Exact and Approximated Log Alignments for Processes with Inter-case Dependencies

Dominique Sommers, Natalia Sidorova, and Boudewijn van Dongen

Department of Mathematics and Computer Science Eindhoven University of Technology, Eindhoven, the Netherlands {d.sommers,n.sidorova,b.f.v.dongen}@tue.nl

Abstract. The execution of different cases of a process is often restricted by inter-case dependencies through e.g., queueing or shared resources. Various high-level Petri net formalisms have been proposed that are able to model and analyze coevolving cases. In this paper, we focus on a formalism tailored to conformance checking through alignments, which introduces challenges related to constraints the model should put on interacting process instances and on resource instances and their roles. We formulate requirements for modeling and analyzing resource-constrained processes, compare several Petri net extensions that allow for incorporating inter-case constraints. We argue that the Resource Constrained ν -net is an appropriate formalism to be used the context of conformance checking, which traditionally aligns cases individually failing to expose deviations on intercase dependencies. We provide formal mathematical foundations of the globally aligned event log based on theory of partially ordered sets and propose an approximation technique based on the composition of individually aligned cases that resolves inter-case violations locally.

Keywords: Petri nets · Conformance checking · Inter-case dependencies · Shared resources.

1 Introduction

Event logs record which activity is executed at which moment of time, and additionally they often include indications which resources were involved in which activity, mentioning the exact person(s) or machine(s). The availability of such event logs enables the use of conformance checking for resource-constrained processes, analyzing not only the single instance control-flow perspective, but also checking whether and where the actual process behavior recorded in an event log deviates from the resource constraints prescribed by a process model.

Process models, and specifically Petri nets with their precise semantics, are often used to describe and reason about the execution of a process. In many approaches, a process model considers a process instance (a case) in isolation from other cases [1]. In practice, however, a process instance is usually subject to interaction with other cases and/or resources, whose availability puts additional constraints on the process execution. In order to expose workflow deviations caused by inter-case dependencies, it is crucial to use models considering multiple cases simultaneously.

There are several approaches to modeling and analysis of processes with inter-case dependencies. In [7] and [12], Petri nets are extended with resources to model availability of durable resources, with multiple cases competing by claiming and releasing these shared resources. To distinguish the cases, v-Petri nets [22] incorporate name creation and management as a minimal extension to classical Petri nets, with the advantage that coverability and termination are still decidable, opposed to more advanced Petri net extensions. The functionality of ν -Petri nets is inherited in other extensions such as Catalog Petri nets [11], synchronizing proclet models [10], resource and instanceaware workflow nets (RIAW-nets) [18], DB-nets [19] and resource constrained ν -Petri nets [24], all with the ability to handle multiple cases simultaneously. For the latter, the cases are assumed to follow the same process, interacting via (abstract) shared resources in a one-to-many relation, i.e., a resource instance can be claimed by one case at a time. More sophisticated extensions allow for cases from various perspectives with manyto-many interactions, via e.g., concepts from databases, shared resources and proclet channels. This may impose, however, problems of undecidability during conformance checking, which we discuss in this work.

Many conformance checking techniques use *alignments* to expose where the behavior recorded in a log and the model agree, which activities prescribed by the model are missing in the log and which log activities should not be performed according to the model [8,3]. The usual focus is on the control flow of the process. In more advanced techniques [6,15,16,17], data and/or resource information is additionally incorporated in the alignments by considering these perspective only after the control flow [15], by balancing the different perspectives in a customizable manner [16] or by considering all perspectives at once [17]. These three types of techniques operate on a case-by-case basis, which can lead to misleading results in case of shared resources, e.g., when multiple cases claim the same resource simultaneously.

In our previous work we considered the execution of all process instances by aligning the complete event log to a resource constrained ν -Petri nets [24]. In this paper, we present our further steps: (1) We compare how the existing Petri net extensions support modeling and analysis of processes with inter-case dependencies by formulating the requirements to such models, and we argue that ν -nets are an appropriate formalism. (2) We employ the poset theory to provide mathematical foundations for aligning the complete event log and exposing deviations of inter-case dependencies; (3) We propose an approximation method for computing optimal alignments in practice, which tackles the limitation of the computational efficiency when computing the complete event log alignment. The approximation method is based on composing alignments for isolated cases first and then resolving inter-case conflicts and deviations in the log locally.

The paper is organized as follows. In Section 2 we introduce basic concepts of the poset theory, Petri nets and event logs. In Section 3 we compare different Petri net extensions. We provide the mathematical foundations of the complete event log alignment in Section 4. Section 5 presents the approximation method for computing alignments. We discuss implications of our work in Section 6.

2 Preliminaries

In this section, we introduce basic concepts related to Petri nets and event logs and present the notations that we will use throughout the paper.

2.1 Multisets and posets

We start with definitions and notation regarding multisets and partially ordered sets.

Definition 1. (Multiset) A multiset m over a set X is $m : X \to \mathbb{N}$. X^{\oplus} denotes the set of all multisets over X. We define the support supp(m) of a multiset m as the set $\{x \in X \mid m(x) > 0\}$. We list elements of the multiset as $[m(x) \cdot x \mid x \in X]$, and write |x| for m(x), when it is clear from context which multiset it concerns.

For two multisets m_1, m_2 over X, we write $m_1 \le m_2$ if $\forall_{x \in X} m_1(x) \le m_2(x)$, and $m_1 < m_2$ if $m_1 \le m_2 \land m_1 \ne m_2$. We define $m_1 + m_2 = [(m_1(x) + m_2(x)) \cdot x \mid x \in X]$, and $m_1 - m_2 = [\max(0, m_1(x) - m_2(x)) \cdot x \mid x \in X]$ for $m_1 \ge m_2$.

Furthermore, $m_1 \sqcup m_2 = [\max(m_1(x), m_2(x)) \cdot x \mid x \in X]$, $m_1 \sqcap m_2 = [\min(m_1(x), m_2(x)) \cdot x \mid x \in X]$.

In some cases, we consider multisets over a set X as vectors of length |X|, assuming an arbitrary but fixed ordering of elements of X.

Definition 2. (*Partial order, Partially ordered set, Antichains*) A partially ordered set (poset) $X = (\bar{X}, \prec_X)$ is a pair of a set \bar{X} and a partial order $\prec_X \subseteq X \times X$. We overload the notation and write $x \in X$ if $x \in \bar{X}$. For $x, y \in X$, we write $x \parallel_X y$ if $x \not\prec y \land y \not\prec x$ and $x \preceq y$ if $x \prec y \lor x = y$.

Given \prec_X , we define \prec_X^+ to be the smallest transitively closed relation containing \prec_X . Thus \prec_X^+ is a partial order with $\prec_X \subseteq \prec_X^+$.

We extend the standard set operations of union, intersection, difference and subsets to posets: for any two posets X and Y, $X \circ Y = (\overline{X} \circ \overline{Y}, (\prec_X \circ \prec_Y)^+)$, with $\circ \in \{\cup, \cap, \setminus\}$ and $Y \subseteq X$ iff $\overline{Y} \subseteq \overline{X}$ and $\prec_Y = \prec_X \cap (\overline{Y} \times \overline{Y})$.

A poset A is an antichain if no elements of A are comparable, i.e., $\forall_{x,y\in A} x || y$. For poset X, $\mathcal{A}(X)$ denotes the set of all antichains $A \subseteq X$, and $\mathcal{A}^+(X)$ is the set of all maximal antichains: $\mathcal{A}^+(X) = \{A \mid A \in \mathcal{A}(X), \forall_{B \in \mathcal{A}(X)} B \subseteq A \implies B = A\}.$

Two special maximal antichains are the minimum and maximum elements of X, defined by $\min(X) = \{x \mid x \in X, \forall_{y \in X} y \not\prec x\} \in \mathcal{A}^+(X) \text{ and } \max(X) = \{x \mid x \in X, \forall_{y \in X} x \not\prec y\} \in \mathcal{A}^+(X).$

We define $X^{<} = \{(\bar{Y}, \prec_Y) \mid \bar{Y} = \bar{X}, \prec_X \subseteq \prec_Y, \forall_{a,b \in Y, a \neq b} a \mid \|Yb\}$ to be the set of totally ordered permutations of X that respect the partial order.

Definition 3. (Interval, prefix and postfix in a poset) With a poset X and two antichains $A, B \in \mathcal{A}(X)$, the closed jhkcbvinterval from A to B is the subposet defined as follows: $[A,B] = (\overline{AB}, \prec_X \cap (\overline{AB} \times \overline{AB}))$ with $\overline{AB} = \{x \mid x \in X, A \preceq x \preceq B\}$, and the half open and open intervals: $(A,B] = [A,B] \setminus A$, $[A,B) = [A,B] \setminus B$ and $(A,B) = [A,B) \setminus A$.

Artificial minimal and maximal elements are denoted as \perp and \top respectively, i.e., $\forall_{x \in X} \perp \prec x \prec \top$. $(\perp, A]$, (\perp, A) , $[A, \top)$ (A, \top) denote the corresponding prefixes and postfixes of an antichain $A \in \mathcal{A}(X)$ in X.

2.2 Petri nets

Petri nets can be used as a tool for the representation, validation and verification of workflow processes to provide insights in how a process behaves [21].

Definition 4. (Labeled Petri nets, Pre-set, Post-set) A labeled Petri net [20] is a tuple $N = (P, T, \mathcal{F}, \ell)$, with sets of places and transitions P and T, respectively, such that $P \cap T = \emptyset$, and a multiset of arcs $\mathcal{F} : (P \times T) \cup (T \times P) \to \mathbb{N}$ defining the flow of the net. $\ell : T \to \Sigma^{\tau} = \Sigma \cup \{\tau\}$ is a labeling function, assigning each transition t a label $\ell(t)$ from alphabet Σ or $\ell(u) = \tau$ for silent transitions.

We assume that the intersection, union and subsets are only defined for two labeled Petri nets N_1 , N_2 where $\forall_{t \in T_1 \cap T_2} \ell_1(t) = \ell_2(t)$.

Given an element $x \in P \cup T$ *, its* pre- and post-set ${}^{\bullet}x (x^{\bullet})$ *are multisets defined by* ${}^{\bullet}x = [\mathcal{F}(y, x) \cdot y \mid y \in P \cup T]$ and $x^{\bullet} = [\mathcal{F}(x, y) \cdot y \mid y \in P \cup T]$ resp.

Definition 5. (*Marking, Enabling and firing of transitions, Reachable markings)* A marking $m \in P^{\oplus}$ of a (labeled) Petri net $N = (P, T, \mathcal{F}, \ell)$ assigns how many tokens each place contains and defines the state of N.

With m and N, a transition $t \in T$ is enabled for firing iff $m \ge {}^{\bullet}t$. We denote the firing of t by $m \xrightarrow{t} m'$, where m' is the resulting marking after firing t and is defined by $m' = m - {}^{\bullet}t + t^{\bullet}$. For a transition sequence $\sigma = \langle t_1, \ldots, t_n \rangle$ we write $m \xrightarrow{\sigma} m'$ to denote the consecutive firing of t_1 to t_n . We say that m' is reachable from m and write $m \xrightarrow{*} m'$ if there is some $\sigma \in T^*$ such that $m \xrightarrow{\sigma} m'$.

 $\mathcal{M}(N) = P^{\oplus}$ and it denotes the set of all markings in net N and $\mathcal{R}(N,m)$ the set of markings reachable in net N from marking m.

Definition 6. (*Place invariant*) Let $N = (P, T, \mathcal{F}, \ell)$ be a Petri net. A place invariant [14] is a row vector $I : \mathbf{P} \to \mathbb{Q}$ such that $I \cdot \mathbf{F} = \mathbf{0}$, with \mathbf{P} and \mathbf{F} vector representations of P and \mathcal{F} . We denote the set of all place invariants as \mathcal{I}_N , which is a linear subspace of \mathbb{Q}^P .

The main property of a place invariant I in a net N with initial marking m_i is that $\forall_{m_1,m_2 \in \mathcal{R}(N,m_i)} I \cdot m_1 = I \cdot m_2$.

Definition 7. (Net system, Execution poset and sequence, Language) A net system is a tuple $SN = (N, m_i, m_f)$, where N is a (labeled) Petri net, and m_i and m_f are respectively the initial and final marking. An execution sequence in a net system SN = (N, m_i, m_f) is a firing sequence from m_i to m_f . Additionally, an execution poset is a poset of transition firings, where each totally ordered permutation is a firing sequence. The language of a net system SN is the set of all execution sequences in SN.

2.3 Event logs

An event log records activity executions as events including at least the occurred activity, the time of occurrence and the case identifier of the corresponding case. Often resources are also recorded as event attributes, e.g., the actors executing the action. It is generally known beforehand in which activities specific *resource roles* R are involved and which resource instances Id_r are involved in the process for each role $r \in R$. We assume that each resource has only one role (function) allowing to execute a predefined number of tasks, and therefore define the set Id_R of resource instances of all roles as the disjoint union of resource instance sets of roles: $Id_R = \bigcup_{r \in R} Id_r$. A *resource instance* $\rho \in Id_R$ with role $r \in R$ is equipped with capacity, making Id_r and Id_R both multisets.

Definition 8. (Event, Event log, Trace) An event e is a tuple (a, t, c, Id'_R) , with an activity $a = \text{activity}(e) \in \Sigma$, a timestamp $t = time(e) \in \mathbb{R}$, a case identifier $c = \text{case}(e) \in \text{Id}_c$ and a multiset of resource instances $\text{Id}'_R = \text{Res}(e) \leq \text{Id}_R$. Such an event represents that activity a occurred at timestamp t for case c and is executed by resource instances from Id'_R belonging to possibly different resource roles.

An event log L is a set of events with partial order \prec_L that respects the chronological order of the events, i.e., $\forall_{e_1,e_2\in L}time(e_1) < time(e_2) \Longrightarrow e_2 \not\prec_L e_1$. An event log can be partitioned into traces, defined as projections e.g., on the case identifiers or on the resources names. For every $c \in \mathrm{Id}_c$, L_c denotes a trace projected on the case identifier c defined by $L_c = (\{e \mid e \in L, \mathrm{case}(e) = c\}, \prec_{L_c})$ with $\prec_{L_c} = \{(e, e') \mid (e, e') \in \prec_L, \mathrm{case}(e) = \mathrm{case}(e') = c\}$.

Alternatively, we write $\langle e_1, e_2, \cdots \rangle$ for an event log which is totally ordered, and $a^{Id'_R}$ and $\underline{a}^{Id'_R}$ for events where the case is identified by the activity color (and bar position) and the time of occurrence is abstracted away from.

For a (labeled) Petri net modeling a process, the transitions' names or labels correspond to the activity names found in the recorded event log.

3 Modeling, analysis and simulation of case handling systems with inter-case dependencies

A classical Petri net models a process execution using transition firings and the corresponding changes of markings without making distinctions between different cases on which the modeled system works simultaneously. To create a case view, Workflow nets [2] model processes from the perspective of a single case. Systems in which cases interact with each other, e.g., by queueing or sharing resources, need to be modeled in a different way. We show from a modeling point how this boils down to multiple cases competing over shared tokens representing resources in a Petri net, which requires an extension on the formalism of the classical Petri nets. In Sec. 3.1, we motivate the requirements by providing examples, after which, in Sec. 3.2, we discuss whether existing Petri net extensions satisfy these requirements. We end, in Sec. 3.3 by proposing a *minimal extension* based on ν -Petri nets [22] that meets each requirement for simulation and analysis of resource-constrained processes.

3.1 Requirements imposed by inter-case dependencies

When modeling systems with inter-case dependencies, i.e., shared resources, simultaneous cases can interfere in each other's processing via the resources, causing inter-case dependencies. To model, simulate and analyze such behavior, the cases and resources,



Fig. 1: Example Petri net N_1 to argue the requirements, with token colors denoting different instances.

represented as tokens in a Petri net, should be handled together and simultaneously in the process model. This introduces the need for case (R1) and resource isolation (R2) as well as durable resources (R3) and case-resource correlations (R4), which regular Petri nets are not capable of. For analysis, like computing alignments (see Section 4), non-invertible functions can cause state-space explosions (R5). We show for each requirement, when not satisfied, how simulation and/or analysis concerning multiple simultaneous cases fails:

- R1 Distinguishable cases are required when dealing with multiple cases. Tokens involved in a firing of a transition should not belong to different cases, unless case batching is used. Mixing tokens from different cases, possible in classical Petri nets, can potentially cause model behavior that is not possible in the modeled system: Suppose we have a simple operation process modeled by Petri net N_1 , shown in Fig. 1, where a patient undergoes an operation involving the activities of preparation (o_p) , assistance (o_a) , closed surgery (o_{sc}) and open surgery (o_{so}) which is followed by closeup (o_c) . We assume case tokens to be indistinguishable. The language of $(N_1, [p_i, 2p_s], [p_f, 2p_s])$ is $\{\langle o_p, o_a, o_{sc} \rangle, \langle o_p, o_{sc}, o_a \rangle, \langle o_p, o_a, o_{so}, o_c \rangle, \langle o_p, o_c \rangle, \langle o_$ $\langle o_p, o_{so}, o_a, o_c \rangle$ and the language of the same net processing two cases with sufficient resources has to consist of all possible interleaving of two traces belonging to single cases. However, $\{\langle o_p, o_a, o_{sc}, o_p, o_a, o_{so}, o_c \rangle\}$ is included in the language of $(N_1, [2p_i, 2p_s], [2p_f, 2p_s])$, which is impossible to obtain by an interleaving of two single cases, as o_c is never enabled after o_{sc} fires. Here and later we use underlined symbols when referring to the second case in examples. From now on, we assume case tokens are distinguishable and we have $m_i(p_i) = (c, c)$;
- R2 Distinguishable resources are required when resource instances are uniquely identifiable. If the tokens in p_s are indistinguishable, $\langle \dots, o_{so}^{\{x\}}, o_{sc}^{\{x\}}, o_c^{\{x\}} \rangle$ belongs to the language of $(N_1, [2p_i, 2p_s], [2p_f, 2p_s])$. However, resource instance x can only be claimed by the second case after it has been released by the first case (by firing transition o_c), hence it should not be included in the language. From now on, we assume resource tokens are distinguishable and we have $m_i(p_s) = (x, y)$;
- R3 Resources are required to be *durable* when having a variable number of cases in the system simultaneously. In N_1 , the resource instances in p_s are modeled to be durable, since these instances are always released after being claimed. However, were arc (o_c, p_s) to be removed, problems arise when observed behavior concerns more than two cases, since after transition o_{so} fired twice, it is never enabled again, causing a deadlock;

Exact and Approximated Log Alignments for Processes with Inter-case Dependencies

- R4 Capturing case-resource correlation is required when dealing with multiple distinguishable cases and resources in order to keep track of which resource handles which case. Without it, the language of $(N_1, [2p_i, 2p_s], [2p_f, 2p_s])$ includes e.g., $\langle \dots, o_{so}^{\{x\}}, o_{so}^{\{y\}}, o_{c}^{\{x\}}, o_{c}^{\{y\}} \rangle$, which is undesirable as resources x and y have switched cases after transition oso is fired twice. Case-resource correlation should ensure, in this case, that transition o_c can only be fired using the same resource as was claimed by firing transition o_{so} ;
- R5 Operations on token values (e.g., guards, arc inscriptions) should be invertible and computable when aligning observed and modeled behavior in order to keep the problem decidable. Consider e.g., that patients enter the process by their name and birthdate v, which is transformed to an identifier c in the first transition by an operation f(v) on (o_p) . When activity o_p is missing for a patient, it is undecidable which value v should be inserted for the firing of o_p when f is not invertible.

3.2 **Existing Petri net extensions**

Several extensions on Petri nets have been proposed focusing on multi-case and/or multi-resource processes able to handle (some) inter-case dependencies. We go over each extension, describing how they satisfy (and violate) requirements listed in Sec. 3.1. We propose an extension, which combines concepts of the described extensions and satisfies all requirements.

Resource constrained workflow nets (RCWF-nets) [12] are Petri nets extended with resource constraints, where resources are durable units: they are claimed and then released again (R3). They define structural criteria for its correctness.

Definition 9. (Resource-constrained workflow net [12]) Let R be a set of resource roles. A net system $N = (P_p \uplus P_r, T, \mathcal{F}_p \uplus \mathcal{F}_r, m_i, m_f)$ is a resource-constrained workflow net (RCWF-net) with the set P_p of production places and the set $P_r = \{p_r \mid p_r \mid r \in \mathbb{N}\}$ $r \in R$ of resource places iff

- $\mathcal{F}_p : (P_p \times T) \cup (T \times P_p) \to \mathbb{N}$ and $\mathcal{F}_r : (P_r \times T) \cup (T \times P_r) \to \mathbb{N}$; $N_p = (P_p, T, \mathcal{F}_p, [m_i(p) \cdot p \mid p \in P_p], [m_f(p) \cdot p \mid p \in P_p])$ is a net system, called the production net of N.

The semantics of Petri nets is extended by having colored tokens on production places (R1) and as resources are shared across all cases, tokens on resource places are colorless $(\neg R2, \neg R4)$. A transition is enabled if and only if there are sufficient tokens on its incoming places using tokens of the same color on production places.

 ν -Petri nets [22] are an extension of Petri nets with pure name creation and name management, strictly surpassing the expressive power of regular Petri nets and they essentially correspond to the minimal object-oriented Petri nets of [13]. In a ν -Petri net, names can be created, communicated and matched which can be used to deal with authentication issues [23], correlation or instance isolation [9]. Name management is formalized by replacing ordinary tokens by distinguishable ones, thus adding color the the Petri net.

Definition 10. $(\nu$ -Petri net [22]) Let Var be a fixed set of variables. A ν -Petri net is a tuple ν -N = $\langle P, T, \mathcal{F} \rangle$, with a set of places P, a set of transitions T with $P \cap T = \emptyset$, and a flow function $\mathcal{F} : (P \times T) \cup (T \times P) \rightarrow Var^{\oplus}$ such that $\forall_{t \in T}, \Upsilon \cap^{\bullet} t = \emptyset \land t^{\bullet} \backslash \Upsilon \subseteq {}^{\bullet} t$, where ${}^{\bullet}t = \bigcup_{p \in P} supp(\mathcal{F}(p,t))$ and $t^{\bullet} = \bigcup_{p \in P} supp(\mathcal{F}(t,p))$. $\Upsilon \subset Var$ denotes a set of places a set of places

special variables ranged by ν, ν_1, \ldots to instantiate fresh names.

A marking of ν -N is a function $m : P \to Id^{\oplus}$. Id(m) denotes the set of names in m, i.e. $Id(m) = \bigcup_{p \in P} supp(m(p))$.

A mode μ of a transition t is an injection μ : $Var(t) \rightarrow Id$, that instantiates each variable to an identifier.

For a firing of transition t with mode μ , we write $m \xrightarrow{t_{\mu}} m'$. t is enabled with mode μ if $\mu(\mathcal{F}(p,t)) \subseteq m(P)$ for all $p \in P$ and $\mu(\nu) \notin Id(m)$ for all $\nu \in \Upsilon \cap Var(t) = supp(\bigcup_{p \in P} \mathcal{F}(p,t))$. The reached state after the firing of t with mode μ is the marking m', given by:

$$m'(p) = m(p) - \mu(\mathcal{F}(p,t)) + \mu(\mathcal{F}(t,p)) \text{ for all } p \in P$$
(1)

We denote T_{μ} to be the set of all possible transition firings.

 ν -Petri nets support instance isolation for cases and resources requiring the tokens involved in a transition firing to have matching colors (R1, R2). Due to the tokens having singular identifiers, correlation between cases and resources can not be captured (\neg R4).

Resource and instance-aware workflow nets (RIAW-nets) [18], are Petri nets combining the notions from above by defining similar structural criteria for handling resource constraints on top of ν -Petri nets. However, the resource places are assumed to only carry black tokens, not allowing for resource isolation and properly capturing the case-resource correlation.

Synchronizing proclets [10] are a type of Petri net that describe the behavior of processes with many-to-many interactions: unbounded dynamic synchronization of transitions, cardinality constraints limiting the size of the synchronization, and history-based correlation of token identities (R1,R2). This correlation is captured by message-based interaction, specifying attributes of a message as correlation attributes (R4). The correlation constraints are C_{init} , C_{match}^{\subseteq} and $C_{match}^{=}$, for initializing the attributes, partially and fully matching them. ν -Petri nets are at the basis of proclets handling multiple objects by separating their respective subnets. While the proclet formalism is sufficient for satisfying all requirements listed above, they extend to many-to-many relations, which lifts the restriction that a resource can only be claimed by a single case.

Object-centric Petri nets [4], similarly to synchronizing proclets, describe the behavior of processes with multiple perspectives and one-to-many and many-to-many relations between the different object types. These nets are a restricted variant of colored Petri nets where places are typed, tokens are identifiable referring to objects (R1,R2), and transitions can consume and produce a variable number of tokens. Correlation can be achieved with additional places of combined types (R4). Again, due to many-to-many relations, our one-to-many restriction on resources is lifted.

Database Petri nets (DB-nets) [19] are extensions of ν -Petri nets with multi-colored tokens that allows for multiple types of objects and their correlation (R1,R2,R4). Additionally, they support underlying read-write persistent storage consisting of a relational

Exact and Approximated Log Alignments for Processes with Inter-case Dependencies

database with full-fledged constraints. Special "view" places in the net are used to inspect the content of the underlying data, while transitions are equipped with database update operations. These are in the general sense not invertible causing undecidability ($\neg R5$).

Catalog Petri nets (CLog-nets) [11] are similar to DB-nets, but without the "write" operations (R1,R2,R4). The queries from view places in DB-nets have been relocated to transition guards, relying solely on the "read-only" modality for a persistent storage, however suffering from the same undecidability problem as these guards are not invertible in the general sense (\neg R5).

3.3 Resource constrained ν -Petri net with fixed color types

We combine conceptual ideas from the extensions described above, by extending RIAWnets, which inherit the modeling restrictions from RCWF-nets and name management from ν -Petri nets, using concepts from DB-nets and CLog-nets.

The resource places from RCWF-nets model the availability of resource instances by tokens, which is insufficient to capture correlation of cases by which they are claimed and released. We propose a minimal extension resource constrained ν -Petri nets (RC ν net) which additionally contain busy places $\bar{P}_r = \{\bar{p}_r \mid r \in R\}$ for each resource role. Token moves from p_r to \bar{p}_r show that the resource gets occupied, and moves from \bar{p}_r to p_r show that the resource becomes available. Also tests whether there are free/occupied resources can be modeled. A structural condition is imposed on the net to guarantee that resources are *durable*, meaning that resources can neither be created nor destroyed. This also implies that in the corresponding net system with initial and final marking m_i and m_f , $m_i(p_r) = m_f(p_r)$ and $m_i(\bar{p}_r) = m_f(\bar{p}_r)$, for any resource role $r \in R$.

Furthermore, similar to DB-nets and CLog-nets, we extend the tokens from carrying single data values to multiple. Where DB-nets and CLog-nets allow for a variable number of predefined color types, we restrict ourselves to two which are strictly typed, to distinguish between both cases and resources.

Definition 11. (Resource-constrained ν -Petri net) Let C^{ε} be the set of case ids Id_c extended with ordinary tokens, i.e., $\varepsilon \in \mathrm{Id}_c$, and $\mathrm{Id}_R^{\varepsilon}$ be the set of resource ids extended with ordinary tokens. A resource-constrained ν -Petri net $N = (P, T, \mathcal{F}, m_i, m_f)$ is a Petri net system with $\mathcal{F} : (P \times T) \cup (T \times P) \to (\operatorname{Var}_c^{\varepsilon} \times \operatorname{Var}_r^{\varepsilon})^{\oplus}$, where Var_c denote case variables and Var_r denote resource variables, allowing for two colored tokens. $P = (P_p \uplus P_r \uplus \overline{P}_r)$, with production places P_p and resource availability and busy places $P_r = \{p_r \mid r \in R\}$ and $\overline{P}_r = \{\overline{p}_r \mid r \in R\}$. The following modeling restrictions are imposed on N for each $r \in R$:

1. $\bullet p_r + \bullet \bar{p}_r = p_r^{\bullet} + \bar{p}_r^{\bullet}$, *i.e.*, $\forall_{t \in T} \mathcal{F}(p_r, t) + \mathcal{F}(\bar{p}_r, t) = \mathcal{F}(t, p_r) + \mathcal{F}(t, \bar{p}_r)$; 2. $m_i(p_r) = m_f(p_r)$ and $m_i(\bar{p}_r) = m_f(\bar{p}_r) = 0$;

A marking of N is a function $m : P \to (C^{\varepsilon} \times R^{\varepsilon})^{\oplus}$ with case ids C and resources R, which is a mapping from places to multisets of colored tokens.

A mode of a transition t is an injection $\mu : (Var_c^{\varepsilon} \times Var_r^{\varepsilon})(t) \to (C^{\varepsilon} \times R^{\varepsilon})$, that instantiates each variable to an identifier.

Proposition 1. The resource-constrained *v*-Petri nets as defined in Def. 11 satisfy requirements R1-R5, i.e., they allow to distinguish cases and resource instances which are durable, and capture case-resource correlation while restricting to operations that are invertible.

Proof. The two-colored strictly typed tokens distinguish both the cases (R1) and resource instances (R2) in the system. The modeling restrictions imposed on the RC ν -net enforce that for each resource role $r \in R$, tokens can only move between p_r and \bar{p}_r , i.e., we have the place invariant (1, 1) on p_r and \bar{p} , implying that $m(p_r) + m(\bar{p}_r) = m_i(p_r)$ for any reachable marking m, and that all resource tokens are returned to p_r when the net reaches its final marking, ensuring that resources are durable (R3). The two colors on tokens residing in \bar{p} capture correlation between cases and resources instances (R4), denoting by which case a resource instance is claimed throughout their interaction. As the transition firing's modes are bijective functions, each operation on N is invertible (R5).

Note that the RC ν -net formalism is a restricted version of DB-nets, CLog-net and synchronizing proclets, as all three can capture the behavior that can be modeled by RC ν -nets. DB-nets and CLog-nets additionally have database operations which we deem not relevant for our purposes. Synchronizing proclets allow for many-to-many interactions, while we assume that a resource instance cannot be shared by several cases at the same time.

4 Complete event logs alignments

Several state-of-the-art techniques in conformance checking use alignments to relate the recorded executions of a process with a model of this process [5]. An alignment shows how a log or trace can be replayed in a process model, which can expose deviations explaining either how the process model does not fit reality or how the reality differs from what should have happened.

Traditionally, this is computed for individual traces, however, as we show in previous work [24], this fails to expose deviations on a multi-case and -resource level in processes with inter-case dependencies as described in Sec. 3.3. In this section, we go over the foundations of alignments in Sec. 4.1 and show how we extend this to compute alignments of complete event logs in Sec.4.2.

4.1 Foundations of alignments

At the core of alignments are three types of moves: log, model, and synchronous moves (cf. Def. 12), indicating, respectively, that an activity from the log can not be mimicked in the process model, that the model requires the execution of some activity not observed in the log, and that observed and modeled behavior of an activity agree.

Definition 12. (Log, model and synchronous moves) Let L be an event log and $N = (P, T, \mathcal{F}, \ell, m_i, m_f)$ be a labeled ν -Petri net with T_{μ} the set of all possible firings in N. We define the set of log moves $\Gamma_l = \{(e, \gg) \mid e \in L\}$, the set of model moves $\Gamma_m = \{(e, p_i) \mid e \in L\}$. Exact and Approximated Log Alignments for Processes with Inter-case Dependencies

 $\{(\gg, t_{\mu}) \mid t_{\mu} \in T_{\mu}\}$ and the set of synchronous moves $\Gamma_s = \{(e, t_{\mu}) \mid e \in L, t_{\mu} \in T_{\mu}, \text{activity}(e) = \ell(t)\}$. As abbreviations, we write $\Gamma_{ls} = \Gamma_l \cup \Gamma_s, \ \Gamma_{lm} = \Gamma_l \cup \Gamma_m, \Gamma_{ms} = \Gamma_m \cup \Gamma_s, \text{ and } \Gamma_{lms} = \Gamma_l \cup \Gamma_m \cup \Gamma_s.$

Log moves and model moves can expose deviations of the real behavior from the model, by an *alignment* (cf. Def. 13) on a net (N, m_i, m_f) and event log L (possibly a single trace) which is a poset of moves from Def. 12 incorporating the event log and execution sequences in N from m_i to m_f :

Definition 13. (Alignment) An alignment $\gamma = \operatorname{align}(N, L)$ of an event log $L = (\overline{L}, \prec_L)$) and a labeled Petri net $N = (P, T, \mathcal{F}, \ell, m_i, m_f)$ is a poset $\gamma = (\overline{\gamma}, \prec_{\gamma})$, where $\overline{\gamma} \subseteq (\Gamma_l \cup \Gamma_s \cup \Gamma_m^{\oplus})$, having the following properties:

1.
$$\gamma \upharpoonright_L = L$$
 and $\prec_L \subseteq \prec_{\gamma \upharpoonright_L}$

2. $m_i \xrightarrow{\gamma \upharpoonright_T} m_f$, *i.e.*, $\forall_{\sigma \in (\gamma \upharpoonright_T)^<}, m_i \xrightarrow{\sigma} m_f$

with alignment projections on the log events $\gamma \upharpoonright_L$ and on the transition firings $\gamma \upharpoonright_{T_{\mu}}$:

$$\gamma \restriction_{L} = \left(\{ e \mid (e, t_{\mu}) \in \gamma \cap \Gamma_{ls} \}, \{ (e, e') \mid ((e, t_{\mu}), (e', t'_{\mu})) \in \prec_{\gamma} \cap (\Gamma_{ls} \times \Gamma_{ls}) \} \right)$$
(2)
$$\gamma \restriction_{T} = \left(\{ t_{\mu} \mid (e, t_{\mu}) \in \gamma \cap \Gamma_{ms} \}, \{ (t_{\mu}, t'_{\mu}) \mid ((e, t_{\mu}), (e', t'_{\mu})) \in \prec_{\gamma} \cap (\Gamma_{ms} \times \Gamma_{ms}) \} \right)$$
(3)

Note the slight difference in the definition of an alignment as opposed to our previous work in [24], where the alignment is simplified from a distributed run to a poset of moves. The process's history of states (markings) as it has supposedly happened in reality can be extracted from the alignment. For the general case, we introduce the *pseudo-firing* of transitions from corresponding alignment's non-log moves in the process model, to obtain a pseudo-marking, which can be unreachable or contain a negative number of tokens:

Definition 14. (*Pseudo-markings*) A pseudo-marking m of a Petri net $N = (P, T, \mathcal{F})$ is a multiset $P \to \mathbb{Z}$, i.e., the assigned number of tokens a place contains can be negative. $\widetilde{\mathcal{M}}(N)$ denotes the set of all pseudo-markings in N.

Definition 15. (*Pseudo-firing of posets*) Let $N = (P, T, \mathcal{F}, m_i, m_f)$ be a RC ν -net and γ be an alignment on N. We define a function $\widetilde{m} : \mathcal{P}(\gamma) \to \widetilde{\mathcal{M}}(N)$, with powerset \mathcal{P} , to obtain the model pseudo-marking of every subposet of γ . For every subposet $\gamma' \subseteq \gamma$, we have for every $p \in P$:

$$\widetilde{m}(\gamma')(p) = m_i(p) + \sum_{(e,t_\mu)\in\gamma': t_\mu\neq\epsilon} \left(\mu\left(\mathcal{F}(t,p)\right) - \mu\left(\mathcal{F}(p,t)\right)\right)$$
(4)

i.e., the pseudo-marking is obtained by firing all the transitions of γ' with corresponding modes. Note that it is not necessarily reachable.

An antichain in an alignment denotes a possible point in time, and therefore a state of the process. By pseudo-firing the respective (open) prefix of the antichain, we obtain the corresponding *pre-* (*or post-*)*antichain marking*:

Definition 16. (*Pre- and post-antichain marking*) Let γ be an alignment and $G \in \mathcal{A}(\gamma)$ an antichain in γ . The pre- (post-)antichain marking defines the marking reached after the pseudo-firing of (\bot, G) ($(\bot, G]$), i.e., $\widetilde{m}((\bot, G))$ ($\widetilde{m}((\bot, G])$).

4.2 Alignments extended to include inter-case dependencies

The foundational work on constructing alignments is presented in [5] and it relies on the synchronous product of the Petri net $N = (P, T, \mathcal{F}, \ell, m_i, m_f)$ modeling a process and a trace Petri net $N_{\sigma} = (P^{(\sigma)}, T^{(\sigma)}, \mathcal{F}^{(\sigma)}, \ell^{(\sigma)}, m_i^{(\sigma)}, m_f^{(\sigma)})$ (a Petri net representation of a trace in the event log). The synchronous product consists of the union of N and N_{σ} , and a transition t_s for each pair of transitions $(t_m, t_l) \in T \times T^{(\sigma)}$ with $\bullet t_s = \bullet t_m + \bullet t_l$ and $t_s = t_m^{\bullet} + t_l^{\bullet}$, iff t_m and t_l share the same label and variables on the incoming arcs, i.e., $\ell(t_m) = \ell^{(\sigma)}(t_l)$ and $Var(t_m) = Var(t_l)$. The alignment is then computed by a depth-first search on the synchronous product net from $m_i + m_i^{(\sigma)}$ to $m_f + m_f^{(\sigma)}$ using the A^* algorithm, with the firings of transition from $T^{(\sigma)}$, T and $T^{(s)}$ corresponding to the log, model and synchronous moves from Def. 12 [5].

With $c: \Gamma_{lms} \to \mathbb{R}^+$ a cost function, usually defined for each $(e, t_\mu) \in \Gamma_{lms}$ as follows:

$$c((e, t_{\mu})) = \begin{cases} 0 & (e, t_{\mu}) \in \Gamma_s \\ 1 & (e, t_{\mu}) \in \Gamma_{lm} \land \ell(t) \neq \tau \\ \epsilon & \ell(t) = \tau \end{cases}$$
(5)

The *optimal alignment* is an alignment γ such that $\sum_{g \in \gamma} c(g) \leq \sum_{g \in \gamma'} c(g)$ holds for any alignment γ' , which prefers synchronous moves over model and log moves. In terms of conformance checking and exposing realistic deviations, the optimal alignment provides the "best" explanation for the relation between observed and modeled behavior.

In Sec. 3.3, we have shown how a RC ν -net is a Petri net formalism with capability of modeling inter-case dependencies and suitability for conformance checking. We extend the alignment problem in order to expose inter-case deviations by adapting the synchronous product net to ν -nets: an RC ν -net and the log ν -net:

Definition 17. (Log ν -Petri net) Given an event log L, a log ν -Petri net $N^{(L)} = (P^{(L)}, T^{(L)}, \mathcal{F}^{(L)}, \ell^{(L)}, m_i^{(L)}, m_f^{(L)})$ is a labeled ν -net constructed as follows. For every $e \in L$, we make a transition $t_e \in T^{(L)}$ with $\ell^{(L)}(t) = \operatorname{activity}(e)$, and for each resource instance $\rho_r \in \operatorname{supp}(\operatorname{Res}(e))$ we make a place $p \in P^{(L)}$ with $\bullet p = \emptyset$, $\bullet p = [|\rho_r| \cdot t]$, $\mathcal{F}^{(L)}(p, t) = [|\rho_r| \cdot (\varepsilon, r)]$ and $m_i^{(L)}(p)((\varepsilon, \rho)) = |\rho|$. Further, for every pair $(e_1, e_2) \in \prec_L$, we make a place $p \in P^{(L)}$ with $\bullet p = [t_{e_1}]$, $p^{\bullet} = [t_{e_2}]$ and

$$\mathcal{F}^{(L)}(t_{e_1}, p) = \mathcal{F}^{(L)}(p, t_{e_2}) = \begin{cases} [(c, \varepsilon)] & \operatorname{case}(e_1) = \operatorname{case}(e_2) \\ [(\varepsilon, \varepsilon)] & otherwise \end{cases}$$
(6)

For every $e^- \in \min(L)$, we make a place $p^- \in P^{(L)}$ with ${}^{\bullet}p^- = \emptyset$, $p^{-\bullet} = [t_{e^-}]$ and $m_i^{(L)}(p^-)((case(e^-),\varepsilon)) = 1$. Similarly, for every $e^+ \in \max(L)$, we make a place $p^+ \in P^{(L)}$ with ${}^{\bullet}p^+ = [t_{e^+}]$, $p^{+\bullet} = \emptyset$ and $m_f^{(L)}(p^+)((case(e^+),\varepsilon)) = 1$.

Computing the *complete event log alignment* is again a matter of finding a path from the initial to the final marking in the synchronous product net, i.e., from $m_i + m_i^{(L)}$ to $m_f + m_f^{(L)}$, for which we can use *any* of the existing methods as described

before. The optimal alignment is again the one with lowest cost. In terms of complexity, the alignment problem with an empty event log and an all-zero cost function can be reduced to the reachability problem for bounded Petri nets from m_i to m_f , which has exponential worst-case complexity[20]. Adding event to the log ν -Petri net and a non-zero costs on moves makes the problem strictly more complex.

Note that while ν -Petri nets are inherently unbounded in general due to the generation of fresh tokens, we can retain boundedness in the context of alignments, since the bound is predicated by the event log and we can get this information by preprocessing it.

For our running example, modeled in Fig. 2, we extend the small operation process from Fig. 1 with an assistant resource during the operation, an intake subprocess (i_s, i_p) involving a general practitioner (GP), and a prescription subprocess with a FIFO waiting room (p_w, w_e, w_l, p_r) , where the prescription can only be written by the GP involved in the intake, if appropriate. Both the intake and operation subprocesses can be skipped via silent transitions τ_1 and τ_3 respectively in N. Fig. 3 shows the recorded event log L of this process which concerns two patients. An optimal complete event log alignment on N and L, computed by the method above is presented in Fig. 4.

5 Approximation by composition and local realignments

Since multiple cases are executed in parallel, computing the alignment on the complete event $\log L$, as described in Section 4, is a computationally expensive task. At the same time, one can see that the multi-case and -resource alignment only deviates from the classical individual alignments when violations occur on the inter-case dependencies, e.g., when a resource is claimed while it is already at maximal capacity.

We can approximate the alignment of a complete event $\log L$ and a Petri net N by using a composition of individually aligned cases. An overview of this method is illustrated in Fig. 7, which we subdivide into two parts, described respectively in Sec. 5.1 and 5.2.

- 1. *L* is decomposed into the individual cases (L_c, L_c) , which are aligned to $N(\gamma_c, \gamma_c)$ and composed using the event log's partial order $\prec_L (\tilde{\gamma})$. The result is not necessarily an alignment as inter-case deviations may be left unresolved;
- 2. We transform this composed alignment into a valid alignment by taking a permutation ($\tilde{\gamma}'$) and realigning parts ($[A_1, B_1], [A_2, B_2], [A_3, B_3]$) of the event log locally to resolve the violations. The approximated alignment (γ^*) is obtained by substituting the realignments ($\gamma_{AB_1}, \gamma_{AB_2}, \gamma_{AB_3}$).

The implementation of both the original method from [24] and the approximation method for computing complete event log alignments is available at gitlab.com/dominiquesommers/mira, including the examples used in this paper and some additional examples.

5.1 Composing individual alignments

For every case $c \in \text{Id}_c$, we have the trace L_c (cf. Def. 8) projected on the case identifier c. As described in Sec.4, the optimal complete event log alignment γ_L consists of individual alignments γ_c , on N and L_c for every $c \in \text{Id}_c$, composed together respecting



Fig. 2: Process model RC ν -net N, with initial and final marking, annotated with circular and square tokens respectively.



Fig. 4: Complete event log alignment γ , with the colors depicting the move types; green, purple, and yellow for synchronous, model, and log moves respectively.



Fig. 5: Composed alignment $\tilde{\gamma}$ with annotated permutation and realignment intervals.



Fig. 6: Approximated alignment γ^* .



Fig. 7: Overview of our approximation method.

the event log's partial order \prec_L , where each γ_c is not necessarily optimal with regard to L_c .

It is computationally less expensive to compute the optimal alignments $\gamma_c = \text{align}(N, L_c)$ for each $c \in \text{Id}_c$ and then approximate γ_L . We create a composed alignment $\tilde{\gamma}$ with the optimal individual alignments and the event log's partial order, as defined in Def. 18. Fig.5 shows the composed alignment for the running example with additional annotations (in red) which we cover later.

Definition 18. (Composed alignment) Given a Petri net N and an event log L with traces L_c for $c \in Id_c$, let $\gamma_c = align(N, L_c)$ be the corresponding optimal individual alignments. The composed alignment $\tilde{\gamma} = \bigcup_{c \in Id_c} \gamma_c$ is the union of individual alignments with the extended partial order on the synchronous moves, defined as the transitive closure of the union of partial orders from the individual alignments and the partial order on moves imposed by the partial order \prec_L of the event log:

$$\prec_{\widetilde{\gamma}} = \left(\bigcup_{c \in \mathrm{Id}_c} \prec_{\gamma_c} \cup \prec_{\gamma_L}\right)^+ \tag{7}$$

with $\prec_{\gamma_L} = \left\{ \left((e, t_\mu), (e', t'_\mu) \right) \mid e \prec_L e', (e, t_\mu), (e', t'_\mu) \in (\gamma \cap \Gamma_{ls}) \right\}.$

Recall that for every sequence $\sigma \in \tilde{\gamma} \upharpoonright_T$ of an alignment $\tilde{\gamma}$, we have $m_i \xrightarrow{\sigma} m_f$, i.e., σ is a firing sequence in N. This property is not guaranteed for a composed alignment, even in the absence of inter-case deviations. In the presence thereof, we say that a composed alignment is *violating* as there exists no such sequence.

Definition 19. (Violating composed alignment) Let $\rho_r \in supp(\mathrm{Id}_R)$ be a resource instance and $\tilde{\gamma} = \bigcup_{c \in \mathrm{Id}_c} \gamma_c$ a composed alignment. We define

$$\mathcal{S}(\tilde{\gamma}) = \{ (\bar{\tilde{\gamma}}', \prec_{\tilde{\gamma}'}) \mid \bar{\tilde{\gamma}} = \bar{\tilde{\gamma}}', \prec_{\tilde{\gamma}} \subseteq \prec_{\tilde{\gamma}'}, \prec_{\tilde{\gamma}'} = (\prec_{\tilde{\gamma}'})^+, \forall_{g \in \tilde{\gamma}} g \not\prec_{\tilde{\gamma}'} g \}$$
(8)

as the set of transitively closed and acyclic antichain permutations of $\tilde{\gamma}$ that respect the partial order $\prec_{\tilde{\gamma}}$.

 $\tilde{\gamma}$ is in violation with any of the resource instances if and only if:

$$\forall_{\widetilde{\gamma}'\in\mathcal{S}(\widetilde{\gamma})}\exists_{G\in\mathcal{A}^+(\widetilde{\gamma}')}\operatorname{viol}(G)\tag{9}$$

with violation criteria viol : $\mathcal{A}^+(\widetilde{\gamma}) \to \mathbb{B}$ defined for each maximal antichain $G \in \mathcal{A}^+(\widetilde{\gamma})$ as follows:

$$\operatorname{viol}(G) = \exists_{\rho_r \in supp(\operatorname{Id}_R)} \left[\widetilde{m}((\bot, G))(p_r)((\varepsilon, \rho_r)) < \sum_{(e, t_\mu) \in G} \mathcal{F}(p_r, t)(\mu^{-1}((\varepsilon, \rho_r))) \right]$$
(10)

i.e., there is no way of firing all transitions in the alignment such that at all times enough capacity is available.

In Fig. 5, antichains meeting the violation criteria are the single moves with an incoming red arc. In Theorem 1 we show that for every sequence of transitions $\sigma \in \widetilde{\gamma} \upharpoonright_T$ in violating composed alignment $\widetilde{\gamma}$, we have $m_i \xrightarrow{\sigma} m_f$, i.e., $\widetilde{\gamma}$ is not firable.

Theorem 1. (A violating composed alignment is not firable) Let $\tilde{\gamma} = \bigcup_{c \in \mathrm{Id}_c} \gamma_c$ be a composed alignment on RC ν -net $N = (P, T, \mathcal{F}, m_i, m_f)$ and event log L, such that $\tilde{\gamma}$ is violating. Then there exists no firing sequence σ in $\tilde{\gamma}$ such that $m_i \xrightarrow{\sigma} m_f$.

Proof. $\widetilde{\gamma}$ is violating, therefore, for every $\widetilde{\gamma}' \in S(\widetilde{\gamma})$, there is a maximal antichain $G \in \mathcal{A}^+(\widetilde{\gamma}')$ and resource instance $\rho_r \in supp(\mathrm{Id}_R)$, such that

$$\widetilde{m}((\bot,G))(p_r)((\varepsilon,\rho_r)) < \sum_{(e,t_\mu)\in G} \mathcal{F}(p_r,t)(\mu^{-1}((\varepsilon,\rho_r)))$$
(11)

$$\widetilde{m}((\bot,G))(p_r)((\varepsilon,\rho_r)) - \sum_{(e,t_\mu)\in G} \mathcal{F}(p_r,t)(\mu^{-1}((\varepsilon,\rho_r))) < 0$$
(12)

hence firing the transitions in G leads to a negative marking for (ε, ρ_r) in place p_r , which is invalid.

With an antichain $G \subseteq \mathcal{A}(\tilde{\gamma})$, we show in Lemma 1 that $\tilde{m}((\bot, G))$ (and $\tilde{m}(\bot, G]$) is reachable if an only if the prefix (\bot, G) $((\bot, G])$ is not violating.

Lemma 1. (A pre- (and post-)antichain marking in a composed alignment is reachable iff the corresponding prefix is not violating.) Let $\tilde{\gamma} = \bigcup_{c \in \mathrm{Id}_c} \gamma_c$ be a composed alignment on RC ν -net $N = (P, T, \mathcal{F}, m_i, m_f)$ and event log L and let $G \in \mathcal{A}(\tilde{\gamma})$ be an antichain. Then the pre- (and post-)antichain marking $\tilde{m}((\bot, G))$ ($\tilde{m}((\bot, G])$) is reachable if and only if (\bot, G) ($(\bot, G]$) is not violating.

Proof. We prove the lemma by proving both sides of the bi-implication:

 $(\implies) m_G = \widetilde{m}((\bot, G))$ is reachable, hence there exists a sequence $\sigma \in (\bot, G)^*$ with $\prec_{(\bot,G)} \subseteq \prec_{\sigma}$ such that $m_i \xrightarrow{\sigma} m_G$. Let $\widetilde{\gamma}' \in \mathcal{S}(\widetilde{\gamma})$ be an antichain permutation with $\prec_{\sigma} \subseteq \prec_{\widetilde{\gamma}'}$. Then by definition of reachable marking, for every maximal antichain $G \in \mathcal{A}^+(\widetilde{\gamma}')$ and every resource instance $\rho_r \in supp(\mathrm{Id}_R)$, we have $\widetilde{m}((\bot,G))(p_r)((\varepsilon,\rho_r)) \ge \sum_{(e,t_u)\in G} \mathcal{F}(p_r,t)(\mu^{-1}((\varepsilon,\rho_r)))$. Thus (\bot,G) is not violating.

 (\Leftarrow) $(\perp,G]$ is not violating, hence there exists a $\widetilde{\gamma}' \in \mathcal{S}((\perp,G])$, such that for all $G' \in \mathcal{A}^+(\widetilde{\gamma}')$ and all $\rho \in supp(\mathrm{Id}_R)$ we have:

$$\widetilde{m}((\bot,G))(p_r)((\varepsilon,\rho_r)) \ge \sum_{(e,t_\mu)\in G} \mathcal{F}(p_r,t)(\mu^{-1}((\varepsilon,\rho_r)))$$
(13)

 $m_i \xrightarrow{\sigma} \widetilde{m}((\perp, G])$ with σ respecting the partial order $\prec_{\widetilde{\gamma}'}$. \Box

5.2 Resolving violations in the composed alignment

Let $\tilde{\gamma}' \in S(\gamma)$ be an antichain permutation of $\tilde{\gamma}$. Then, by Def. 19, we have a set of violating maximal antichains (which is empty when $\tilde{\gamma}$ is not violating) where the corresponding transitions are not enabled. Instead of needing to align the complete event log, we show that we can resolve violations locally around such antichain. For each violating antichain G, there exists an interval $[A, B] \subseteq \tilde{\gamma}'$ with $A \preceq G \preceq B$ such that [A, B] is alignable, formally defined in Def. 20.

Definition 20. (Alignable interval) Let $\gamma = \bigcup_{c \in \mathrm{Id}_c} \gamma_c$ be a composed alignment on RC ν -net $N = (P, T, \mathcal{F}, m_i, m_f)$ and event log L, and let $A, B \in \mathcal{A}(\gamma)$ be two antichains. We say that the interval [A, B] is alignable if and only if $m_B = \widetilde{m}((\bot, B])$ is reachable from $m_A = \widetilde{m}((\bot, A))$, i.e., $m_A \xrightarrow{*} m_B$, assuming m_A is reachable.

Note that $[\min(\gamma'), \max(\gamma')]$ is always an alignable interval. We use our running example to show that it can be taken locally around G instead, e.g., $[\{\underline{i}_s\}, \{i_p\}]$ with $G = \{\underline{i}_s\}$ (cf. Fig. 5). Note how the violation can be resolved by substituting [A, B] by a subalignment from $m_A = \widetilde{m}((\bot, A))$ to $m_B = \widetilde{m}((\bot, B])$.

In order to prove statements that do not depend on a chosen realignment mechanism, we now assume that there exists a function $f_{\widetilde{\gamma}} : \mathcal{A}^+(\widetilde{\gamma}) \to \mathcal{P}(\widetilde{\gamma})$ that produces an alignable interval [A, B] for an arbitrary $G \in \mathcal{A}^+(\widetilde{\gamma})$.

$$W(\widetilde{\gamma}'_{V}) = \{ [\min(\gamma_{v}), \max(\gamma_{v})] \mid \gamma_{v} \subseteq \widetilde{\gamma}'_{V}, \forall_{g \in \gamma_{v}, g' \in \widetilde{\gamma}'_{V} \setminus \gamma_{v}} g \|_{\widetilde{\gamma}'_{V}} g', \qquad (14)$$

$$\forall_{g \in \gamma_{v}} \exists_{g' \in \gamma_{v}} g \not\|_{\gamma_{v}} g' \}$$

with $\widetilde{\gamma}'_V = \bigcup_{G \in \mathcal{A}^+(\widetilde{\gamma}')} f_{\widetilde{\gamma}'}(G)$, denotes the set of alignable intervals covering every violating antichain in $\widetilde{\gamma}'$, and it is annotated in red for the running example in Fig. 5, with the three intervals $[\{\underline{i_s}\}, \{i_p\}], [\{\underline{o_p}\}, \{o_f\}]$, and $[\{\underline{w_e}\}, \{\tau\}]$ covering the violating antichains $\{i_s\}, \{o_p\}, \{o_{so}\}, \text{ and } \{w_e\}$.

We resolve the violations in $\tilde{\gamma}'$ by substituting every interval $[A, B] \in W(\tilde{\gamma}'_V)$ by an alignment γ_{AB} on N and $[A, B] \upharpoonright_L$ from $m_A = \tilde{m}((\bot, A))$ to $m_B = \tilde{m}((\bot, B])$.

Since, for now, we assume that every interval f(G) is alignable, a subalignment γ_{AB} exists. The approximated alignment $\gamma^* = (\bar{\gamma}^*, \prec_{\gamma^*})$ is then defined as follows:

$$\bar{\gamma}^* = \bigcup_{[A,B]\in W(\tilde{\gamma}'_V)} \bar{\gamma}_{AB} \cup (\bar{\tilde{\gamma}} \setminus \bar{\tilde{\gamma}}'_V) \tag{15}$$

$$\prec_{\gamma^*} = \left(\bigcup_{[A,B]\in W(\widetilde{\gamma}'_V)} \prec_{\gamma_{AB}} \cup \{(g_1,g_2) \mid g_1,g_2 \in \gamma \setminus \widetilde{\gamma}'_V,g_1 \prec_{\widetilde{\gamma}'} g_2\}\right)^+$$
(16)

 γ^* for the running example is shown in Fig. 6 with substituted realignments for the intervals annotated in red from Fig. 5. Note that γ^* is an approximation of the optimal alignment γ from Fig. 4 as $c(\gamma^*) \ge c(\gamma)$, due to the local realignments. In Theorem 2 we show that γ^* is a valid alignment.

Theorem 2. $(\gamma^* \text{ is an alignment.})$ Let $\widetilde{\gamma} = \bigcup_{c \in \mathrm{Id}_c} \gamma_c$ be a composed alignment on RC ν -net $N = (P, T, \mathcal{F}, m_i, m_f)$ and event log L and let $\widetilde{\gamma}' \in S(\widetilde{\gamma})$ be an antichain permutation of $\widetilde{\gamma}$, with $W(\widetilde{\gamma}'_V)$ the set of alignable intervals covering every violating antichain in $\widetilde{\gamma}'$.

 $\gamma^* = (\bar{\gamma}^*, \prec_{\gamma^*})$, following Eqs. 15 and 16, is a valid alignment, i.e., it has properties (1), (2) and (3) from Def. 13.

Proof. We prove that γ^* is an alignment by induction on the size of $W(\tilde{\gamma}'_V)$. For the base case with $|W(\tilde{\gamma}'_V)| = 0$, we have $\bar{\gamma}^* = \bar{\tilde{\gamma}}$ and $\prec_{\gamma^*} = \prec_{\tilde{\gamma}'}$. By definition, $\tilde{\gamma}_L^{-} = \bar{L}$ and $\prec_L \subseteq \prec_{\tilde{\gamma}\uparrow_L}$. Furthermore, since $|W(\tilde{\gamma}'_V)| = 0$, we know that for all $G \in \mathcal{A}^+(\tilde{\gamma}')$,

we have $\neg \operatorname{viol}(G)$, implying that $m_i \xrightarrow{\widetilde{\gamma}'} m_f$.

Let us assume that γ^* is an alignment for $|W(\widetilde{\gamma}'_V)| = w$. We prove the statement for $W'(\widetilde{\gamma}'_V) = W(\widetilde{\gamma}'_V) \cup \{[A, B]\}$ with $|W'(\widetilde{\gamma}'_V)| = w + 1$ and $[A, B] \in \min(W'(\widetilde{\gamma}'_V))$. For every maximal antichain $G \in \mathcal{A}^+((\bot, A))$ before A, i.e., $G \prec A$, we have $\neg \operatorname{viol}(G)$, which we prove by contradiction. Assume $\operatorname{viol}(G)$, then by our assumption of the existence of $f_{\widetilde{\gamma}'}$, there is an alignable interval $[A', B'] \subseteq \widetilde{\gamma}'$ with $A' \preceq G \preceq B'$, thus, by $G \prec A$, we have $[A', B'] \prec [A, B]$, implying that $[A, B] \notin \min(W'(\widetilde{\gamma}'))$ which is a contradiction. By Lemma 1 and the assumption that $f_{\widetilde{\gamma}'}(G)$ is an alignable interval, $m_i \xrightarrow{*} m_A \xrightarrow{*} m_B$ and [A, B] can be substituted by γ_{AB} without violations in $(\bot, B]$, completing the proof.

5.3 Obtaining minimal local alignable intervals

We propose a method to find an antichain permutation of a composed alignment $\tilde{\gamma}$ together with the intervals $W(\tilde{\gamma}_V)$ such that all violations can be resolved by realigning these intervals as described in Sec. 5.2. For computational efficiency, we choose to minimize the number of moves in the intervals that need to be realigned.

We formulate this as an Integer Linear Programming (ILP) problem. The objective of the ILP problem is to adjust the partial order of $\tilde{\gamma}$, such that alignable intervals can be identified around violating antichains, preferring intervals with fewer moves.

Let there be a (possibly arbitrary) fixed order in $\tilde{\gamma}$ and Id_R such that each element has a unique index, i.e., for every $1 \leq i \leq n_{\tilde{\gamma}}, \tilde{\gamma}(i)$ and $(e(i), t_{\mu}(i))$ both denote the i^{th} move in $\tilde{\gamma}$, with $n_{\tilde{\gamma}} = |\tilde{\gamma}|$. Furthermore, for every $1 \leq j \leq n_r$, $\mathrm{Id}_R(j)$ denotes the j^{th} resource instance, with $n_r = |supp(\mathrm{Id}_R)|$.

Let **R** be a $n_{\tilde{\gamma}} \times n_{\tilde{\gamma}}$ matrix, with **R** defined for every two indices $1 \le i, j \le n_{\tilde{\gamma}}$ such that \mathbf{R}_{ij} is a binary value denoting $(\tilde{\gamma}(i), \tilde{\gamma}(j)) \in \prec_{\tilde{\gamma}}$. For each $c \in \mathrm{Id}_c$, we introduce the set I_c of indices corresponding to moves in $\tilde{\gamma} \upharpoonright_{\gamma_c}$. Furthermore, we use $[1..n] = \{1, \ldots, n\}$ as an abbreviation for the set of all indices from 1 to n.

The set of minimal alignable intervals containing all violations, denoted by $W(\tilde{\gamma}'_V)$, with $\tilde{\gamma}'_V$ given by

$$\widetilde{\gamma}'_{V} = \bigcup_{i,j \in [1..n_{\widetilde{\gamma}}]: \mathbf{X}_{ij} - \mathbf{R}_{ij} = 1} [\widetilde{\gamma}(j), \widetilde{\gamma}(i)]$$
(17)

where \mathbf{X} denotes the new partial order relation between alignment moves which respects the resources capacities and provides the solution to

Minimize
$$\sum_{i,j\in[1..n_{\tilde{\gamma}}]} (1-\mathbf{R}_{ij}) \mathbf{R}_{ji} \mathbf{X}_{ij} + \epsilon \cdot (1-\mathbf{R}_{ij}) (1-\mathbf{R}_{ji}) \mathbf{X}_{ij}$$
(18)

subject to

A

$$\forall_{i,j\in[1..n_{\widetilde{\gamma}}]} \qquad \qquad \mathbf{X}_{ij}\in\{0,1\} \tag{19}$$

$$\forall_{c \in \mathrm{Id}_c} \forall_{i,j \in I_c} \qquad \qquad \mathbf{X}_{ij} = \mathbf{R}_{ij} \tag{20}$$

$$\forall_{i,j\in[1..n_{\widetilde{\gamma}}]} \qquad \mathbf{R}_{ij} + (1 - \mathbf{X}_{ij}) - \mathbf{X}_{ji} \le 1$$
(21)

$$\mathbf{X}_{ij,k\in[1..n_{\tilde{\gamma}}]} \qquad \qquad \mathbf{X}_{ij} + \mathbf{X}_{jk} - \mathbf{X}_{ik} \le 1$$
(22)

$$\forall_{i \in [1..n_{\tilde{\gamma}}]} \qquad (1 - \mathbf{X}_{i \bullet}) \mathbf{C}^{\downarrow} - \mathbf{X}_{\bullet i}^{T} \mathbf{C}^{\uparrow} \le \mathbf{k} \qquad (23)$$

with \mathbf{C}^{\downarrow} and \mathbf{C}^{\uparrow} both $n_{\tilde{\gamma}} \times n_r$ matrices counting how many resource instances are claimed and released respectively for every alignment move. Both are defined for every $i \in [1..n_{\tilde{\gamma}}]$ and $k \in [1..n_r]$ with $(e, t_{\mu}) = \tilde{\gamma}(i)$ and $\rho_r = \mathrm{Id}_R(k)$:

$$\mathbf{C}_{ik}^{\downarrow} = \mathcal{F}(p_r, t)((\varepsilon, \mu^{-1}(\rho_r))) \text{ and } \mathbf{C}_{ik}^{\uparrow} = \mathcal{F}(t, p_r)((\varepsilon, \mu^{-1}(\rho_r)))$$
(24)

and capacity vector \mathbf{k} of length n_r , defined as $\mathbf{k}_k = |\mathrm{Id}_R(k)|$ for every $k \in [1..n_r]$.

X provides the solution of a new partial order of moves in $\tilde{\gamma}$ such that all violations are resolved and the least number of partial order relations is removed. For the running example, the additional arcs from the solution X are shown in red in Fig. 5.

We refer to App. A for the correctness proof of the ILP problem, where we show (1) the effectiveness of each constraint, (2) that there always exists a solution, (3) that the optimal solution has zero cost if and only if the composed alignment is not violating, and (4) that each interval obtained in $W(\tilde{\gamma}'_V)$ is alignable.

6 Conclusion

We have formulated the requirements for modeling and analyzing processes with intercase dependencies and argued that our previously proposed Petri net extension named Resource Constrained ν -Petri nets meets them. This paper continues on work presented in [24], where we showed that the traditional methods of aligning observed behavior with the modeled one fall short when dealing with coevolving cases, as they consider isolated cases only. The technique we present here aligns multiple cases simultaneously, exposing violations on inter-case dependencies. We developed and implemented an approximation technique based on a composition of individual alignments and local resolution of violations, which is an important advancement for the use of the technique in practice.

There can be ambiguity in the interpretation of the exposed violations, e.g., was the activity executed but not recorded, executed by an "incorrect" resource instance, or not executed at all? In [24], we briefly touched upon relaxations of the synchronous product model as a means to improve the deviations' interpretability. One such relaxation

19

helps to detect situations when a step required by the model was skipped in a process execution, and the resources needed for the step were not available at the time when it should have been executed. Adding "resource-free" model moves for transitions allows to capture such deviations. Such special moves, when present in the alignment, reduce the ambiguity and provide a better explanation, e.g., that the activity was not executed at all, rather than it might also have been executed but not recorded. For future work, we plan to extend and formalize the relaxations, and evaluate the insights obtained with the alignments based on a real-life case study.

Acknowledgments. This work is done within the project "Certification of production process quality through Artificial Intelligence (CERTIF-AI)", funded by NWO (project number: 17998).

References

- 1. Wil M.P. van der Aalst. The application of Petri nets to workflow management. *Journal of circuits, systems, and computers*, 8(01):21–66, 1998.
- Wil M.P. van der Aalst. Data science in action. In *Process mining*, pages 3–23. Springer, 2016.
- Wil M.P. van der Aalst, Arya Adriansyah, and Boudewijn F. van Dongen. Replaying history on process models for conformance checking and performance analysis. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(2):182–192, 2012.
- Wil M.P. van der Aalst and Alessandro Berti. Discovering object-centric Petri nets. Fundamenta informaticae, 175(1-4):1–40, 2020.
- Arya Adriansyah. Aligning observed and modeled behavior. PhD thesis, Mathematics and Computer Science, 2014.
- Mahdi Alizadeh, Xixi Lu, Dirk Fahland, Nicola Zannone, and Wil M.P. van der Aalst. Linking data and process perspectives for conformance analysis. *Computers & Security*, 73:172– 193, 2018.
- Kamel Barkaoui and Laure Petrucci. Structural analysis of workflow nets with shared resources. 1998.
- Josep Carmona, Boudewijn F. van Dongen, Andreas Solti, and Matthias Weidlich. Conformance checking. Springer, 2018.
- Gero Decker and Mathias Weske. Instance isolation analysis for service-oriented architectures. In 2008 IEEE International Conference on Services Computing, volume 1, pages 249–256. IEEE, 2008.
- Dirk Fahland. Describing behavior of processes with many-to-many interactions. In *International Conference on Applications and Theory of Petri Nets and Concurrency*, pages 3–24. Springer, 2019.
- 11. Silvio Ghilardi, Alessandro Gianola, Marco Montali, and Andrey Rivkin. Petri nets with parameterised data: modelling and verification (extended version). *arXiv preprint arXiv:2006.06630*, 2020.
- 12. Kees van Hee, Natalia Sidorova, and Marc Voorhoeve. Resource-constrained workflow nets. *Fundamenta Informaticae*, 71(2, 3):243–257, 2006.
- Olaf Kummer. Undecidability in object-oriented Petri nets. In *Petri Net Newsletter*. Citeseer, 2000.
- 14. K. Lautenbach. Liveness in Petri Nets. Bonn Interner Bericht ISF. Selbstverl. GMD, 1975.

Exact and Approximated Log Alignments for Processes with Inter-case Dependencies

- Massimiliano de Leoni and Wil M.P. van der Aalst. Aligning event logs and process models for multi-perspective conformance checking: An approach based on integer linear programming. In *Business Process Management*, pages 113–129. Springer, 2013.
- Felix Mannhardt, Massimiliano de Leoni, Hajo A Reijers, and Wil M.P. van der Aalst. Balanced multi-perspective checking of process conformance. *Computing*, 98(4):407–437, 2016.
- 17. Azadeh S Mozafari Mehr, Renata M de Carvalho, and Boudewijn F. van Dongen. Detecting privacy, data and control-flow deviations in business processes. In *International Conference on Advanced Information Systems Engineering*, pages 82–91. Springer, 2021.
- Marco Montali and Andrey Rivkin. Model checking Petri nets with names using data-centric dynamic systems. *Formal Aspects of Computing*, 28(4):615–641, 2016.
- Marco Montali and Andrey Rivkin. Db-nets: On the marriage of colored petri nets and relational databases. In *Transactions on Petri Nets and Other Models of Concurrency XII*, pages 91–118. Springer, 2017.
- Tadao Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.
- 21. James Lyle Peterson. Petri net theory and the modeling of systems. Prentice Hall PTR, 1981.
- 22. Fernando Rosa-Velardo and David de Frutos-Escrig. Decision problems for Petri nets with names. *arXiv preprint arXiv:1011.3964*, 2010.
- Fernando Rosa-Velardo, David de Frutos-Escrig, and Olga Marroquín-Alonso. On the expressiveness of mobile synchronizing Petri nets. *Electronic Notes in Theoretical Computer Science*, 180(1):77–94, 2007.
- Dominique Sommers, Natalia Sidorova, and Boudewijn F. van Dongen. Aligning event logs to resource-constrained ν-Petri nets. In *International Conference on Applications and The*ory of Petri Nets and Concurrency, pages 325–345. Springer, 2022.

A Correctness of the ILP problem

We first show the effectiveness of each constraint:

- Constraint 20 ensures that the original partial order of the individual alignments is preserved. Note that this also ensures g ⊀ g for every g ∈ γ;
- Const. 21 enforces that when a relation $g \prec g'$ is removed, the opposite $g' \prec g$ is added:

$$\mathbf{R}_{ij} = 1 \wedge \mathbf{X}_{ij} = 0 \implies \mathbf{X}_{ij} = 1 \tag{25}$$

- Const. 22 enforces that the transitive closure is covered:

$$\mathbf{X}_{ij} = \mathbf{X}_{jk} = 1 \implies \mathbf{X}_{ik} = 1 \tag{26}$$

With Const. 22 together with $g \not\prec g$ for every $g \in \tilde{\gamma}$ from Const. 20, there can be no loops in the solution of the ILP problem;

 Const. 23 enforces that the solution is not violating with regard to any resource instance capacities, which we show in Lemma 2.

Lemma 2. (Constraint 23 ensures there are no violations) With a composed alignment $\tilde{\gamma} = \bigcup_{c \in \mathrm{Id}_c} \gamma_c$ and \mathbf{X} a solution to the ILP problem formulated above. The permutation $\tilde{\gamma}' = (\tilde{\gamma}', \prec_{\tilde{\gamma}'})$ of $\tilde{\gamma}$ following the partial order of \mathbf{X} , i.e., $\tilde{\gamma}' = \tilde{\gamma}$ and $\prec_{\tilde{\gamma}'} = \{(\tilde{\gamma}'(i), \tilde{\gamma}'(j)) \mid i, j \in [1..n_{\tilde{\gamma}}], \mathbf{X}_{ij} = 1\}$, is not violating (cf. Def. 19).

Proof. First, let us rewrite Const. 23 to

$$\forall_{i \in [1..n_{\tilde{\gamma}}]} \forall_{k \in [1..n_r]} (1 - \mathbf{X}_{i\bullet}) \mathbf{C}_{\bullet k}^{\downarrow} - \mathbf{X}_{i\bullet}^T \mathbf{C}_{\bullet k}^{\uparrow} \le \mathbf{k}_k = |\mathrm{Id}_R(k)|$$
(27)

For ever move index $i \in [1..n_{\tilde{\gamma}}]$, we can define $G_i = \{\tilde{\gamma}(j) \mid j \in [1..n_{\tilde{\gamma}}], \mathbf{X}_{ij} = \mathbf{X}_{ji} = 0\}$ to be the maximal antichain in \mathbf{X} that contains $\tilde{\gamma}(i)$. $(1 - \mathbf{X}_{ij})$ and \mathbf{X}_{ij}^T) relate to G_i as follows:

$$(1 - \mathbf{X}_{ij}) = 1 \iff \widetilde{\gamma}(j) \in (\bot, G_i] \text{ and } \mathbf{X}_{ij}^T = 1 \iff \widetilde{\gamma}(j) \in (\bot, G_i)$$
 (28)

We can now rewrite the Eq. 27 to match the property in Eq. 9, with abbreviations $C_{\rho_r}^{\downarrow}((e,t_{\mu})) = \mathcal{F}(p_r,t)((\varepsilon,\mu^{-1}(\rho_r)))$ and $C_{\rho_r}^{\downarrow}((e,t_{\mu})) = \mathcal{F}(t,p_r)((\varepsilon,\mu^{-1}(\rho_r)))$. Note that $\mathbf{C}_{jk}^{\downarrow} = C_{\mathrm{Id}_R(k)}^{\downarrow}(\widetilde{\gamma}(j))$ and $\mathbf{C}_{jk}^{\uparrow} = C_{\mathrm{Id}_R(k)}^{\uparrow}(\widetilde{\gamma}(j))$ for every $j \in [1..n_{\widetilde{\gamma}}]$ and $k \in [1..n_r]$. For every $i \in [1..n_{\widetilde{\gamma}}]$ and $k \in [1..n_r]$ with $\rho = \mathrm{Id}_R(k)$, the following holds:

$$(1 - \mathbf{X}_{i\bullet})\mathbf{C}_{\bullet k}^{\downarrow} - \mathbf{X}_{i\bullet}^{T}\mathbf{C}_{\bullet k}^{\uparrow} \le |\rho|$$
(29)

$$\iff \sum_{j \in [1..n_{\gamma}]} \left((1 - \mathbf{X}_{ij}) \mathbf{C}_{jk}^{\downarrow} - \mathbf{X}_{ij}^{T} \mathbf{C}_{jk}^{\uparrow} \right) \le |\rho|$$
(30)

$$\iff \sum_{g \in (\perp,G_i]} C_{\rho}^{\downarrow}(g) - \sum_{g \in (\perp,G)} C_{\rho}^{\uparrow}(g) \le |\rho| \quad \text{(By Eq. 28)} \quad (31)$$

$$\iff \sum_{g \in (\perp,G_i)} \left(C_{\rho}^{\downarrow}(g) - C_{\rho}^{\uparrow}(g) \right) + \sum_{g \in G} C_{\rho}^{\downarrow}(g) \le |\rho|$$
(32)

$$\iff |\rho| - \sum_{g \in (\perp, G_i)} \left(C_{\rho}^{\downarrow}(g) - C_{\rho}^{\uparrow}(g) \right) + \sum_{g \in G} C_{\rho}^{\downarrow}(g) \ge 0$$
(33)

$$\iff \qquad \qquad \widetilde{m}((\bot,G))(p_r)((\varepsilon,\rho_r)) + \sum_{g \in G} C_{\rho}^{\downarrow}(g) \ge 0 \qquad \text{(By Def. 15)} \quad (34)$$

| _ |
|---|
| |
| |
| |

Next, we go over two important properties of the ILP problem. In Lemma 3, we show that a solution respecting the constraints always exists for any composed alignment $\tilde{\gamma}$, and in Lemma 4 we show that if and only if $\tilde{\gamma}$ is not violating as defined in Def. 19, the cost of the solution is 0.

Lemma 3. (*There always exists a solution to the ILP problem.*) With a composed alignment $\tilde{\gamma} = \bigcup_{c \in \mathrm{Id}_c} \gamma_c$ and the ILP problem formulated as above, there exists a solution for **X** such that all constraints hold.

Proof. We show by construction that there is always a solution \mathbf{X}' to the ILP problem, which respects all constraints. Let there be a (possibly arbitrary) order in Id_c, such that c_x denotes the x^{th} case identifiers for every $x \in [1..n_c]$, with $n_c = |\text{Id}_c|$.

$$\mathbf{X}' = \begin{bmatrix} \mathbf{R}_{c_1} & \mathbf{1} & \cdots & \mathbf{1} \\ \mathbf{0} & \mathbf{R}_{c_2} & \cdots & \mathbf{1} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{R}_{c_{n_c}} \end{bmatrix}$$
(35)

with $\mathbf{R}_c = \mathbf{R}_{I_c I_c}$ for each $c \in \mathrm{Id}_c$ the $|\gamma_c| \times |\gamma_c|$ submatrix containing only the elements of the case.

- Const. 20 X' trivially respects this constraint as it contains \mathbf{R}_c for each $c \in \mathrm{Id}_c$;
- Const. 21 Let $i \in I_{c_x}$ and $j \in I_{c_y}$ be two indices. If x = y, this trivially holds, since \mathbf{X}' contains \mathbf{R}_{c_x} . Otherwise, by construction of \mathbf{X}_{ij} , $\mathbf{R}_{ij} = 1 \land \mathbf{X}_{ij} = 0 \implies j < i \implies \mathbf{X}_{ii} = 1$;
- Const. 22 Let $i, j, k \in I_{c_x}, I_{c_y}, I_{c_z}$ respectively. When x = y = z, this trivially holds, since \mathbf{X}' contains \mathbf{R}_{c_x} which is transitively closed. Otherwise, we know by construction of \mathbf{X}' that with $x \neq y$, $\mathbf{X}'_{ij} = 1 \iff x < y$ and $\mathbf{X}'_{ij} = 0 \iff x > y$ which we use two prove the two cases:

23

- 24 D. Sommers et al.
 - (1) $\mathbf{X}'_{ij} = 1 \land x \neq y \implies x < y$. $\mathbf{X}'_{jk} = 1 \implies y \le z \implies x < z \implies$ $\mathbf{X}'_{ik} = 1;$ (2) $\mathbf{X}'_{jk} = 1 \land y \neq z \implies y < z$. $\mathbf{X}'_{ij} = 1 \implies x \le y \implies x < z \implies$ $\mathbf{X}'_{ik} = 1.$
 - Hence $\mathbf{\tilde{X}}'$ is transitively closed;

Const. 23 For every $x \in [1..n_c]$ and every $i \in I_{c_x}$, let $J_i = \{j \mid j \in [1..n_{\widetilde{\gamma}}], 1 - \mathbf{X}'_{ij} = 1\}$ and $J'_i = \{j \mid j \in [1..n_{\widetilde{\gamma}}], \mathbf{X}'_{ij}^T = 1\}$, note that $J' \subseteq J$. We know by construction of \mathbf{X}' that for every $y \in [1..n_c]$ and every $j \in I_{c_y}, 1 - \mathbf{X}'_{ij} = 1 \implies y \leq x$ and

 $\mathbf{X}_{ii}^{T} = 1 \implies y \leq x$. Therefore

$$\forall_{y \in [1..n_c] \setminus \{x\}} I_{c_y} \subseteq J' \lor I_{c_y} \cap J' = \emptyset$$
(36)

holds. Since γ_{c_y} is an alignment for every $y \in [1..n_c]$, we have for every resource index $k \in [1..n_r], \sum_{j \in I_{c_n}} (\mathbf{C}_{jk}^{\downarrow} - \mathbf{C}_{jk}^{\uparrow}) = 0$ which together imply that

$$\sum_{j\in J} \mathbf{C}_{jk}^{\downarrow} - \sum_{j\in J'} \mathbf{C}_{jk}^{\uparrow} = \sum_{y\in[1..n_c]} \left(\sum_{j\in J\cap I_{c_y}} \mathbf{C}_{jk}^{\downarrow} - \sum_{j\in J'\cap I_{c_y}} \mathbf{C}_{jk}^{\uparrow} \right)$$
$$= \sum_{j\in J\cap I_{c_x}} \mathbf{C}_{jk}^{\downarrow} - \sum_{j\in J'\cap I_{c_x}} \mathbf{C}_{jk}^{\uparrow} \le |\mathrm{Id}_R(k)|$$
(37)

since γ_{c_x} is a non-violating alignment.

Lemma 4. (Cost=0 if and only if the composed alignment is not violating.) With a composed alignment $\tilde{\gamma} = \bigcup_{c \in \mathrm{Id}_c} \gamma_c$, the solution to the ILP problem formulated above has zero cost if and only if $\tilde{\gamma}$ is not violating as defined in Def. 19, i.e., Eq. 9 does not hold for $\widetilde{\gamma}$.

Proof. We prove the two sides of the bi-implication:

- (\implies) Assuming the cost is zero, we have for every $i, j \in [1..n_{\tilde{\gamma}}], \mathbf{R}_{ij} = 1 \implies \mathbf{X}_{ij} = 1$ by the objective function, and therefore $\{(\widetilde{\gamma}(i),\widetilde{\gamma}(j)) \mid i,j \in [1..n_{\widetilde{\gamma}}], \mathbf{R}_{ij} =$ 1} = $\prec_{\mathbf{R}} \subseteq \prec_{\mathbf{X}} = \{ (\widetilde{\gamma}(i), \widetilde{\gamma}(j)) \mid i, j \in [1..n_{\widetilde{\gamma}}], \mathbf{X}_{ij} = 1 \}$. Since X respects Const. 23, **R** is not violating either;
- (\Leftarrow) Assuming $\tilde{\gamma}$ is not violating, there exists a $\tilde{\gamma}' \in \mathcal{S}(\tilde{\gamma})$, such that the resource capacities are respected. Any X such that $\prec_{\widetilde{\gamma}} \subseteq \prec_{\widetilde{\gamma}'} \subseteq \prec_X$ results in zero cost, since for every $i, j \in [1..n_{\tilde{\gamma}}]$, $\mathbf{R}_{ij} = 1 \implies \mathbf{X}_{ij} = 1$. Furthermore, the cost can not be negative as $(1 - \mathbf{R}_{ij})\mathbf{X}_{ij} \ge 1$ for every $i, j \in [1..n_{\widetilde{\gamma}}]$.

Lastly, to fulfill the assumption in Sec. 5 that the intervals around violating antichains are alignable, we show in Lemma 5 that this is the case for each interval in $W(\widetilde{\gamma}'_V).$

Lemma 5. (Each interval $[A, B] \in W(\widetilde{\gamma}'_V)$ is alignable) Let $\widetilde{\gamma} = \bigcup_{c \in \mathrm{Id}_c} \gamma_c$ be a composed alignment and $W(\widetilde{\gamma}'_V)$ with $\widetilde{\gamma}'_V = \bigcup_{i,j \in [1..n_{\widetilde{\gamma}}], \mathbf{X}_{ij} - \mathbf{R}_{ij} = 1}[\widetilde{\gamma}(j), \widetilde{\gamma}(i)]$ the set of intervals obtained from the ILP problem formulated above with solution X. For every $[A, B] \in W(\widetilde{\gamma}'_V)$, [A, B] is alignable, i.e., (by Def. 20) $m_A \xrightarrow{*} m_B$ with $m_A = \widetilde{m}((\bot, A))$ and $m_B = \widetilde{m}((\bot, B])$.

Proof. Let $[A, B] \in W(\tilde{\gamma}'_V)$ be any interval in $W(\tilde{\gamma}'_V)$. We prove that $m_A \xrightarrow{*} m_B$ by construction of a subalignment $\gamma_{AB} = (\bar{\gamma}_{AB}, \prec_{\gamma_{AB}})$ constructed as follows:

$$\bar{\gamma}_{AB} = \bigcup_{(e,t_{\mu})\in[A,B]\cap\Gamma_s} \{(\gg,t_{\mu}), (e,\gg)\}$$
(38)

$$\begin{aligned} \prec_{\gamma_{AB}} &= \left(\{((\gg, t_{\mu}(i)), (\gg, t_{\mu}(j))) \mid i, j \in [1..n_{\gamma}], \gamma(i), \gamma(j) \in [A, B] \setminus \Gamma_{l}, \mathbf{X}_{ij} = 1\} \\ & (39) \\ & \cup \{((e, t_{\mu}), (e', t'_{\mu})) \mid (e, t_{\mu}), (e', t'_{\mu})) \in \prec_{\gamma'_{L}}, (e, \gg), (e', \gg) \in \gamma_{AB}\})^{+} \end{aligned}$$

i.e., all synchronous moves are split into a corresponding model and log move and the partial order for model moves is defined by **X** and for the log moves by the event log's partial order \prec_L . There is a $\sigma \in N(\gamma_{AB})^*$ such that $m_A \xrightarrow{\sigma} m_B$ because of two properties of γ_{AB} :

- By Const. 23 and Lemma 2, $\forall_{G \in \mathcal{A}^+(\gamma_{AB}), \rho \in supp(\mathrm{Id}_R)} \neg \operatorname{viol}(G)$;
- For every $c \in \mathrm{Id}_c$, we have, by construction of γ_{AB} and the ILP's constraints, $\gamma_{AB} \upharpoonright_{T_c} = [A, B] \upharpoonright_{T_c}$, with $T_c = \{t_{\mu} \mid t_{\mu} \in T_{\mu}, \exists_{v_c \in Var_c, v_r \in Var_r}) \mu((v_c, v_r)) = (c, *)\}$. Therefore, $\gamma_{AB} \upharpoonright_c = \gamma_{AB} \cap \Gamma_c$ is an alignment, with $\Gamma_c = \{(e, t_{\mu}) \mid (e, t_{\mu}) \in \gamma_{AB}, \operatorname{case}(e) = c \lor t_{\mu} \in T_c\}$.