Glocal Conformance Checking

Alessandro Burigana[1], Alessandro Gianola[2(✉)], Marco Montali[1], and Sarah Winkler[1]

[1] Free University of Bozen-Bolzano, Bolzano, Italy
{burigana,montali,winkler}@inf.unibz.it
[2] INESC-ID/Instituto Superior Técnico, Universidade de Lisboa, Lisbon, Portugal
alessandro.gianola@tecnico.ulisboa.pt

Abstract. Conformance checking is a process mining task where observed process executions are compared with the conduct prescribed by a process model. Even in the case where multiple parties interact in the process, it is typically assumed that the process itself provides a monolithic, global description of the overall, expected behaviour. However, it may very well be the case that such a global process is not explicitly available, and that each agent comes with its own local view of the process, while the overall behaviour is only be implicitly obtained by composing such local views. In this paper, we provide for the first time a formal framework for *glocal conformance checking*, where a global observed trace of a multi-party process is related to local Data Petri nets, each representing the subjective view of each participating agent. We formulate conformance checking in this multi-agent setting as an alignment problem, and show how it can be tackled by "acting locally, and thinking globally", that is, pairing local alignments with a suitable global compatibility condition. We then observe that in this setting, cost functions must take the context of activities in the global trace into account, which is realised through a new schema of regular expression-based cost functions. We pair the foundational investigation of the problem with a proof-of-concept SMT-based implementation.

Keywords: Conformance checking · collaborative processes · local alignments · data Petri nets · SMT encodings

1 Introduction

Conformance checking is a central process mining task, in which observed process executions are compared with the conduct prescribed by a reference process model [3]. In this spectrum, alignment-based techniques have gained particular attention, given their ability to provide, in the case of a non-conforming observed

Gianola was partially supported by national funds through FCT, Fundação para a Ciência e a Tecnologia, under projects UIDB/50021/2020 (DOI:10.54499/UIDB/50021/2020). The other authors acknowledge the UNIBZ project ADAPTERS and the PRIN MIUR project PINPOINT Prot. 2020FNEB27. This research was partly supported by the project eProcess, no. 14375, supported by COMPETE2030-FEDER-00570400.

A. Marrella et al. (Eds.): BPM 2024, LNCS 14940, pp. 75–92, 2024.
https://doi.org/10.1007/978-3-031-70396-6_5

trace, fine-grained insights on where the trace deviates from the reference model. This assumes that the model provides a monolithic, global description of the expected behaviour. However, processes are typically jointly executed by multiple actors or parties (called *agents* in this paper), each having a *partial knowledge* of the overall process (see, e.g., [1,2,5,6]) or, in an interorganisational setting, concurrently enacting *interacting process components* (see, e.g., [4,9,11,12]).

In fact, it may well be that a global process is not explicitly available, and that each agent comes with its own *local process*, while the overall behaviour is only implicit in the composition of these local views. In such a multi-agent setting, when an execution does not evolve as expected, agents formulate hypotheses about what could have gone wrong from their *epistemic* point of view, before consolidating their local reconstructions, discarding those that are incompatible. For example, suppose a customer sends an order cancellation to a seller, and later unexpectedly receives the order from a carrier. The customer may hypothesize that the carrier misbehaved, only to discover later, through knowledge consolidation with the seller, that the seller forgot to withdraw the shipment.

In this paper, we provide for the first time a formal framework to tackle this form of epistemic conformance checking, which we call *glocal conformance checking* to reflect the need of "acting locally, and thinking globally", as explained next. From the modelling point of view, we consider the setting where local views are described as Data Petri Nets (DPNs [10,13]) operating over overlapping tasks, and complemented by the very general compatibility condition dictating that local process views must agree on the execution of shared tasks. Notably, this general condition reconstructs, in spirit, concrete compatibility conditions such as those used to define the semantics of hybrid process models consisting of procedural and declarative components that synchronise on common tasks [1].

Our first contribution is to formalise glocal conformance checking as an alignment problem, where the principle of "act locally, and think globally" is substantiated as follows: on the one hand, local process alignments are computed on projections of the global observed trace, and on the other, they are subject to global compatibility condition ensuring that they do not contradict each other on shared tasks. As a second contribution, we observe that in this multi-agent setting, cost functions for alignment deviations (due to model or log moves) must take the context of activities in the global trace into account, to reflect the fact that since agents have only partial knowledge of the overall process, they are often constrained to act "bona fide". To meet this requirement, we put forward a new schema of regular expression-based cost functions, where the cost of model and log moves in a given position i of the global trace is defined based on a regular expression evaluated over the prefix of that trace up to i. Our last contribution is to pair this foundational investigation of the problem with a proof-of-concept tool called GloCoMoT, based on Satisfiability Modulo Theories (SMT). GloCoMoT builds on the CoCoMoT SMT-based framework for data-aware alignments over (monolithic) DPNs [1], and lifts it to the setting of glocal conformance checking combined with costs based on regular expressions. This appears as a natural choice, given the fact that dealing with data requires

alternative techniques to the classical ones used for pure control-flow models, such as those based on state-space exploration [16]. In fact, in the presence of data such techniques would need to explore an infinite state space even when the underlying process control-flow is bounded.

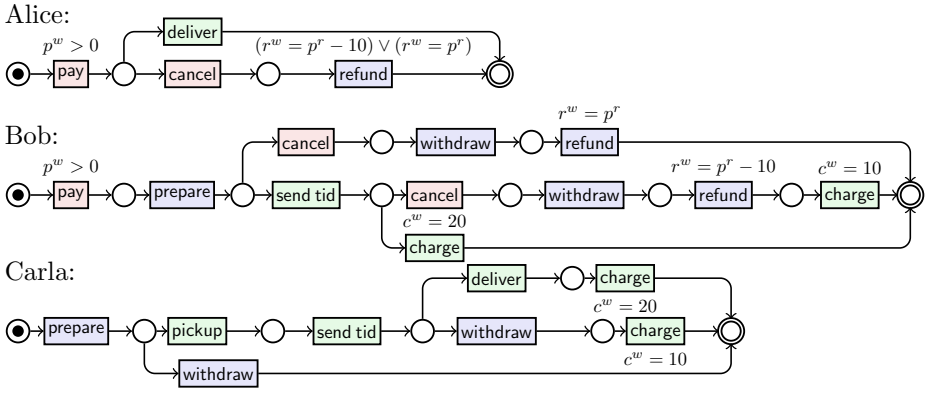
The paper is organised as follows. We start with a motivating example Sect. 2, and the necessary preliminaries in Sect. 3. The technical framework of glocal conformance is developed in Sect. 4, while context-dependent costs are tackled in Sect. 5. Finally, before concluding, we report our implementation in Sect. 6.

2 Motivating Example

We describe next a motivating example that illustrates the main characteristics and challenges to be tackled when dealing with *glocal conformance checking*.

Example 1. Consider three agents Alice, Bob, and Carla, respectively playing the roles of customer, seller, and carrier in an order-to-delivery process. Each agent comes with a local process specification enacted by the role. Some activities are local to an agent, while others may be shared with one or more other agents. In the latter case, all the agents sharing the activity need to agree on its execution.

Specifically, Alice can order and pay for a product of price p; the product is then delivered, unless she cancels, in which case she is refunded an amount r. Bob receives the customer's payment and prepares the order for shipment. In the happy path, he gets a tracking id and is charged amount c by the carrier. If the customer cancels, Bob withdraws from the carrier and refunds the customer; however, the refund is only partial if cancellation happens after the tracking id was created, because in this case he is partially charged by the carrier. Carla is notified about the prepared package, picks up the package, creates a tracking id, delivers the package, and charges Bob, the seller; in case the seller withdraws, charging depends on whether pickup already happened. We formalise these processes as three distinct DPNs N_A, N_B, and N_C:



We assume that agents see only activities that occur in their process models, not the remaining ones (e.g., the activities visible to Alice are {pay,

deliver, cancel, refund}). Moreover, suppose that every activity is triggered by one agent, as indicated by the colors above: red for Alice, blue for Bob, green for Carla. This is only for visualization purposes, and does not affect the semantics.

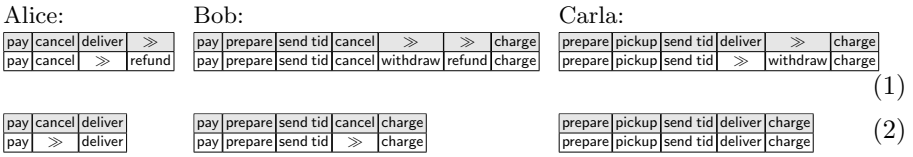
Consider now the following scenario where Alice cancelled the order but Bob forgot to send a withdraw message to Carla, and the cancellation happened after Carla had already picked up the package (which Alice cannot know about):
$\mathbf{e} = \langle \text{pay}\,\{p \mapsto 100\}, \text{prepare}, \text{pickup}, \text{send tid}, \text{cancel}, \text{deliver}, \text{charge}\,\{c \mapsto 20\}\rangle$

If we filter $\mathbf{e}$ with respect to the activities that are visible to the individual agents, we get the following three *projections*:

Alice: $\mathbf{e}_A = \langle \text{pay}\,\{p \mapsto 100\}, \text{cancel}, \text{deliver}\rangle$
Bob: $\mathbf{e}_B = \langle \text{pay}\,\{p \mapsto 100\}, \text{prepare}, \text{send tid}, \text{cancel}, \text{charge}\,\{c \mapsto 20\}\rangle$
Carla: $\mathbf{e}_C = \langle \text{prepare}, \text{pickup}, \text{send tid}, \text{deliver}, \text{charge}\,\{c \mapsto 20\}\rangle$

The agents can now check whether their respective projections on $\mathbf{e}$ indeed conform to their own local processes. If not, they can compute local alignments to understand which deviations occur, and where. However, in doing so, they cannot arbitrarily operate in autonomy, only selecting alignments based on their epistemic stance and minimisation of local distances. A further step is actually needed, where the agents confront with each other to merge their local knowledge into something that globally makes sense. Specifically, they should only retain those local alignments that agree on what should have happened, that is, are *compatible* with each other on the common tasks.

Let us now consider two possible sets of local alignments, each ensuring compatibility. First, the agents may reconstruct that withdrawal should have happened, corresponding to the alignments (1) below (the log component is shown on top, data values omitted for readability). Another explanation is that cancellation did not happen: then all other events can be perfectly matched (2):



When evaluating which of these two sets of alignments is preferrable, one may simply adopt the common approach to minimize the number of mismatches (≫). Under this assumption, (2) has a smaller overall cost. However, this might not be the preferred solution, because e.g. the analyst may want to express that if payments are correctly recorded then so are cancellations, as both are triggered by Alice. This would call for a refined definition of costs for Alice's tasks, defined based on the *context* in which tasks occur in the log. For example, the analysis could indicate that the cost of a log move on cancellation should be increased in case it is applied in a position that comes after a payment.

The example points out the two main ingredients we need to account for when dealing with conformance checking in such a distributed setting:

- We need to formally define how to project a global trace into local traces, and how to reconcile local alignments based on a suitable notion of compatibility.
- We need to support flexible cost functions based on (regular) expressions evaluated on the observed trace, taking position information into account.

The remainder of the paper is dedicated to devising a framework supporting these two ingredients, in a formal and operational way.

3 Background and Preliminaries

We use Data Petri nets (DPNs) for modeling multi-perspective processes, adopting the same formalization as in [7,13]. We fix a set of *sorts* $\Sigma = \{\texttt{bool}, \texttt{int}, \texttt{rat}\}$ with associated domains of booleans $\mathcal{D}(\texttt{bool}) = \mathbb{B}$, integers $\mathcal{D}(\texttt{int}) = \mathbb{Z}$, and rationals $\mathcal{D}(\texttt{rat}) = \mathbb{Q}$. A set of *process variables* V is *sorted* if there is a function $sort\colon V \to \Sigma$ assigning a sort to each variable $v \in V$. For a set of variables V, we consider two disjoint sets of annotated variables $V^r = \{v^r \mid v \in V\}$ and $V^w = \{v^w \mid v \in V\}$ to be respectively read and written by process activities, as explained below, and we assume $sort(v^r) = sort(v^w) = sort(v)$ for every $v \in V$. For a sort $\sigma \in \Sigma$, V_σ denotes the subset of $V^r \cup V^w$ of annotated variables of sort σ. To manipulate variables, we consider expressions c defined as follows:

$$c ::= V_{\texttt{bool}} \mid \mathbb{B} \mid n \geq n \mid r \geq r \mid r > r \mid c \wedge c \mid \neg c$$
$$n ::= V_{\texttt{int}} \mid \mathbb{Z} \mid n + n \mid -n \qquad\qquad r ::= V_{\texttt{rat}} \mid \mathbb{Q} \mid r + r \mid -r$$

The set of constraints over variables V is denoted $\mathcal{C}(V)$; they are used for read and write operations in process activities.

Definition 1 (DPN). *A tuple $N = (P, T, F, \ell, A, V, guard)$ is a* Petri net with data *(DPN), where:*

- *(P, T, F, ℓ) is a Petri net with two non-empty disjoint sets of places P and transitions T, a flow relation $F : (P \times T) \cup (T \times P) \to \mathbb{N}$ and a labeling function $\ell : T \to A \cup \{\tau\}$, where A is a finite set of activity labels and τ is a special symbol denoting silent transitions;*
- *V is a set of typed process variables; and*
- *$guard\colon T \to \mathcal{C}(V^r \cup V^w)$ is a guard assignment; for $t \in T$ with $\ell(t) = \tau$ we assume that $guard(t)$ does not use variables in V^w.*

Transition guards serve to simultaneously read and write variables. For instance, a transition with guard $(x^r > 3)$ can only be taken if the current value of variable x is greater than 3 (the superscript r indicates that the guard is on the current, or *read*, variable). On the other hand, a guard $(x^w > 1) \wedge (x^r < 4)$ requires that the current value of x is smaller than 4 and, at the same time, it non-deterministically *writes* to x a new value that is greater than 1 (superscripts w refer to *written* values). Note that transition guards with disjunctions like in Example 1 can be simulated by using multiple transitions between the same places.

As customary, given $x \in P \cup T$, we use ${}^\bullet x := \{y \mid F(y,x) > 0\}$ to denote the *preset* of x and $x^\bullet := \{y \mid F(x,y) > 0\}$ to denote the *postset* of x. In order to refer to the variables read and written by a transition t, we use the notations $read(t) = \{v \mid v^r \in \mathcal{V}\text{⊣}\nabla(guard(t))\}$ and $write(t) = \{v \mid v^w \in \mathcal{V}\text{⊣}\nabla(guard(t))\}$.

To represent the current values of variables, we consider a *state variable assignment*, i.e., a (possibly partial) function α that assigns a value (of the right type) to each variable in V. We denote by $v\textsc{dom}(\alpha)$ the domain of α. A *state* in a DPN N is a pair (M, α) constituted by a marking $M\colon P \to \mathbb{N}$ for the underlying Petri net (P, T, F, ℓ), plus a total state variable assignment α. Therefore, a state simultaneously accounts for the control flow progress and for the current values of all variables in V, as specified by α. We fix one state (M_I, α_0) as *initial*, where M_I is the initial marking of the underlying Petri net and α_0 specifies the initial value of all variables in V. Similarly, we denote the final marking as M_F, and call *final* any state of the form (M_F, α_F) for some α_F.

A *transition variable assignment* is a partial function β with $\textsc{dom}(\beta) \subseteq V^r \cup V^w$ that assigns a value to annotated variables, namely $\beta(x) \in \mathcal{D}(sort(x))$, with $x \in V^r \cup V^w$. Transition variable assignments are used to specify how variables change as the result of activity executions (cf. Definition 2).

We now define when a Petri net transition may fire from a given state.

Definition 2 (Transition firing). *A transition $t \in T$ is* enabled *in state (M, α) if there exists a transition variable assignment β such that:*

- $\textsc{dom}(\beta) = \mathcal{V}\text{⊣}\nabla(guard(t))$*: β is defined for the variables in the guard;*
- $\beta(v^r) = \alpha(v)$ *for every $v \in read(t)$, i.e., β is as α for read variables;*
- $\beta \models guard(t)$*, i.e., β satisfies the guard; and*
- $M(p) \geq F(p,t)$ *for every $p \in {}^\bullet t$.*

An enabled transition may fire*, producing a new state (M', α'), s.t. $M'(p) = M(p) - F(p,t) + F(t,p)$ for every $p \in P$, and $\alpha'(v) = \beta(v^w)$ for every $v \in write(t)$, and $\alpha'(v) = \alpha(v)$ for every $v \notin write(t)$. A pair (t, β) as above is called (valid)* transition firing*, and we denote its firing by $(M, \alpha) \xrightarrow{(t,\beta)} (M', \alpha')$.*

Informally, a transition firing between the current state (M, α) and the next state (M', α') is a couple (t, β) where: *i)* $t \in T$ is a transition that is enabled in the 'token game' sense of standard Petri nets; *ii)* β is a function connecting the values of the read variables (matching the values assigned by α in the current state) to the values of the write variables (matching the values assigned by α' in the next state); *iii)* β satisfies the guard associated to t. For Petri nets without data variables, we omit β in transition firings for readability.

Based on this single-step transition firing, we say that a state (M', α') is *reachable* in a DPN with initial state (M_I, α_0) iff there exists a sequence of valid transition firings of the form $\mathbf{f} = \langle (t_1, \beta_1), \ldots, (t_n, \beta_n) \rangle$ such that $(M_I, \alpha_0) \xrightarrow{(t_1,\beta_1)} \ldots \xrightarrow{(t_n,\beta_n)} (M', \alpha')$. Such a sequence $\mathbf{f}$ is called a *process run* of N if $(M_I, \alpha_0) \xrightarrow{\mathbf{f}} (M_F, \alpha_F)$ for some α_F, i.e., if the run leads to a final state. For any kind of sequence $\mathbf{x} = \langle x_1, \ldots, x_n \rangle$ and $1 \leq i \leq n$, we write $\mathbf{x}|_{<i}$ for the prefix $\langle x_1, \ldots, x_{i-1} \rangle$, and $\mathbf{x}|_{\leq i}$ for $\langle x_1, \ldots, x_i \rangle$. As in [7,14], we restrict to DPNs

where a final state is reachable. We denote the set of transition firings of N by $\mathcal{F}(N)$, and the set of process runs by $Runs(N)$.

For instance, the three processes depicted in Example 1 are three different DPNs, where the set of variables is $V = \{p, r, c\}$, all of type `rat`. Initial markings are those where a single token is in the places marked by a token in the picture; and final markings are the singleton markings where one token is in the place marked by a double border. One could also consider the three nets as just one big DPN, but in the sequel we will not adopt this view.

4 Glocal Conformance Checking

In this section, we formalise the problem of *glocal conformance checking*, dealing with problems like the one explained in Sect. 2.

Global and Local Traces. We assume that a finite set $\mathcal{A} := \{a_1, \ldots, a_n\}$ of n agents is given, and associate to each agent a_i a process model N_i specified as a DPN. The set of all DPNs is abbreviated by $\mathcal{N} := \{N_1, \ldots, N_n\}$. We moreover assume that $N_i = (P_i, T_i, F_i, \ell_i, A_i, V, guard_i)$ for all $i \in \{1, \ldots, n\}$, with $P_i \cap P_j = \emptyset$ and $T_i \cap T_j = \emptyset$ for $1 \leq i < j \leq n$. Importantly:

- The sets A_i of actions are *not* disjoint across agents. This is key in out approach. Conceptually, when transitions of different DPNs are labelled by the same action, then that action is *shared* among the agents associated to those DPNs (in the epistemic sense formally captured later in Definition 3). Depending on how the different labelling functions ℓ_i are defined, we can thus capture local actions, actions shared by some agents, and global actions shared by all. We write $A := \bigcup_{i=1}^{n} A_i$ for the shared action vocabulary.
- All DPNs share the same set V of process variables. Every DPN maintains *its own local assignment* for such variables, consistently with the notion of DPN state and transition firing defined before. Variables exclusively used by a single agent, or by a subset of agents, can be seamlessly supported, by ensuring that the DPNs of the other agents do not use them.

A good modelling principle is that whenever two agents operate over the same variable, they write that variable only through actions they share, and using exactly the same guard. This guarantees that whenever the variable gets updated, all agents using that variable consistently update their own state. When this modelling principle is respected, variables effectively behave like *global variables*.

Consider an agent, and a trace possibly containing actions not visible to that agent. As a first fundamental building block, we need to define how the trace gets projected into a local trace representing the epistemic view of the agent. Let a *global transition sequence* be a sequence $\langle f_1, f_2, ..., f_m \rangle$ such that each f_k, $1 \leq k \leq m$, is a transition firing of some N_j, i.e., there are $M_k, M'_k, \alpha_k, \alpha'_k$ and $1 \leq j \leq n$ such that $(M_k, \alpha_k) \xrightarrow{f_k} (M'_k, \alpha'_k)$ is a valid transition firing of N_j. Note that a global transition sequence is merely a sequence of tuples, we do not require that the firings can be subsequently enabled in any model.

Definition 3 (Run projection). *Given a global transition sequence* $\mathbf{f} = \langle f_1, \ldots, f_m \rangle$*, the* projection $\mathbf{f}_{a_j}$ *of* $\mathbf{f}$ *on agent* a_j *is the maximal subsequence in the shared action set, i.e., the maximal subsequence* $\langle f_{k_1}, \ldots, f_{k_s} \rangle$ *of* $\mathbf{f}$ *such that* $f_{k_r} = (t_r, \beta_r)$ *and* $\ell(t_r) \in A_j$*, for all* $1 \leq r \leq s$*.*

For different DPNs to properly support the execution of common, global traces over the overall set of actions occurring in those DPNs, we need to identify a suitable notion of compatibility. The literature abounds of frameworks that deal with processes under the responsibility of different agents, tackling their interoperability. A classical setting is the one where a monolithic process model is decomposed into sub-components or views that are assigned to process stakeholders, who are completely unaware of all the events pertaining tasks that are not part of their own views [6,9]. This also connects back to a line of research pursued to ensure the realisability of multi-party choreographic processes, considering that they must be enacted in a decentralized way through interaction of the parties, which in turn call for ensuring that they locally have enough visibility of the current state of affairs when required to perform some task [5,12]. In such a setting, open nets [2] can be used to tackle, bottom-up, the construction of a global process from such local specifications through suitable notions of compositions. In a more recent line of research, hybrid process models have been put forward to handle the recurring situation where the same process instance needs to simultaneously go through multiple, concurrent (data-aware) processes [1]. When relating our setting to this long-standing literature, three points deserve emphasis: *(i)* local DPNs are data-aware, a feature that is only present in [1]; *(ii)* local DPNs do not share places, while they may share transitions; *(iii)* to tackle glocal conformance checking, we are interested in traces, and do not need to relate local DPNs to an explicit, global model (such as an orchestrator or a choreography). With these three points in mind, we provide next two suitable notions of compatibility.

Definition 4 (Compatibility). *DPNs* $N_1, \ldots, N_n$ *are*

- weakly compatible *if there is a global transition sequence* $\widehat{\mathbf{f}}$ *such that for all* j *with* $1 \leq j \leq n$*, the projection* $\widehat{\mathbf{f}}_{a_j}$ *is a run of* N_j*; and*
- strongly compatible *if for every run* $\mathbf{f} \in \mathit{Runs}(N_i)$ *and* $1 \leq i \leq n$*, there is a global transition sequence* $\widehat{\mathbf{f}}$ *such that* $\mathbf{f}$ *coincides with* $\widehat{\mathbf{f}}_{a_i}$ *and for all* j *with* $1 \leq j \leq n$*, the projection* $\widehat{\mathbf{f}}_{a_j}$ *is a run of* N_j*.*

Intuitively, weak compatibility demands that there is a combination of runs from each DPN that is not contradictory. This, in turn, indicates that every DPN has at least one local behaviour that can be suitably complemented by compatible local behaviours of the other DPNs. However, under weak compatibility not all runs in a local DPN may have corresponding compatible runs in the others. This is instead what is guaranteed by the more restrictive notion of strong compatibility: that *every* run of one of the DPNs can be embedded in a global, combined run whose local projections can be executed by all DPNs.

Note that since the set of runs $Runs(N_i)$ is assumed to be nonempty for all N_i, strong compatibility implies weak compatibility.

Definition 4 provides an abstract compatibility criterion that reconstructs: *(i)* the execution semantics of hybrid models where process components synchronise on their common actions [1]; *(ii)* the composition semantics of open nets, in the special case where nets only have common transitions and no open place.

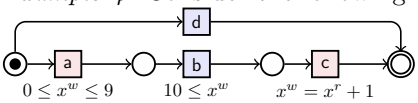
Example 2. The three Petri nets from Example 1 are strongly compatible: We show this for runs of Alice, the other cases are similar. Two cases can be distinguished:

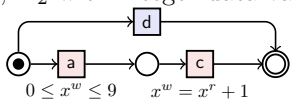
- Any run of the form $\mathbf{f}_1 = \langle \mathsf{pay}\,\{p^w \mapsto x\}, \mathsf{deliver}\rangle$ of N_A, for some $x \in \mathbb{Q}$, is embedded in the global sequence $\widehat{f_1} = \langle \mathsf{pay}\,\{p^w \mapsto x\}, \mathsf{prepare}, \mathsf{pickup}, \mathsf{send\ tid}, \mathsf{deliver}, \mathsf{charge}\,\{c^w \mapsto 20\}\rangle$, which projects to runs for Bob and Carla.
- A run $\mathbf{f}_2 = \langle \mathsf{pay}\,\{p^w \mapsto x\}, \mathsf{cancel}, \mathsf{refund}\,\{p^r \mapsto x, r^w \mapsto y\}\rangle$ for some $x, y \in \mathbb{Q}$ is embedded in $\widehat{f_2} = \langle \mathsf{pay}\,\{p^w \mapsto x\}, \mathsf{prepare}, \mathsf{cancel}, \mathsf{withdraw}, \mathsf{refund}\,\{p^r \mapsto x, r^w \mapsto y\}\rangle$ which can be projected to valid runs for Bob and Carla as well.

We next give one counterexample to weak compatibility, and one example that shows how compatibility can be hampered by data variables that are used by two agents, but written in transitions visible to one agent, i.e., the above-mentioned modelling principle is not followed.

Example 3. Consider three very simple Petri nets without data N_1, N_2, N_3 that admit only a single run each, namely $Runs(N_1) = \{\langle \mathsf{a}, \mathsf{b}\rangle\}$, $Runs(N_2) = \{\langle \mathsf{b}, \mathsf{c}\rangle\}$, and $Runs(N_3) = \{\langle \mathsf{c}, \mathsf{a}\rangle\}$. While any two of them are strongly compatible, the three of them are not even weakly compatible: N_1 requires that b happens after a, and N_2 that c happens after b, but N_3 that c happens *before* a.

Example 4. Consider the following DPNs N_1, N_2 with integer data variable x:





They are weakly compatible due to the global transition sequence $\widehat{\mathbf{f}} = \langle \mathsf{d}\rangle$, but not strongly. In fact, $\widehat{\mathbf{f}}$ is the *only* global transition sequence that projects to runs for both agents: any other such sequence would need to have the form

$$\langle (\mathsf{a}, \{x^w \mapsto r\}), (\mathsf{b}, \{x^w \mapsto s\}), (\mathsf{c}, \{x^r \mapsto t, x^w \mapsto t+1\})\rangle$$

for some $r, s, t \in \mathbb{Z}$. However, to project to a run of N_1, we would need $t = s$, but for N_2 we would need $t = r$, which is impossible by the constraints. In fact, by dropping transition d, the DPNs would not even be weakly compatible.

Next, we define the notion of event log. Given a set S, we denote by S^* the set of sequences of elements from S. An *event* is a pair (b, α) for $b \in A$ an activity label and α a (typically partial) state variable assignment, associating values to variables in V.

Definition 5 (Event log). *Given a set $\mathcal{E}$ of events, a* log trace $\mathbf{e} \in \mathcal{E}^*$ *is a sequence of events in $\mathcal{E}$ and an* event log L *is a multiset of log traces.*

We say that a log trace L is *local to agent* a_i (or to the DPN N_i) if all the activity labels in L are in A_i. If a log trace $\mathbf{e}$ is not local to any agent in $\mathcal{A}$ then $\mathbf{e}$ is a *global log trace*. Analogously, we call a *global event log* a multiset of global log traces. Intuitively, a global event log is a multi-set of log traces that correspond to the execution of activities of the global system emerging from the (concurrent) executions of the single processes controlled by the different agents.

For instance, $\mathbf{e}$ in Example 1 is a global log trace, while $\mathbf{e}_A$, $\mathbf{e}_B$, and $\mathbf{e}_C$ are local to the respective agent. Contrary to standard conformance checking, in our setting a global log trace cannot be compared to a global process model, because no such model exists; and due to the presence of data, a global model cannot be simply obtained as a cross-product of the local models. Therefore, we compute alignments with respect to the local models, after projecting global log traces to what the single agents can see. To this end, we proceed in three steps: *(i)* we recall the standard notion of data-aware alignments for DPNs, *(ii)* we use this notion to compute local alignments on trace projections, *(iii)* we fuse local alignments into a global viewpoint through a compatibility criterion that matches well the model-level one given in Definition 4, finally getting a "glocal" result.

Standard Alignments. Standard conformance checking aims at constructing an *alignment* of a given log trace $\mathbf{e}$ with respect to a given DPN N, by matching events in $\mathbf{e}$ against transition firings in a process run. Since not every event can be put in correspondence with a transition firing, a "skip" symbol $\gg$ is used. Let $\mathcal{E}^{\gg} = \mathcal{E} \cup \{\gg\}$ and the extended set of transition firings $\mathcal{F}(N)^{\gg} = \mathcal{F}(N) \cup \{\gg\}$.

Given a DPN N and a set $\mathcal{E}$ of events, a pair $(e, f) \in \mathcal{E}^{\gg} \times \mathcal{F}(N)^{\gg} \setminus \{(\gg, \gg)\}$ is called a *move*. A move (e, f) is a *log move* if $e \in \mathcal{E}$ and $f = \gg$; a *model move* if $e = \gg$ and $f \in \mathcal{F}(N)$; and a *synchronous move* if $(e, f) \in \mathcal{E} \times \mathcal{F}(N)$. Let $Moves_N$ be the set of all moves. For a sequence of moves $\gamma = (e_1, f_1), \ldots, (e_m, f_m)$, the log and model projections are obtained by restricting to one component and dropping $\gg$ symbols. Precisely, the *log projection* $\gamma|_L$ of γ is the maximal subsequence $\mathbf{e}$ of $\langle e_1, \ldots, e_m \rangle$ such that $\mathbf{e} \in \mathcal{E}^*$; and the *model projection* $\gamma|_M$ of γ is the maximal subsequence $\mathbf{f}$ of $\langle f_1, \ldots, f_m \rangle$ such that $\mathbf{f} \in \mathcal{F}(N)^*$.

Definition 6 (Alignment). *For a DPN N and a log trace $\mathbf{e}$ local to N, a sequence of moves γ is an* N-alignment *if $\gamma|_L = \mathbf{e}$; it is* complete *if $\gamma|_M \in Runs(N)$.*

Note that the notion of *N-alignment* in Definition 6 depends on N, since alignments can vary if different DPN models are considered.

Example 5. Consider Example 1 and $\mathbf{e}_A = \langle \mathsf{pay}\,\{p \mapsto 100\}, \mathsf{cancel}, \mathsf{deliver}\rangle$. The following sequences γ_1 and γ_2 are complete N_A-alignments of $\mathbf{e}_A$:

$$\gamma_1 = \begin{array}{|c|c|c|c|}\hline \mathsf{pay}\ p \mapsto 100 & \mathsf{cancel} & \mathsf{deliver} & \gg \\ \hline \mathsf{pay}\ p^w \mapsto 100 & \mathsf{cancel} & \gg & \mathsf{refund}\ r^w \mapsto 90 \\ \hline \end{array} \qquad \gamma_2 = \begin{array}{|c|c|c|}\hline \mathsf{pay}\ p \mapsto 100 & \mathsf{cancel} & \mathsf{deliver} \\ \hline \mathsf{pay}\ p^w \mapsto 100 & \gg & \mathsf{deliver} \\ \hline \end{array}$$

Here, alignments are depicted as tables where the cells in the top row correspond to log trace events, and cells in the bottom row correspond to transitions in the model run. Cells of trace events contain activity labels and event variable assignments; in cells for model transitions we list the written variables (instead of the entire transition variable assignment, for the sake of readability).

Given a log trace $\mathbf{e}$ local to N, $\mathit{Align}(N, \mathbf{e})$ denotes the set of complete N-alignments for a log trace $\mathbf{e}$. A *cost function* is a mapping $\kappa\colon \mathit{Moves}_N \to \mathbb{R}^+$ that assigns a cost to every move. This is naturally extended to alignments:

Definition 7 (Cost). *Given N, $\mathbf{e}$ and $\gamma = (e_1, f_1), \ldots, (e_n, f_n) \in \mathit{Align}(N, \mathbf{e})$, the* cost *of γ is obtained by summing up the costs of its moves, that is, $\kappa(\gamma) = \sum_{i=1}^{n} \kappa(e_i, f_i)$. Moreover, γ is* optimal for $\mathbf{e}$ *if $\kappa(\gamma)$ is minimal among all complete alignments for $\mathbf{e}$, namely there is no $\gamma' \in \mathit{Align}(N, \mathbf{e})$ with $\kappa(\gamma') < \kappa(\gamma)$.*

Glocal Alignments. Next, we define local alignments by combining standard alignments with log trace projections on the different agents.

Definition 8 (Log trace projection). *Given a global log trace $\mathbf{e}_G = \langle e_1, \ldots, e_m\rangle$, let the* projection of $\mathbf{e}_G$ on agent a_i *be $👁_{a_i}(\mathbf{e}_G) = \langle e_{j_1}, \ldots, e_{j_s}\rangle$, where $\langle e_{j_1}, \ldots, e_{j_s}\rangle$ is the maximal subsequence of $\mathbf{e}$ such that for each h, $1 \le h \le s$, we have $e_{j_h} = (b, \alpha)$ and b is in the set A_i of activity labels of agent a_i.*

In Example 1, $\mathbf{e}_A$, $\mathbf{e}_B$, and $\mathbf{e}_C$ are projections of the global trace $\mathbf{e}$ to the respective agent. Now given a global log trace $\mathbf{e}_G \in L_G$, we want to compute its alignments considering one of the process models N_i for some agent a_i. We call this a *local alignment of $\mathbf{e}_G$ with respect to a_i*. For instance, the alignments in Example 6 are local alignments of the global log trace $\mathbf{e}$ with respect to Alice. Formally, we get:

Definition 9 (Local alignment). *Given a global trace $\mathbf{e}_G$, γ_{a_i} is called a* local alignment of $\mathbf{e}_G$ with respect to agent a_i *if γ_{a_i} is an N_i-alignment of the projection $👁_{a_i}(\mathbf{e}_G)$ on agent a_i.*

Since agents share actions, they can observe and interpret the alignments done by others, by limiting themselves to those actions they know about.

Definition 10 (Alignment projection). *For $\gamma = \langle (e_1, f_1), \ldots, (e_m, f_m)\rangle$ a local alignment of a global log trace $\mathbf{e}_G$ with respect to agent a_i, the* projection of γ on agent a_j *is $\mathit{proj}_{a_j}(\gamma) := 👁_{a_j}(\gamma|_M)$.*

The projection of a local alignment to agent a_j thus selects all moves that are also valid for agent a_j. We are now able to define compatibility as a form of "knowledge alignment", dictating that two agents agree on two local alignments they are respectively pondering, if the mutual projections are identical.

Definition 11 (Compatible alignments). *Two local alignments γ_{a_i} for a_i and γ_{a_j} for a_j are* compatible *if $proj_{a_j}(\gamma_{a_i}) = proj_{a_i}(\gamma_{a_j})$.*

We extend this notion to sets of alignments and agents: given a log trace $\mathbf{e}_G$ and a set of alignments $\{\gamma_1, \ldots, \gamma_n\}$ such that for all i, $1 \leq i \leq n$, γ_i is a local alignment of agent a_i and $\mathbf{e}_G$, we call this set *compatible* if they are pairwise compatible. Since this set carries at once a local meaning for each agent, paired with compatibility at the global level, we also call $\{\gamma_1, \ldots, \gamma_n\}$ a *glocal set of alignments* for $\mathbf{e}_G$ and $\mathcal{N}$ in this case. Finally, we show that, if we start from local DPNs that mutually agree on some shared action sequence (that is, they are weakly compatible in the sense of Definition 4), then it is always possible to find a glocal set of alignments (that is, local alignments that are pairwise compatible in the sense of Definition 11).

Theorem 1. *Let $N_1, \ldots, N_n$ be weakly compatible DPNs, and $\mathbf{e}_G$ a global log trace. Then there is a glocal set of alignments $\{\gamma_1, \ldots, \gamma_n\}$ for $\mathbf{e}_G$ and $\mathcal{N}$.*

Proof. By weak compatibility, there exists a global transition sequence $\widehat{\mathbf{f}}$ such that for all i, $1 \leq i \leq n$, the projection $\blacktriangleleft_{a_i}(\widehat{\mathbf{f}})$ is a run of N_i. For an agent a_i, $1 \leq i \leq n$, let $\blacktriangleleft_{a_i}(\widehat{\mathbf{f}}) = \langle (t_{i,1}, \beta_{i,1}), \ldots, (t_{i,m_i}, \beta_{i,m_i}) \rangle$, and $\blacktriangleleft_{a_i}(\mathbf{e}_G) = \langle (e_{i,1}, \alpha_{i,1}), \ldots, (e_{i,k_i}, \alpha_{i,k_i}) \rangle$. Now consider the sequence of moves

$$\gamma_i = \begin{array}{|c|}\hline \gg \\ \hline t_{i,1}\ \beta_{i,1} \\ \hline \end{array} \cdots \begin{array}{|c|c|}\hline \gg & e_{i,1}\ \alpha_{i,1} \\ \hline t^i_{i,m_i}\ \beta_{i,m} & \gg \\ \hline \end{array} \cdots \begin{array}{|c|}\hline e_{i,k}\ \alpha_{i,k_i} \\ \hline \gg \\ \hline \end{array}.$$

By definition, γ_i is a local alignment of $\mathbf{e}_G$ with respect to a_i, and it is complete because $\blacktriangleleft_{a_i}(\widehat{\mathbf{f}})$ is a run of N_i. To verify that $\gamma_1, \ldots, \gamma_n$ is a glocal set of alignments, it remains to show that for two agents a_i, a_j, the alignments γ_i and γ_j are compatible, that is, $\blacktriangleleft_{a_j}(\gamma_i|_M) = \blacktriangleleft_{a_i}(\gamma_j|_M)$. By construction of γ_i, γ_j from $\widehat{\mathbf{f}}$, this is equivalent to $\blacktriangleleft_{a_j}(\blacktriangleleft_{a_i}(\widehat{\mathbf{f}})) = \blacktriangleleft_{a_i}(\blacktriangleleft_{a_j}(\widehat{\mathbf{f}}))$, which holds because both sides of the equation describe the maximal subsequence of $\widehat{\mathbf{f}}$ in which all transitions have a label in $A_i \cap A_j$. □

Example 6. Consider again Example 1 and the following three alignments γ_A, γ_B, and γ_C (where send tid is abbreviated to tid):

pay $p \mapsto 100$	cancel	deliver		
pay $p^w \mapsto 100$	$\gg$	deliver		

pay $p \mapsto 100$	prepare	send tid	cancel	charge $c \mapsto 20$
pay $p^w \mapsto 100$	prepare	send tid	$\gg$	charge $c^w \mapsto 20$

prepare	pickup	send tid	deliver	charge $c \mapsto 20$
prepare	pickup	send tid	deliver	charge $c^w \mapsto 20$

These are complete local alignments of $\mathbf{e}$ with respect to the three agents. We have $proj_{B}(\gamma_A) = proj_{A}(\gamma_B) = \langle \mathsf{pay}\{p^w \mapsto 100\} \rangle$, and $proj_{C}(\gamma_A) = proj_{A}(\gamma_C) = \langle \mathsf{deliver} \rangle$, and $proj_{C}(\gamma_B) = proj_{B}(\gamma_C) = \langle \mathsf{prepare}, \mathsf{send\ tid}, \mathsf{charge}\{c^w \mapsto 20\} \rangle$, so the alignments are compatible.

Theorem 1, based on weak compatibility, states only existence of a set of glocal alignments, but gives no optimality guarantee, as the next example illustrates.

Example 7. Consider again Example 4, log trace $\mathbf{e}_G = \langle (\mathsf{a}, \{x \mapsto 1\}), (\mathsf{c}, \{x \mapsto 11\}) \rangle$ and

$\gamma_1 = \gamma_2 =$

$\gg$	a $x \mapsto 1$	c $x \mapsto 11$
d	$\gg$	$\gg$

$\gamma'_1 =$

a $x \mapsto 1$	c $x \mapsto 11$
a $x^w \mapsto 1$	c $x^w \mapsto 2$

$\gamma'_2 =$

a $x \mapsto 1$	$\gg$	c $x \mapsto 11$
a $x^w \mapsto 1$	b $x \mapsto 10$	c $x^w \mapsto 11$

Then γ_1, γ_2 form a glocal set of alignments, and all such sets have this shape up to reordering moves. Let κ assign cost 1 to log and model moves, and to synchronous moves cost ∞ in case of different labels, and cost 0 (resp. cost 1) if labels match but variables are assigned the same (resp. different) values. Then $\kappa(\gamma_1) = \kappa(\gamma_2) = 3$, but $\kappa(\gamma_1') = \kappa(\gamma_2') = 1$. In fact γ_1', γ_2' are optimal local alignments, but do not form a glocal set. This shows that if DPNs are weakly but not strongly compatible, glocal alignments can be suboptimal for all agents.

We conclude this section by showing that strong compatibility ensures existence of a glocal set of alignments that satisfies some (local) optimality guarantee:

Theorem 2. *Let $N_1, \ldots, N_n$ be strongly compatible DPNs, $\mathbf{e}_G$ a global log trace, and $1 \leq k \leq n$. Then there is a local alignment γ_i of $\mathbf{e}_G$ with respect to agent a_i, for every i, $1 \leq i \leq n$, such that $\gamma_1, \ldots, \gamma_n$ is a glocal set of alignments, and γ_k is optimal for $\mathbb{O}_{a_k}(\mathbf{e}_G)$.*

Proof. Let $\mathbf{f}_k = \gamma_k|_M$, which is a run of N_k. By strong compatibility, there exists a global transition sequence $\widehat{\mathbf{f}}$ such that for all i, $1 \leq i \leq n$, the projection $\mathbb{O}_{a_i}(\widehat{\mathbf{f}})$ is a run of N_i, and $\mathbf{f}_k$ coincides with $\mathbb{O}_{a_k}(\widehat{\mathbf{f}})$. For all i such that $1 \leq i \leq n$ and $i \neq k$, one can define γ_i as in the proof of Theorem 1, and the alignments are compatible for the same reason. □

5 Context-Aware Cost Functions

We next delve into the second research question pertaining glocal conformance checking, namely the design of cost functions that can flexibly prescribe different costs of moves depending on their context in an alignment. To that end, we consider the natural approach of defining context by a regular expression, evaluated on the observed trace up to the considered instant (i.e., the trace prefix).

Regular Expressions. We recap regular expressions to fix notation. Let Γ be an alphabet, ϵ the empty string, $\circ$ denote string concatenation, and Γ^* the set of finite strings over Γ. The set of regular expressions over Γ, denoted by $RE(\Gamma)$, is defined by the grammar $e ::= \boldsymbol{\epsilon} \mid a \mid \boldsymbol{.} \mid e \circ e \mid e + e \mid e^{\boldsymbol{*}}$ where $a \in \Gamma$. For $A, B \subseteq \Gamma^*$, let $A \circ B = \{a \circ b \mid a \in A \text{ and } b \in B\}$. The language $\mathcal{L}(e)$ of a regular expression e is given by $\mathcal{L}(\boldsymbol{\epsilon}) = \{\epsilon\}$, $\mathcal{L}(a) = \{a\}$, $\mathcal{L}(\boldsymbol{.}) = \Gamma$, $\mathcal{L}(e_1 \circ e_2) = \mathcal{L}(e_1) \circ \mathcal{L}(e_2)$, $\mathcal{L}(e_1 + e_2) = \mathcal{L}(e_1) \cup \mathcal{L}(e_2)$, and $\mathcal{L}(e^{\boldsymbol{*}}) = \bigcup_{k \geq 0} \mathcal{L}(e^k)$, where $\mathcal{L}(e^k)$ is inductively defined by $\mathcal{L}(e^0) = \{\epsilon\}$ and $\mathcal{L}(e^{k+1}) = \mathcal{L}(e) \circ \mathcal{L}(e^k)$. We usually omit the concatenation operator for succinctness.

Let again A be the joint activity vocabulary of all agents, and V their set of process variables. Below, we will consider regular expressions over the alphabet $\Gamma = A \times \mathcal{C}(V)$. Though the alphabet is in general infinite, only finitely many symbols occur in the relevant regular expressions, so that we will in fact be able to work with a finite alphabet. For instance, the following are regular expressions in $RE(A \times \mathcal{C}(V))$, for A and $\mathcal{C}(V)$ as in Example 1:

- $e_1 = \mathbf{.}^*(\mathsf{pay}, \top)\mathbf{.}^*(\mathsf{refund}, p = w)$ describes an event sequence containing a pay event later followed by a refund event where the entire payment is refunded;
- $e_2 = ((\mathsf{pay}, p > 100)(\mathsf{cancel}, \top)\mathbf{.})^*$ captures sequences made of an arbitrary number of subsequences that start with a payment larger than 100 directly followed by cancellation, followed by some arbitrary events.

For readability, we write below $\neg a$ for $a \in A$ for the regular expression $(a_1, \top) + \ldots + (a_k, \top)$ if $A \backslash \{a\} = \{a_1, \ldots, a_k\}$. Moreover, we call a regular expression r *single-letter* if either $r \in \Gamma \cup \{\mathbf{.}\}$, or $r = r_1 + r_2$ for single-letter expressions r_1, r_2. Obviously, single-letter expressions match only words of length 1.

Cost Patterns. Next, we define cost patterns to capture prefixes of traces. The intuition is the following. We are scanning the observed trace (projection), to understand how it aligns with a given DPN. We are in position i, pondering a model or log move on event e. To define the cost of such a move, we inspect the prefix of the log trace up to position i, matching it against a regular expression that defines the relevant context for that move when assigning a given cost to it. Different contexts can be then linked to different costs for the same move. To this end, an event (b, α) *matches* a tuple $(a, c) \in A \times \mathcal{C}(V)$ if $a = b$ and $\alpha \models c$, i.e., the activity labels coincide and the state variable assignment α satisfies the constraint c. Accordingly, a sequence $\langle e_1, \ldots, e_m \rangle$ of events *matches* sequence $\langle (a_1, c_1), \ldots, (a_n, c_n) \rangle \in (A \times \mathcal{C}(V))^*$ if $m = n$ and e_i matches (a_i, c_i) for all i.

Definition 12 (Alignment cost scheme). *A* cost pattern *is a triple* $P = \langle r, p, k \rangle$ *where* $r \in RE(A \times \mathcal{C}(V))$, p *is a single-letter regular expression over* $A \times \mathcal{C}(V)$, *and* $k \in \mathbb{Q}_{\geq 0} \cup \{\infty\}$. *An* alignment cost scheme *is a triple* $\Pi = \langle \Lambda_L, \Lambda_M, d \rangle$ *where* Λ_L, Λ_M *are lists of cost patterns, and* $d \in \mathbb{Q}_{\geq 0} \cup \{\infty\}$.

E.g., for $e_1 = \mathbf{.}^*(\mathsf{pay}, \top)\mathbf{.}^*(\mathsf{refund}, p = w)$, the tuple $\langle e_1, (\mathsf{cancel}, \top), 100 \rangle$ is a cost pattern. We say that an event sequence $\langle e_1, \ldots, e_m \rangle$ with $m \geq 1$ *matches* cost pattern $P = \langle r, p, k \rangle$ if $\langle e_1, \ldots, e_{m-1} \rangle$ matches r and e_m matches p. Finally, an alignment cost scheme defines the following cost function:

Definition 13 (Context-dependent cost). *An alignment cost scheme* $\Pi = \langle \Lambda_L, \Lambda_M, d \rangle$ *induces the* context-dependent cost *function* κ_Π*: For* $\gamma = (e_1, f_1), \ldots, (e_{i-1}, f_{i-1}), (e_i, f_i)$ *a sequence of moves,* $i \geq 1$, *the cost* $\kappa_\Pi(e_i, f_i)$ *of move* (e_i, f_i) *is as follows:*

- *If* (e_i, f_i) *is a log move and* $\langle r, p, k \rangle$ *is the first pattern in* Λ_L *that matches* $\gamma_{\leq i}|_L$ *then* $\kappa_\Pi(e_i, f_i) = k$*; if no pattern matches then* $\kappa_\Pi(e_i, f_i) = d$.
- *If* (e_i, f_i) *is a model move and* $\langle r, p, k \rangle$ *is the first pattern in* Λ_M *that matches* $\gamma_{\leq i}|_L$ *then* $\kappa_\Pi(e_i, f_i) = k$*; if no pattern matches then* $\kappa_\Pi(e_i, f_i) = d$.
- *If* (e_i, f_i) *is a synchronous move,* $e_i = (b, \alpha)$, $f_i = (t_i, \beta_i)$, $\ell(t_i) = b$ *and* $\alpha(v) = \beta(v^w)$ *for all* $v \in \text{DOM}(\alpha)$ *then* $\kappa_\Pi(e_i, f_i) = 0$*, otherwise* $\kappa_\Pi(e_i, f_i) = \infty$.

Finally, $\kappa_\Pi(\gamma) = \sum_{i=1}^{n} \kappa_\Pi(\mathbf{m}_i)$.

For instance, for any $k \in \mathbb{N}$, the pattern $P_k = \langle \mathbf{.}^*, \mathbf{.}, k \rangle$ matches every trace and assigns cost 1. So the very simple alignment cost scheme $\langle [P_1], [P_2], \infty \rangle$

assigns cost 1 to all log moves and 2 to all model moves: for example, in γ_A from Example 6, this alignment cost scheme would assign cost 1 to log move $(\mathsf{cancel}, \gg)$, where the log prefix $(\mathsf{pay}, p = 100)$ is the context of $(\mathsf{cancel}, \emptyset)$.

One may wonder why we do not evaluate the context against the alignment prefix, instead of the observed trace prefix. The reason is twofold. First, alignments are hypotheses on what should have happened, while the observed trace provides factual knowledge. Second, alignments are chosen based on move costs, hence our approach is consequential: given an observed trace, we are able, in principle, to compute all the move costs, *then* using them to ponder the overall costs of alignments. This could be seen as a stance in line with the "in log we trust" motto from [15]. Here, this simply means that we trust that what we have in the log has been effectively recorded, and thus provides a factual context for defining costs. However, it does not imply that the recording is faithful to reality.

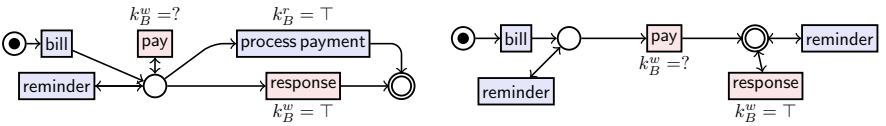
We next revisit Example 1 and show how context-dependent cost functions can be used to achieve the desired notion of an optimal alignment.

Example 8. We assume that the logging system is reliable but agents may fail to behave according to the normative process. A high cost for a particular move expresses here that the respective misbehaviour is severe, but cost ∞ that it only happens if there is no other choice. Hence, we accept alignments with cost ∞ only if there is no better solution. In this setting an optimal alignment is an apologetic scenario that explains how agents misbehaved the least. We define different cost functions for Alice, Bob, and Carla, and for every agent, moves have cost 0 if they use an activity which is not triggered by themselves: let thus $P_A^{nb} = \langle \bullet^*, \mathsf{deliver} + \mathsf{refund}, 0\rangle$ for Alice, $P_B^{nb} = \langle \bullet^*, \mathsf{pay} + \mathsf{cancel} + \mathsf{send\,tid} + \mathsf{charge}, 0\rangle$ for Bob, and $P_C^{nb} = \langle \bullet^*, \mathsf{prepare} + \mathsf{withdraw}, 0\rangle$ for Carla (where the superscript nb abbreviates *not my business*). Moreover, for Alice we fix cost ∞ for log moves with cancel that are not preceded by $\mathsf{deliver}$, to reflect the duty of customers to abstain from cancellations after delivery; let thus $P_A^{\mathsf{c}} = \langle (\neg\mathsf{deliver})^*, \mathsf{cancel}, \infty\rangle$. For Bob, model moves with $\mathsf{withdraw}$ have cost 100 if the log contains a cancellation that is not followed by withdrawal, so let $P_B^{\mathsf{w}} = \langle \bullet^*\mathsf{cancel}(\neg\mathsf{withdraw})^*, \mathsf{withdraw}, 100\rangle$. When using the cost schemata

$$\Pi_A = \langle [P_A^{nb}, P_A^{\mathsf{c}}], [P_A^{nb}], 1\rangle \quad \Pi_B = \langle [P_B^{nb}], [P_B^{nb}, P_B^{\mathsf{w}}], 1\rangle \quad \Pi_C = \langle [P_C^{nb}], [P_C^{nb}], 1\rangle$$

for alignments (1), we get cost 0 for Alice, 101 for Bob, and 2 for Carla, so 103 in total; for alignments (2), we get cost ∞ for Alice, and 0 for Bob and Carla, so ∞ in total. Thus alignments (1) are preferable (and optimal).

Example 9. Consider a payment process where a seller, Bob, sends a bill to a customer, Alice, who is supposed to pay. However, payments do not always succeed, so Bob might not know about them. If he believes to not have received a payment, he sends reminders, until Alice sends an email response with a payment confirmation. This is modelled by the below DPNs for Bob (left, triggered activities in blue) and Alice (right, activities in red). Here, k_B is an "epistemic" variable that indicates whether Bob knows about Alice's payment; it is non-deterministically written by the constraint $k_B^w = ?$ of pay:



The DPNs are weakly compatible. Consider trace $\mathbf{e} = \langle \mathsf{bill}, \mathsf{pay}\,\{k_B \mapsto \bot\}, \mathsf{reminder}\rangle$ with $\mathbf{e}_B = \mathbf{e}_A = \mathbf{e}$, admitting the following two pairs of compatible alignments $\gamma_{B,1}, \gamma_{A,1}$ and $\gamma_{B,2}, \gamma_{A,2}$ for Bob and Alice, respectively:

$$\gamma_{B,1} = \begin{array}{|c|c|c|c|}\hline \mathsf{bill} & \mathsf{pay}\,\{k_B \mapsto \bot\} & \mathsf{reminder} & \gg \\ \hline \mathsf{bill} & \mathsf{pay}\,\{k_B \mapsto \top\} & \mathsf{reminder} & \mathsf{process\ payment} \\ \hline \end{array} \quad \gamma_{A,1} = \begin{array}{|c|c|c|}\hline \mathsf{bill} & \mathsf{pay}\,\{k_B \mapsto \bot\} & \mathsf{reminder} \\ \hline \mathsf{bill} & \mathsf{pay}\,\{k_B \mapsto \top\} & \mathsf{reminder} \\ \hline \end{array}$$

$$\gamma_{B,2} = \begin{array}{|c|c|c|c|}\hline \mathsf{bill} & \mathsf{pay}\,\{k_B \mapsto \bot\} & \mathsf{reminder} & \gg \\ \hline \mathsf{bill} & \mathsf{pay}\,\{k_B \mapsto \bot\} & \mathsf{reminder} & \mathsf{response}\,\{k_B \mapsto \top\} \\ \hline \end{array} \quad \gamma_{A,2} = \begin{array}{|c|c|c|c|}\hline \mathsf{bill} & \mathsf{pay}\,\{k_B \mapsto \bot\} & \mathsf{reminder} & \gg \\ \hline \mathsf{bill} & \mathsf{pay}\,\{k_B \mapsto \bot\} & \mathsf{reminder} & \mathsf{response}\,\{k_B \mapsto \top\} \\ \hline \end{array}$$

When assigning cost 1 for every log and model move, as well as the synchronous pay move with mismatching data, we have $\kappa(\gamma_{B,1}) = 2$ and $\kappa(\gamma_{A,1}) = 1$; and $\kappa(\gamma_{B,2}) = 1$ and $\kappa(\gamma_{A,2}) = 1$, hence little difference with respect to alignment cost. However, the first alignment pair seriously changes the course of action, in that, for the alignments to be compatible, it is required to change $k_B \mapsto \top$ in the pay action, to make the model move with process payment possible.

Instead, let $\Pi_B = \langle [], [P^{\mathsf{res}}, P^{\mathsf{pay}}], 1\rangle$ with $P^{\mathsf{res}} = \langle \mathbf{.}^*, \mathsf{response}, 0\rangle$, and $P^{\mathsf{pay}} = \langle \mathbf{.}^*, \mathsf{process\ payment}, 100\rangle$, expressing that response has no cost for Bob as it is triggered by Alice and process payment model moves have high cost. For Alice, $\Pi_A = \langle [P^{\mathsf{res},1}, P^{\mathsf{res},2}], [], 1\rangle$ with $P^{\mathsf{res},1} = \langle \mathbf{.}^*(\mathsf{pay}, \neg k_B)\mathsf{reminder}, \mathsf{response}, 1\rangle$ and $P^{\mathsf{res},2} = \langle (\neg \mathsf{reminder})^*, \mathsf{response}, 100\rangle$, expressing that response model moves have low cost after pay with $k_B = \bot$ and a subsequent reminder, but high cost if no reminder occurred. This reflects that, if Alice receives a reminder after payment, she *knows* that Bob is not aware of her payment. Then $\kappa(\gamma_{B,1}, \gamma_{A,1}) = 102$ and $\kappa(\gamma_{B,2}, \gamma_{A,2}) = 1$, so $\gamma_{B,2}, \gamma_{A,2}$ are clearly preferred.

6 Implementation

We implemented our approach on top of the data-aware conformance checker CoCoMoT, a command-line tool written in Python, that uses SMT encodings to find optimal alignments for DPNs [7]. This is a natural choice considering that, in our setting, agents employ data-aware processes. In the restricted case where agents use pure control-flow Petri nets without data, other alignment-based conformance checking approaches may be explored, such as those based on synchronous products [3]. In the sequel, we refer to our extended tool as GloCoMoT (https://bitbucket.org/gconformance/glocal). In contrast to CoCoMoT, (i) besides the global trace, GloCoMoT takes n DPNs $N_1, \ldots, N_n$ as input, and searches for alignments $\gamma_1, \ldots, \gamma_n$ for each model, using a similar approach as in standard CoCoMoT, ensuring in addition that $\gamma_1, \ldots, \gamma_n$ are compatible (Definition 11). (ii) In addition, GloCoMoT takes n alignment cost schemes (Definition 12) as input, one for each DPN, and uses the resulting cost functions in the encoding. Below, we describe these two changes on the implementation level.

Compatibility Encoding. We encode compatibility for each pair of alignments γ_i, γ_j of agents a_i, a_j with activity sets A_i, A_j separately. In the CoCoMoT encoding [7],

variables $\mathtt{S}^i_1, \ldots \mathtt{S}^i_{k_i}$ and $\mathtt{S}^j_1, \ldots \mathtt{S}^j_{k_j}$ encode the transition sequences of agent a_i and a_j, respectively. For $\overline{\ell}_i = \ell(\mathtt{S}^i_1), \ldots \ell(\mathtt{S}^i_{k_i})$ and $\overline{\ell}_j = \ell(\mathtt{S}^j_1), \ldots \ell(\mathtt{S}^j_{k_j})$ the respective sequences of labels, we encode compatibility by encoding the length L of the *longest common subsequence* of $\overline{\ell}_i$ and $\overline{\ell}_j$, and check whether L equals the number of labels in $\overline{\ell}_i$ that are in A_j, and the number of labels in $\overline{\ell}_j$ in A_i.

Context-Dependent Cost Functions. GloCoMoT parses alignment cost schemes from command line arguments, and encodes them in *log move penalty* P_L and *model move penalty* P_M (cf. [7, Def. 11]): Given an alignment cost schema $\Pi = \langle \Lambda_L, \Lambda_M, d \rangle$, a log trace $\mathbf{e} = \langle e_1, \ldots, e_m \rangle$ and a process run $\mathbf{f} = \langle f_1, \ldots, f_k \rangle$, $P_L(i)$ returns the cost of a log move with e_i, and $P_M(i, j)$ the cost of a model move with f_j in an alignment where $e_1, \ldots, e_i$ is the prefix of the log trace already consumed. The encodings of P_L and P_M are denoted by $[P_L]$ and $[P_M]$. For P_L, given $\Lambda_L = \langle r_1, p_1, d_1 \rangle, \ldots, \langle r_l, p_l, d_l \rangle$, we set $[P_L]_i = d_q$ if q is the smallest number such that r_q matches $\langle e_1, \ldots, e_{i-1} \rangle$ and p_q matches e_i, or $[P_L]_i = d$ otherwise. For P_M, one needs to encode a case distinction. Let $\langle r_1, p_1, d_1 \rangle, \ldots, \langle r_o, p_o, d_o \rangle$ be all patterns in Λ_M such that r_α matches $\langle e_1, \ldots, e_i \rangle$ for all $1 \leq \alpha \leq o$. Then we set $[P_M]_{i,j} = ite(matches((\mathtt{S}_j, \mathtt{X}_j), p_1), d_1, \ldots ite(matches((\mathtt{S}_j, \mathtt{X}_j), p_o), d_o, d) \ldots)$ where $matches((\mathtt{S}_j, \mathtt{X}_j), p)$ simply encodes that for $p = (a, c)$, the label of $\mathtt{S}_j$ is a and the data variables $\mathtt{X}_j$ satisfy the constraint c. We tested our implementation on the examples in this paper; all test files are available in the repository.

7 Conclusions

We proposed a foundational framework, paired with a proof-of-concept implementation, for conformance checking in the common setting where process agents come with their "local" knowledge of the process. Local alignments formulated by agents may contradict each other, so we define a suitable compatibility condition for what we call "glocal" conformance. In this setting, it is especially useful to define the cost of moves depending on their context in the trace, for which we propose a new regular expression-based approach.

Our framework provides the first stepping stone towards *epistemic conformance*, in which subjective knowledge of process stakeholders is taken in consideration. In particular, we believe our contribution provides a solid basis to extract, from glocal alignments, explanations and root causes for alignments. Along this line, we intend to refine the framework by differentiating visibility and ownership about the execution of activities. We also plan to explore settings where agent processes can not only interact synchronously on common actions, but must obey global contractual/choreography/interaction rules, which calls for refining the compatibility notion used here. Moreover, the compatibility notions introduced here to guarantee the existence of glocal alignments are undecidable to check for general DPNs; it would be worth studying whether they become decidable for DPNs with *finite summary* [8].

References

1. Alman, A., Maggi, F.M., Montali, M., Patrizi, F., Rivkin, A.: A framework for modeling, executing, and monitoring hybrid multi-process specifications with bounded global-local memory. Inf. Syst. **119**, 102271 (2023)
2. Baldan, P., Corradini, A., Ehrig, H., Heckel, R.: Compositional semantics for open Petri nets based on deterministic processes. MSCS **15**(1), 1–35 (2005)
3. Carmona, J., van Dongen, B.F., Solti, A., Weidlich, M.: Conformance Checking - Relating Processes and Models. Springer, Cham (2018)
4. Compagnucci, I., Corradini, F., Fornari, F., Re, B.: A study on the usage of the BPMN notation for designing process collaboration, choreography, and conversation models. Bus. Inf. Syst. Eng. **66**(1), 43–66 (2024)
5. Decker, G., Weske, M.: Local enforceability in interaction Petri nets. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 305–319. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-75183-0_22
6. Eshuis, R., Hull, R., Sun, Y., Vaculín, R.: Splitting GSM schemas: a framework for outsourcing of declarative artifact systems. Inf. Syst. **46**, 157–187 (2014)
7. Felli, P., Gianola, A., Montali, M., Rivkin, A., Winkler, S.: Data-aware conformance checking with SMT. Inf. Syst. **117** (2023)
8. Felli, P., Montali, M., Winkler, S.: Soundness of data-aware processes with arithmetic conditions. In: Franch, X., Poels, G., Gailly, F., Snoeck, M. (eds.) CAiSE 2022. LNCS, vol. 13295, pp. 389–406. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-07472-1_23
9. Hildebrandt, T., Mukkamala, R.R., Slaats, T.: Safe distribution of declarative processes. In: Barthe, G., Pardo, A., Schneider, G. (eds.) SEFM 2011. LNCS, vol. 7041, pp. 237–252. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-24690-6_17
10. de Leoni, M., Felli, P., Montali, M.: A holistic approach for soundness verification of decision-aware process models. In: Trujillo, J.C., et al. (eds.) ER 2018. LNCS, vol. 11157, pp. 219–235. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-00847-5_17
11. Lichtenstein, T., Weske, M.: Execution semantics for process choreographies with data. In: Di Francescomarino, C., Burattin, A., Janiesch, C., Sadiq, S. (eds.) BPM 2023. LNBIP, vol. 490, pp. 90–106. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-41623-1_6
12. Lohmann, N., Wolf, K.: Realizability is controllability. In: Laneve, C., Su, J. (eds.) WS-FM 2009. LNCS, vol. 6194, pp. 110–127. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14458-5_7
13. Mannhardt, F.: Multi-perspective process mining. Ph.D. thesis, Technical University of Eindhoven (2018)
14. Mannhardt, F., de Leoni, M., Reijers, H., van der Aalst, W.: Balanced multi-perspective checking of process conformance. Computing **98**(4), 407–437 (2016)
15. Rogge-Solti, A., Senderovich, A., Weidlich, M., Mendling, J., Gal, A.: In log and model we trust? A generalized conformance checking framework. In: La Rosa, M., Loos, P., Pastor, O. (eds.) BPM 2016. LNCS, vol. 9850, pp. 179–196. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45348-4_11
16. van Zelst, S.J., Bolt, A., van Dongen, B.F.: Computing alignments of event data and process models. Trans. Petri Nets Other Model. Concurr. **13**, 1–26 (2018)