Complexity of Alignments on Sound Free-Choice Workflow Nets

Christopher T. Schwanen[1][0000−0002−3215−7251],
Wied Pakusa[2][0009−0004−6302−4445], and
Wil M. P. van der Aalst[1][0000−0002−0955−6940]

[1] Chair of Process and Data Science (PADS),
RWTH Aachen University, Aachen, Germany
{schwanen,wvdaalst}@pads.rwth-aachen.de
[2] Federal University of Applied Administrative Sciences, Brühl, Germany
Wied.Pakusa@hsbund.de

Abstract An optimal alignment consists of a minimal number of edit operations (deletions and insertions) to fit an observed event trace with a process model. In conformance checking, alignments are used to quantify in how far reality deviates from the predefined business norm and constitute probably the most important tool. In practice, however, it has frequently been observed that finding optimal alignments is computationally expensive. In this paper, we extend the proof of the Shortest Sequence Theorem for live, bounded, free-choice Petri nets to make it also applicable to moves in alignments on this model class. This way, we are able to show that computing alignments on sound free-choice workflow nets is NP-complete. While this still rules out an efficient algorithm, our result opens the door for a new set of tools to attack the alignment problem which go beyond the standard reachability approach used in most implementations. Eventually, we will demonstrate that soundness alone is not a sufficient criterion by proving that computing alignments on general safe and sound workflow nets is PSPACE-complete and thus indeed incurring immense algorithmic costs on more general model classes.

Keywords: Process Mining · Conformance Checking · Alignments · Computational Complexity · Workflow Nets · Free-Choice Petri Nets

1 Introduction

The goal of *conformance checking* is to compare the observed behavior of a process against a reference model, typically given as Petri net or BPMN diagram. This allows the identification of inefficiencies, regulatory violations, and so on. As of today, *alignments* are considered to be the state-of-the-art technique in conformance checking; we refer to [8] for a thorough introduction to this field.

The system in Figure 1 specifies a process with two observable events a and b in the form of a Petri net. We view a Petri net as both, a formal model defining a process, and as a *language acceptor*. In both cases, we agree on an *initial* and

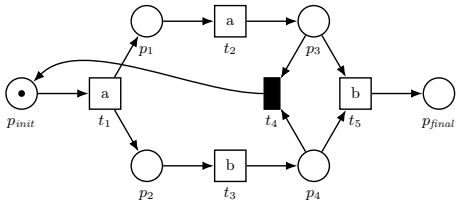

Figure 1: Example of a Petri net (sound free-choice workflow net)

a *final* marking and consider runs of the model (so called *firing sequences*) from the initial to the final marking. For each run, the transition labels form a finite *word*, also called a *trace*. Transitions can also be unlabeled (those are called *silent*). The collection of generated words constitutes the *language* accepted by the Petri net. For the net in Figure 1, the initial marking is highlighted (one single token in place p_{init}) and the intended final marking has a single token in p_{final}. Relative to these markings, the Petri net accepts the words *aabb* and *abab*, but not *abaa*. The accepted language is $(aab|aba)^+b$.

Dissimilarity is quantified by "*aligning*" a given trace against the model, which means that we *insert* and *delete* activities into and from the trace until it fits with the model. The goal is to find an *optimal alignment* which minimizes the number of required change operations. This is formalized as follows: we assume that the business process and the trace are executed and generated concurrently. While the model evolves from its initial to its final marking, and, concurrently, the trace from its first to its last letter, there are three different types of *moves*:

1. the system *and* the trace take one step *synchronously*, i.e., the system fires a transition t labelled with a and the trace moves on via its next letter a; this is called a *synchronous* move (a, t);
2. the system fires t, but the trace maintains its state; this corresponds to an *insert* operation where the label of t (which might be empty) is inserted into the trace; such move is called *model move* and is denoted by $(\gg, t)$ where $\gg$ is a distinguished "no-move" symbol;
3. dually, the model can stay in its current state while only the trace proceeds one letter further which corresponds to a *deletion* operation; such steps are called *log moves* and can be written as $(a, \gg)$.

Figure 2 shows two alignments for a trace $w = abaa$ and the system from Figure 1. The first row indicates the progress in the trace, while the last two rows contain the labels and transitions fired by the model. For the first alignment, we have four insert operations (model moves) including a silent one (t_4 is unlabeled, indicated by τ). The trace generated by the model is *abaaabb*. The model trace *abab* in the second alignment results in one deletion (log move) and one insertion (model move), which is optimal regarding the number of change operations.

It has frequently been observed that finding optimal alignments is computationally hard. Given the pivotal role of alignments, this situation calls for a thorough and systematic analysis of the complexity of the alignment problem. In

a	b	$\gg$	$\gg$	a	a	$\gg$	$\gg$
a	b	a	τ	a	a	b	b
t_1	t_3	t_2	t_4	t_1	t_2	t_3	t_5

(a) Four synchronous moves, three model moves (inserts), one silent model move

a	b	a	$\gg$	a
a	b	a	b	
t_1	t_3	t_2	t_5	$\gg$

(b) Three synchronous moves, one model move (insert), one log move (deletion)

Figure 2: Two possible alignments for $w = abaa$

particular, for practical applications it would be beneficial to know which parameters of the process model influence the complexity of the alignment problem and, thus, which algorithmic approaches are most promising on certain inputs. Surprisingly, such an analysis is, to date, still missing. In this work, we make several important steps towards such a complexity-theoretic classification of alignments.

In [26, 27], we studied the complexity of alignments on process trees. We showed that the alignment problem is in NP for general process trees and in P for process trees with unique labels. We now take one step further by considering *live, bounded, free-choice Petri nets*, a model class strongly generalizing process trees, but still known for its good algorithmic properties. Our first main result is to show that alignments on this class can also be computed in NP (see Section 5). This is a significant improvement over the case of safe Petri nets where the alignment problem is PSPACE-complete (see Section 7). Our proof is based on an extension of the *Shortest Sequence Theorem* for live, bounded, free-choice Petri nets [14] by which we can bound the length of an optimal alignment polynomially. To complement the NP upper bound, we also show NP-hardness via a reduction from the membership problem for shuffle languages (see Section 5). This gives NP-completeness for alignments on important model classes studied in process mining, such as sound free-choice workflow nets or process trees (see Section 6). Interestingly, for both classes, the reachability problem is known to be in P, which means that the alignment problem is provably harder (assuming $\mathsf{P} \neq \mathsf{NP}$).

Our second main contribution is to show that the *free-choice* assumption is crucial. We prove that on safe and sound workflow nets, the alignment problem is PSPACE-complete (see Section 7). For this, we significantly extend a construction for simulating a PSPACE-computation by a live and safe Petri net. This shows that the alignment problem is indeed intractable on safe and sound workflow nets, the standard model class used in process mining.

2 Related Work

The groundwork for conformance checking was laid in [24] where metrics based on a technique known as *token-based replay* were proposed. Although token-based replay is efficient from an algorithmic point of view, it fails to provide accurate results in presence of common model constructs. This is why alignments, introduced by [3], have replaced token-based replay as "the gold standard" in conformance checking [9]. An overview of recent developments is given in [8, 9].

It is ubiquitously mentioned that computing alignments (with respect to the standard algorithms based on A^*) suffers from the so-called *state explosion problem* [cf. 30]. In general, it has been recognized that the reachability problem for Petri nets is a lower bound for the complexity [8, 9]. Most relevant for us are complexity bounds for classes of safe Petri nets (which means that places can hold at most one token). For safe Petri nets, it was shown that reachability is PSPACE-complete [19]. Later on, it turned out that almost all interesting computational problems on safe Petri nets are PSPACE-hard. A comprehensive overview of results is given in [10, 17, 18]. Because of the high complexity, more restricted classes of Petri nets with better algorithmic properties have been studied as well. Here, *free-choice* Petri nets form the most relevant example on which important computational problems become tractable, see [13].

Different algorithmic approaches were investigated to improve scalability of alignment computations. E.g., in [5] the authors use symbolic representations of the (exponential) state space to reduce the memory footprint of the alignment computation. Another common approach is to encode alignments into related problems for which well-adapted algorithms exists. This idea is investigated, for example, in [21] where alignment computations are represented as a planning problem. Another angle is to improve heuristics for the A^*-algorithm: in [15], for instance, Petri net theory and linear programming are combined to improve the runtime on several benchmarks significantly. Besides this, also approximative algorithms have been proposed, see, e.g., [28] for a scheme based on integer linear programming and [29] for a genetic method to compute optimal alignments.

However, these studies do not provide guarantees and we did hardly find any source that investigates the algorithmic complexity of alignments. A statement on specific model classes is provided in [6, 7]. There, the authors showed that computing alignments is NP-complete on the class of safe Petri nets when the length of permissible alignments is limited. To the best of our knowledge, the approach in [27] was the first to truly break the PSPACE-barrier when computing alignments on the class of process trees. When further restricting process trees to only have unique labels, the complexity drops from NP even further to P [26]. Beyond that, we are not aware of any results studying the complexity of alignments over different process models.

Finally, we like to mention work on the related *error correction problem* for regular languages. A first efficient algorithm for this problem was given by Wagner in the 1970s [31], a complexity-theoretic analysis can be found in [23]. For context-free grammars (CFGs), a polynomial-time error-correction algorithm can be found in [4]. Our work might be interesting for the error correction problem as well, as it provides a new perspective on the complexity of the problem for other presentations of regular languages.

3 Preliminaries

Let $\mathbb{N} \coloneqq \{0, 1, 2, \ldots\}$ denote the natural numbers. For any tuple a, $\pi_i(a)$ denotes the *projection* on its ith element, i.e., $\pi_i \colon A_1 \times \cdots \times A_n \to A_i, (a_1, \ldots, a_n) \mapsto a_i$.

Definition 1 (Multiset). A *multiset* M over a set A is a function $M\colon A \to \mathbb{N}$; thus, for any $a \in A$, $M(a)$ indicates how often a is contained in the multiset M. The set of all multisets over A is given by $\mathbb{N}^A$. We also use the notation $\left[a^{M(a)} \mid a \in A\right]$ for a multiset $M \in \mathbb{N}^A$. Any set A can also be considered a multiset by assigning 1 to each item. For multisets $M, M' \in \mathbb{N}^A$, we use the standard notation for functions, e.g., $M + M'$, $M \leq M'$, etc. The *support* of a multiset $M \in \mathbb{N}^A$, denoted by $\langle M \rangle$, is the set of distinct elements contained in M, i.e., $\langle M \rangle \coloneqq \{a \in A \mid M(a) > 0\}$. For multisets $M \in \mathbb{N}^A$ and $M' \in \mathbb{N}^B$ over two sets A and B, $M \oplus M'$ denotes their addition after extending them by 0.

Definition 2 (Sequence, Permutation). *Sequences* with index set I over a set A are denoted by $\sigma = \langle a_i \rangle_{i \in I} \in A^I$. The *length* of a sequence σ is written as $|\sigma|$ and the set of all finite sequences over A is denoted by A^*. For a sequence $\sigma = \langle a_i \rangle_{i \in I} \in A^I$, the notation $\sum \sigma$ is used as a shorthand for $\sum_{i \in I} a_i$. Given two sequences σ and σ', $\sigma \cdot \sigma'$ (or $\sigma\sigma'$ in short) denotes the concatenation of the two sequences. The restriction of a sequence $\sigma \in A^*$ to a set $B \subseteq A$ is the subsequence $\sigma|_B$ of σ consisting of all elements in B. A function $f\colon A \to B$ can be applied to a sequence $\sigma \in A^*$ given the recursive definition $f(\langle\rangle) \coloneqq \langle\rangle$ and $f(\langle a \rangle \cdot \sigma) \coloneqq \langle f(a) \rangle \cdot f(\sigma)$. The *multiset representation* of a sequence $\sigma \in A^*$, denoted by $\boldsymbol{\sigma}$, is defined by $\boldsymbol{\sigma}(a) \coloneqq \left|\sigma|_{\{a\}}\right|$ for every $a \in A$, and $\langle \boldsymbol{\sigma} \rangle$ provides the *support* of σ. A sequence $\sigma' \in A^*$ is a *permutation* of σ if and only if $\boldsymbol{\sigma'} = \boldsymbol{\sigma}$.

Definition 3 (Alphabet). An *alphabet* Σ is a finite, non-empty set of *labels*.

Definition 4 (Petri Net). Let Σ be an alphabet. A *Petri net* N is a bipartite directed graph $N = (P, T, F, \ell)$ where P and T, $P \cap T = \emptyset$ are disjoint finite sets of vertices and $F \subseteq (P \times T) \cup (T \times P)$ is the set of arcs. In a Petri net, P is called the set of *places*, T the set of *transitions*, and F the *flow relation*. In addition, $\ell\colon T \to \Sigma \cup \{\tau\}$ is a *labeling function*. A transition $t \in T$ is *labeled* if $\ell(t) \in \Sigma$ and it is *silent* if $\ell(t) = \tau$. Given a vertex $v \in P \cup T$, its *pre-set* $\bullet v$ and *post-set* $v\bullet$ are defined by $\bullet v \coloneqq \{u \in P \cup T \mid (u, v) \in F\}$ and $v\bullet \coloneqq \{u \in P \cup T \mid (v, u) \in F\}$. With regard to a place (transition), its pre- and post-set are also called *input* and *output transitions* (*places*).

Definition 5 (Marking, System, Firing Rule). Given a Petri net $N = (P, T, F, \ell)$, a *marking* $M \in \mathbb{N}^P$ is a multiset where $M(p)$ is the number of *tokens* at place $p \in P$. A place $p \in P$ is *marked* at M if $M(p) > 0$. The pair (N, M) of a Petri net $N = (P, T, F, \ell)$ and a marking $M \in \mathbb{N}^P$ is called a *system*. A transition $t \in T$ is *enabled* in M, denoted by $(N, M)[t\rangle$, if and only if each of its input places $p \in \bullet t$ is marked, i.e., $\forall p \in \bullet t\colon M(p) > 0$. An enabled transition may *fire*, denoted by $(N, M)[t\rangle(N, M')$, and firing results in a new marking M':

$$M'(p) = \begin{cases} M(p) - 1 & \text{if } p \in \bullet t \wedge p \notin t\bullet, \\ M(p) + 1 & \text{if } p \notin \bullet t \wedge p \in t\bullet, \\ M(p) & \text{otherwise.} \end{cases}$$

A sequence of transitions $\sigma = \langle t_i \rangle_{i=1}^n \in T^*$ is called a *firing sequence* of (N, M) if for every transition t_i of the sequence holds that $(N, M_{i-1})[t_i\rangle$ and

$(N, M_{i-1})[t_i\rangle(N, M_i)$ where $M_0 = M$ and $M_n = M'$. Firing such a sequence is denoted by $(N, M)[\sigma\rangle(N, M')$. The empty sequence $\langle\rangle$ is always enabled and firing the empty sequence leaves the marking unchanged, i.e., $(N, M)[\langle\rangle\rangle(N, M)$. A marking M' is *reachable* if a firing sequence $\sigma \in T^*$ exists such that M' is the resulting marking, i.e., $(N, M)[\sigma\rangle(N, M')$. The set of all reachable markings of (N, M) is denoted by $[N, M\rangle \coloneqq \left\{M' \in \mathbb{N}^P \mid \exists \sigma \in T^* \colon (N, M)[\sigma\rangle(N, M')\right\}$.

Definition 6 (Boundedness, Safeness). Given a $k \in \mathbb{N}$, a system (N, M_0) with $N = (P, T, F, \ell)$ is *k-bounded* if k is a bound for any reachable marking, i.e., $\forall M \in [N, M_0\rangle \colon \forall p \in P \colon M(p) \leq k$. (N, M_0) is *safe* if it is 1-bounded.

Definition 7 (Accepting System). An *accepting system* (N, M_{init}, M_{final}) extends a system (N, M_{init}), $N = (P, T, F, \ell)$, with a *final marking* $M_{final} \in \mathbb{N}^P$.

Definition 8 (Complete Firing Sequence, Trace, Behavior and Language of a System). Let $S = (N, M_{init}, M_{final})$ be an accepting system with $N = (P, T, F, \ell)$ and $\ell \colon T \to \Sigma \cup \{\tau\}$ over Σ. A firing sequence $\sigma \in T^*$ is a *complete firing sequence* of S if $(N, M_{init})[\sigma\rangle(N, M_{final})$. The set of complete firing sequences $\phi(S) \coloneqq \{\sigma \in T^* \mid (N, M_{init})[\sigma\rangle(N, M_{final})\}$ is the *behavior* of S. Considering the labels of a complete firing sequence, this is referred to as a *trace* $\sigma \in \Sigma^*$. The set of all traces $\mathcal{L}(S) \coloneqq \{\ell(\sigma)|_\Sigma \mid \sigma \in \phi(S)\}$ is the *language* of S.

Definition 9 (Easy Soundness). An accepting system (N, M_{init}, M_{final}) is *easy sound* if and only if the final marking is reachable, i.e., $M_{final} \in [N, M_{init}\rangle$.

4 Alignments

Alignments juxtapose an observed trace with a complete firing sequence of the process model. For this, activities in the trace are compared in pairs with the transitions of the complete firing sequence. These pairs are called *moves*:

Definition 10 (Moves). Let Σ be an alphabet and $S = (N, M_{init}, M_{final})$ an accepting system with Petri net $N = (P, T, F, \ell)$ and labeling function $\ell \colon T \to \Sigma \cup \{\tau\}$. Furthermore, let $\gg$ be a distinguished *"no move" symbol*. A *move* is an ordered pair $(a, t) \in (\Sigma \cup \{\gg\}) \times (T \cup \{\gg\})$. We distinguish between three types of *legal* moves: The move (a, t) is a

- *synchronous move* if $a \in \Sigma$, $t \in T$, and $a = \ell(t)$,
- *log move* if $a \in \Sigma$ and $t = \gg$,
- *model move* if $a = \gg$ and $t \in T$.

All other moves are considered illegal. A model move $(\gg, t)$ is a *silent move* if $\ell(t) = \tau$. The set $LM_{\Sigma,S}$ denotes all legal moves between Σ and S:

$$LM_{\Sigma,S} \coloneqq \{(a, t) \in \Sigma \times T \mid a = \ell(t)\} \cup \Sigma \times \{\gg\} \cup \{\gg\} \times T.$$

An *alignment* is a sequence of legal moves whose first components form the observed trace and whose second components form a complete firing sequence of the process model (ignoring the $\gg$-symbol and τ-labels), formally:

Definition 11 (Alignment). Let Σ be an alphabet, $\sigma \in \Sigma^*$ a trace, and $S = (N, M_{init}, M_{final})$ an accepting system with $N = (P, T, F, \ell)$ and $\ell \colon T \to \Sigma \cup \{\tau\}$. An *alignment* $\gamma \in {LM_{\Sigma,S}}^*$ between σ and S is a sequence of moves such that $\sigma = \pi_1(\gamma)|_\Sigma$ and $\pi_2(\gamma)|_T \in \phi(S)$. $\Gamma_{\sigma,S}$ denotes all alignments between σ and S:

$$\Gamma_{\sigma,S} \coloneqq \{\gamma \in {LM_{\Sigma,S}}^* \mid \pi_1(\gamma)|_\Sigma = \sigma \wedge \pi_2(\gamma)|_T \in \phi(S)\}.$$

In easy-sound systems, a trivial alignment always exists: first generate the input trace via log moves, and then generate a firing sequence from the initial to the final marking via model moves. This trivial alignment corresponds to the worst possible scenario: the model and the trace have nothing in common. However, we are really interested in *optimal* alignments which *maximize* the synchronization between the trace and the model. This is formalized by assigning costs to moves and then finding an alignment with minimal costs.

Definition 12 (Alignment Cost). Let $S = (N, M_{init}, M_{final})$ be an easy-sound system, i.e., $\phi(S) \neq \emptyset$, with $N = (P, T, F, \ell)$ and $\ell \colon T \to \Sigma \cup \{\tau\}$, and let $LM_{\Sigma,S}$ be the set of all legal moves between Σ and S. An *alignment cost function* is a function $c \colon LM_{\Sigma,S} \to \mathbb{Q}_{\geq 0}$. The cost of an alignment $\gamma \in {LM_{\Sigma,S}}^*$ is given by the sum of costs of each move in the sequence, i.e., $\sum c(\gamma)$.

The *standard cost function* $c \colon LM_{\Sigma,S} \to \mathbb{Q}_{\geq 0}$ is defined as

$$(a,t) \mapsto c(a,t) \coloneqq \begin{cases} 0 & a \in \Sigma \wedge t \in T \wedge a = \ell(t), \text{ or } a = \gg \wedge \ell(t) = \tau, \\ 1 & a \in \Sigma \wedge t = \gg, \qquad\quad \text{or } a = \gg \wedge \ell(t) \in \Sigma. \end{cases}$$

Definition 13 (Optimal Alignment). Let S be an easy-sound system, i.e., $\phi(S) \neq \emptyset$, and let $c \colon LM_{\Sigma,S} \to \mathbb{Q}_{\geq 0}$ be an alignment cost function. Given a trace $\sigma \in \Sigma^*$, an alignment $\gamma_{opt} \in \Gamma_{\sigma,S}$ is *optimal* if and only if no other alignment between σ and S has lower costs, i.e., $\sum c(\gamma_{opt}) = \min_{\gamma \in \Gamma_{\sigma,S}} \{\sum c(\gamma)\}$.

Of course, computing an optimal alignments is a *functional* optimization problem, the corresponding decision problem is the following:

Problem 1 (Alignment (Align)*).*

Input: An alphabet Σ, an easy-sound system (N, M_{init}, M_{final}) with Petri net $N = (P, T, F, \ell)$ and labeling function $\ell \colon T \to \Sigma \cup \{\tau\}$, a trace $\sigma \in \Sigma^*$ over Σ, and a cost function $c \colon LM_{\Sigma,S} \to \mathbb{Q}_{\geq 0}$.

Question: Given $k \in \mathbb{Q}_{\geq 0}$, is there an alignment $\gamma \in \Gamma_{\sigma,S}$ with $\sum c(\gamma) \leq k$?

For our purposes the (binary) decision problem Align is more adequate, since we are interested in classifying its computational complexity. However, it is also clear that an algorithm for the decision version can be transformed into an algorithm for the functional variant by performing a binary search on the cost threshold k. This only requires a polynomial number of calls to the decision algorithm and does not change the complexity class of the problem.

5 Align on Live, Bounded, Free-Choice Systems

First, we turn our attention to *live, b-bounded, free-choice systems* (LBFC-systems, for short). These nets have been thoroughly studied in Petri net theory and enjoy nice structural properties [cf. 13]. In our main result (Theorem 4), we show that for each trace and each LBFC-system there exists an optimal alignment of polynomial length. From this, we obtain a simple guess-and-verify NP-algorithm for the alignment problem which reduces the complexity from PSPACE to NP.

Definition 14 (Live, b-Bounded, Free-Choice (LBFC) Systems). A system (N, M_0) over $N = (P, T, F, \ell)$ is a live, b-bounded, free-choice system (LBFC-system) if and only if it is

live, if:	$\forall M \in [(N, M_0)\rangle, \forall t \in T\colon \exists M' \in [(N, M)\rangle\colon (N, M')[t\rangle$
b-bounded, if:	$\forall M \in [(N, M_0)\rangle, \forall p \in P\colon M(p) \leq b$
free-choice, if:	$\forall (s, t) \in F\colon {\bullet t} \times s{\bullet} \subseteq F$.

In this paper, we consider LBFC-systems with respect to a *fixed* value of $b \in \mathbb{N}$, i.e., b is a constant which does not vary along different inputs. Implicitly, we think of $b = 1$ (safe systems), but any other fixed value for b is possible too.

Our NP-algorithm for LBFC-systems is based on the following key property: whenever a marking M' can be reached from M in an LBFC-system, then there *exists some* connecting firing sequence of polynomial length. This was first shown in [14], but we rely on the textbook by the same authors [13].

Theorem 1 (Shortest Sequence Theorem [13, Theorem 9.17]). *Let (N, M_0) be an LBFC-system with n transitions and let M be a reachable marking. Then, there is a firing sequence σ such that $(N, M_0)[\sigma\rangle(N, M)$ and*

$$|\sigma| \leq b \cdot \frac{n \cdot (n+1) \cdot (n+2)}{6}.$$

Unfortunately, the Shortest Sequence Theorem cannot be directly applied to alignments as it only guarantees the *existence* of a "short" firing sequence. In particular, when we start from an arbitrary firing sequence and then, using the above theorem, pass over to a short one, this new sequence can be completely different from the original sequence. This is problematic in context of alignments, since the "short" sequence could contain different transitions, potentially with much higher costs of moves. In a nutshell, a "shorter" sequence is not necessarily a "cheaper" sequence. Luckily, the result can be stated in a more general form.

Theorem 2 (Shortest Sequence Theorem – Generalized Form). *Let (N, M_0) with $N = (P, T, F, \ell)$ and bound b be an LBFC-system. Moreover, let $\sigma \in T^*$ and $M_1, M_2 \in [N, M_0\rangle$ such that $(N, M_1)[\sigma\rangle(N, M_2)$. Then, there exists a firing sequence $\sigma' \in T^*$ with $\boldsymbol{\sigma' \leq \sigma}$, $(N, M_1)[\sigma'\rangle(N, M_2)$ and*

$$|\sigma'| \leq b \cdot \frac{|T| \cdot (|T|+1) \cdot (|T|+2)}{6}.$$

To prove Theorem 2, we go through the machinery of [13] while making necessary changes to results and proofs. For those parts that do not require any adaptation, we refer to [13] for details. The proof consists of two steps. First, we consider the case of *T-systems*, a subclass of free-choice systems where each place has at most one input and one output transition. In such systems, whenever a transition is enabled, it cannot be disabled by firing any other transition.

Definition 15 (T-Net, T-System). A Petri net $N = (P, T, F, \ell)$ is a *T-net* if each place has at most one input and one output transition, i.e., $\forall p \in P\colon |\bullet p| \leq 1, |p\bullet| \leq 1$. A system (N, M_0) is a *T-system* if N is a T-net.

The syntactic restrictions of T-nets allow us to rearrange firing sequences in such a way that the same sets of transitions are repeatedly fired until, eventually, more and more transitions die out and the set of occurring transitions becomes smaller and smaller (assuming b-boundedness).

Definition 16 (Biased Firing Sequence [cf. 13, Definition 3.22]**).** A firing sequence $\sigma \in T^*$ is *biased* if for all $t_1, t_2 \in \langle \boldsymbol{\sigma} \rangle$, $t_1 \neq t_2$, it holds that $\bullet t_1 \cap \bullet t_2 = \emptyset$.

Note that firing sequences in T-systems are always biased. In biased sequences, we can move each occurring transition to an initial segment:

Lemma 1 ([13, Lemma 3.24 p. 56]**).** *Let (N, M_0) be a system with $N = (P, T, F, \ell)$ and let $\sigma \in T^*$ be a biased firing sequence. Then, there exists a permutation $\rho = \sigma_1\sigma_2$ of σ with $\boldsymbol{\rho} = \boldsymbol{\sigma}$ such that $\exists M \in \mathbb{N}^P\colon (N, M_0)[\sigma\rangle(N, M) \wedge (N, M_0)[\sigma_1\sigma_2\rangle(N, M)$ and no transition occurs more than once in σ_1 and every transition that occurs in σ_2 occurs also in σ_1, i.e., $\langle \boldsymbol{\sigma_2} \rangle \subseteq \langle \boldsymbol{\sigma_1} \rangle$.*

For the following lemma we need to generalize the result from [13]:

Lemma 2 (Generalized Form of [13, Lemma 3.25]**).** *Let (N, M_0) with $N = (P, T, F, \ell)$ and bound b be an LBFC-system and let $(N, M_0)[\sigma\rangle(N, M)$ for a firing sequence $\sigma \in T^*$ and a marking M such that σ is biased and non-empty. Then, there exists $\rho = \sigma_1\sigma_2$ with $\boldsymbol{\rho} \leq \boldsymbol{\sigma}$ such that*

- *$(N, M_0)[\sigma\rangle(N, M)$ and $(N, M_0)[\sigma_1\sigma_2\rangle(N, M)$,*
- *each transition occurs at most b times in σ_1, and*
- *$\langle \boldsymbol{\sigma_1} \rangle \supset \langle \boldsymbol{\sigma_2} \rangle$ (set of occurring transitions decreases).*

Proof. First, we repeatedly apply Lemma 1 in order to obtain a permutation of σ which is of the form $\rho_1\rho_2 \cdots \rho_n$ with $\rho_i \neq \langle\rangle$ with the following properties:

- for all $i = 1, \ldots, n$, no transition occurs more than once in ρ_i, and
- for all $i < n$, $\langle \boldsymbol{\rho_i} \rangle \supseteq \langle \boldsymbol{\rho_{i+1}} \rangle$.

If $n \leq b$, we are done: simply choose $\sigma_2 = \langle\rangle$ and $\sigma_1 = \rho_1 \cdots \rho_n$. So, let us assume $n \geq b + 1$. If $\langle \boldsymbol{\rho_1} \rangle \supset \langle \boldsymbol{\rho_{b+1}} \rangle$, we can stop our argument at this point as well, since then $\sigma_1 = \rho_1 \cdots \rho_b$ and $\sigma_2 = \rho_{b+1} \cdots \rho_n$ would have the desired properties. So, let us also assume that $\langle \boldsymbol{\rho_1} \rangle = \langle \boldsymbol{\rho_{b+1}} \rangle$. Choose the maximal m such that $\langle \boldsymbol{\rho_1} \rangle = \langle \boldsymbol{\rho_m} \rangle$ (note that $b + 1 \leq m \leq n$). Let $X \coloneqq \langle \boldsymbol{\rho_1} \rangle = \langle \boldsymbol{\rho_m} \rangle \subseteq T$. Since

each transition in X occurs precisely once in each of the ρ_i, we know that for each place the same number of tokens is added or removed after firing each of the subsequences ρ_i. Since each place can hold at most b tokens (since the net is b-bounded) this means that firing all transitions in X must leave the number of tokens in each place invariant. But this, in turn, implies that each of the ρ_i, $1 \leq i \leq m$ does not modify the initial marking M_0. Hence, we can simply choose $\sigma_1 = \rho_1$ and $\sigma_2 = \rho_{m+1} \cdots \rho_n$ which completes our proof. □

This already implies the Shortest Sequence Theorem for T-systems. For later use, let us state the concrete implications in a generalized form of what the authors in [13] call the *Biased Sequence Lemma*:

Lemma 3 (Biased Sequence Lemma, Generalized Form of [13, Lemma 3.26]**).** *Let (N, M_0) with $N = (P, T, F, \ell)$ and bound b be an* LBFC*-system and let $(N, M_0)[\sigma\rangle(N, M)$ for a firing sequence $\sigma \in T^*$ and a marking M such that σ is biased and non-empty. Let k denote the number of (distinct) transitions in σ, i.e., $k \coloneqq |\langle \boldsymbol{\sigma} \rangle|$. Then, there exists a firing sequence ρ such that $\boldsymbol{\rho} \leq \boldsymbol{\sigma}$, $(N, M_0)[\rho\rangle(N, M)$, and*

$$|\rho| \leq b \cdot \frac{k \cdot (k+1)}{2}.$$

Proof. By repeatedly applying Lemma 2, we find $\rho = \rho_1 \rho_2 \cdots \rho_k$, $\boldsymbol{\rho} \leq \boldsymbol{\sigma}$ where

- $(N, M_0)[\rho_1 \rho_2 \cdots \rho_k\rangle(N, M)$,
- each transition occurs at most b times in ρ_i, and
- ρ_i contains at most $(k + 1 - i)$ many different transitions.

Hence, $|\rho| \leq b \cdot \sum_{i=1}^{k} i = b \cdot k \cdot (k+1)/2$ as claimed. □

Next, we consider the case of general free-choice systems. To proceed, we need to introduce a couple of notions and definitions. In what follows, we implicitly refer to some LBFC-system (N, M_0) with $N = (P, T, F, \ell)$.

Definition 17 (Cluster [cf. 13, Definition 4.4]**).** For two transitions $t, t' \in T$ we let $t \sim t'$ if $\bullet t = \bullet t'$. This gives rise to an equivalence relation $\sim$ on T whose equivalence classes $[t]$ are called *clusters*.

Definition 18 (Conflict Order [cf. 13, Definition 9.8]**).** A *conflict order* $\preceq \subseteq T \times T$ is any partial order on T such that two transitions $t, t' \in T$ are comparable if and only if $[t] = [t']$, i.e., $\bullet t \cap \bullet t' \neq \emptyset$. The corresponding strict partial order is denoted by $\prec$, i.e., $t \prec t'$ if and only if $t \preceq t'$ and $t \neq t'$.

To put it differently, a conflict order is composed of separate linear orderings on the clusters of the LBFC-system. The key idea (and main challenge) in the proof of the *Shortest Sequence Theorem* is to show that each firing sequence can be rearranged in such a way that the permuted firing sequence *is ordered* with respect to *some* conflict order with which the firing sequence *agrees*:

Definition 19 (Ordered Firing Sequence [cf. 13, Definition 9.8]**).** A firing sequence $\sigma \in T^*$ *is ordered* with respect to a conflict order $\preceq$ if for all $t \prec t'$ there is no occurrence of t in σ after an occurrence of t'. Furthermore, σ *agrees* with $\preceq$ if for each cluster c either no transition of c occurs in σ or the last transition of c that occurs in σ is the maximal transition in c (according to $\preceq$).

The central result is the following Theorem from [13] which, conveniently, we can use without modification:

Theorem 3 ([13, Proposition 9.16]**).** *Let (N, M_0) with $N = (P, T, F, \ell)$ be an* LBFC*-system. Moreover, let $(N, M_0)[\sigma\rangle(N, M)$ for a firing sequence $\sigma \in T^*$ and let $\preceq$ be any conflict order which agrees with σ. Then, there exists a $\preceq$-ordered permutation ρ of σ such that $(N, M_0)[\rho\rangle(N, M)$.*

With Theorem 3 and the Biased Sequence Lemma (Lemma 3) we can finally prove our generalized version of the Shortest Sequence Theorem.

Proof (Theorem 2). First, note that for $M \in [N, M_0\rangle$ the system (N, M) is an LBFC-system as well. Hence, it suffices to shorten a firing sequence σ of the form $(N, M_0)[\sigma\rangle(N, M)$. Moreover, due to Theorem 3, we can assume that σ is $\preceq$-ordered for some conflict order $\preceq$ (if not, we can permute σ accordingly).

We iteratively split σ up into parts σ_i, $i = 1, \ldots, k$, i.e., $\sigma = \sigma_1\sigma_2 \cdots \sigma_k$, with respect to the following property: σ_i is a maximal prefix of $\sigma_i \cdots \sigma_k$ such that σ_i is biased. We claim that the number of distinct transitions in σ_{i+1} is smaller than the number of distinct transitions in σ_i. This is because the first transition t' of σ_{i+1} must be in the same cluster $t' \in [t]$ of some transition $t \neq t'$ that occurs in σ_i (because of the maximality of σ_i). Since σ is ordered, $t \prec t'$ and t cannot occur in $\sigma_{i+1} \cdots \sigma_k$. Hence, σ_i contains at most $(|T| - i + 1)$ distinct transitions. In particular, $k \leq |T|$. We have $(N, M_{i-1})[\sigma_i\rangle(N, M_i)$ where $M_k = M$. By Lemma 3 we find firing sequences σ_i', $\boldsymbol{\sigma_i'} \leq \boldsymbol{\sigma_i}$ of length at most $b \cdot (|T| - i + 1) \cdot (|T| - i + 2)/2$ with $(N, M_{i-1})[\sigma_i'\rangle(N, M_i)$. The shortened sequence $\sigma' = \sigma_1'\sigma_2' \cdots \sigma_k'$, $\boldsymbol{\sigma'} \leq \boldsymbol{\sigma}$ has length at most

$$\frac{b}{2} \cdot \sum_{i=1}^{|T|} (|T| - i + 1) \cdot (|T| - i + 2) = \frac{b}{2} \cdot \frac{|T| \cdot (|T| + 1) \cdot (|T| + 2)}{3},$$

and satisfies $(N, M_0)[\sigma'\rangle(N, M)$, which completes our proof. □

With this result, we can bound the length of sequences of consecutive model moves in alignments. This, in turn, allows us to show that LBFC-systems always have optimal alignments of polynomial length:

Theorem 4 (Alignments in LBFC-Systems). *Let $S = (N, M_{init}, M_{final})$ be an* LBFC*-system with $N = (P, T, F, \ell)$, bound b, and labeling function $\ell \colon T \to \Sigma \cup \{\tau\}$ and let $\sigma \in \Sigma^*$ be a trace over the alphabet Σ. Then, there exists an optimal alignment $\gamma \in \Gamma_{\sigma,S}$ between σ and S such that*

$$|\gamma| \leq (|\sigma| + 1) \cdot \left(b \cdot \frac{|T| \cdot (|T| + 1) \cdot (|T| + 2)}{6} + 1 \right).$$

Proof. Let $\gamma = \langle \gamma_i \rangle_{i=1}^{|\gamma|} \in \Gamma_{\sigma,S}$ be an optimal alignment of minimal length. We show that $|\gamma|$ satisfies the above inequality.

Let us denote the length of the trace σ by $q = |\sigma|$. Since $\pi_1(\gamma)|_\Sigma = \sigma$, we can find a subsequence $\langle \gamma_i \rangle_{i \in I}$ of γ with $I \subseteq \{1, \ldots, |\gamma|\}$, $I = \{i_1, i_2, \ldots, i_q\}$, $i_1 < i_2 < \cdots < i_q$ and such that $\pi_1(\langle \gamma_i \rangle_{i \in I}) = \sigma$. We use the positions $i_1, i_2, \ldots, i_q \in I$ in order to split γ up into $q+1$ parts:

$$\delta_0 = \langle \gamma_1, \ldots, \gamma_{i_1} \rangle,\ \delta_1 = \langle \gamma_{i_1+1}, \ldots, \gamma_{i_2} \rangle,\ \ldots,\ \delta_q = \langle \gamma_{i_q+1}, \ldots, \gamma_{|\gamma|} \rangle.$$

For an illustration, see Figure 3. Next, we show that for each $j \in \{0, \ldots, q\}$ we have $|\delta_j| \leq b \cdot |T| \cdot (|T|+1) \cdot (|T|+2)/6 + 1$. If we can verify this, the claim follows. Pick some $j \in \{0, \ldots, q\}$ and let

$$M_j \in \mathbb{N}^P :\ (N, M_{init})[\pi_2(\langle \gamma_1, \ldots, \gamma_{i_j} \rangle)|_T \rangle (N, M_j),$$
$$M'_j \in \mathbb{N}^P :\ (N, M_{init})[\pi_2(\langle \gamma_1, \ldots, \gamma_{i_{j+1}-1} \rangle)|_T \rangle (N, M'_j),$$

where for the cases $j = 0$ we let $M_0 \coloneqq M_{init}$ and for $j = q$ we let $M'_q \coloneqq M_{final}$. In words, M_j is the marking that we obtain from M_{init} by firing the transitions in γ of the first j parts, i.e., $\pi_2(\langle \gamma_1, \ldots, \gamma_{i_j} \rangle)|_T$, and M'_j is the marking that we get by firing the transitions in the first $j+1$ parts, i.e., $\pi_2(\langle \gamma_1, \ldots, \gamma_{i_{j+1}-1} \rangle)|_T$, except for the very last transition from the move γ_{i_j}. The motivation for looking at these two markings is as follows:

- by definition, we can reach the marking M'_j from marking M_j by firing the intermediate sequence $\pi_2(\langle \gamma_{i_j+1}, \ldots, \gamma_{i_{j+1}-1} \rangle)$,
- each of the moves in this sequence $\langle \gamma_{i_j+1}, \ldots, \gamma_{i_{j+1}-1} \rangle$ is of the form $(\gg, t)$, i.e., we only move in the system, but not in the trace,
- the length of this sequence is $|\delta_j| - 1$.

Hence, it suffices to show that the sequence $\langle \gamma_{i_j+1}, \ldots, \gamma_{i_{j+1}-1} \rangle$ is of length at most $b \cdot |T| \cdot (|T|+1) \cdot (|T|+2)/6$. To see this, we make use of Theorem 2. In fact, by this result we know that from $\pi_2(\langle \gamma_{i_j+1}, \ldots, \gamma_{i_{j+1}-1} \rangle)$ we could construct, by deleting and rearranging transitions, a firing sequence which leads from M_j to M'_j in the underlying system of length at most $b \cdot |T| \cdot (|T|+1) \cdot (|T|+2)/6$. Since *all* moves in $\langle \gamma_{i_j+1}, \ldots, \gamma_{i_{j+1}-1} \rangle$ are of the form $(\gg, t)$, we could lift the necessary deletion and rearrangement steps to the level of γ without doing any harm to the alignment properties. Also note that since we only delete and rearrange moves, the costs of the alignment do not increase. Since γ was chosen to be an optimal alignment of minimal length, the claim follows. □

Theorem 4 yields an NP-strategy for the alignment problem on LBFC-systems: on input S, σ, and k, where S is an accepting LBFC-system, σ is a trace as above, and $k \in \mathbb{Q}_{\geq 0}$ denotes a threshold, the problem is to decide whether some alignment $\gamma \in \Gamma_{\sigma,S}$ exists with costs $\sum c(\gamma) \leq k$. We make use of Theorem 4 and non-deterministically construct an alignment γ of length at most $(|\sigma|+1) \cdot (b \cdot |T| \cdot (|T|+1) \cdot (|T|+2)/6 + 1)$ and verify that: (1) γ is a valid alignment between σ and S, and (2) its costs $\sum c(\gamma)$ do not exceed k. By Theorem 4, an optimal

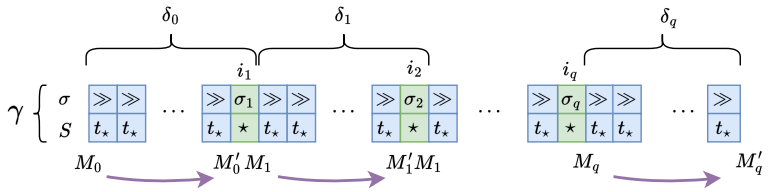

Figure 3: Decomposing the alignment γ into parts δ_j at non-model move positions i_j. The proof idea is to shorten model move sequences (purple arrows) to connecting sequences of length at most $b \cdot |T| \cdot (|T|+1) \cdot (|T|+2)/6$. $t_\star$ is a placeholder for any transition in T, $\star$ for a transition in T or the no-move symbol $\gg$.

alignment between σ and S is among the potential candidates. Note that (since b is a constant) the length of γ is bounded polynomially in σ and S. Moreover, it is easy to verify that γ is a valid alignment (just simulate the system S and σ accordingly) and to check that its costs do not exceed k. Hence, we obtain:

Corollary 1. *On the class of* LBFC*-systems,* ALIGN *is in* NP.

6 ALIGN on Sound Free-Choice Workflow Nets

In this section, we focus on sound free-choice workflow nets. Workflow nets are characterized by a distinct source place and a distinct sink place and are thus tightly related to accepting systems. In particular, sound free-choice workflow nets are not only considered a subclass of LBFC-systems for which we have shown above that ALIGN $\in$ NP, but also of cyclic LBFC-systems (i.e., their initial marking is always reachable) for which reachability is in P [12]. This raises hope for an efficient algorithm for ALIGN—contrary to reachability on LBFC-systems, which is NP-complete [16]—; yet, we show that ALIGN remains NP-hard.

Definition 20 (Workflow Net). A Petri net $N = (P, T, F, \ell)$ with labeling function $\ell \colon T \to \Sigma \cup \{\tau\}$ is a *workflow net* if

- there is a single *source place* $p_{init} \in P$, i.e., $\{p_{init}\} = \{p \in P \mid \bullet p = \emptyset\}$,
- a single *sink place* $p_{final} \in P$, i.e., $\{p_{final}\} = \{p \in P \mid p\bullet = \emptyset\}$, and
- every vertex $v \in P \cup T$ of the Petri net is on a path from p_{init} to p_{final}.

Implicitly, in workflow nets, the initial marking is $M_{init} = [p_{init}]$, and the final marking is $M_{final} = [p_{final}]$. The *short-circuited net* $\bar{N} \coloneqq (P, T \cup \{\bar{t}\}, F \cup \{(p_{final}, \bar{t}), (\bar{t}, p_{init})\}, \ell \cup \{(\bar{t}, \ell(\bar{t}) \coloneqq \tau)\})$, $\bar{t} \notin T$, is an important tool to transfer the behavioral characteristics of LBFC-systems to workflow nets.

Definition 21 (Soundness [cf. 1, Theorem 11]**).** A workflow net is *sound* if and only if the *short-circuited* net is live and bounded.

Note that a sound free-choice workflow net is always safe [2, Lemma 1]. Using Definitions 20 and 21 and Corollary 1, we immediately obtain NP-membership:

Corollary 2. *On the class of sound free-choice workflow nets,* Align *is in* NP.

We use a small detour via a closely related problem to show that NP is indeed also a lower bound. The *membership problem* determines whether a trace is part of the language of a given Petri net, i.e., whether it occurs as the labeling of a complete firing sequence of a given accepting system.

Problem 2 (Membership (Member)*).*
Input: An alphabet Σ, an accepting system $S = (N, M_{init}, M_{final})$ with $N = (P, T, F, \ell)$ and labeling function $\ell\colon T \to \Sigma \cup \{\tau\}$, and a trace $\sigma \in \Sigma^*$.
Question: Is $\sigma \in \mathcal{L}(S)$?

It is easy to see that Member is a special case of Align where we look for a *perfect* alignment with costs 0.

Lemma 4. Member *is polynomial-time reducible to* Align.

Proof. Let Σ, $S = (N, M_{init}, M_{final})$, $N = (P, T, F, \ell)$, $\ell\colon T \to \Sigma \cup \{\tau\}$, and $\sigma \in \Sigma^*$ be an input of the Member problem. For the reduction to Align we make use of the standard cost function and set $k = 0$. In effect, we are looking for a *perfect* alignment between σ and S (i.e., with costs 0), which can only consist of synchronous moves and silent model moves. Thus, it requires that the trace σ can be obtained as the labeling of a firing sequence of the system S from M_{init} to M_{final}. This is precisely the decision problem Member. There is, however, a small issue: the system S is not necessarily easy-sound, but this is required for inputs of Align. To solve this, we add a new transition $t_{\blacktriangleright\!\blacktriangleright\!|}$ with ${}^\bullet t_{\blacktriangleright\!\blacktriangleright\!|} = M_{init}$, $t_{\blacktriangleright\!\blacktriangleright\!|}{}^\bullet = M_{final}$ and a new label not present in σ. This ensures easy-soundness, but, if this transition is taken in some alignment, we neither have a synchronous nor silent move, and thus the costs are at least 1. □

To show NP-hardness of Align, we use the fact that a live, safe T-system can emulate a *shuffle language* for which Member is NP-complete [22, 32]. Since T-systems are free-choice, we can combine Theorem 5 with Corollaries 1 and 2, to conclude the same for lbfc-systems and sound free-choice workflow nets.

Theorem 5. *On the class of live and safe T-systems,* Align *is* NP*-hard. Even when this class is further restricted to acyclic systems,* Align *remains* NP*-hard.*

Proof. Let $x \sqcup\!\sqcup y \coloneqq \{v_1 w_1 \cdots v_k w_k \mid x = v_1 \cdots v_k, y = w_1 \cdots w_k, v_i, w_i \in \Sigma^*, 1 \leq i \leq k\}$ be the *shuffle* of two words $x, y \in \Sigma^*$ and let $\mathcal{L}_1 \sqcup\!\sqcup \mathcal{L}_2 \coloneqq \bigcup\{w_1 \sqcup\!\sqcup w_2 \mid w_1 \in \mathcal{L}_1, w_2 \in \mathcal{L}_2\}$ be the shuffle of two languages $\mathcal{L}_1, \mathcal{L}_2 \subseteq \Sigma^*$. Furthermore, let $w \in \Sigma^*$ be a word and let $\mathcal{L}(w_1 \sqcup\!\sqcup w_2 \sqcup\!\sqcup \cdots \sqcup\!\sqcup w_n)$ be the *shuffle language* over words $w_1, w_2, \ldots, w_n \in \Sigma^*$. There exists a process tree T only using the sequence and parallel operator with $\mathcal{L}(T) = \mathcal{L}(w_1 \sqcup\!\sqcup w_2 \sqcup\!\sqcup \cdots \sqcup\!\sqcup w_n)$ [cf. 27]. Using the construction in [34], T can be transformed to a safe and sound workflow net that is also a T-system and acyclic. Because Member for a shuffle language (i.e., deciding whether $w \in \mathcal{L}(w_1 \sqcup\!\sqcup w_2 \sqcup\!\sqcup \cdots \sqcup\!\sqcup w_n)$) is already NP-complete [22, 32], Align is NP-hard on the class of live and safe (acyclic) T-systems by Lemma 4.

Corollary 3. *On the class of (acyclic)* lbfc*-systems and on the class of sound (acyclic) free-choice workflow nets,* Align *is* NP*-complete.*

7 ALIGN on General Safe and Sound Workflow Nets

We finally show that the free-choice assumption is needed to break the PSPACE barrier: In Theorem 8 and Corollary 5, we prove that ALIGN on safe and sound workflow nets is PSPACE-complete.

As a preparatory step, we show that the alignment problem is PSPACE-complete on the class of safe systems (Theorem 7). The usual approach for computing optimal alignments is via a reduction to the reachability problem in Petri nets. Therefore, we express the input trace itself in form of a Petri net:

Definition 22 (Trace System). Let $\sigma \in \Sigma^*$ be a trace over Σ. Its *trace system* $\mathcal{T}(\sigma) \coloneqq (N, M_{init}, M_{final})$ is an accepting system with $N = (P, T, F, \ell)$ where

- $P \coloneqq \{p_i \mid 0 \leq i \leq |\sigma|\}$ is the set of places,
- $T \coloneqq \{t_i \mid 1 \leq i \leq |\sigma|\}$ is the set of transitions,
- $F \coloneqq \bigcup_{1 \leq i \leq |\sigma|} \{(p_{i-1}, t_i), (t_i, p_i)\}$ is the flow relation,
- $\ell \colon T \to \Sigma$ is the labeling function such that $\ell\big(\langle t_i \rangle_{i=1}^{|\sigma|}\big) = \sigma$,
- $M_{init} = [p_0]$ is the initial marking, and
- $M_{final} = [p_{|\sigma|}]$ is the final marking.

Now, both, the process model and the trace are represented by a Petri net and can be combined using the synchronous product, which was introduced in [3] and is a special case of the product of Petri nets introduced in [33].

Definition 23 (Synchronous Product). Let Σ be an alphabet and let $S_1 = (N_1, M_{1,init}, M_{1,final})$ and $S_2 = (N_2, M_{2,init}, M_{2,final})$ be two accepting systems with $N_1 = (P_1, T_1, F_1, \ell_1)$, $N_2 = (P_2, T_2, F_2, \ell_2)$, $\ell_1 \colon T_1 \to \Sigma \cup \{\tau\}$, and $\ell_2 \colon T_2 \to \Sigma \cup \{\tau\}$ where P_1, T_1, P_2, and T_2 are pairwise disjoint sets. Furthermore, let $\gg \notin \Sigma, T_1, T_2$ be a distinguished no-move symbol. The *synchronous product* of S_1 and S_2, denoted by $S_1 \otimes S_2$, is the accepting system $S_1 \otimes S_2 \coloneqq (N, M_{init}, M_{final})$ with the Petri net $N \coloneqq (P, T, F, \ell)$ and labeling function $\ell \colon T \to \Sigma \cup \{\tau\}$ where

- $P \coloneqq P_1 \cup P_2$,
- $T \coloneqq \{(t_1, t_2) \in T_1 \times T_2 \mid \ell_1(t_1) = \ell_2(t_2)\} \cup (T_1 \times \{\gg\}) \cup (\{\gg\} \times T_2)$,
- $F \coloneqq \{(p, (t_1, t_2)) \in P \times T \mid (p, t_1) \in F_1 \vee (p, t_2) \in F_2\}$
  $\cup \{((t_1, t_2), p) \in T \times P \mid (t_1, p) \in F_1 \vee (t_2, p) \in F_2\}$,
- $(t_1, t_2) \mapsto \ell(t_1, t_2) \coloneqq \begin{cases} \ell_1(t_1) & t_1 \in T_1, \\ \ell_2(t_2) & t_1 \notin T_1, \end{cases}$
- $M_{init} \coloneqq M_{1,init} \oplus M_{2,init}$, and $M_{final} \coloneqq M_{1,final} \oplus M_{2,final}$.

As shown in [3], complete firing sequences in the synchronous product correspond to alignments between the trace and the model.

Proposition 1 ([3, Theorem 4.3.5]). *Given a trace σ and an accepting system S as process model, complete firing sequences of their synchronous product correspond to alignments between σ and S, i.e., $\Gamma_{\sigma,S} = \mathcal{L}(\mathcal{T}(\sigma) \otimes S)$.*

Furthermore, the product structure directly transfers to the reachability set:

Proposition 2 ([3, Theorem 4.3.4, 33, Theorem 4.1])**.** *Given two systems S_1 and S_2, any combination $M_1 \oplus M_2$ of a reachable marking $M_1 \in [S_1\rangle$ and a reachable marking $M_2 \in [S_2\rangle$ is a reachable marking in the synchronous product $S_1 \otimes S_2$, i.e., $\forall M_1 \in [S_1\rangle, M_2 \in [S_2\rangle \colon M_1 \oplus M_2 \in [S_1 \otimes S_2\rangle$ and vice versa.*

Corollary 4. *Given a b_1-bounded system S_1 and a b_2-bounded system S_2, their synchronous product $S_1 \otimes S_2$ is $\max\{b_1, b_2\}$-bounded.*

We now draw the connection to the *reachability problem* and its cost-variant:

Problem 3 (Reachability (Reach)*).*
Input: A system (N, M_0) with $N = (P, T, F, \ell)$ and a marking $M \in \mathbb{N}^P$.
Question: Is $M \in [N, M_0\rangle$?

Problem 4 (Minimum-Cost Reachability (MinCostReach)*).*
Input: A system (N, M_0) with a Petri net $N = (P, T, F, \ell)$, a marking $M \in \mathbb{N}^P$, a cost function $c \colon T \to \mathbb{Q}_{\geq 0}$ and a number $k \in \mathbb{Q}_{\geq 0}$.
Question: Is there a $\sigma \in T^*$ such that $(N, M_0)[\sigma\rangle(N, M)$ and $\sum c(\sigma) \leq k$?

Since we can also assign costs of 0 to any transition, MinCostReach is a generalization of Reach and we have:

Lemma 5. Reach *is polynomial-time reducible to* MinCostReach.

We can now show that MinCostReach on safe systems is in PSPACE:

Theorem 6. *On the class of safe systems,* MinCostReach *can be decided in polynomial space (in short:* MinCostReach $\in$ PSPACE*).*

Proof. To find a *deterministic* PSPACE algorithm for MinCostReach, we use Savitch's Theorem [25]: for every *non-deterministic* PSPACE algorithm, there also exists an equivalent *deterministic* algorithm.

Let (N, M_0), $N = (P, T, F, \ell)$, c, M, and t be an input of the MinCostReach problem. That is, (N, M_0) is a safe system with a Petri net $N = (P, T, F, \ell)$, $c \colon T \to \mathbb{Q}_{\geq 0}$ is a cost function, $M \in \mathbb{N}^P$ is a marking, and $t \in \mathbb{Q}_{\geq 0}$ is a cost limit. The algorithm stores a marking $\overline{M}$, which is initially set to $\overline{M} = M_0$, and a cost value $\bar{c}$, initially set to $\bar{c} = 0$. Note that since (N, M_0) is safe, a marking of N can be stored in polynomial space. As long as $\overline{M} \neq M$, the algorithm non-deterministically chooses a transition $t \in T$ enabled in marking $\overline{M}$ and computes the marking $\overline{M}'$ after firing t, i.e., $(N, \overline{M})[t\rangle(N, \overline{M}')$. Then, the stored marking is set to $\overline{M} := \overline{M}'$ and the stored cost value is set to $\bar{c} := \bar{c} + c(t)$.

If M is not reachable, the algorithm does not necessarily terminate. Thus, we add a counter which counts the number of fired transitions. Because (N, M_0) is safe, it has at most $2^{|P|}$ reachable markings. Therefore, we can stop if more than $2^{|P|} - 1$ transitions were fired. If the algorithm reaches the marking M and $\bar{c} \leq t$, it stops and M can be reached within the cost limit. If the counter exceeds $2^{|P|} - 1$ or $\bar{c}$ exceeds t, M cannot be reached within the cost limit. □

Since Reach on safe systems is PSPACE-complete [10, 11], we can conclude:

Lemma 6. *On the class of safe systems,* MinCostReach *is* PSPACE*-complete.*

Proof. On the class of safe systems, Reach is PSPACE-complete [10, 11]. By Lemma 5, MinCostReach is PSPACE-hard. In combination with Theorem 6, MinCostReach is PSPACE-complete on the class of safe systems. □

The next result allows us to transfer the PSPACE-bound to Align.

Lemma 7. Align *is polynomial-time reducible to* MinCostReach.

Proof. Let Σ, $S = (N, M_{init}, M_{final})$, $N = (P, T, F, \ell)$, $\ell \colon T \to \Sigma \cup \{\tau\}$, $\sigma \in \Sigma^*$, $c \colon LM_{\Sigma,S} \to \mathbb{Q}_{\geq 0}$, and k be an input of the Align problem. That is, Σ is an alphabet representing the set of activities, $S = (N, M_{init}, M_{final})$ is an easy-sound system, i.e., $\phi(S) \neq \emptyset$, with the Petri net $N = (P, T, F, \ell)$ and labeling function $\ell \colon T \to \Sigma \cup \{\tau\}$, $\sigma \in \Sigma^*$ is a trace over the alphabet Σ, and $c \colon LM_{\Sigma,S} \to \mathbb{Q}_{\geq 0}$ is a function which assigns costs to each legal move between Σ and S.

According to [3], finding an optimal alignment between σ and S is identical to finding a cost-minimal complete firing sequence in the synchronous product net $\mathcal{T}(\sigma) \otimes S$. Let $\mathcal{T}(\sigma) \otimes S \coloneqq (N', M'_{init}, M'_{final})$ where $N' \coloneqq (P', T', F', \ell')$ and in particular $T' \subseteq LM_{\Sigma,S}$. Therefore, a solution to MinCostReach with a system (N', M'_{init}) where $N' = (P', T', F', \ell')$, a marking M'_{final}, a cost function c, and a number k as input is also a solution to Align. □

Lemma 8. MinCostReach *is polynomial-time reducible to* Align.

Proof. Let (N, M_0), M, c, and k be an input for the MinCostReach problem, i.e., (N, M_0) is a system with Petri net $N = (P, T, F, \ell)$, $M \in \mathbb{N}^P$ the target marking, $c \colon T \to \mathbb{Q}_{\geq 0}$ a cost function, and $k \in \mathbb{Q}_{\geq 0}$ a threshold. It is to decide if M can be reached from M_0 with costs at most k.

To map this to an input of Align, we make use of the empty trace $\sigma = \langle\rangle$. Aligning the empty trace corresponds to finding a firing sequence from the initial marking M_0 to the final marking M with minimal costs. Similarly as in the proof of Lemma 4, we have one technical problem: the system (N, M_0, M) might not be easy-sound. Again, we can solve this by adding a transition $t_{\blacktriangleright\!\blacktriangleright}$ which allows the system to move from M_0 to M in one step. By making this transition very expensive (at least $k + 1$), we ensure easy-soundness, and, in case M is not reachable from M_0, the optimal alignment has costs at least $k + 1$. □

Altogether, this yields our first main result of this section:

Theorem 7. *On the class of safe systems,* Align *is* PSPACE*-complete.*

Proof. A trace system is safe by definition, the synchronous product considered in Lemma 7 is also safe according to Corollary 4. Hence, with Lemma 6 we have Align $\in$ PSPACE on the class of safe systems. Due to Lemmas 6 and 8, Align is also PSPACE-hard on the class of safe systems and thus PSPACE-complete. □

We finally turn our attention to safe and *sound* workflow nets, i.e., we add the liveness assumption. The question is whether this property suffices to reduce the complexity of the alignment problem. This turns out not to be the case:

Theorem 8. *There is a polynomial time algorithm which transforms a deterministic Turing machine $\mathcal{M}$ with polynomial space bound $p(n)$ and an input w into a safe and sound workflow net $S = (N, M_{init}, M_{final})$ and a trace σ such that, with respect to the standard cost function, S and σ can be aligned with 0 costs if and only if $\mathcal{M}$ accepts w.*

Proof. We extend the construction in [10, Theorem 4]. First, we make some assumptions on $\mathcal{M}$ which can be guaranteed by preprocessing. When the (deterministic, single tape) Turing machine $\mathcal{M}$ is started with input w, during the computation, the head of $\mathcal{M}$ only moves between positions 0 and $p(n)$, where $n = |w|$, starting at position 0. Moreover, each computation is finite, i.e., the machine $\mathcal{M}$ never repeats a configuration. In particular, $\mathcal{M}$ halts on every input and either accepts or rejects. Finally, there is precisely one accepting and one rejecting configuration of the machine $\mathcal{M}$. To guarantee this, one can implement a subroutine such that, whenever the machine $\mathcal{M}$ enters a final state (accepting or rejecting), then the tape is cleared, the head moves back to position 0, and the machine enters a unique accepting or rejecting state, respectively.

Let $\mathcal{M} = (K, \Sigma, \Gamma, \delta, q_0, q_+, q_-, \perp)$ be a deterministic Turing machine where K is the set of states, Σ the input alphabet, Γ the tape alphabet, $\delta\colon K \setminus \{q_+, q_-\} \times \Gamma \to K \times \Gamma \times \{-1, 1, 0\}$ the transition function (-1 (1) means the head moves one position to the left (right), and 0 means no move), $q_0 \in K$ the initial state, $q_+ \in K$ the unique accepting state, $q_- \in K$ the unique rejecting state, and $\perp \in \Gamma \setminus \Sigma$ the blank symbol. To encode the computation of $\mathcal{M}$ on input w, we define a workflow net $N = (P, T, F, \ell)$ with a set of places P consisting of:

- the initial place p_{init} and the final place p_{final},
- a place p^K_q for each state $q \in K$ (a token in p^K_q indicates that in the current configuration $\mathcal{M}$ is in state q),
- a place p^H_i for each possible head position $i \in \{0, \ldots, p(n)\}$ (a token in p^H_i indicates that in the current configuration the head is at position i),
- a place $p^C_{i,a}$ for each possible tape cell content, i.e., for each combination of a valid position $i \in \{0, \ldots, p(n)\}$ and a tape symbol $a \in \Gamma$ (a token in $p^C_{i,a}$ indicates that in the current configuration tape cell i holds symbol a).

With this preparation, we identify configurations of $\mathcal{M}$ with markings of N. To simulate the computation, we introduce the following set of transitions T:

- One transition $t_{\blacktriangleright}$ that yields the initial configuration, i.e., $\bullet t_{\blacktriangleright} = \{p_{init}\}$ and $t_{\blacktriangleright}\bullet = \{p^K_{q_0}\} \cup \{p^H_0\} \cup \{p^C_{i,w_i} \mid i < |w|\} \cup \{p^C_{i,\perp} \mid |w| \leq i \leq p(n)\}$.
- One transition $t_{\checkmark}$ that finalizes the computation when the (unique) accepting configuration is reached, i.e., $t_{\checkmark}\bullet = \{p_{final}\}$ and $\bullet t_{\checkmark} = \{p^K_{q_+}\} \cup \{p^H_0\} \cup \{p^C_{i,\perp} \mid i \leq p(n)\}$. This transition is labeled by $\checkmark$ and there is no other transition labeled by $\checkmark$. In the same way, we add one transition $t_{\times}$ that finalizes the computation when the (unique) rejecting configuration is reached.
- For each possible computational step, we introduce a distinct transition: for each state $q \in K$, symbol $a \in \Gamma$ with $\delta(q, a) = (q', b, d)$, and for each head position $i \in \{0, \ldots, p(n)\}$, we introduce a transition $t[q, a, i]$ that models

the configuration change which occurs when $\mathcal{M}$ is in state q, the head is at position i, and reads the symbol a. More precisely, $\bullet t[q,a,i] = \{p_q^K, p_i^H, p_{i,a}^C\}$ and $t[q,a,i]\bullet = \{p_{q'}^K, p_{i+d}^H, p_{i,b}^C\}$.

By construction, we can trigger the simulation from the initial marking $M_{init} = [p_{init}]$ by firing $t_{\blacktriangleright}$. This generates the initial configuration of the computation of $\mathcal{M}$ on w. Since the machine $\mathcal{M}$ is deterministic, from that point onward there is at most one transition of the form $t[q,a,i]$ that can be fired in the current marking/configuration. This transition, in turn, updates the marking/configuration according to the transition function of $\mathcal{M}$. Since we have prepared $\mathcal{M}$ in such a way that the computation is acyclic, we will finally reach the unique accepting or rejecting configuration from which we can fire $t_{\checkmark}$ or $t_{\times}$, respectively, to reach the final marking $M_{final} = [p_{final}]$. Giving the one-to-one correspondence between markings of N and configurations of $\mathcal{M}$, the resulting net is safe (every reachable marking corresponds to a configuration in the sense described above and such markings only hold at most one token per place).

The problem is that N is not sound. In general, it might be that a transition of the form $t[q,a,i]$ can never fire simply because the computation of $\mathcal{M}$ on w does never run into an enabling configuration. Also, we can either fire $t_{\checkmark}$ or $t_{\times}$ from the initial marking, but not both. To overcome this, we first add a new place p_{aux} indicating when the net is in *auxiliary* mode. Then, we add two new transitions $t^0_{\checkmark}$ and $t^0_{\times}$ which can fire at the initial marking and activate $t_{\checkmark}$ and $t_{\times}$: we set $\bullet t^0_{\checkmark} = \{p_{init}\}$ and $t^0_{\checkmark}\bullet = \bullet t_{\checkmark}$ and analogously for $t^0_{\times}$. In other words, $t^0_{\checkmark}$ and $t^0_{\times}$ produce the two unique markings where precisely $t_{\checkmark}$ or $t_{\times}$ is enabled, respectively. So, we can reach the final marking by firing either of them (note that transitions of the form $t[q,a,i]$ are not enabled in these terminal configurations of $\mathcal{M}$). Second, for each transition $t[q,a,i]$ we add two new transitions $t^0[q,a,i]$ and $t^1[q,a,i]$ which activate and deactivate $t[q,a,i]$ from the initial marking, i.e.,

$$\bullet t^0[q,a,i] = \{p_{init}\}, \quad \text{and} \quad t^0[q,a,i]\bullet = \{p_{aux}\} \cup \bullet t[q,a,i],$$
$$\bullet t^1[q,a,i] = \{p_{aux}\} \cup t[q,a,i]\bullet, \quad \text{and} \quad t^1[q,a,i]\bullet = \{p_{final}\}.$$

In this way, we can move via the sequence $\langle t^0[q,a,i], t[q,a,i], t^1[q,a,i]\rangle$ from the initial to the final marking and fire $t[q,a,i]$ along the way. However, there is one subtlety we have to discuss. In contrast to the case of $t_{\checkmark}$ and $t_{\times}$, the manual activation and firing of $t[q,a,i]$ might enable transitions different from $t^1[q,a,i]$. In fact, if the head does not move in state q while reading a, three tokens would be produced by $t[q,a,i]$ which would allow the net to fire another transition of the form $t[q',b,i]$. However, all such transitions are conservative in the sense that they do not alter the total count of tokens (they all consume and produce three tokens). Thus, the total count of tokens remains three which means that we will never be able to fire one of the final transitions $t_{\checkmark}$ or $t_{\times}$. Since $\mathcal{M}$ does not allow cyclic computations, eventually we get stuck in the simulation component after firing one transition of the form $t[q',b,i]$ which moves the head to the left or right (note that for the new cell, we are missing a token in a place $p^C_{a,i+d}$ for $d \in \{-1,1\}$). In this setting we can simply fire $t^1[q',b,i]$ which removes the

tokens produced by $t[q', b, i]$ and enters the final marking. Finally, all introduced auxiliary transitions get the extra label ◘. Note that we cannot mix a proper simulation of the computation of $\mathcal{M}$ on w triggered by firing $t_{\blacktriangleright}$ with a transition of the form $t^1[q, a, i]$ since this transition requires a token in p_{aux}. Altogether, by this second extension each transition in the workflow net can be fired from the initial marking and we maintain the property to always reach the final marking.

Finally, let $\sigma := \langle \checkmark \rangle$. Then, we claim that σ and the safe and sound workflow net S can be aligned with costs 0 (wrt. the standard cost function) if and only if $\mathcal{M}$ accepts input w. Clearly, if $\mathcal{M}$ accepts w, the simulation triggered by firing $t_{\blacktriangleright}$ simulates the computation via silent transitions of the form $t[q, a, i]$ until the single transition $t_{\checkmark}$ with label ✔ can eventually be fired synchronously with the symbol ✔ in σ. This does not incur any costs since we only have one synchronous move and several silent moves in the net. If, on the other hand, the computation of $\mathcal{M}$ on w is rejecting, there is no way to align σ with costs 0. In fact, if any of the auxiliary transitions is used, this will immediately lead to a model move since σ does not contain the symbol ◘. If, on the other hand the proper simulation of $\mathcal{M}$ on w is started via $t_{\blacktriangleright}$, then this will end up in a completely silent run ending with $t_{\blacktriangleright}$ and we would require a log move for the symbol ✔ in the trace. □

Together with PSPACE-membership for safe systems (Theorem 7), we get:

Corollary 5. *On the class of safe and sound workflow nets,* Align *is* PSPACE*-complete.*

8 Conclusion

We proved that the high algorithmic costs for computing alignments are unavoidable: an efficient algorithm for alignments on sound workflow nets (the standard model in process mining) does not exist (assuming $\mathsf{P} \neq \mathsf{PSPACE}$). Furthermore, we derived better algorithmic bounds for important model classes, such as the class of sound free-choice workflow nets which, in turn, includes all process trees.

Our results also show that for a complexity-theoretic understanding of alignments, we cannot simply refer to the reachability problem. For instance, on live, bounded, free-choice systems, the reachability and alignment problem are both NP-complete (see [16] and Section 5). However, if we further assume cyclicity (i.e., the initial marking is reachable from any other marking), the complexity of reachability drops to P while the alignment problem remains NP-complete (see [12] and Section 5). In particular, this complexity gap also holds for sound free-choice workflow nets as well as simpler model classes like Partially Ordered Workflow Language (POWL) models [20] or process trees [27].

For future research, we want to investigate in how far the bounds of the *Shortest Sequence Theorem* can be further improved on sound free-choice workflow nets. Potentially, this would allow us to generalize our ILP encoding for process trees from [27] to sound free-choice workflow nets and can lead to a much more efficient alignment algorithm on this class in practice. On the more theoretical level, we want to dive deeper into the complexity structure of alignments, in particular, regarding the influence of different parameters of the models.

References

[1] W. M. P. van der Aalst. "Verification of workflow nets." In: *Application and Theory of Petri Nets 1997*. ICATPN 1997. Ed. by P. Azéma and G. Balbo. Vol. 1248. LNCS. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997, pp. 407–426. ISBN: 978-3-540-69187-7. DOI: 10.1007/3-540-63139-9_48.

[2] W. M. P. van der Aalst. "Workflow Verification: Finding Control-Flow Errors Using Petri-Net-Based Techniques." In: *Business Process Management. Models, Techniques, and Empirical Studies*. Ed. by W. M. P. van der Aalst, J. Desel, and A. Oberweis. Vol. 1806. LNCS. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000. Chap. 11, pp. 161–183. ISBN: 978-3-540-45594-3. DOI: 10.1007/3-540-45594-9_11.

[3] A. Adriansyah. "Aligning observed and modeled behavior." PhD thesis. Technische Universiteit Eindhoven, 2014. ISBN: 978-90-386-3574-3. DOI: 10.6100/IR770080.

[4] A. V. Aho and T. G. Peterson. "A Minimum Distance Error-Correcting Parser for Context-Free Languages." In: *SIAM Journal on Computing* 1.4 (1972), pp. 305–312. DOI: 10.1137/0201022.

[5] V. Bloemen, J. van de Pol, and W. M. P. van der Aalst. "Symbolically Aligning Observed and Modelled Behaviour." In: *Application of Concurrency to System Design*. ACSD 2018. IEEE Computer Society, 2018, pp. 50–59. ISBN: 978-1-5386-7013-2. DOI: 10.1109/ACSD.2018.00008.

[6] M. Boltenhagen, T. Chatain, and J. Carmona. "Generalized Alignment-Based Trace Clustering of Process Behavior." In: *Application and Theory of Petri Nets and Concurrency*. PETRI NETS 2019. Ed. by S. Donatelli and S. Haar. Vol. 11522. LNCS. Cham: Springer International Publishing, 2019, pp. 237–257. ISBN: 978-3-030-21571-2. DOI: 10.1007/978-3-030-21571-2_14.

[7] M. Boltenhagen, T. Chatain, and J. Carmona. "Optimized SAT encoding of conformance checking artefacts." In: *Computing* 103.1 (2021), pp. 29–50. DOI: 10.1007/s00607-020-00831-8.

[8] J. Carmona, B. F. van Dongen, A. Solti, and M. Weidlich. *Conformance Checking. Relating Processes and Models*. Cham: Springer International Publishing, 2018. ISBN: 978-3-319-99413-0. DOI: 10.1007/978-3-319-99414-7.

[9] J. Carmona, B. F. van Dongen, and M. Weidlich. "Conformance Checking: Foundations, Milestones and Challenges." In: *Process Mining Handbook*. Ed. by W. M. P. van der Aalst and J. Carmona. Vol. 448. LNBIP. Cham: Springer International Publishing, 2022. Chap. 5, pp. 155–190. ISBN: 978-3-031-08847-6. DOI: 10.1007/978-3-031-08848-3_5.

[10] A. Cheng, J. Esparza, and J. Palsberg. "Complexity Results for 1-safe Nets." In: *Foundations of Software Technology and Theoretical Computer Science*. FSTTCS 1993. Ed. by R. K. Shyamasundar. Vol. 761. LNCS. Berlin, Heidelberg: Springer, 1993, pp. 326–337. ISBN: 978-3-540-48211-6. DOI: 10.1007/3-540-57529-4_66.

[11] A. Cheng, J. Esparza, and J. Palsberg. "Complexity results for 1-safe nets." In: *Theoretical Computer Science* 147.1-2 (1995), pp. 117–136. DOI: 10.1016/0304-3975(94)00231-7.

[12] J. Desel and J. Esparza. "Reachability in cyclic extended free-choice systems." In: *Theoretical Computer Science* 114.1 (1993), pp. 93–118. DOI: 10.1016/0304-3975(93)90154-L.

[13] J. Desel and J. Esparza. *Free Choice Petri Nets*. Cambridge Tracts in Theoretical Computer Science 40. Cambridge: Cambridge University Press, 1995. ISBN: 978-0-521-01945-3. DOI: 10.1017/CBO9780511526558.

[14] J. Desel and J. Esparza. "Shortest Paths in Reachability Graphs." In: *Journal of Computer and System Sciences* 51.2 (1995), pp. 314–323. DOI: 10.1006/jcss.1995.1070.

[15] B. F. van Dongen. "Efficiently Computing Alignments. Using the Extended Marking Equation." In: *Business Process Management*. BPM 2018. Ed. by M. Weske, M. Montali, I. Weber, and J. vom Brocke. Vol. 11080. LNCS. Cham: Springer International Publishing, 2018, pp. 197–214. ISBN: 978-3-319-98647-0. DOI: 10.1007/978-3-319-98648-7_12.

[16] J. Esparza. "Reachability in live and safe free-choice Petri nets is NP-complete." In: *Theoretical Computer Science* 198.1-2 (1998), pp. 211–224. DOI: 10.1016/S0304-3975(97)00235-1.

[17] J. Esparza. "Decidability and complexity of Petri net problems — An introduction." In: *Lectures on Petri Nets I: Basic Models: Advances in Petri Nets*. Ed. by W. Reisig and G. Rozenberg. Vol. 1491. LNCS. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 374–428. ISBN: 978-3-540-49442-3. DOI: 10.1007/3-540-65306-6_20.

[18] J. Esparza and M. Nielsen. "Decidability Issues for Petri Nets." In: *BRICS Report Series* 1.8 (1994). DOI: 10.7146/brics.v1i8.21662.

[19] N. D. Jones, L. H. Landweber, and Y. E. Lien. "Complexity of some problems in Petri nets." In: *Theoretical Computer Science* 4.3 (1977), pp. 277–299. DOI: 10.1016/0304-3975(77)90014-7.

[20] H. Kourani and S. J. van Zelst. "POWL: Partially Ordered Workflow Language." In: *Business Process Management*. BPM 2023. Ed. by C. Di Francescomarino, A. Burattin, C. Janiesch, and S. Sadiq. Vol. 14159. LNCS. Cham: Springer Nature Switzerland, 2023, pp. 92–108. ISBN: 978-3-031-41620-0. DOI: 10.1007/978-3-031-41620-0_6.

[21] M. de Leoni and A. Marrella. "Aligning Real Process Executions and Prescriptive Process Models through Automated Planning." In: *Expert Systems with Applications* 82 (2017), pp. 162–183. DOI: 10.1016/j.eswa.2017.03.047.

[22] A. Mansfield. "On the computational complexity of a merge recognition problem." In: *Discrete Applied Mathematics* 5.1 (1983), pp. 119–122. DOI: 10.1016/0166-218X(83)90021-5.

[23] G. Pighizzini. "How Hard Is Computing the Edit Distance?" In: *Information and Computation* 165.1 (2001), pp. 1–13. DOI: 10.1006/inco.2000.2914.

[24] A. Rozinat and W. M. P. van der Aalst. "Conformance checking of processes based on monitoring real behavior." In: *Information Systems* 33.1 (2008), pp. 64–95. DOI: 10.1016/j.is.2007.07.001.

[25] W. J. Savitch. "Relationships Between Nondeterministic and Deterministic Tape Complexities." In: *Journal of Computer and System Sciences* 4.2 (1970), pp. 177–192. DOI: 10.1016/S0022-0000(70)80006-X.

[26] C. T. Schwanen, W. Pakusa, and W. M. P. van der Aalst. "A Dynamic Programming Approach for Alignments on Process Trees." In: *Process Mining Workshops. ICPM 2024 International Workshops, Lyngby, Denmark, October 14–18, 2024, Revised Selected Papers*. ICPM 2024. Ed. by A. Delgado and T. Slaats. Vol. 533. LNBIP. Cham: Springer Nature Switzerland, 2025, pp. 84–97. ISBN: 978-3-031-82224-7. DOI: 10.1007/978-3-031-82225-4_7.

[27] C. T. Schwanen, W. Pakusa, and W. M. P. van der Aalst. "Process Tree Alignments." In: *Enterprise Design, Operations, and Computing*. EDOC 2024. Ed. by J. Borbinha, T. Prince Sales, M. Mira Da Silva, H. A. Proper, and M. Schnellmann. Vol. 15409. LNCS. Cham: Springer International Publishing, 2025. ISBN: 978-3-031-78337-1. DOI: 10.1007/978-3-031-78338-8_16.

[28] F. Taymouri and J. Carmona. "A Recursive Paradigm for Aligning Observed Behavior of Large Structured Process Models." In: *Business Process Management*. BPM 2016. Ed. by M. La Rosa, P. Loos, and O. Pastor. Vol. 9850. LNCS. Cham: Springer International Publishing, 2016, pp. 197–214. ISBN: 978-3-319-45348-4. DOI: 10.1007/978-3-319-45348-4_12.

[29] F. Taymouri and J. Carmona. "Model and Event Log Reductions to Boost the Computation of Alignments." In: *Data-Driven Process Discovery and Analysis*. SIMPDA 2016. Ed. by P. Ceravolo, C. Guetl, and S. Rinderle-Ma. Vol. 307. LNBIP. Cham: Springer International Publishing, 2018, pp. 1–21. ISBN: 978-3-319-74160-4. DOI: 10.1007/978-3-319-74161-1_1.

[30] A. Valmari. "The State Explosion Problem." In: *Lectures on Petri Nets I: Basic Models: Advances in Petri Nets*. Ed. by W. Reisig and G. Rozenberg. Vol. 1491. LNCS. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 429–528. ISBN: 978-3-540-49442-3. DOI: 10.1007/3-540-65306-6_21.

[31] R. A. Wagner. "Order-n Correction for Regular Languages." In: *Communications of the ACM* 17.5 (1974), pp. 265–268. DOI: 10.1145/360980.360995.

[32] M. K. Warmuth and D. Haussler. "On the Complexity of Iterated Shuffle." In: *Journal of Computer and System Sciences* 28.3 (1984), pp. 345–358. DOI: 10.1016/0022-0000(84)90018-7.

[33] G. Winskel. "Petri Nets, Algebras, Morphisms, and Compositionality." In: *Information and Computation* 72.3 (1987), pp. 197–238. DOI: 10.1016/0890-5401(87)90032-0.

[34] S. J. van Zelst and S. J. J. Leemans. "Translating Workflow Nets to Process Trees: An Algorithmic Approach." In: *Algorithms* 13.11 (2020). DOI: 10.3390/a13110279.