M_0) contains no dead transitions. An *extended* WFN sound is live and bounded. A WFN can represent a process behaviour by associating task labels to some transitions.

Definition 8. A labelled WFN is a four-tuple (N, M_0, Σ, L), where Σ is a finite set of tasks labels and $L: T \rightarrow \Sigma$ is the labelling function.

C. Problem statement and model qualities

1) Process discovery

The standard formulation of the process discovery problem for WFN is given below.

Definition 9. Given a workflow log $\lambda = \{\sigma_1, \sigma_2, \ldots, \sigma_r\}$, where $\sigma_i \in \Sigma^*$ generated by an unknown sound WFN, the discovery problem consists of building a sound WFN N using $|\Sigma|$ transitions, i.e, the labelling function is bijective. N must replay properly all the traces in λ (from $M_0 = [i]$, every trace makes reach $M_r = [o]$). The number of places is unknown.

Although λ may be a multiset of traces, only different traces are considered because we are interested in the language represented in λ.

2) Quality measures

The quality of the discovered **N** can be evaluated through four measures: *fitness, precision, simplicity,* and *generalisation* [3, 7]. Regarding the two first measures, fitness (replay fitness) evaluates the number of traces that are not represented in N: $(\lambda \backslash \mathcal{L}(N))$, whilst precision assesses the overrepresentation of λ in N: $(\mathcal{L}(N) \backslash \lambda)$, i.e. the surplus language (see Figure 3). In general, most of the discovery methods yield models that represent the entire event log.

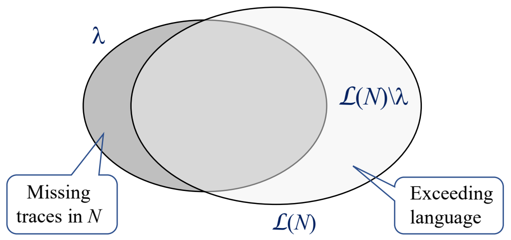

Fig. 3 Relation between λ and the language of N

III. Building highly precise WFN

A. Approach

The proposed method obtains a subclass of WFN with reduced exceeding language; such a model is composed of subnets as shown in Figure 1. These subnets are WFN that may include cycles.

The technique presented in this paper focuses on the discovery of a WFN N that executes once every event; this net has no cycles; this method is a base component of the complete method.

The approach held for this discovery technique consists of building two WFN, which are composed by merging the transitions labelled with the same event symbols, as described in Section I. The steps are informally presented through the following example.

Example 1. Consider the event log λ_1={xabcy, xcbay}; the steps are as follows.

First, consider the event precedence relation $R_<$ in the traces of λ_1. Figure 4 shows this relation as a graph (Σ, $R_<$). Then, from this relation one must obtain the WFN $N_<$ shown in Figure 5, which executes the traces in λ_1 and some other traces. Notice that $N_<$ is a PN state machine.

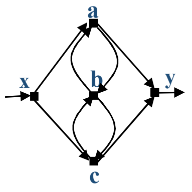

Fig. 4. $R_<$ of λ_1

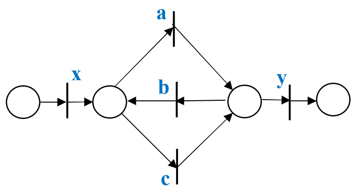

Fig. 5. WFN $N_<$ corresponding to $R_<$.

Then, the WFN N_C that constraints the firing of each transition once is built. It is shown in Figure 1.b; N_C replays λ_1, but also other traces; indeed, the concurrent events allow the execution of 3! different traces that fire all the transitions once. In this example N_C is a PN marked graph.

Afterwards, the WFN N that replays λ_1 is obtained by the parallel composition of $N_<$ and N_C: N=$N_<$||N_C. N is shown in Figure 6. Notice that N_C constrains the events in $N_<$ to be fired once. In this example $\mathcal{L}$(N)\ λ_1=$\varnothing$. Finally, a further analysis on the structure of N may lead to a simpler WFN equivalent to N (Figure 1.a).

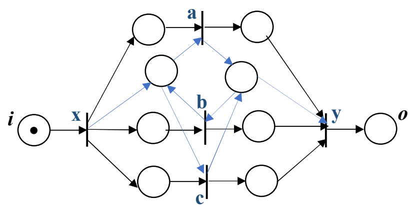

Fig. 6. Final WFN N=$N_<$||N_C

B. *Building a WFN that fires n transitions*

Now, we will reformulate the discovery problem for the subclass of WFN dealt.

1) Discovery sub-problem

The discovery problem given in Definition 9 is constrained by the following assumptions:
i) $|\sigma_1|=|\sigma_2|= \dots =|\sigma_r|=|\Sigma|$;
ii) Every σ_i starts with x and ends with the y; x,y$\in\Sigma$.

Thus, the WFN N to discover has no cycles and execute once every event in Σ to replay all the traces in λ.

2) Building $N_<$

We are addressing the case where $N_<$ is a class of PN state machine, which has to be built from the event precedence relationship in the traces of λ. First, we need to express the precedence between events in the traces.

Definition 10. The relation $R_<\subseteq\Sigma\times\Sigma$, denoted also as <, specifies the precedence relation between events in traces. $R_<=\{(a,b) \mid a$ precedes b in σ_i, $\forall\sigma_i \in \lambda\}$; $a\, R_<\, b$ is written also as $a < b$. $R_<$ can be can be represented by a digraph $G_<= (\Sigma, R_<)$.

In Example 1, $R_<$={($\varnothing$, x), (x,a), (a,b), (b,c), (c,y), (x,c), (c,b), (b,a), (a,y), (y, $\varnothing$)}; it is represented by the graph in Figure 3.

Definition 11. $N_<$ is a sound WFN that has the structure of a PN state machine, where all the transitions that have different label of Σ must be fired exactly once.

Now, $N_<$ can be obtained by relying transitions labelled with events in Σ by considering $R_<$ as causal dependencies between events [e_i, e_j] $\forall$(e_i, e_j) $\in R_<$. A causal dependency [e_i, e_j] is represented by a place in the PN by relating transitions with labels e_i and e_j [8].

A simple procedure to build $N_<$ from $R_<$, which represents all the causal dependencies [e_i, e_j], is proposed.

Step 1. Consider only |Σ| transitions. Then, $\forall$(e_i, e_j) $\in R_<$ rely the transitions through a place; use the output place of e_i if it already exists.

In Example 1, if the events precedence derived from σ_1=xabcy are first taken, the net built is shown in Figure 7. Then, the remainder dependencies in $R_<$ issued from σ_2=xcbay are added; $N_<$ is shown in Figure 8. Notice that the obtained $N_<$, although the paths follow the precedence of the events in the traces, does not replay the traces in λ_1; it is deadlocked since And-join substructures are created; thus, it is not a PN state machine.

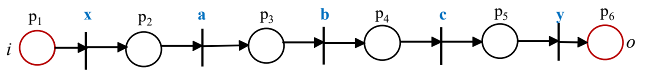

Fig. 7. Partial building of $N_<$ from σ_1 of Example 1.

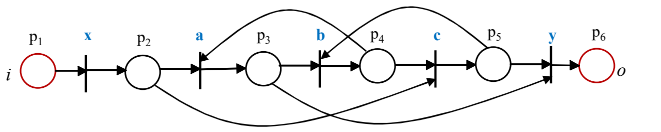

Fig. 8. $N_<$ derived from $R_<$ of Example 1 using |Σ| transitions

Step 2. The WFN $N_<$ built using |Σ| transitions must be transformed into a PN state machine. This can be performed using a simple structural rewriting procedure [9] described below.

Rewriting N<. For every transition t such that $|{}^\bullet t|>1$ and L(t)=e_j, the dependency [e_i, e_j] must be rewritten by adding a new transition t' where L(t') = e_j; then, new arcs are added from ${}^\bullet$t to t' and from t' to $t^\bullet$. In this example, the rewritten net shown in Figure 9 has four additional transitions; this WFN replays the traces in λ_1.

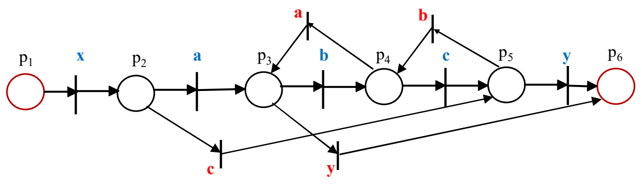

Fig. 9. $N_<$ with duplicated transitions

Step 3. The obtained $N_<$ can be reduced by applying a minimisation procedure for finite automata [10]. Although $N_<$ is close to a finite automaton since it is a PN state machine, the next state function δ must be completed to be analogous to a deterministic finite automaton (DFA). Then, it can be handled by a standard minimisation procedure.

The extension of δ consists of a) adding self-loop transitions to places: for every event e_i that leads to a place p_k, a new transition corresponding to $\delta(p_k, e_i)=p_k$ is added; b) adding to places, output (sink) transitions corresponding to events not allowed in the traces. In the example, the completed $N_<$ is shown Figure 10; it fulfils the features of an FDA.

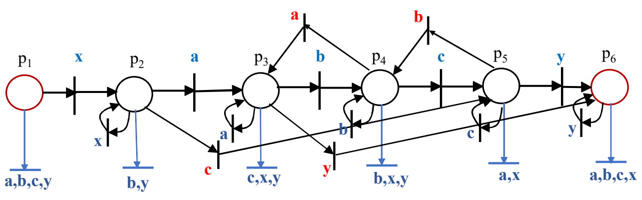

Figure 10. Completed $N_<$.

In the example, the application of such a minimisation procedure determines that places p_2 and p_4 are indistinguishable; also, places p_3 and p_5. These equivalences lead to the $N_<$ shown in Figure 5; this model is bisimilar to the above model.

Proposition 1. $N_<$ replays all the traces in λ.
Proof. $R_<$ captures all the possible event precedence in the traces of λ; then, *Step1* translates such precedence using $|\Sigma|$ transitions. In *Step 2*, the rewriting of $N_<$ using new transitions with duplicated labels guarantees the event precedence and avoids the And-join substructures that cause deadlocks; furthermore, this transformation leads to a PN state machine. Finally, *Step 3* obtains a reduced model that preserves the language of $N_<$. □

Remark. $N_<$ is derived through a simple procedure from the event precedence relation $R_<$; it fulfils closely the standard definition of DFA for applying the minimisation procedure; transitions in loops and sink transitions are included to complete the next state function. However, this FDA has a special semantics in which all the events must be executed before reaching the state (place) *o*; thus, transitions in loops may not be fired if the associated event has been already fired.

Previous steps for building N< can be summarised in the procedure given below (Algorithm 1).

Algorithm 1. Building $N_<$

```
Input: $\Sigma$, $R_<$
Output: $N_<$, $\Sigma_<$
1. $P_< \leftarrow \varnothing$; $T_< \leftarrow \varnothing$;
2. $\forall i \in [1, |\Sigma|+1]$,
     $P_< \leftarrow P_< \cup \{p_i\}$;            // Defining $P_<$
3. $\forall e_j \in \Sigma$,
     $T_< \leftarrow T_< \cup \{t_j\}$;             // Defining $T_<$
4. $L(t_1) \leftarrow x$; $L(t_{|\Sigma|}) \leftarrow y$;        // Transition labelling
5. $\forall j \in [2, \Sigma-1]$,
     $L(t_j) \leftarrow e_j \in \Sigma \setminus \{x, y\}$
6. $F_< \leftarrow \{(p_1, t_1), (t_{|\Sigma|}, p_{|\Sigma+1|})\}$; // Flow definition
7. $R_< \leftarrow R_< \setminus \{(\varnothing, x), (y, \varnothing)\}$;
8. $\forall (e_k, e_m) \in R_<$,                //from event precedence
     $F_< \leftarrow F_< \cup \{(t_r, p_u), (p_u, t_s)\} \mid L(t_r)=e_k, L(t_s)=e_m$;
         Reuse $p_u$ for every treated $(e_k, \bullet)$
                                    // $N^1=(P_<, T_<, F_<, L)$
9. $u \leftarrow |\Sigma|+1$; $\Sigma_< \leftarrow \Sigma$;          // Duplicate transitions
   $\forall t_j \in T_< \mid |{}^\bullet t_j|>1$
      $T_< \leftarrow T_< \cup \{t_u\}$; $L(t_u) \leftarrow L(t_j)$ ; // Duplicate label
      $\Sigma_< \leftarrow \Sigma \cup_M L(t_j)$        //Multiset union
      $F_< \leftarrow F_< \cup \{(p_v, t_u), (t_u, t_j^\bullet)\} \mid p_v \in {}^\bullet t_j$;
      $F_< \leftarrow F_< \setminus \{(p_v, t_j)\}$;
      $u \leftarrow u+1$;
10. $N^2 \leftarrow$Complete-δ$(P_<, T_<, F_<, L)$
11. $N_< \leftarrow$Minimise$(N^2)$
12. Return $N_<$, $\Sigma_<$
```

3) Building N_C

In Example 1 the built $N_<$ has $|\Sigma|$ transitions; however, in general, the minimized net may contain more transitions using replicated labels from Σ as shown in the following example.

Example 2. Consider the log λ_2={xabcy, acbay, xbcay}; the corresponding event precedence graph $G_<$ and the obtained $N_<$ are shown in Figure 11.

Remark. Since $N_<$ has transitions labelled with duplicated event symbols, then N_C requires also the same number of transitions. In this example, N_C is shown in Figure 12.

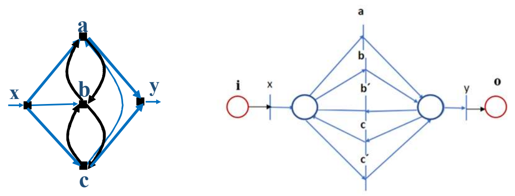

Fig. 11. $G_<$ and $N_<$ of Example 2.

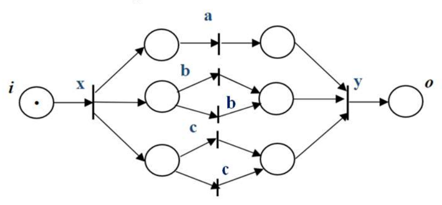

Fig. 12. N_C of Example 2.

After the previous remark on the number of transitions in $N_<$ we can state the procedure to build N_C.

Definition 12. Let $\Sigma=\{x, y, e_1, e_2, \dots, e_{n-2}\}$ and $\Sigma_<$ be the alphabet and the multiset on $\Sigma\backslash\{x, y\}$ used in $N_<$ respectively. Considering $\Sigma_<$ lexicographically ordered, N_C is constructed using $|\Sigma_<|+2$ transitions and $2|\Sigma|-2$ places as follows:

i) Add to F the arcs (p_1, x) and (y, p_n)

ii) $\forall e_i \in \Sigma_<, \forall e_i=e_{i+1}$, add to F the arcs (x, p_k) and (p_k, e_i), where $k=2,\dots |\Sigma-1|$

iii) $\forall e_i \in \Sigma_<, \forall e_i=e_{i+1}$, add to F the arcs (e_i, p_s) and (p_s, y), where $s=|\Sigma|, \dots 2(|\Sigma|-1)$.

In Example 2, the N_C of Figure 12 is built from $\Sigma_<=\{a, b, b, c, c\}$. It is clear that N_C allows the firing of $|\Sigma|-2$ different labelled transitions once. A procedure can be derived straightforward from Definition 11 (Algorithm 2).

Algorithm 2. Building N_C

```
Input:   $\Sigma = \{x, y, e_1, e_2, \dots, e_{n-2}\}$, $\Sigma_< = \{e_1, e_2, \dots, e_i\}$
Output : $N_C = (P_C, T_C, F_C)$,  $L_C$
1.  $P_C \leftarrow \emptyset$; $T_C \leftarrow \emptyset$;
2.  $r \leftarrow 2|\Sigma| - 2$; $s \leftarrow |\Sigma_<| + 2$;    // $|P_C|$ and $|T_C|$
3.  $\forall i \in [1, r]$,
        $P_C \leftarrow P_C \cup \{p_i\}$          //Defining the set of places
4. $\forall j \in [1, s]$,
        $T \leftarrow T_C \cup \{t_j\}$          // Defining the set of transitions
5. $L_C(t_1) \leftarrow x$ ; $L_C(t_s) \leftarrow y$;     // Labeling of transitions
6. $\forall j \in [2, s-1]$
        $L_C(t_j) \leftarrow$ j-th item of $\Sigma_<$; // Labeling of transitions
7. $F_C \leftarrow \{(p_1, t_1), (t_s, p_r)\}$;
8. $\forall k \in [2, s/2]$
        $F_C \leftarrow F_C \cup \{(t_1, p_k)\}$; // Arcs from x
9. $\forall k \in [(s/2)+1, s-1]$
        $F_C \leftarrow F_C \cup \{(p_k, t_s)\}$; // Arcs to y
10. $\forall k \in [2, r/2]$
      $\forall j \in [2, s/2]$
        $F_C \leftarrow F_C \cup \{(p_k, t_j), (t_j, p_u) \mid u=k+(r-1)/2\}$;
                          // Arcs relating $e_i \in \Sigma_<$
11. Return $N_C = (P_C, T_C, F_C)$,  $L_C$
```

4) The final WFN N

N is obtained by the parallel composition of $N_<$ and N_C ($N_<||N_C$), where the places i and o, and the transitions that have the same label in both nets are merged. Figure 13 shows the resulting net $N = N_<||N_C$ of Example 2, where $\mathcal{L}(N)$ ={xabcy, acbay, xbcay, xacby}; the last trace is not in λ_2.

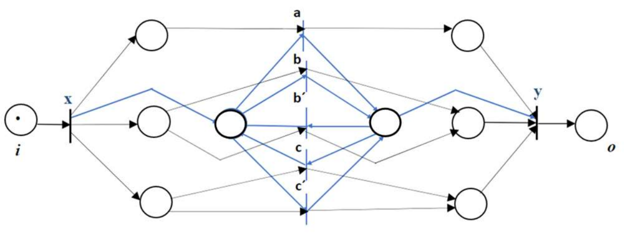

Figure 13. Parallel composition $N_<||N_C$ of Example 2.

Proposition 2. The discovered WFN N built using the described technique replays the event log λ and a reduced quantity of traces not contained in λ.

Proof. $N_<$ is created to execute the traces in λ. Since $N_<$ has a minimal number of places, it may contain cycles; hence it may replay other traces. However, N_C constraints the firing of $|\Sigma|$ transitions with different labels in each trace execution. □

Eventually, N can be simplified according to the substructures created. Removal of implicit places is the structural simplification currently applied. The discovery procedure is summarised in Algorithm 3.

Algorithm 3. Discovering N

```
Input: $\lambda$, $\Sigma$
Output : $N$
1. Obtain $R_<$ from $\lambda$;
2. Determine $N_<$;  Determine $N_C$;
3. Perform the composition $N = N_<|| N_C$;
3. Return $N$
```

IV. Discussion

A. Precision vs. simplicity

The proposed approach for discovering N aims to fulfil the event precedence $R_<$ issued from λ through $N_<$, which may represent more traces than those in λ; then, N_C avoids the execution of some of these traces. It is clear that the final structure of N, even with simplifications, can be complex when $|\lambda|$ and $|\Sigma|$ are large; this is the cost of building highly precise models.

In contrast, methods that pursue generalization of the discovered models yield structurally simple models, which allow the execution of many more traces. For example, the complete model N of Example 2, shown in Figure 14, has a precision of 0.75, since $|\mathcal{L}(N)|=4$. On the other hand, the WFN of Figure 5, obtained through Cominer [11], replays the log λ_2; it is evidently a simpler model, but its precision is 0.5. Below two more examples are included.

Example 3. For the log λ_3={xabcdey, xacdbey} the WFN obtained with the proposed method is shown in Figure 14.a; it has a precision equal to 1.0. The model obtained through Cominer is depicted in Figure 14.b; it replays three traces, then its precision is 0.666.

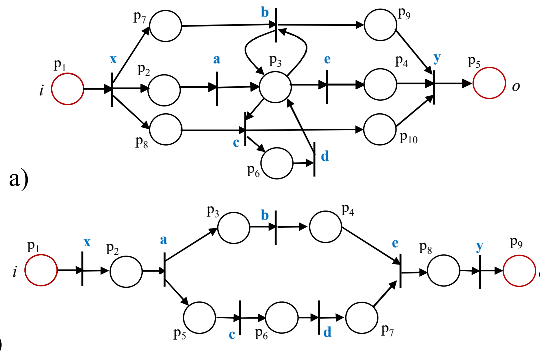

Figure 14. WFN of the Example 3.

Example 4. For the log λ_3={xabcdy, xbadcy} the WFN obtained with the proposed method is shown in Figure 15.a; its precision is 0.5 because $|\mathcal{L}(N)|$=4. Instead, the model obtained with Cominer, shown in Figure 15.b, has a precision of 0.1666 since it replays 12 different traces.

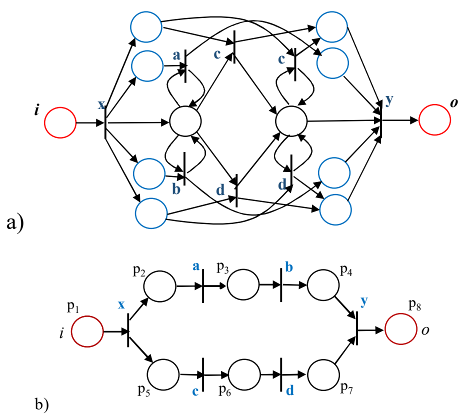

Fig. 15. WFN of Example 4.

B. *Dealing with iterative traces*

The processing event logs that include traces involving cycles in the WFN can be performed in the steps outlined below.

First, the iterative subtrace is detected and the cyclic part is removed. For example, the trace *xabfbcdy* executes a cycle *bfb*; then, the trace without iteration becomes *xabcdy*.

Second, $N_<$ is obtained from the event log containing traces without iterations, and later the cyclic part is added.

Afterwards, N_C must be conceived to allow either an arbitrary number of executions of cycles found in the traces or the execution of the maximum number of cycles in each iterative trace in the log. In the first case, the cycling subsequence is not restricted by N_C; thus, the language of N will be infinite. In the second case, the iterations are enabled and bounded by N_C, but some places must be emptied when the place o is marked.

C. *On testing the method*

The proposed method is at an early stage, thus, it is suitable to perform the tests under a rediscovery scheme, where known sound WFNs are used to generate artificial logs, which fulfil the current assumptions. This strategy allows testing the technique in a controlled manner by proposing models with diverse structures and sizes.

V. Conclusions

We have presented a method that addresses the problem of discovering highly precise WFN. The proposed approach deals with a subclass of models that seems to be restrictive, but such models are often discovered when event data from business processes are treated; a block net is built from the behaviour of a variant in the process. To our knowledge, there are no other techniques that pursue the reduction of the exceeding language of the discovered WFNs.

The core of the proposed technique focuses on the enforcement in $N_<$ of the event precedence $R_<$ stated by the traces of λ. $R_<$ induces tracking pre-emptively the traces to avoid other behaviours. Then, N_C constraints, even more, the language of $N_<$. Consequently, the method attains the purpose of reducing the surplus language of the final model N.

Current research addresses the case where there are traces that represent iterative behaviour, which leads to cycles in the WFN. Also, the issue where clusters of traces do not have identical event alphabet but share many event symbols is studied.

VI. Acknowledgment

Author Marisol Vera-Arellano is supported by CONACYT (Mexico) under Grant No. 1075533.

VII. References

[1] Van der Aalst, W., Weijters, T. and Maruster, L. Workflow mining: discovering process models from event logs. *IEEE Trans. on Knowledge and Data Eng.*, 16(9), pp.1128-1142. (2004).

[2] W. P. van der Aalst, Process Mining: Discovery, Conformance and Enhancement of Bussiness Processes, Springer 2011.

[3] Estrada-Vargas, A., Lopez-Mellado, E. and Lesage, J-J. A Comparative Analysis of Recent Identification Approaches for Discrete-Event Systems. *Mathematical Problems in Engineering*, 2010, pp.1-21.

[4] Cabasino, M., Darondeau, P., Fanti, M. and Seatzu, C. Model identification and synthesis of discrete-event systems. In: M. Zhou, H. Li and M. Weijnen, ed., *Contemporary Issues in Systems Science and Engineering*. IEEE/Wiley Press Book Series. (2013).

[5] López-Mellado, E., X. Morán-Soltero. Counting Finite Transition Sequences of Block Structured Workflow Nets, IFAC-PapersOnLine, Volume 53, Issue 4, 2020, Pages 193-198,

[6] Barragán-Pérez, R. E. López-Mellado, Qualitative Assessment of the Exceeding Behaviour of Discovered Petri Nets, IFAC-PapersOnLine,Volume 51, Issue 7, 2018, Pages 172-178.

[7] Tapia-Flores, T., E. López-Mellado, A. P. Estrada-Vargas, and J. J. Lesage (2018), “Discovering Petri Net Models of Discrete Event Processes by Computing T-invariants”. IEEE Transactions on Automation Science and Engineering. 05/2018; 15(3):992-1003, DOI:10.1109/TASE.2017.2682060.

[8] Pomares-Angelino, R., López-Mellado, E. Discovering petri nets including silent transitions. A repairing approach based on structural patterns. *Discrete Event Dyn Syst* 32, 291–315 (2022). https://doi.org/10.1007/s10626-021-00358-w

[9] Hopcroft, J.E., J. D. Ullman Introduction to Automata Theory, Languages and Computation. Addison-Wesley, 1976.

[10] Tapia-Flores, T., Rodríguez-Pérez, E., and López-Mellado, E. Discovering process models from incomplete event logs using conjoint occurrence classes. *ATAED@Petri Nets/ACSD*, pp. 31-46. (2016).