Model and Event Log Reductions to Boost the Computation of Alignments

Farbod Taymouri and Josep Carmona

Universitat Politècnica de Catalunya, Barcelona (Spain)
{taymouri,jcarmona}@cs.upc.edu

Abstract. The alignment of observed and modeled behavior is a pivotal issue in process mining because it opens the door for assessing the quality of a process model, as well as the usage of the model as a precise predictor for the execution of a process. This paper presents a novel technique for reduction of a process model based on the notion of *indication*, by which, the occurrence of an event in the model reveals the occurrence of some other events, hence relegating the later set as less important information when model and log alignment is computed. Once indications relations are computed in the model, both model and log can be reduced accordingly, and then fed to the state of the art approaches for computing alignments. Finally, the (macro)-alignment derived is expanded in these parts containning high-level events that represent a set of indicated events, by using an efficient algorithm taken from bioinformatics that guarantees optimality in the local parts of the alignment. The implementation of the presented techniques shows a significant reduction both in computation time and in memory usage, the latter being a significant barrier to apply the alignment technology on large instances.

1 Introduction

Nowadays many systems generate *event logs*, which are footprints left by process executions. *Process mining* delves into this information, and examines it to extract, analyze and enhance evidence-based process models [10]. One of the challenges in process mining is how to align a process model to a set of traces forming an event log. Given a trace representing a real process execution, an *optimal alignment* provides the best trace the process model can provide to imitate the observed trace [1]. Alignments are crucial for important metrics like fitness, precision and generalization [1,2].

This paper presents a model-based technique for reduction of a process model and observed behavior that both preserves the semantics of the process model and retains the information of the original observed behavior as much as possible. The technique is meant to fight the main problem current approaches for alignment computation have: the complexity both in space and time. The overall idea relies on the notion of *indication* between activities of the process model when it is represented as a Petri net. An indication relation between a set of transitions (indicated set) and another transition (indicator) denotes a causal

firing relation in the model, which expresses that the presence in any model's sequence of the indicator transition requires the presence of the indicated set as well. The notion of indication is inspired from the *reveals* relation from [3]. We use a well-known technique to find logically independent parts of a graph (known as SESEs in [7]), which are then used to gather indication relations efficiently. These relations dictate which parts of a process model are abstracted as a single, high-level node. Once the model is reduced, the observed trace to align is projected (hence, reduced as well) into the reduced model's alphabet. This way, not only the model but also the trace are reduced, which in turn makes the alignment techniques to be significantly alleviated, specially for well-structured process models where many indication relations may exist. Once alignments are computed, the final step is also an interesting contribution of this paper: to cast the well-known Needleman–Wunsch algorithm [6] to expand locally each high-level part of the alignment computed, using the indication relation.

2 Related Work

The seminal work in [1] proposed the notion of alignment, and developed a technique to compute optimal alignments for a particular class of process models. For each trace σ in the log, the approach consists on exploring the synchronous product of model's state space and σ. In the exploration, the shortest path is computed using the A^* algorithm, once costs for model and log moves are defined. The approach is implemented in ProM, and can be considered as the state-of-the-art technique for computing alignments. Several optimizations have been proposed to the basic approach to speed up and improve memory consumption. The recent work in [9] proposed a divide and conquer strategy based on Integer Linear Programming (ILP) approach to compute approximate alignments. Despite its memory and time efficiency, it cannot guarantee the obtention of an (optimal) alignment.

The work in [5] presented a decomposition approach using SESEs for conformance checking of the model and observed behavior. The proposed approach decomposes a given model to smaller parts via SESE and then applies conformance checking for each part independently. This technique is very efficient, but the result is decisional (a yes/no answer on the fitness of the trace). Recently [12] proposed a new approach which provides an algorithm that is able to obtain such an optimal alignment from the decomposed alignments if this is possible, which is called proper optimal alignment. Otherwise, it produces a so-called pseudo-alignment which as in the case of [9], may not be executable in the net.

The seminal work [4] first introduced the notion of *reveals relation*, which determines that whenever an action a happens, then the occurrence of another action b is inevitable. The notion of indication of this paper is inspired on the reveals relation.

The Refined Process Structure Tree (RPST), proposed by [11], is a graph parsing technique that provides well-structured parts of a graph. The resulting parse tree is unique and modular, i.e., local change in the local workflow graph

results in a local change of the parse tree. It can be computed in linear time using the method proposed in [8] which is based on the triconnected components of a given biconnected graph. The proposed approach only works with single sink, single source workflow graphs which hampers its applicability to real world problems with many sink, source nodes. The work in [7] presents a more efficient way to compute RPST which can deal with multiple source, sink workflow graphs.

3 Preliminaries

A *Petri Net* is a 3-tuple $N = \langle P, T, \mathcal{F} \rangle$, where P is the set of places, T is the set of transitions, $P \cap T = \emptyset$, $\mathcal{F} : (P \times T) \cup (T \times P) \to \{0, 1\}$ is the flow relation. Marking of a Petri net represents the number of tokens each place has. Given a node $x \in P \cup T$, its pre-set and post-set (in graph adjacency terms) are denoted by ${}^\bullet x$ and $x^\bullet$ respectively. WF-net is a Petri net where there is a place *start* (denoting the initial state of the system) with no incoming arcs and a place *end* (denoting the final state of the system) with no outgoing arcs, and every other node is within a path between *start* and *end*. Fig. 1(a) represents a WF-net. Given an alphabet of events $T = \{t_1, \ldots, t_n\}$, a trace is a word $\sigma \in T^*$ that represents a finite sequence of events. An *event log* $L \in \mathcal{B}(T^*)$ is a multiset of traces[1]. An *alignment* is represented by a two-row matrix where the top and bottom rows represent moves on log and the model respectively. For example given trace $t_1 t_4 t_2 t_5 t_8$ and the model in Fig. 1(a), an example of alignment is:

$$\alpha = \begin{array}{|c|c|c|c|c|c|c|c|} \hline t_1 & \bot & t_4 & t_2 & t_5 & \bot & t_8 \\ \hline t_1 & t_2 & t_4 & \bot & t_5 & t_7 & t_8 \\ \hline \end{array}$$

Let $F \subseteq E$ represents a set of edges of a directed graph $\langle V, E, \ell \rangle$, $G_F = \langle V_F, F \rangle$ is the subgraph formed by by F if V_F is the smallest set of nodes such that G_F is a subgraph. A node in V_F is *boundary* with respect to G_F if it is connected to nodes in V_F and in $V - V_F$, otherwise it is *interior*. A boundary node u of G_F is an *entry* node if no incoming edge of u belongs to F or if all outgoing edges of u belong to F. A boundary node v of G_F is an *exit* node of G_F if no outgoing edge of v belongs to F or if all incoming edges of v belong to F. G_F with one entry and one exit node is called *SESE*. If a SESE contains only one edge it is called trivial. A SESE of G is called *canonical* if it does not overlap with any other SESEs of G, but it can be nested or disjoint with other SESEs. For example in Fig. 1(b) all SESEs are canonical, S_2 and S_4 are nested, S_3 and S_2 are disjoint. A WF-net can be viewed as a Workflow graph if no distinctions are made between its nodes. WF-graph of Fig. 1(a) is presented in Fig. 1(b). Let G be a graph, then its *Refined Process Structure Tree* (RPST) is the set of all canonical SESEs of G. Because canonical fragments are either nested or disjoint, they form a hierarchy. In a typical RPST, the leaves are trivial SESE and the root is the whole graph. Fig. 1(c) is the RPST of WF-graph in Fig. 1(b),

[1] $\mathcal{B}(A)$ denotes the set of all multisets of the set A.

S_1 which is the entire graph is at root and leaves are trivial SESEs which only contain one edge.

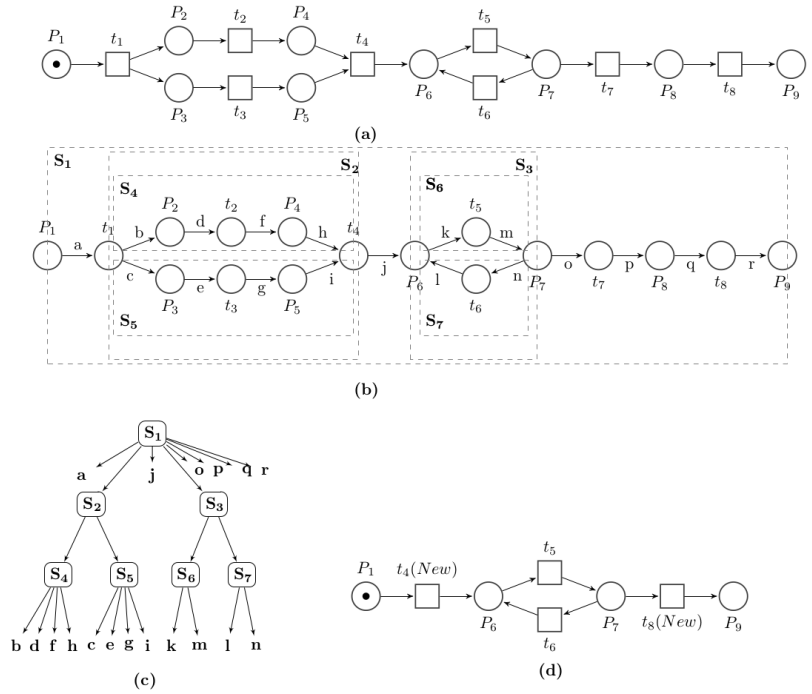


Fig. 1: (a) WF-net,(b) Workflow graph,(c) RPST, (d) Reduced WF-net

4 Overall Framework

Given a process model N, represented by a Petri net, and σ as observed behavior, the strategy of this paper is sketched in Fig. 2. We now provide descriptions of each stage.

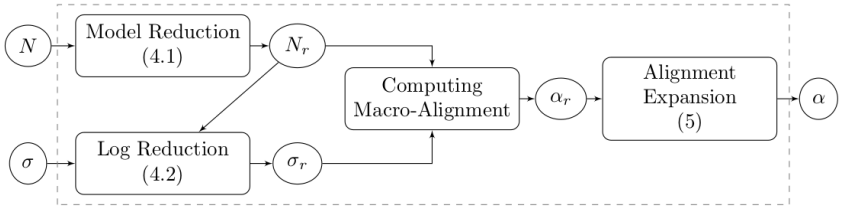


Fig. 2: Overall framework for boosting the computation of alignments

– *Model Reduction*: N will be reduced based on the notion of *indication* relation which results in N_r. It contains some abstract events representing the indicators of certain indicated sets of transitions. Section 5.1 explains it in detail.
– *Log Reduction*: Using the indication relations computed in the model, σ is projected into the remaining labels in N_r, resulting in σ_r. Section 5.2 describes this step.
– *Computing Alignment*: Given N_r and σ_r, approaches like [1] and [9] can be applied to compute alignments. At this point because both N_r and σ_r contain abstract events, the computed alignment will have them as well. We call it *macro-alignment*.
– *Alignment Expansion*: For each abstract element of a macro-alignment, the modeled and observed indications are confronted. Needleman–Wunsch algorithm [6] is adapted to compute optimal alignments. Section 6 will be centered on this.

5 Reduction of Model and Observed Behavior

5.1 The Indication Relation

Let us consider the model in Fig. 1(a). For any sequence of the model, whenever transition t_4 fires it is clear that transitions t_1, t_3, and t_2 have fired as well. Formally:

Definition 1 (Indication Relation). *Let $N = \langle P, T, \mathcal{F} \rangle$, $\forall t \in T$, indication is defined as a function, $I(t)$ where, $I : T \to [P(T)^+]^+$ $I(t)$ such that for any sequence $\sigma \in \mathcal{L}(N)$, if $t \in \sigma$ then $I(T) \in \sigma$. If $I(t) = \omega_1\omega_2...\omega_n$, then elements of ω_m precede the elements of ω_n in σ for $1 \leq m < n$. $I(t)$ is called linear if it contains only singleton sets, i.e. $\forall \omega_i \in I(t), |\omega_i| = 1$ otherwise it is non-linear.*

For example in Fig. 1(a), $I(t_4) = \{t_1\}\{\{t_2\},\{t_3\}\}\{t_4\}$ (non-linear) and $I(t_8) = \{t_7\}\{t_8\}$ (linear). SESEs are potential candidates for identifying indication relations inside a WF-net: the exit node of a SESE is the potential indicator of the nodes inside the SESE. Since entry/exit nodes of a SESE can be either place or transitions, SESEs are categorized as (P,P), (P,T), (T,P) or (T,T). In case the SESE is linear, indication relations can be extracted easily and the corresponding SESE is reduced (see Fig.3).

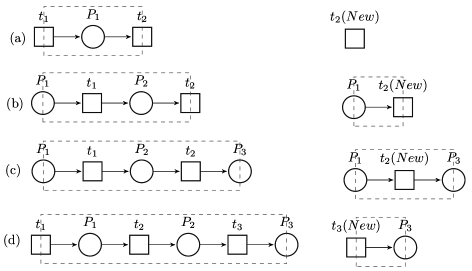


Fig. 3: Linear SESEs and their reduction.

Non-linear cases are decomposed into linear ones so that indication relations can be computed directly on the linear components extracted. After that, the indication relation of the corresponding linear SESEs are computed and they are

reduced as well. This procedure should be done with caution to avoid reaching a deadlock situation. Hence a post verification must be done after reduction of these linear parts. Informally, the verification is only needed for particular type of linear SESEs ((T,T)), and consists on validating the property of the SESE after the reduction. Notice the verification is necessary in these cases because, non-linear SESEs may contain linear indications at nested level, but which cannot be extracted due to choice or loop constructs (see See Fig. 4).

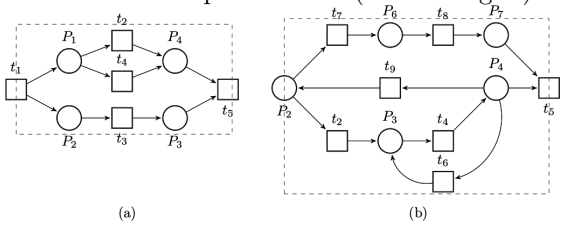


Fig. 4: (a) Non-Linear (T,T), (b) Non-Linear (P,T). Here indication relations cannot be computed at the whole SESE level, i.e., in (a) t_5 does not indicate neither t_2 nor t_4.

The reduction schema is depicted in Fig. 5. From the RPST, a top-down approach is applied that searches for indication-based reductions that do preserve the language of the initial model, once the net is expanded back, i.e., the language of the model must be preserved after reduction. As mentioned above, the reduction of non-linear SESEs must be done alongside by a post verification; for instance, Fig. 6 shows that in spite of the indication arising from SESE S_2, the net cannot be reduced without changing the language. To put it another way, this reduction will cause a deadlock in the reduced model, and hence must be avoided. Looking at the reduced result in Fig. 6, transition $t_5(New)$ will never fire because after the reduction. Notice that the reduction can be applied more than once till saturation (hence the arc back from the node "Reduced WF-net" to the node "WF-net").

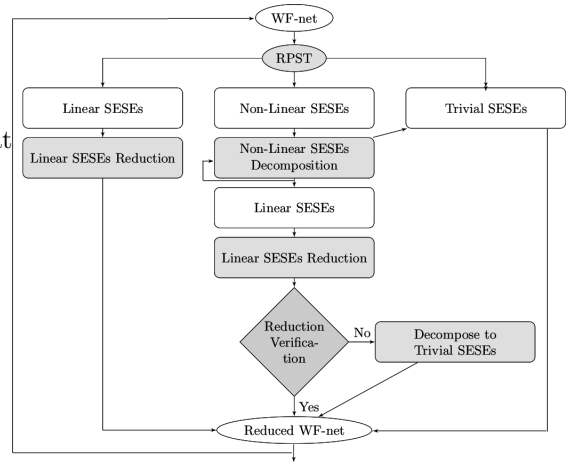


Fig. 5: Schema for reduction of a WF-net.

Fig. 7 shows an example (for the sake of simplicity only linear SESEs are shown). Obviously, SESE S_2 is inherently a linear SESE but the rest come from the decomposition of non-linear SESEs. The reduction schema is as follows: Since S_2 is inherently a linear SESE, hence it can be reduced easily according to Fig. 3 without any post verification. The rest of linear SESEs also will be reduced

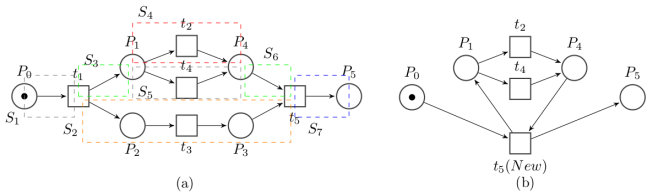
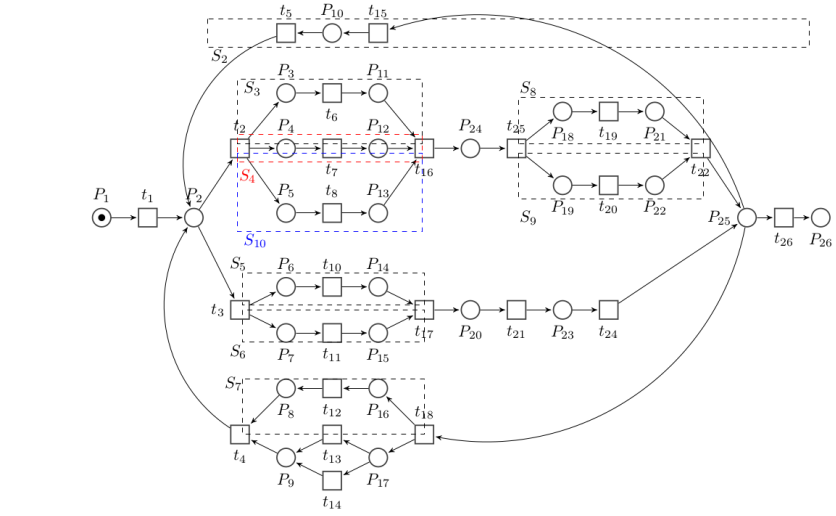


Fig. 6: Incorrect indication-based reduction: a deadlock is introduced.

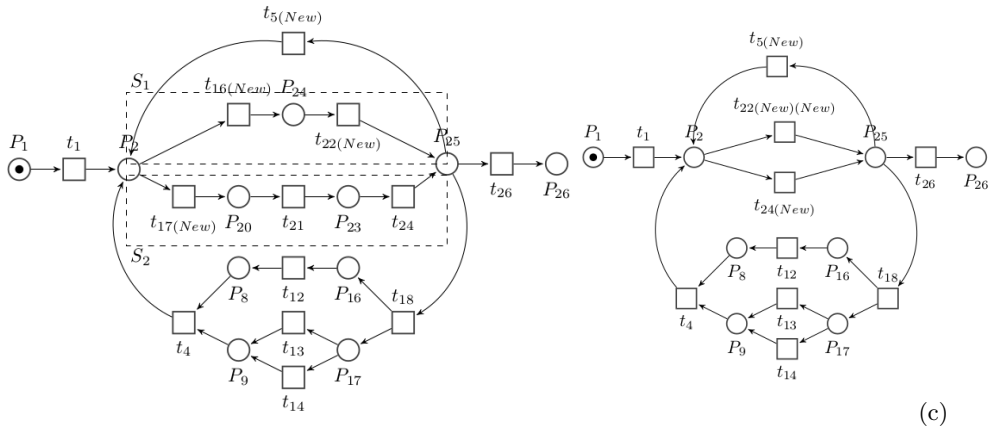
accordingly and post verification will be done after each reduction to check that no deadlock arises. One can see all reductions will be pass the verification, except for S_7, whose reduction induces a deadlock. Applying the reduction once, results in Fig. 7(b). As mentioned earlier, the reduction can be applied more than once until no reduction can be made. Fig. 7(c) is the reduction of the model in Fig. 7(b) and it is clear that no more reduction can be made from this model.

5.2 Reduction of Observed Behavior

Given a reduced model N_r and σ, we show how to produce σ_r. We will use the reduced model in Fig. 7(b) and the trace $\sigma_1 = t_1t_5t_3t_{11}t_{10}t_{21}t_6t_2t_7t_{16}t_{25}t_{19}t_{20}t_{26}$. The indication of $t_{5(New)}$ in Fig. 7(b) which is linear, equals to $\{t_5\}\{t_{15}\}$. So the *observed indication* for this abstract node is $\sigma_{1_{\downarrow_{I(t_{5(new)})}}} = t_5$. After computing the observed indication the reduced trace is $t_1t_{5(new)}t_3t_{11}t_{10}t_{21}t_6t_2t_7t_{16}t_{25}$ $t_{19}t_{20}t_{26}$. For $t_{17(New)}$, $I(t_{17(New)}) = \{t_3\}\{\{t_{10}\},\{t_{11}\}\}\{t_{17}\}$, which is non-linear and merged of two linear indications, $I_1(t_{17(New)})$=$\{t_3\}\{t_{10}\}\{t_{17}\}$ and$I_2(t_{17(New)})$= $\{t_3\}\{t_{11}\}\{t_{17}\}$. So the projection must be done for each linear indication separately, $\sigma_{1_{\downarrow_{I_1(t_{17(New)})}}} = t_3t_{10}$ and $\sigma_{1_{\downarrow_{I_2(t_{17(New)})}}} = t_3t_{11}$, removing transitions t_3, t_{10}, t_{11} and t_{17} from the current trace (notice that t_{17} does not appear originally, hence it is not projected). Finally, we need to insert $t_{17(New)}$ into the reduced trace; it will be inserted at the position of t_{10}, because the end transition of the abstract node, i.e. t_{17} did not happened in σ, and t_{10} happened last in σ. Therefore the reduced trace so far is $t_1t_{5(new)}t_{17(new)}t_{21}t_6t_2t_7t_{16}t_{25}$ $t_{19}t_{20}t_{26}$. By applying this process for the rest of abstract nodes ($t_{16(New)}$, $t_{22(New)}$), we reach $\sigma_r = t_1t_{5(new)}t_{17(new)}t_{21}t_{16(New)}$ $t_{22(New)}t_{26}$.



(a)



(b) (c)

Fig. 7: (a) Process model, (b) One-time reduced (c) Two-times reduced.

6 Expansion through Local Optimal Indication Alignments

After reducing a given process model and corresponding observed behavior, we can use current methods for computing alignments [1,9] to align N_r and σ_r, deriving α_r. For example the following is the macro alignment of $\sigma_{1r} = t_1 t_{5(new)} t_{17(new)} t_{21} t_{16(New)}\ t_{22(New)} t_{26}$ and the model in Fig. 7(b).

$$\alpha_r = \begin{array}{|c|c|c|c|c|c|c|c|} \hline t_1 & t_{5(New)} & t_{17(New)} & t_{21} & \perp & \perp & t_{16(New)} & t_{22(New)} & t_{26} \\ \hline t_1 & \perp & t_{17(New)} & t_{21} & t_{24} & t_{5(New)} & t_{16(New)} & t_{22(New)} & t_{26} \\ \hline \end{array}$$

When mapped to linear indications, indication of an abstract node and the corresponding observed indication are both sequence of events; hence for each linear combination of modeled/observed indication, we can adapt the dynamic programming approach from [6] (used in bioinformatics) to align two sequences. As example, we use indication of $t_{17(New)}$ and its observed indication computed in the previous section.

Table 1: Aligning modeled and observed indications

		t_3	t_{11}
	0	-1	-2
t_3	-1	0	-1
t_{11}	-2	-1	0
t_{17}	-3	-2	-1

(a)

		t_3	t_{10}
	0	-1	-2
t_3	-1	0	-1
t_{10}	-2	-1	0
t_{17}	-3	-2	-1

(b)

α_1=

t_3	t_{11}	$\perp$
t_3	t_{11}	t_{17}

α_2=

t_3	t_{10}	$\perp$
t_3	t_{10}	t_{17}

To achieve this goal, we create a table for each linear indication, where the first row and column are filled by observed and abstract node indications respectively, as depicted in Table 1(a), 1(b). The second row and second column are initialized with numbers starting from 0,-1,-2,..., they are depicted in yellow color. The task then is to fill the remaining cells as follows:
$SIM(t_i, t_j) = MAX(SIM(t_{i-1}, t_{j-1}) + s(t_i, t_j), SIM(t_{i-1}, t_j) - 1, SIM(t_i, t_{j-1}) - 1)$
Where $SIM(t_i, t_j)$ represents the similarity score between t_i and t_j. $s(t_i, t_j)$ is the substitution score for aligning t_i and t_j, it is 0 when they are equal and -1 otherwise.

The final step in the algorithm is the trace back for the best alignment. In the above mentioned example, one can see the bottom right hand corner in for example Table 1, score as -1. The important point to be noted here is that there may be two or more alignments possible between the two example sequences. The current cell with value -1 has immediate predecessor, where the maximum score obtained is diagonally located and its value is 0. If there are two or more values which points back, suggests that there can be two or more possible alignments. By continuing the trace back step by the above defined method, one would reach to the 0th row, 0th column. Following the above described steps, alignment of two sequences can be found.

Alignments can be represented by a sequence of paired elements, for example $\alpha_1 = (t_3, t_3)(t_{11}, t_{11})(\perp, t_{17})$, $\alpha_2 = (t_3, t_3)(t_{10}, t_{10})(\perp, t_{17})$ and final alignment which represent the non-linear indication is $\alpha = (t_3, t_3)\{(t_{11}, t_{11}), (t_{10}, t_{10})\}(\perp, t_{17})$. This information is booked for each abstract node.

After computing local alignments for abstract nodes, we can use them to expand corresponding abstract nodes in a given α_r. The policy of expansion depends on whether the abstract node is in synchronous or asynchronous move.

In α_r, $t_{17(New)}$ is in a synchronous move so we can expand it by its local alignment, which results in:

α=

t_1	$t_{5(New)}$	t_3	t_{11}	t_{10}	$\perp$	t_{21}	$\perp$	$\perp$	$t_{16(New)}$	$t_{22(New)}$	t_{26}
t_1	$\perp$	t_3	t_{11}	t_{10}	t_{17}	t_{21}	t_{24}	$t_{5(New)}$	$t_{16(New)}$	$t_{22(New)}$	t_{26}

The same story also happens for $t_{16(New)}$ and $t_{22(New)}$, which results in:

$$\alpha=\begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|}\hline t_1 & t_{5(New)} & t_3 & t_{11} & t_{10} & \perp & t_{21} & \perp & \perp & t_6 & t_2 & t_7 & \perp & \perp & t_{16} & t_{25} & t_{19} & t_{20} & \perp & t_{26}\\\hline t_1 & \perp & t_3 & t_{11} & t_{10} & t_{17} & t_{21} & t_{24} & t_{5(New)} & \perp & t_2 & t_7 & t_6 & t_8 & t_{16} & t_{25} & t_{19} & t_{20} & t_{22} & t_{26}\\\hline\end{array}$$

On the other hand $t_{5(New)}$ in α_r is a asynchronous move both on the model and observed trace. The policy of expansion is to expand move on log and move on model independently. To put it in another way, move on log will be expanded using observed indication and move on model will be expanded using the abstract node' indication, which results:

$$\alpha=\begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|}\hline t_1 & t_5 & t_3 & t_{11} & t_{10} & \perp & t_{21} & \perp & \perp & \perp & t_6 & t_2 & t_7 & \perp & \perp & t_{16} & t_{25} & t_{19} & t_{20} & \perp & t_{26}\\\hline t_1 & \perp & t_3 & t_{11} & t_{10} & t_{17} & t_{21} & t_{24} & t_{15} & t_5 & \perp & t_2 & t_7 & t_6 & t_8 & t_{16} & t_{25} & t_{19} & t_{20} & t_{22} & t_{26}\\\hline\end{array}$$

7 Experiments

The technique presented in this paper has been implemented in Python as a prototype tool. The tool has been evaluated over different family of examples, alongside with the state of the art techniques for computing alignments [9] ($ILP.R$), [1] (A^*). We used benchmark datasets from [9], [5], and a new dataset.

Reduction of Models. Table 2 provides the results of one-time reduction by applying the proposed method to benchmark datasets. Significant reductions are found often. Obviously one can see that the results of reduction is more representative for models without loops or contain small loops, like (*Banktransfer*).

Table 2: Reduced benchmark datasets

Model	$\|P\|$ (Before)	$\|T\|$ (Before)	$\|\sigma\|_{avg}$ (Before)	$\|P\|$ (After)	$\|T\|$ (After)	$\|\sigma\|_{avg}$ (After)
prAm6	363	347	31	175(52%)	235(32%)	22(29%)
prBm6	317	317	43	188(40%)	225(29%)	33(23%)
prCm6	317	317	42	188(40%)	225(29%)	33(21%)
prDm6	529	429	248	270(49%)	248(42%)	148(40%)
prEm6	277	275	98	180(35%)	205(26%)	75(23%)
prFm6	362	299	240	181(50%)	172(42%)	137(42%)
prGm6	357	335	143	195(45%)	221(34%)	94(34%)
M_1	40	39	13	25(37%)	28(28%)	9(30%)
M_2	34	34	17	26(23%)	28(18%)	13(23%)
M_3	108	123	37	76(30%)	98(20%)	29(21%)
M_4	36	52	26	31(14%)	48(8%)	23(11%)
M_5	35	33	34	27(23%)	27(18%)	28(18%)
M_6	69	72	53	51(26%)	59(18%)	43(19%)
M_7	65	62	37	43(34%)	46(26%)	28(24%)
M_8	17	15	17	6(65%)	7(53%)	9(47%)
M_9	47	55	44	26(45%)	39(29%)	34(23%)
M_{10}	150	146	58	91(39%)	105(28%)	42(28%)
Bank-transfer	121	114	58	61(46%)	72(37%)	38(34%)

Table 3: Replaying of Computed Step-Sequences for ILP.R

Model	Cases	Replay% (Before) ILP.R	Replay% (After) ILP.R
prAm6	1200	100%	100%
prBm6	1200	100%	100%
prCm6	500	100%	100%
prDm6	1200	100%	100%
prEm6	1200	100%	100%
prFm6	1200	100%	100%
prGm6	1200	100%	100%
M_1	500	94.2%	86%
M_2	500	95.4%	86.2%
M_3	500	98%	88.8%
M_4	500	90%	81%
M_5	500	94.8%	95.2%
M_6	500	98.6%	90.8%
M_7	500	97.2%	96%
M_8	500	100%	100%
M_9	500	100%	98.8%
M_{10}	500	100%	99.8%
Bank-transfer	2000	97.25%	88.9%

Quality of Alignments. Since the alignment technique $ILP.R$ may be approximate, Table 3 provides an overview of how many of the computed alignments

can be replayed for $ILP.R$ method when combined with the technique of this paper[2].

Comparing with Original Alignments. Table 4 reports the evaluation of the quality of the results for both approaches [1], [9] with and without applying the technique of this paper. Columns ED/Jaccard report the edit/Jaccard distance between the sequences computed, while MSE columns reports the mean square root error between the corresponding fitness values. Edit distances are often large, but interestingly this has no impact on the fitness, since when expanding abstract nodes although the final position may differ, the model still can replay the obtained sequences very often.

Table 4: Quality of Computed Step-Sequences

Model	ED (A^*vs $EXP.R.A^*$)	Jaccard (A^*vs $EXP.R.A^*$)	MSE (A^*vs $EXP.R.A^*$)	ED (ILP.R vs EXP.R.ILP.R)	Jaccard (ILP.R vs EXP.R.ILP.R)	MSE (ILP.R vs EXP.R.ILP.R)
prAm6	7.49	0	0.065	9.25	0.017	0.00081
prBm6	7.87	0	0	18.31	0	0
prCm6	8.65	0.016	0.005	11.60	0.0019	0.00646
prDm6	NA	NA	NA	93.28	0.0101	0.00041
prEm6	37.14	0	0.02	37	0	0
prFm6	NA	NA	NA	67	0.013	0.0074
prGm6	NA	NA	NA	77	0.011	0.00064
M_1	4	0.085	0.021	4	0.025	0.0165
M_2	6	0.012	0.0193	6	0	0.018
M_3	8	0.046	0.021	5	0.011	0.016
M_4	4	.12	0.028	2	0.015	0.025
M_5	11	0.0022	0.0045	15	0.00024	0.0103
M_6	NA	NA	NA	12	0.0012	0.0088
M_7	NA	NA	NA	15	0.0027	0.019
M_8	4	0.073	0.039	4	0.0078	0.035
M_9	NA	NA	NA	3	0.0044	0.0085
M_{10}	NA	NA	NA	13	0.00038	0.012
Bank-transfer	18	0.031	0.025	13	0.0118	0.0067

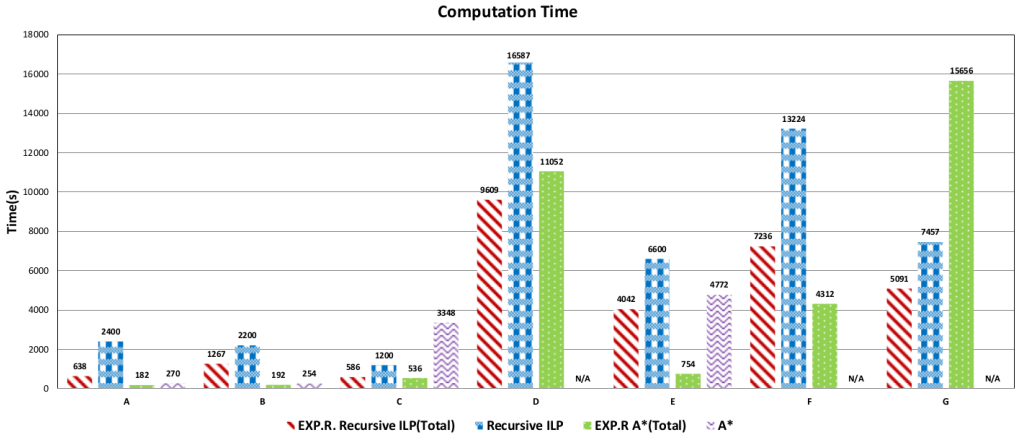
Memory Usage. The memory usage of computing alignments using [1], is reduced significantly. For large models, *prDm6, prFm6, prGm6,* M_6, M_{10}, it can only compute alignments if applied in combination of the technique of this paper.

Computation Time Comparison. Fig. 8(a)-(b) report computation times for BPM-2013 and other benchmark datasets respectively. It is evident that A^* approach combined with the proposed method is significantly faster than the other approach in nearly all datasets except (*prGm6, prDm6,* M_6, M_{10}). Still A^* approach cannot compute alignments for models M_6 *and* M_{10} even after applying the presented technique, and in that case the combination of $ILP.R$ with the presented technique is the best choice.

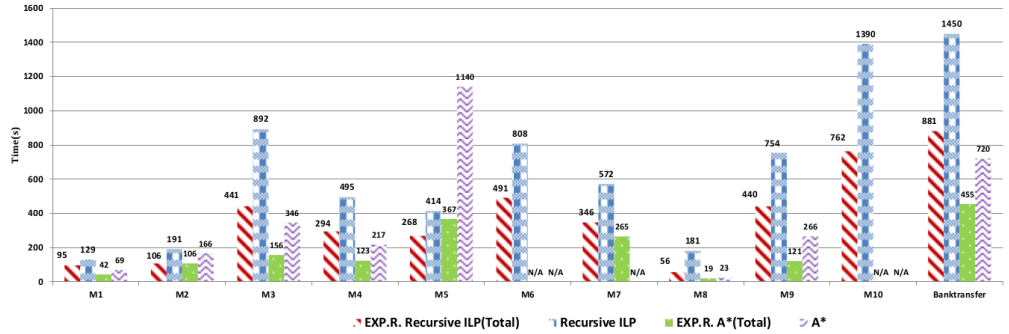
8 Conclusion and Future Work

We have presented a technique that can be used to significantly alleviate the complexity of computing alignments. The technique uses the indication relation

[2] Expanded alignments provided by A^* were replayed 100% for all datasets



(a)



(b)

Fig. 8: Computation time for (a) BPM2013 and (b) synthetic datasets.

to abstract unimportant parts of a process model so that global computation of alignments focus on a reduced instance. The reduced part of computed alignments then will be expanded to represent local deviations as well. Experiments are provided that witness the capability of the technique when used in combination with state-of-the-art approaches for alignment computation. Future work will be devoted to apply the technique on more unstructured inputs.

References

1. Arya Adriansyah. *Aligning observed and modeled behavior*. PhD thesis, Technische Universiteit Eindhoven, 2014.
2. Arya Adriansyah, Jorge Munoz-Gama, Josep Carmona, Boudewijn F. van Dongen, and Wil M. P. van der Aalst. Measuring precision of modeled behavior. *Inf. Syst. E-Business Management*, 13(1):37–67, 2015.
3. Sandie Balaguer, Thomas Chatain, and Stefan Haar. Building occurrence nets from reveals relations. *Fundam. Inform.*, 123(3):245–272, 2013.

4. Stefan Haar. Unfold and Cover: Qualitative Diagnosability for Petri Nets. In *Proceedings of the 46th IEEE Conference on Decision and Control (CDC'07)*, pages 1886–1891, New Orleans, LA, USA, United States, 2007. IEEE Control System Society.
5. Jorge Munoz-Gama, Josep Carmona, and Wil M. P. Van Der Aalst. Single-entry single-exit decomposed conformance checking. *Inf. Syst.*, 46:102–122, December 2014.
6. Saul B. Needleman and Christian D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443 – 453, 1970.
7. Artem Polyvyanyy, Jussi Vanhatalo, and Hagen Völzer. Simplified computation and generalization of the refined process structure tree. In *7th International Conference on Web Services and Formal Methods*, WS-FM'10, pages 25–41, Berlin, Heidelberg, 2011.
8. Robert E. Tarjan and Jacobo Valdes. Prime subprogram parsing of a program. In *Proceedings of the 7th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '80, pages 95–105, New York, NY, USA, 1980. ACM.
9. Farbod Taymouri and Josep Carmona. A recursive paradigm for aligning observed behavior of large structured process models. In *14th International Conference of Business Process Management (BPM), Rio de Janeiro, Brazil, September 18 - 22*, 2016.
10. Wil M. P. van der Aalst. *Process Mining - Discovery, Conformance and Enhancement of Business Processes*. Springer, 2011.
11. Jussi Vanhatalo, Hagen Völzer, and Jana Koehler. The refined process structure tree. In *Proceedings of the 6th International Conference on Business Process Management*, BPM '08, pages 100–115, Berlin, Heidelberg, 2008. Springer-Verlag.
12. H. M. W. Verbeek and W. M. P. van der Aalst. *Merging Alignments for Decomposed Replay*, pages 219–239. Springer International Publishing, Cham, 2016.