CITYSPIN

Cyber-Physical Social Systems for City-wide Infrastructures

Deliverable 5.1: Contextually rich business process model language

Authors	:	Claudio Di Ciccio, Alessio Cecconi
Dissemination Level	:	Public
Due date of deliverable	:	31.03.2018 (v1)
Actual submission date	:	April 6, 2018
Work Package	:	5. Process Mining and Monitoring on Linked Data
Type	:	Report
Version	:	1.0

Abstract

This document provides a formal framework that underlies process modelling concepts to be used in the context of WP5. This allows for a clear understanding of the notions used to both discover and present the information treated in the context of the work package. In particular, we focus on declarative process models, which are reportedly the most suitable ones to depict highly flexible and context-dependent processes. We resort on an existing well-established declarative modelling language and extend it so as to make its behavioural rules account for additional perspectives including data, time, and resources, which are crucial for the description of context-dependent constraints of cyber-physical systems. We investigate its syntax, semantics, and properties that its constraints enjoy.

FFG

Project Funded by FFG – IKT der Zukunft Programme

Project Number: 861213 **Start date: 01.10.2017** **Duration: 30 months**

History

Version	Date	Reason	Revisited by
0.1	23.11.2017	Draft for syntax and semantics	CDC
0.2	01.03.2018	First framework draft	CDC
0.3	19.03.2018	Preliminaries completed	CDC, AC
0.4	22.03.2018	Porting on project template	AC
0.5	28.03.2018	Theorems finalised	CDC
0.6	29.03.2018	Proofs finalised, introduction	CDC
0.7	30.03.2018	Research outlook	CDC
0.8	31.03.2018	Executive summary, summary	CDC
0.9	03.04.2018	Cleansing	CDC
1.0	05.04.2018	Proof-reading, finalisation	CDC, AC

Author List

Project Partner	Name(Initial)	Contact Information
WU Wien	Alessio Cecconi (AC)	cecconi@ai.wu.ac.at
WU Wien	Claudio Di Ciccio (CDC)	claudio.di.ciccio@ai.wu.ac.at

Executive Summary

As its first task, WP5 aims at enriching a process modelling language with contextual information to understand the embedding of the processes in their physical and social context. Based on the use cases derived from D3.1 and the requirements elicited in collaboration with the industry partners, we have observed that declarative modelling is the best suitable language to be extended to that end, for two main reasons: *(i)* declarative process models are constraint-based, i.e., based on temporal rules, hence better suitable to represent conditions under which certain behaviour is observed in an if-then manner, and *(ii)* they best depict flexible, semi-structured processes, such as the ones seen in the context of the project.

So far declarative process modelling approaches have been proposed that let the existing well-established language Declare be extended to contain information on control-flow plus data, resources, and time. Such additional perspectives are crucial to encompass the multiple viewpoints of cyber-physical social systems. The objective of this deliverable is thus to study an overarching formal framework that allows for a unifying definition of concepts related to Multi-Perspective Declare (MP-Declare). We begin with the definition of what event logs are, to establish the properties and characteristics of the structure upon which declarative constraints are verified. Thereafter we describe standard, single-perspective Declare. We thus define how to extend it towards the checking of conditions over data, resource, time information at the points in time in which they are activated and exerted. To allow for comparisons of attributes between different points in time, and achieve the full-extent of MP-Declare, we investigate how to represent correlation conditions. The full MP-Declare framework is expressed via linear temporal logics extended with universal quantification over data attributes. We prove that the expansion of Declare with check and correlation conditions is such that monotonicity properties are enjoyed. We conclude the deliverable showing that metrics used to assess the extent to which MP-Declare constraints preserve monotonicity properties, which is essential to provide an approach in future work that discovers such constraints from data in an iterative manner.

The outcome of this deliverable is preparatory for the subsequent task of process discovery on linked-data, which will be reported on in next deliverable D5.2.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Project goal

The goal of the CitySPIN project is to create a platform for cyber-physical social systems in order to facilitate innovative Smart City infrastructure services. The analysis and monitoring of real data stemming from different sources of smart city infrastructures and services is a pillar of the overarching objective of the project.

1.2 Work package goal

The high-level goal of this WP is to enable process mining and process understanding on Linked Data sources that aggregate, enrich, and filter existing raw data. The operations to that extent are meant to be both off-line, in order to carry out process analytics on historical information and provide insights on the existing processes, and on-line, so as to enable adaptive monitoring on facts as they happen within running process instances.

1.3 Relation to other work packages

WP5 cooperates with the other work packages as follows. First and foremost, process mining and adaptive monitoring will be conducted on data stemming from the Scalable Linked Data Platform, designed, realised and implemented in WP4. The tools generated will be integrated and evaluated in the context of WP7. The process analytics and monitoring of real world information will be subject to secure secure/privacy-aware data access (WP6). The showcase of the project outcome including process analytics and monitoring will be based upon use cases and scenarios elicited and illustrated in the context of WP3.

1.4 Deliverable goal

The goal of Deliverable D5.1 is to establish a formal framework that underlies process modelling concepts. This allows for a clear understanding of the notions used to both discover and present the information treated in the context of WP5. In particular, we focus on declarative process models, which are reportedly the most suitable ones to depict highly flexible and context-dependent processes. This framework acts as a conceptual input to the forthcoming deliverable D5.2, in which those notions will be utilised as a foundational ground for the actual discovery of process models from event data.

1.5 Deliverable structure

We begin this deliverable with Section 2, in which the basic notions adopted in declarative process mining are described. Later Section 3 shows an extension to those concepts so as to encompass other perspectives than pure temporal control-flow rules between tasks solely identified by their name. In this way, information on data, resources, and time among others are part of the expressions dictating the behavioural rules of the business process. Properties are proven to be enjoyed by those extended models that guarantee either monotonicity or anti-monotonicity with respect to their expansion to encompass multiple perspectives. Section 4 shows that those properties are preserved by the declarative process discovery metrics support and confidence. Finally, Section 5 draws the plans for future research in the context of this

project based on the current findings, and Section 6 briefly summarises the contribution of this deliverable.

2 Preliminaries

In this section, we describe fundamentals of event logs, declarative process modelling, multi-perspective extensions and automated model discovery.

2.1 Event Logs

Business processes executed via a Business Process Management System (BPMS) typically record process executions into data stores – the so-called *logging* [1]. We name *event* in particular a system-recorded information bit reporting on the execution of an activity (henceforth interchangeably named also task) during the execution of a process. Events are labelled with so-called *event classes*, i.e., task names [2]. We assume that each event uniquely corresponds to the execution of a single task. This assumption builds upon the work on event class reconciliation of Baier et al. [3, 4, 5]. We thus associate every event with its event class, i.e., a symbol representing the task carried out. Additionally, we require events to explicitly refer to the running process instance during which they were registered, i.e., their *case* identifier. Events can convey other information than the task they pertain to, in order to specify the situation and the context in which they occurred, as well as their side-effect on the running process instance. Examples of this additional information are explicit references to the operating resources, or time-stamp. Organisational knowledge can help us to identify resource assignment rules as well as control-flow rules that only apply to certain groups of resources, e.g., to specific roles [6]. Time-stamps are reportedly of use for performance measurement [7, 8]. A finite sequence of events representing the execution of a single process instance from the beginning to the end is named *trace*. We remark here that the concept of sequence entails an assumption on the way events are collected: they are subject to total order within every trace – typically in line with their time-stamping. An *event log* is a collection of traces.

Nomenclature and data structures follow the extensible storage format OpenXES, especially developed for the purpose of storing event data [9]. Recently, with *RelationalXES (RXES)* a relational database schema for storing event log data has been introduced [10]. The RXES schema stores traces in an event log as tables with identifiers. RXES provides a full implementation of OpenXES using the database as a back-end. The database schema used in RXES allows for a significant reduction of redundancy by storing frequently occurring attributes only once rather than repeating them at each occurrence. In this deliverable, we consider an event log to be available in form of the RXES schema and therefore to be stored in a conventional relational database. For the sake of readability we use a denormalised event log table like the one in Table 1. The event log data of Table 1 captures attributes such as the unique event number, the case identifier of the trace, the label of the task corresponding to the event, as well as additional date-formatted, text, and numeric attributes reporting timestamps, resource information, and location, respectively. Based on this underlying data scheme, we need to specify the relevant data fields that should be analysed for declarative constructs.

Definition 1 (Event log)**.** *An* event log *is a tuple* $L(E, \Sigma, C, \ell, \iota, \preccurlyeq, \widehat{\Delta}, \widehat{\alpha})$*, where*
$E \triangleq \{\varepsilon_1, \ldots, \varepsilon_{|E|}\}$ *is a finite set of constant symbols named* events*;*
$\Sigma \triangleq \{a_1, \ldots, a_{|\Sigma|}\}$ *is a finite non-empty set of constant symbols named* activities*;*
$C \triangleq \{\rho_1, \ldots, \rho_{|C|}\}$ *is a finite set of constant symbols named* cases*;*

Table 1: Event log excerpt stored in a denormalised relational database table

Event	Case	Task	Time	Clerk	Location	Score
1	*1*	*a*	2015-11-06 15:31:03	John	Lab 4	12
2	*1*	*b*	2015-11-06 15:35:12	John	Lab 4	12
3	*1*	*c*	2015-11-06 15:37:22	Jane	Office 2	60
4	*2*	*b*	2015-11-06 16:22:45	Judith	Lab 4	90
5	*2*	*c*	2015-11-06 16:45:12	Judith	Lab 4	100
6	*2*	*d*	2015-11-07 09:00:01	Jane	Office 2	100
…	…	…	…	…	…	

$\ell : E \to \Sigma$ *is the* labelling function *connecting events to activities;*
$\iota : E \to C$ *is the surjective* instance function *associating events to cases;*
$\preccurlyeq \subseteq E \times E$ *is the reflexive* total order *relation over events; we shall denote as* $\prec$ *the corresponding strict total order, excluding from* $\preccurlyeq$ *the* $(\varepsilon, \varepsilon)$ *pairs;*
$\widehat{\Delta} \triangleq \{\Delta_1, \ldots, \Delta_{n+m}\} \triangleq \widehat{\Delta_\leqslant} \cup \widehat{\Delta_=}$ *is the* universe of interpretation, *consisting of* $\widehat{\Delta_\leqslant} \triangleq \{\Delta_{\leqslant_1}, \ldots, \Delta_{\leqslant_n}\}$, *the set of* ordered (interpretation) domains, *and* $\widehat{\Delta_=} \triangleq \{\Delta_{=_1}, \ldots, \Delta_{=_m}\}$, *the set of* unordered (interpretation) domains, *with* $m, n \in \mathbb{N}$ *and* $\Delta_\leqslant \cap \Delta_= = \varnothing$. *For* $\widehat{\Delta_\leqslant}$ *a total order relation* $\leqslant$ *is defined. For* $\widehat{\Delta_=}$ *only an equivalence relation* $=$ *is defined instead.*
$\widehat{\alpha} \triangleq \widehat{\alpha_\leqslant} \cup \widehat{\alpha_=} \triangleq \{\alpha_1, \ldots, \alpha_{n+m}\}$ *is the* universe of attribute functions, *consisting of* $\widehat{\alpha_\leqslant} \triangleq \{\alpha_{\leqslant_1}, \ldots, \alpha_{\leqslant_n}\}$ *where* $\alpha_{\leqslant_k} : E \to \Delta_{\leqslant_k} \cup \{\bot\}$ *for* $1 \leqslant k \leqslant n$ *and* $\widehat{\alpha_=} \triangleq \{\alpha_{=_1}, \ldots, \alpha_{=_m}\}$ *where* $\alpha_{=_l} : E \to \Delta_{=_l} \cup \{\bot\}$ *for* $1 \leqslant l \leqslant m$, *associating events to* attribute values *that are of ordered and unordered domains, respectively; the special symbol* $\bot$ *denotes absence of knowledge about the attribute value.*

Example 1. *In the event log of Table 1, every event is reported as a line of the table and identified by a unique, increasing number (1, 2, 3, 4, 5, . . .). The total order* $\leqslant$ *is implicitly represented by the increasing associated number. To ease the readability, we henceforth denote the example events as* $\varepsilon 1$, $\varepsilon 2$, $\varepsilon 3$, *etc. In the example, e.g.,* $\varepsilon 1 \prec \varepsilon 2$, $\varepsilon 1 \preccurlyeq \varepsilon 2$, $\varepsilon 2 \preccurlyeq \varepsilon 2$, $\varepsilon 1 \preccurlyeq \varepsilon 3$, $\varepsilon 1 \prec \varepsilon 3$. *In Table 1, also cases are assigned with a unique, increasing number (1, 2, . . .). For the sake of readability, we shall denote them as* $\rho 1$, $\rho 2$, *etc. In the example, events* $\varepsilon 1$, $\varepsilon 2$, *and* $\varepsilon 3$, *are mapped to case* $\rho 1$, *i.e.,* $\iota(\varepsilon 1) = \iota(\varepsilon 2) = \iota(\varepsilon 3) = \rho 1$, *whilst* $\iota(\varepsilon 4) = \iota(\varepsilon 5) = \iota(\varepsilon 6) = \rho 2$. *The activity set of the example is* $\Sigma = \{a, b, c, d, \ldots\}$. *We observe that events* $\varepsilon 2$ *and* $\varepsilon 4$ *are both associated to task b, i.e.,* $\ell(\varepsilon 2) = \ell(\varepsilon 4) = b$, *whereas* $\ell(\varepsilon 1) = a$. *An ordered interpretation domain is the one of time-stamps. Event* $\varepsilon 1$ *is associated to its time-stamp ("2015-11-06 15:31:03") by an attribute function, represented in Table 1 under the* `Time` *column. Rooms given as location ("Lab 4", "Office 2", . . .) are elements of an unordered interpretation domain, that is the one of premises of the organisation running the process. Event* $\varepsilon 1$ *is associated to "Lab 4" by an attribute function, represented in Table 1 under the* `Location` *column.*

In the following, all those events that are labelled as a will be referred to as $\varepsilon(a)$, i.e., $\varepsilon(a) = \{\varepsilon \in E : \ell(\varepsilon) = a\}$. Considering the example of Table 1, $\varepsilon(a)$ is the event identified by $\varepsilon 1$, i.e., $\varepsilon(a) = \{\varepsilon 1\}$, whereas $\varepsilon(b) = \{\varepsilon 2, \varepsilon 4\}$. Furthermore, we will interchangeably use a post-fix dot-separated notation for functions over events, so $\iota(\varepsilon) \doteq \varepsilon.\iota$, $\ell(\varepsilon) \doteq \varepsilon.\ell$, and $\alpha(\varepsilon) \doteq \varepsilon.\alpha$ for all $\alpha \in \widehat{\Delta}$. For instance, we shall write $\varepsilon 1.\texttt{Clerk} =$ "John".

Definition 2 (Trace). *Given an* event log $L = (E, \Sigma, C, \ell, \iota, \preccurlyeq, \widehat{\Delta}, \widehat{\alpha})$, ι *defines an equivalence relation partitioning* E *in equivalence classes* $[\rho] \triangleq \{\varepsilon \mid \iota(\varepsilon) = \rho\}$ *for every* $\rho \in C$. *Given* L *and an equivalence class* $[\rho]$, *a* trace $\tau \in T$ *is the restriction of* L *on events of* $[\rho]$, *i.e., a tuple*

$\tau \triangleq (E_\tau, \Sigma_\tau, C_\tau, \ell_\tau, \iota_\tau, \preccurlyeq_\tau, \widehat{\Delta}_\tau, \widehat{\alpha}_\tau)$ *where*

$$E_\tau \triangleq [\rho], \quad \Sigma_\tau \triangleq \Sigma \cap \{\ell(\varepsilon) | \varepsilon \in E_\tau\}, \quad C_\tau \triangleq \{\rho\},$$
$$\ell_\tau \triangleq \ell \cap (E_\tau \times \Sigma_\tau), \quad \iota_\tau \triangleq \iota \cap (E_\tau \times C_\tau), \quad \preccurlyeq_\tau \triangleq \preccurlyeq \cap (E_\tau \times E_\tau),$$
$$\widehat{\Delta}_\tau \triangleq \widehat{\Delta} \cap \bigcup_{\alpha \in \widehat{\alpha}} \{\alpha(\varepsilon) | \varepsilon \in E_\tau\},$$
and
$$\widehat{\alpha}_\tau \triangleq \widehat{\alpha} \cap (E_\tau \times \widehat{\Delta}_\tau).$$

Example 2. *In the event log of Table 1 the partition of the log into traces is signalled by a separating line. Traces τ_1 and τ_2 are associated to cases $\rho 1$ and $\rho 2$, respectively.*

Applying an order- and label-preserving isomorphism from traces to sequences of activities, we obtain *string-traces*, henceforth denoted as σ. With a slight overload of nomenclature we shall name string-traces as simply traces when clear from the context.

Example 3. *From the log of Example 1 we obtain (string-)traces $\sigma_1 = \langle a, b, c \rangle$ and $\sigma_2 = \langle b, c, d \rangle$. Additional example traces are $\sigma_3 = \langle a, a, b, c \rangle$, $\sigma_4 = \langle b, b, c, d \rangle$, $\sigma_5 = \langle a, b, c, b \rangle$, and $\sigma_6 = \langle a, b, a, c \rangle$.*

2.2 Declarative Process Modelling

Declarative models represent processes by restrictions over the permissible behaviour. The restricting rules are named *constraints*, which express those conditions that must be satisfied throughout the process execution. Modelling languages like Declare [11] provide a standard *repertoire* of *templates*, i.e., constraints parametrised over activities. Table 2 lists the constraint templates that will be considered in this deliverable, along with their activations and targets. Without loss of generality, we extend the notion of activation and target to the tasks assigned to those parameters. Typical examples of Declare constraints are, e.g., Existence1(a) and Response(b, c). The former applies the Existence1 existence template on task a (target), and states that a must occur in every trace of the process. The latter applies the Response relation template on tasks b (activation) and c (target), imposing that if b occurs, then c must occur later on in the same trace. Intuitively, activations determine the circumstances triggering the constraint. Targets are the consequential conditions being imposed upon the occurrence of the activations. Table 2 lists activations and targets for every considered constraint template. In this article, we analyse constraints of arity not higher than 2, because they constitute the subset of constraints discovered by the majority of declarative process miners [12, 13, 14]. Nevertheless, the presented approach can be seamlessly extended to the case of constraints of higher arity.

2.2.1 Linear Temporal Logic over finite traces

Because constraints are meant to exert conditions over subsequent or precedent statuses reached at different discrete stages of the process execution, Declare semantics are formalised using (propositional) Linear Temporal Logic over finite traces (LTL$_f$) [15, 16, 17, 18], as shown in Table 2. Formulae of LTL$_f$ ψ_1 and ψ_2 are built from a set of propositional symbols and closed under boolean connectives and *temporal modalities*. $\mathbf{F}$, $\mathbf{X}$, $\mathbf{G}$, and $\mathbf{U}$ are modalities with the following meanings: formula $\mathbf{F}\,\psi_1$ means that ψ_1 holds some time in the future, $\mathbf{X}\,\psi_1$ means that ψ_1 holds in the next position, $\mathbf{G}\,\psi_1$ says that ψ_1 holds forever in the future, and lastly, $\psi_1\,\mathbf{U}\,\psi_2$ means that some time in the future ψ_2 will hold and until that moment ψ_1 holds. Dual past counterparts are $\mathbf{O}$, $\mathbf{Y}$ and $\mathbf{S}$, where $\mathbf{O}\,\psi_1$ means that ψ_1 holds some time in the past, $\mathbf{Y}\,\psi_1$ means that ψ_1 holds in the previous instant, and $\psi_1\,\mathbf{S}\,\psi_2$ means that ψ_1 has held some time in the past and since that moment ψ_2 holds.

Table 2: Semantics for Declare *templates in LTL$_f$.*

Template	LTL$_f$ semantics	Activation	Target
Existence1(x)	$\mathbf{F}(x)$	–	x
RespondedExistence(x, y)	$\mathbf{G}(x \rightarrow (\mathbf{O}\, y \vee \mathbf{F}\, y))$	x	y
Response(x, y)	$\mathbf{G}(x \rightarrow \mathbf{F}\, y)$	x	y
AlternateResponse(x, y)	$\mathbf{G}(x \rightarrow \mathbf{X}(\neg x\, \mathbf{U}\, y))$	x	y
ChainResponse(x, y)	$\mathbf{G}(x \rightarrow \mathbf{X}\, y)$	x	y
Precedence(x, y)	$\mathbf{G}(y \rightarrow \mathbf{O}\, x)$	y	x
AlternatePrecedence(x, y)	$\mathbf{G}(y \rightarrow \mathbf{Y}(\neg y\, \mathbf{S}\, x))$	y	x
ChainPrecedence(x, y)	$\mathbf{G}(y \rightarrow \mathbf{Y}\, x)$	y	x
NotRespondedExistence(x, y)	$\mathbf{G}(x \rightarrow \neg(\mathbf{O}\, y \vee \mathbf{F}\, y))$	x	y
NotResponse(x, y)	$\mathbf{G}(x \rightarrow \neg\, \mathbf{F}\, y)$	x	y
NotPrecedence(x, y)	$\mathbf{G}(y \rightarrow \neg\, \mathbf{O}\, x)$	y	x
NotChainResponse(x, y)	$\mathbf{G}(x \rightarrow \neg\, \mathbf{X}\, y)$	x	y
NotChainPrecedence(x, y)	$\mathbf{G}(y \rightarrow \neg\, \mathbf{Y}\, x)$	y	x

Definition 3 (Syntax of LTL$_f$). *An LTL$_f$ formula ψ is inductively defined in Backus Naur Form (BNF) as:*

$$\psi ::= \Phi \mid \neg\psi \mid \psi \wedge \psi' \mid \psi \vee \psi' \mid \psi \rightarrow \psi' \mid \mathbf{X}\, \psi \mid \mathbf{F}\, \psi \mid \psi\, \mathbf{U}\, \psi' \mid \mathbf{Y}\, \psi \mid \mathbf{O}\, \psi \mid \psi\, \mathbf{S}\, \psi' \mid \mathbf{G}\, \psi$$

where Φ is a propositional formula over traces, $\mathbf{X}$ is the next *operator, $\mathbf{F}$ is* eventually, *$\mathbf{U}$ is* until, *$\mathbf{O}$ is* once, *$\mathbf{Y}$ is* previously, *$\mathbf{S}$ is* since, *$\mathbf{G}$ is* globally.

Definition 4 (Semantics of LTL$_f$). *Let relation $\prec_1$ denote the immediate antecedence between two events $\varepsilon, \varepsilon' \in E_\tau$: $\varepsilon \prec_1 \varepsilon'$ if and only if (i) $\varepsilon \neq \varepsilon'$, (ii) $\varepsilon \prec \varepsilon'$, and (iii) there exists no ε'' s.t. $\varepsilon \neq \varepsilon''$ and $\varepsilon \prec \varepsilon''$. Semantics of LTL$_f$ is defined by the satisfaction relation $\models$, inductively defined over the structure of the formula, given a trace τ and an event $\varepsilon \in E_\tau$, as follows:*

1) $\tau, \varepsilon \models \Phi$ if ε satisfies Φ (in terms of propositional entailment);
2) $\tau, \varepsilon \models \neg\psi$ if it is not true that $\tau, \varepsilon \models \psi$;
3) $\tau, \varepsilon \models \psi_1 \wedge \psi_2$ if $\tau, \varepsilon \models \psi_1$, and $\tau, \varepsilon \models \psi_2$;
4) $\tau, \varepsilon \models \mathbf{X}\, \psi$ if there exists $\varepsilon' \in \tau$ s.t. $\varepsilon \prec_1 \varepsilon'$ and $\tau, \varepsilon' \models \psi$;
5) $\tau, \varepsilon \models \mathbf{Y}\, \psi$ if there exists $\varepsilon' \in \tau$ s.t. $\varepsilon' \prec_1 \varepsilon$ and $\tau, \varepsilon' \models \psi$;
6) $\tau, \varepsilon \models \psi_1\, \mathbf{U}\, \psi_2$ if there exists $\varepsilon' \in \tau$ such that (i) $\varepsilon \preccurlyeq \varepsilon'$, (ii) $\tau, \varepsilon' \models \psi_2$, and (iii) for every $\varepsilon'' \in \tau$ such that $\varepsilon \preccurlyeq \varepsilon'' \prec \varepsilon'$, we have $\tau, \varepsilon'' \models \psi_1$;
7) $\tau, \varepsilon \models \psi_1\, \mathbf{S}\, \psi_2$ if there exists $\varepsilon' \in \tau$ such that (i) $\varepsilon' \preccurlyeq \varepsilon$, (ii) $\tau, \varepsilon' \models \psi_2$, and (iii) for every $\varepsilon'' \in \tau$ such that $\varepsilon \preccurlyeq \varepsilon'' \prec \varepsilon'$, we have $\tau, \varepsilon'' \models \psi_1$. (iv) for every $\varepsilon'' \in \tau$ such that $\varepsilon' \prec \varepsilon'' \preccurlyeq \varepsilon$, we have $\tau, \varepsilon'' \models \psi_1$.

The other operators are obtained as combinations of the ones defined above [17]. Their semantics can be derived by recalling that: $\psi_1 \vee \psi_2 \equiv \neg(\neg\psi_1 \wedge \neg\psi_2)$; $\psi_1 \rightarrow \psi_2 \equiv \neg\psi_1 \vee \psi_2$; $\mathbf{F}\, \psi \equiv \top\, \mathbf{U}\, \psi$; $\mathbf{O}\, \psi \equiv \top\, \mathbf{S}\, \psi$; $\mathbf{G}\, \psi \equiv \neg\, \mathbf{F}\, \neg\psi$.

Definition 5. *Given a set of propositional terms $\widehat{\Phi}$, a* restricted *LTL$_f$ formula ψ over $\widehat{\Phi}$, denoted as $\psi[\widehat{\Phi}]$, is an LTL$_f$ formula defined as in Defs. 3 and 4 whose every propositional term Φ is s.t. $\Phi \in \widehat{\Phi}$.*

As clarified in [19], temporal modalities enjoy the property of *monotonicity* [20]. In the context of logics, monotonicity guarantees that if a formula ψ entails ψ', then the models of ψ are also models of ψ' ($\psi \models \psi'$). On the contrary, antimonotonicity guarantees that $\psi \dashv \psi'$ under the same conditions. Given formulae ψ and ψ', then:

1. a unary operator $\circledcirc$ is monotonic iff $\circledcirc\psi \models \circledcirc\psi'$;
2. a unary operator $\circledcirc$ is antimonotonic iff $\circledcirc\psi \dashv \circledcirc\psi'$;
3. a binary operator $\oplus$ is monotonic iff $\psi \models \psi \oplus \psi'$;
4. a binary operator $\oplus$ is antimonotonic iff $\psi \dashv \psi \oplus \psi'$.

Monotonicity holds not only for temporal modalities but also for $\vee$, $\wedge$ and $\rightarrow$, whereas $\neg$ is antimonotonic.

2.2.2 Standard Declare

In standard Declare propositional formulae Φ are literals solely referring to the label of events, say $\Phi = a$. We shall therefore consider also string-traces as an alternative input to traces for the verification of standard Declare constraints. Consider, e.g., the Response constraint $\mathbf{G}(a \rightarrow \mathbf{F}\, b)$ and string-traces $\sigma_1, \ldots, \sigma_6$ of Example 3. The constraint indicates that whenever an event ε_1 labelled as *a occurs*, an event ε_2 labelled as *b* must eventually *follow*. It is satisfied by all events of σ_1, σ_2, σ_3, σ_4, and σ_5, but not by σ_6 because the second occurrence of *a* is not followed by any occurrence of *b*.

Definition 6 (Declarative Template). *A* declarative template *is a predicate* $\Xi_{/n} \in \widehat{\Xi}$ *where* n *denotes its* arity, *i.e., its number of* parameters *[21, 22]. When clear from the context, subscript "*$_{/n}$*" shall be omitted. We indicate the parameters of a declarative template as* $par(\Xi)$. *Semantics of a template* $\Xi_{/n}$ *is expressed by an LTL*$_f$ *formula* $\psi\lfloor X \rfloor$ *restricted to parametric propositional statements* $X = par(\Xi)$. *To explicitly refer to its parameters we shall also denote a template as* $\Xi(x, y, \ldots)$. *A template of arity* 1 *is an* existence template. *A template of arity* 2 *or more is a* relation template. *The semantics follow Def. 4, with the following specialisation of rule 1:*

1. $\tau, \varepsilon \models x$ *if it is true that* $\ell(\varepsilon) = x$.

Definition 7 (Declare Repertoire, Activation, Target). *A finite set of declarative templates constitutes a* repertoire *such as the one of* Declare *in Table 2. Let* $\widehat{\Xi}$ *denote the* repertoire *of* Declare. *Parameters of* Declare *templates belong to either the category of* activation, $par_{\bullet}(\Xi)$, *or* target, $par_{\Rightarrow}(\Xi)$: $par(\Xi) = par_{\bullet}(\Xi) \cup par_{\Rightarrow}(\Xi)$, *with* $par_{\bullet}(\Xi) \cap par_{\Rightarrow}(\Xi) = \varnothing$. *A parameter* x *is an activation for template* Ξ, *i.e.,* $x \in par_{\bullet}(\Xi)$, *if and only if the following holds:* $\mathbf{G}((\neg x) \rightarrow \Xi(x, \ldots)) \equiv \mathbf{true}$.

Notice that we characterise activations by the principle of *ex falso quod libet*: if an activation occurs, there can be violations or not. If the activation does not occur, the constraint is certainly satisfied – the so-called vacuous satisfaction [23]. In standard Declare any parameter that is not an activation, is a target, as shown in Table 2.[1]

In standard Declare, we interpret parameters as activities. A standard Declare constraint in particular is an assignment of a template under this interpretation, thus binding every parameter to a specific activity.

Definition 8 (Standard Declare constraint). *Let* β_Σ *be a parameter assignment for a declarative template* Ξ, *i.e., a function* $\beta_\Sigma : par(\Xi) \rightarrow \Sigma$. *A* Declare *constraint* $\mathcal{C}$ *is a* Declare *template* $\Xi_{/n}$ *of arity* n *whose parameters are assigned by* $\mathscr{A}_\Sigma$: $\mathcal{C} \triangleq \Xi(\mathscr{A}_\Sigma(x_1), \ldots, \mathscr{A}_\Sigma(x_n))$.

Declare constraints exert conditions on the execution of (so-called) *target* tasks, when *activation* tasks occur. For example, a is the activation task of the Response(a, b) constraint

[1] Some Declare relation templates are such that both parameters are activation and target at the same time, e.g., Succession(x, y). They are named *mutual* [14]. However, they can be reformulated as a pair of templates where the activation and target are separated, e.g., Succession(x, y) is the conjunction of Response(x, y) and Precedence(x, y) [24].

$\mathbf{G}(a \rightarrow \mathbf{F}\, b)$ and b is the target, because the execution of a forces b to be eventually executed, whilst if no a occurs, then Response(a, b) is fulfilled. An activation of a constraint leads to either a *fulfilment* or a *violation*. Consider again $\mathbf{G}(a \rightarrow \mathbf{F}\, b)$. In σ_1, the constraint is activated and fulfilled twice, whereas in σ_3 it is activated and fulfilled only once. In σ_4 it is activated twice and the second activation leads to a violation (b does not occur subsequently).

3 Multi-Perspective Declare

A central shortcoming of languages like Declare is the fact that templates are not directly capable of expressing the connection between the behaviour and other perspectives of the process. Consider the example of a loan application process. The process analyst would be interested in learning about constraints such as the following: **Activation conditions:** When a loan was requested and *account balance* $>$ 14 500 euros, the loan was subsequently granted in 95% of the cases; **Correlation conditions:** When a loan was requested, the loan was subsequently granted and *amount requested* $=$ *amount granted* in 95% of the cases; **Target conditions:** When a loan was requested, the loan was subsequently granted in 95% of the cases by a specific member of the financial board; **Temporal conditions:** When a loan was requested, the loan was subsequently granted *within the next 30 days* in 95% of the cases.

The importance of complex constraints that integrate activation, correlation, target and temporal dependencies has been emphasised by prior research and has led to the definition of a multi-perspective version of Declare [25]. In the remainder of the deliverable, we will refer to standard Declare as *Single-Perspective* Declare (*SP*-Declare) to distinguish it from MP-Declare.

To move from SP-Declare to MP-Declare, we detach the assignment of template parameters from the sole task labels, i.e., we enrich SP-Declare constraints with *conditions* over other perspectives than the pure control flow. We thus introduce the following comparison formulae: check formulae and correlation formulae. We shall consider at first check formulae only and then extend the discussion to correlation formulae.

3.1 Attribute check for MP-Declare

We introduce the notion of check atoms to cater for expressions such as $\texttt{Time} < $ 2015-11-06 17:00:00 or $\texttt{Clerk} =$ Judith, referring to the example of Example 1. Check atoms are inherently single propositional terms. They indeed consist of comparisons with constant symbols, which can be reduced to propositional statements provided that we consider the valuation of LTL$_f$ sentences over the active domain of the database in which the event log is stored, in other words, assuming traces to be Herbrand interpretations of declarative constraints [26].

The concept of check formulae is provided to conjunct check atoms and propositional symbols denoting an activity, e.g., $b \wedge \texttt{Time} <$ 2015-11-06 17:00:00 $\wedge\ \texttt{Clerk} =$ Judith. A new pre-interpretation shall serve the purpose of associating parameters of templates not directly to activities, but to propositional formulae like the aforementioned ones, following the rationale of multi-valued model checking [27, 28]. Likewise, the new assignment shall bind parameters not only to activities, but also to attributes and elements of the domain. The interpretation of the new formulae, i.e., the MP-Declare templates, will thus change accordingly, so as to take into account the broader structure over which predicates have to be evaluated.

Formally, we define as *check atom* a predicate $\varsigma(\alpha, \delta)$ where arguments are meant to be

$\alpha \in \widehat{\alpha}$ and $\delta \in \widehat{\Delta}$ as per Def. 1 in the following form:

$$\varsigma ::= \alpha \circledast \delta; \qquad \circledast ::= \ominus \mid \otimes; \qquad \ominus ::= \neq \mid =; \qquad \otimes ::= \leqslant \mid \geqslant \mid < \mid > \mid \ominus. \tag{1}$$

Notice that for `Time`-attribute values an ordering relation is defined, whereas it is not the case for `Clerk`. It is, e.g., undesirable to have formulations like `Clerk` $<$ Judith. Therefore, we introduce the notion of *well-defined check atom*, intended to ensure that comparable elements are confronted. A *well-defined check atom* is such that its production rule $\circledast ::= \ominus \mid \otimes$, $\circledast$ is expressed as $\otimes$ only if $\alpha \in \widehat{\alpha_{\leqslant}} \subseteq \widehat{\alpha}$. In the following we assume that used check atoms are well-defined.
Thereupon, we define the *check formula* as follows:

$$\Phi_{\varsigma} ::= a \mid \Phi_{\varsigma} \wedge \varsigma. \tag{2}$$

In the following we name *check expansion* the production of the non-terminal rewriting rule, i.e., $\Phi'_{\varsigma} \leftarrow \Phi_{\varsigma} \wedge \varsigma$.
We denote the language of check formulae Φ_{ς} as $\mathscr{L}_{\varsigma}$. Taking inspiration from the formalisation introduced in [22], we define the pre-interpretation of MP-Declare as follows.

Definition 9 (MP-Declare check pre-interpretation). *Let $\Xi \in \widehat{\Xi}$ be a declarative template in the* Declare *repertoire as per Defs. 6 and 7. An* MP-Declare check pre-interpretation *is a function $\mathscr{I}_{\varsigma} : par(\Xi) \rightarrow \mathscr{L}_{\varsigma}$. We shall denote a template Ξ pre-interpreted by $\mathscr{I}_{\varsigma}$ as ${}_{\mathscr{I}_{\varsigma}}\Xi$.*

In light of the above definition, given the Declare template Response(x, y), its activation can be pre-interpreted as, e.g., $a1 \wedge \alpha 1 < \delta 1 \wedge \alpha 2 = \delta 2$. An assigned activation can be: $a \wedge$ `Time` $<$ 2015-11-06 17:00:00 $\wedge$ `Clerk` $=$ Judith. Its target can be pre-interpreted as, e.g., $a2 \wedge \alpha 3 = \delta 3$, so as to be later assigned with $b \wedge$ `Location` $=$ Lab 4.

3.1.1 Property of monotonicity

Check expansion enjoys the property of monotonicity. We will see however that it leads to different results when applied to the activation or the target of templates. We begin the discussion with the following.

Lemma 1 (Monotonicity of check expansion). *Let ς be a check atom, Φ_{ς} and $\Phi'_{\varsigma} \doteq \Phi_{\varsigma} \wedge \varsigma$ be check formulae. It follows that $\Phi'_{\varsigma} \models \Phi_{\varsigma}$.*

Lemma 1 is a direct consequence of the monotonicity of $\wedge$. Notice that indeed in the log of Example 1 both $\varepsilon 2$ and $\varepsilon 4$ satisfy `Time` $<$ 2015-11-06 17:00:00, whilst only $\varepsilon 2$ satisfies `Time` $<$ 2015-11-06 17:00:00 $\wedge$ `Clerk` $=$ Judith.

Theorem 1 (Anti-monotonicity of check expansion over activation). *Let $\Xi \in \widehat{\Xi}$ be a* Declare *template and $x \in par_{\bullet}(\Xi)$ its activation. Let $\mathscr{I}_{\varsigma}$ and $\mathscr{I}'_{\varsigma}$ be check pre-interpretations such that $\Phi_{\varsigma} \doteq \mathscr{I}_{\varsigma}(x)$, $\Phi'_{\varsigma} \doteq \mathscr{I}'_{\varsigma}(x) \doteq \Phi_{\varsigma} \wedge \varsigma$, where ς is a check atom. and $\mathscr{I}'_{\varsigma}(y) = \mathscr{I}'_{\varsigma}(y)$ for every $y \in par(\Xi) \backslash \{x\}$. It follows that ${}_{\mathscr{I}_{\varsigma}}\Xi \models {}_{\mathscr{I}'_{\varsigma}}\Xi$.*

Proof (sketch). Let us consider the RespondedExistence template (henceforth Res.Ex for short), namely $\mathbf{G}(x \rightarrow (\mathbf{O}\, y \vee \mathbf{F}\, y))$. Under pre-interpretation $\mathscr{I}_{\varsigma}$ we have that ${}_{\mathscr{I}_{\varsigma}}$Res.Ex $\doteq \mathbf{G}(\neg \Phi_{\varsigma} \vee \mathbf{O}\, \mathscr{I}_{\varsigma}(y) \vee \mathbf{F}\, \mathscr{I}_{\varsigma}(y))$. Under pre-interpretation $\mathscr{I}'_{\varsigma}$ we have that ${}_{\mathscr{I}'_{\varsigma}}$Res.Ex $\doteq \mathbf{G}(\neg \Phi'_{\varsigma} \vee \mathbf{O}\, \mathscr{I}_{\varsigma}(y) \vee \mathbf{F}\, \mathscr{I}_{\varsigma}(y))$ because by hypothesis $\mathscr{I}'_{\varsigma}(y) = \mathscr{I}_{\varsigma}(y)$. Thus ${}_{\mathscr{I}_{\varsigma}}$Res.Ex $\models$ ${}_{\mathscr{I}'_{\varsigma}}$Res.Ex by monotonicity of $\mathbf{G}$, $\mathbf{O}$, $\mathbf{F}$, $\vee$, and anti-monotonicity of $\neg$. □

The same proof scheme can be applied on the other Declare templates in Table 2, with the exception of Existence1 as it has no activation. Indeed, in all such templates' formulae the activation lies under $\mathbf{G}$ and $\neg$, which are respectively monotonic and anti-monotonic. Intuitively, by Theorem 1 it follows that events leading to satisfaction cannot decrease if additional check formulae are concatenated to an activation. For example, $\mathbf{G}((b \wedge \texttt{Time} < \text{2015-11-06 17:00:00}) \rightarrow \mathbf{F}(d))$, an MP-Declare constraint stemming from the Response template, is satisfied by the events of trace τ_2, i.e., $\varepsilon 4$, $\varepsilon 5$, and $\varepsilon 6$. $\mathbf{G}((b \wedge \texttt{Time} < \text{2015-11-06 17:00:00} \wedge \texttt{Clerk} = \text{Judith}) \rightarrow \mathbf{F}(d))$ is instead satisfied by the events of both τ_1 and τ_2, because no activity is carried out by Judith in τ_1.

Theorem 2 (Monotonicity of check expansion over non-negated target)**.** *Let $\Xi \in \widehat{\Xi}$ be a* Declare *template and $y \in par_{\Rightarrow}(\Xi)$ its target under an even number of negations. Let $\mathscr{I}_\varsigma$ and $\mathscr{I}'_\varsigma$ be check pre-interpretations such that $\Phi_\varsigma \doteq \mathscr{I}_\varsigma(y)$, $\Phi'_\varsigma \doteq \mathscr{I}'_\varsigma(y) \doteq \Phi_\varsigma \wedge \varsigma$, where ς is a check atom. and $\mathscr{I}'_\varsigma(x) = \mathscr{I}'_\varsigma(x)$ for every $x \in par(\Xi)\backslash\{y\}$. It follows that ${}_{\mathscr{I}'_\varsigma}\Xi \models {}_{\mathscr{I}_\varsigma}\Xi$.*

Proof (sketch). Let us consider again Res.Ex $\doteq \mathbf{G}(x \rightarrow (\mathbf{O}\, y \vee \mathbf{F}\, y))$. Under pre-interpretation $\mathscr{I}_\varsigma$ we have that ${}_{\mathscr{I}_\varsigma}$Res.Ex $\doteq \mathbf{G}(\neg\mathscr{I}_\varsigma(x) \vee \mathbf{O}\,\Phi_\varsigma \vee \mathbf{F}\,\Phi_\varsigma)$. Under pre-interpretation $\mathscr{I}'_\varsigma$ we have that ${}_{\mathscr{I}'_\varsigma}$Res.Ex $\doteq \mathbf{G}(\neg\mathscr{I}_\varsigma(x) \vee \mathbf{O}\,\Phi'_\varsigma \vee \mathbf{F}\,\Phi'_\varsigma)$ because by hypothesis $\mathscr{I}'_\varsigma(x) = \mathscr{I}_\varsigma(x)$. Therefore ${}_{\mathscr{I}'_\varsigma}$Res.Ex $\models {}_{\mathscr{I}_\varsigma}$Res.Ex by monotonicity of $\mathbf{G}$, $\mathbf{O}$, $\mathbf{F}$, and $\vee$. □

The same proof scheme can be applied on the other Declare templates, provided that the hypothesis of even negations on y applies. For example, $\mathbf{G}(b \rightarrow \mathbf{F}(c \wedge \texttt{Location} = \text{Office 2}))$, stemming from the Response template again, is satisfied by the events of both τ_1 and τ_2. $\mathbf{G}(b \rightarrow \mathbf{F}(c \wedge \texttt{Location} = \text{Office 2} \wedge \texttt{Clerk} = \text{Jane}))$, is instead satisfied by events of τ_1 only.

Corollary 2.1. *If $y \in par_{\Rightarrow}(\Xi)$ is under an odd number of negations, check expansion is anti-monotonic.*

The proof can proceed analogously to Theorem 2, yet considering that the target lies under non-monotonic operator $\neg$. Theorem 2 intuitively entails that events leading to satisfaction cannot increase if additional check formulae are concatenated to a target, which is the opposite consequence of concatenating check formulae to activations. Notice that for templates negating targets, such as NotRespondedExistence, the opposite holds (Corollary 2.1): the more the check formulae on the target, the higher the number of events supporting them.

Intuitively, we see that constraints of these sorts are Declare templates assigned with activities for labels and linking attributes to domain values. We remark here that semantics can still be expressed in LTL$_f$, and more precisely through an LTL$_f$ formula $\psi\lfloor\Phi_\varsigma\rfloor$ restricted to parametric propositional statements Φ_ς – cf. Eq. (2).

3.2 Full extent of MP-Declare

The level of expressiveness discussed so far is limited by the fact that no comparison between attributes of two events can be made, as demonstrated by Abiteboul et al. [29]. For instance, with pre-interpreted templates like the ones of Def. 9 we cannot express, e.g., that if a is executed, then b will be carried out eventually afterwards by a different resource than the one doing a. The values of the compared attributes would be variables in fact, hence going beyond the expressiveness of propositional languages. To cater for it, we need to extend the adopted logic to a fragment of First-Order Linear-Temporal Logic over finite traces (LTL-FO$_f$) [15, 30, 31, 32], as follows: $\forall\nu.\, \mathbf{G}((a \wedge \texttt{Clerk} = \nu) \rightarrow \mathbf{F}(b \wedge \texttt{Clerk} \neq \nu))$. Notice that the bounded variable ν serves to express any clerk that carried out a, to be different from the clerk executing b. We also remark that in this research we do not consider the full extent of LTL-FO$_f$

as universal quantifiers $\forall$ are used solely over globally-scoped variables, i.e., variables which are assigned with a single value that holds globally over all events [15]. To include such statements, we shall introduce the notions of *(i) correlation atoms*, revisiting the grammar of check atoms to include variables ν, *(ii) correlation condition*, joining under conjunction multiple correlation atoms, and finally *(iii)* a *rewriting rule* to achieve the full MP-Declare pre-interpretation $\mathscr{I}$. Pre-interpretations sort out formaulae in a subset of LTL-FO$_f$ that can then be assigned with tasks, attributes and domain constants, so as to be finally evaluated over traces of event logs. We remark that correlation conditions can be expressed only in the presence of defined activation and target, such as in the case of Response and unlike AtMostOne, e.g.

A *correlation atom* is a predicate $\zeta(\alpha, \nu)$ with $\alpha \in \widehat{\alpha}$ and ν a free variable in the following form:

$$\zeta ::= \alpha \circledast \nu; \qquad \circledast \text{ as in (1).} \tag{3}$$

Examples of assigned correlation atoms are $\texttt{Time} > \nu$ or $\texttt{Location} \neq \nu$. We denote as $[\![=]\!]$ the rewriting rule $\circledast \mapsto =$. Thus $[\![=]\!](\zeta) \doteq \alpha = \nu$. Therefore, e.g., $[\![=]\!](\texttt{Time} > \nu)$ is $\texttt{Time} = \nu$.

A *correlation condition* $\varphi_{\rightleftharpoons}$ consists of logical conjunctions of correlation atoms:

$$\varphi_{\rightleftharpoons} \triangleq \mathbf{true} \wedge \bigwedge_{i=1}^{r} \zeta_i(\alpha_i, \nu_i) \tag{4}$$

with $r \in \mathbb{N} \cup \{0\}$. We observe that formula (4) evaluates to $\mathbf{true}$ if $r = 0$. We shall omit conjunctions when trivially entailed by the remainder of the formula. With slight abuse of notation we extend the notion of $[\![=]\!]$ over conditions, e.g., $[\![=]\!](\varphi_{\rightleftharpoons}) \triangleq \mathbf{true} \wedge \bigwedge_{i=1}^{r} [\![=]\!](\zeta_i)(\alpha_i, \nu_i)$. Given, e.g., a correlation condition $\texttt{Time} > \nu_1 \wedge \texttt{Clerk} \neq \nu_2$, then $[\![=]\!](\texttt{Time} > \nu_1 \wedge \texttt{Clerk} \neq \nu_2)$ is $\texttt{Time} = \nu_1 \wedge \texttt{Clerk} = \nu_2$. In the following, we name as *correlation expansion* the $\wedge$-concatenation of correlation atoms.

Definition 10 (MP-Declare pre-interpretation)**.** *Let $\Xi(x_1, \ldots, x_n) \in \widehat{\Xi}$ be a* Declare *template of arity $n > 1$ where $par_{\bullet}(\Xi)$ is the non-empty set of activation parameters, and $par_{\Rightarrow}(\Xi)$ is the non-empty set of target parameters, as per Defs. 6 and 7. Let $\mathscr{I}_{\varsigma}$ be a check pre-interpretation as per Def. 9. Given a correlation condition $\varphi_{\rightleftharpoons}$, an* MP-Declare pre-interpretation *is a function defined as follows:*

$$\mathscr{I}(x) = \begin{cases} \mathscr{I}_{\bullet} ::= \mathscr{I}_{\varsigma}(x) \wedge [\![=]\!](\varphi_{\rightleftharpoons}) & \text{if } x \in par_{\bullet}(\Xi); \\ \mathscr{I}_{\Rightarrow} ::= \mathscr{I}_{\varsigma}(x) \wedge \varphi_{\rightleftharpoons} & \text{if } x \in par_{\Rightarrow}(\Xi). \end{cases}$$

*We shall denote an MP-*Declare *pre-interpreted template as ${}_{\mathscr{I}}\Xi$.*

We remark that the distinction made between activation and target within the definition of $\mathscr{I}$'s is legit because the distinction refers to parameters of parametric predicates, hence prior to any interpretation or assignment thereof. An example of MP-Declare pre-interpreted template is $\mathbf{G}((a \wedge \texttt{Clerk} = \nu) \rightarrow \mathbf{F}(b \wedge \texttt{Clerk} \neq \nu))$.

Notice that ν is a free variable that can assume any value. We shall thus express MP-Declare templates as sentences, i.e., formulae whose variables are all bound to a first-order quantification. The quantification needed to our extent is $\forall$. A valuation function is thus needed that maps variables to constant symbols. In our interpretation, it is meant to map variables to symbols of domain Δ. The mapping of variable ν to constant δ will be denoted as $[\delta \leftarrowtail \nu]$. We recall that different values can be mapped to a variable along the trace, i.e., depending on the event over which the valuation takes place [33]: this distinguishes it from the pre-interpretation function.

Definition 11 (MP-Declare pre-constraint). *Let $\Xi_{/n}$ be a declarative template of arity n in the* Declare *repertoire as per Defs. 6 and 7. An* MP-pre-interpreted Declare template *is a* Declare *template $\Xi_{/n}$ whose n parameters are pre-interpreted by $\mathscr{I}$, $\Xi(\mathscr{I}(x_1), \ldots, \mathscr{I}(x_n))$. By Def. 10, it consists of an LTL-FO$_f$ formula with free variables $\nu_1, \ldots, \nu_m$ with $m \in \mathbb{N}$ having the following syntax:*

$$\psi ::= \Phi \mid \neg\psi \mid \psi \wedge \psi' \mid \psi \vee \psi' \mid \psi \rightarrow \psi' \mid \mathbf{X}\,\psi \mid \mathbf{F}\,\psi \mid \psi\,\mathbf{U}\,\psi' \mid \mathbf{Y}\,\psi \mid \mathbf{O}\,\psi \mid \psi\,\mathbf{S}\,\psi' \mid \mathbf{G}\,\psi \tag{5}$$

$$\Phi ::= \mathbf{true} \mid a \mid \Phi \wedge \alpha \circledast \delta \mid \Phi \wedge \alpha \circledast \nu \tag{6}$$

$$\circledast ::= \ominus \mid \otimes; \qquad \ominus ::= \neq \mid =; \qquad \otimes ::= \leqslant \mid \geqslant \mid < \mid > \mid \ominus. \tag{7}$$

Equation (5) *matches the syntax definition of Def. 3. Therefore we specialise semantics by replacing rule 1 with the following four:*

1(a). $\tau, \varepsilon \models \mathbf{true}$;

1(b). $\tau, \varepsilon \models a$ *if it holds true that* $\ell(\varepsilon) = a$;

1(c). $\tau, \varepsilon \models \alpha \circledast \delta$ *if it holds true that* $\alpha(\varepsilon)$ *is in* $\circledast$*-relation with constant* δ *(let* $\circledast$ *be equality, inequality, or order comparison);*

1(d). $\tau, \varepsilon \models \alpha \circledast \nu$ *if it holds true that* $\alpha(\varepsilon)$ *is in* $\circledast$*-relation with* $[\delta \leftarrowtail \nu]$*, where* $[\delta \leftarrowtail \nu]$ *is the application of a valuation function upon event* ε *on variable* ν*, returning* δ.

An MP-Declare pre-constraint *is the universal closure in prenex form of* $\Xi(\mathscr{I}(x_1), \ldots, \mathscr{I}(x_n))$ *[34], i.e., such that all free variables* $\nu_1, \ldots, \nu_m$ *are bound to universal quantification prepended to the formula:* $\forall \nu_1, \ldots, \nu_m . \Xi(\mathscr{I}(x_1), \ldots, \mathscr{I}(x_n))$. *We shall overload* $\mathscr{I}$ *with its inductive extension on sets of parameters:* $\mathscr{I}(x_1, \ldots, x_m) = (\mathscr{I}(x_1), \ldots, \mathscr{I}(x_m))$. *We will henceforth compactly indicate a pre-constraint as* ${}^{\forall}_{\mathscr{I}}\Xi$. *Its semantics extend the aforementioned ones with the following valuation rule:*

0. $\tau, \varepsilon \models \forall \nu . \psi$ *if it is true that* ψ *holds for any valuation of* ν*, i.e., for any* δ *resulting from* $[\delta \leftarrowtail \nu]$.

Intuitively, an MP-Declare pre-constraint is the conceptual link from standard Declare templates to MP-Declare constraints, because they interpret parameters as a conjunction of conditions imposed over activities and context information stemming from event attributes. Given two MP-Declare interpretation functions, $\mathscr{I}_1$ and $\mathscr{I}_2$, examples of MP-Response pre-constraints are:

$${}^{\forall}_{\mathscr{I}_1}\Xi \doteq \forall \nu_1 . \mathbf{G}((a_1 \wedge \alpha_1 < \delta_1 \wedge \alpha_2 = \nu_1) \rightarrow \mathbf{F}(a_2 \wedge \alpha_2 \neq \nu_1)); \tag{8}$$

$${}^{\forall}_{\mathscr{I}_2}\Xi \doteq \mathbf{G}(a_1 \rightarrow \mathbf{F}(a_2 \wedge \alpha_1 = \delta_1)). \tag{9}$$

To associate activation and target parameters to the respective sub-formulae of a pre-constraint, we introduce the notions of *activation formula* and *target formula* as follows.

Definition 12 (Activation formula, target formula). *Given a* Declare *template* Ξ*, and an* MP-Declare *pre-constraint* ${}^{\forall}_{\mathscr{I}}\Xi$ *defined by pre-interpretation* $\mathscr{I}$*, the* activation formula *of* ${}^{\forall}_{\mathscr{I}}\Xi$ *is* $\Phi_{\bullet} = \mathscr{I}(par_{\bullet}(\Xi))$. *The* target formula *of* ${}^{\forall}_{\mathscr{I}}\Xi$ *is* $\Phi_{\Rightarrow} = \mathscr{I}(par_{\Rightarrow}(\Xi))$.

The activation formula and target formula of pre-constraint ${}^{\forall}_{\mathscr{I}_1}\Xi$ in example (8) are $a_1 \wedge \alpha_1 < \delta_1 \wedge \alpha_2 = \nu_1$ and $a_2 \wedge \alpha_2 \neq \nu_1$ respectively.

3.2.1 Property of monotonicity

Keeping in mind the property of monotonicity enjoyed by check expansions, and their effect on pre-interpretations of activations and targets, we show here that correlation expansion enjoys

(i) monotonicity, if the target is non-negated, as in the case of Response, or *(ii)* anti-monotonicity otherwise, as in the case of NotResponse.

Theorem 3 (Monotonicity of correlation expansion with non-negated targets)**.** *Let* $\Xi \in \widehat{\Xi}$ *be a* Declare *template,* $x \in par_{\bullet}(\Xi)$ *its activation, and* $y \in par_{\Rightarrow}(\Xi)$ *its target under an even number of negations. Let* $\mathscr{I}$ *and* $\mathscr{I}'$ *be pre-interpretations such that (i)* $\Phi \doteq \mathscr{I}(x)$*, (ii)* $\Phi' \doteq \mathscr{I}'(x) \doteq \Phi \wedge [\![=]\!](\varphi_{\rightleftharpoons})$*, (iii)* $\Psi \doteq \mathscr{I}(y)$*, and (iv)* $\Psi' \doteq \mathscr{I}'(y) \doteq \Psi \wedge \varphi_{\rightleftharpoons}$*. It follows that* ${}^{\forall}_{\mathscr{I}'}\Xi \models {}^{\forall}_{\mathscr{I}}\Xi$.

Proof (sketch). We consider Res.Ex $\doteq \mathbf{G}(x \rightarrow (\mathbf{O}\, y \vee \mathbf{F}\, y))$ as in previous proof sketches. As a base case, we consider $\varphi_{\rightleftharpoons} \doteq \mathbf{true}$. Under these circumstances, ${}^{\forall}_{\mathscr{I}'}\Xi \equiv {}^{\forall}_{\mathscr{I}}\Xi$. For the inductive case, we consider that *(i)* $\mathscr{I}$ contains variables $\nu_1, \ldots, \nu_{m-1}$ for some $m \in \mathbb{N}$, and we assume w.l.o.g. that *(ii)* $\Phi' \doteq \Phi \wedge \alpha = \nu_m$, and *(iii)* $\Psi' \doteq \Phi \wedge \alpha \circledast \nu_m$. Under pre-interpretation $\mathscr{I}$ we have that ${}^{\forall}_{\mathscr{I}}$Res.Ex $\doteq \forall \nu_1 \ldots \nu_{m-1}.\, \mathbf{G}(\neg\Phi \vee \mathbf{O}\,\Psi \vee \mathbf{F}\,\Psi)$. Under pre-interpretation $\mathscr{I}'$ we have that ${}^{\forall}_{\mathscr{I}'}$Res.Ex $\doteq \forall \nu_1 \ldots \nu_{m-1}, \nu_m.\, \mathbf{G}(\neg\Phi \vee \neg(\alpha = \nu_m) \vee \mathbf{O}(\Psi \wedge \alpha \circledast \nu_m) \vee \mathbf{F}(\Psi \wedge \alpha \circledast \nu_m))$. Let us assume per absurdo that for some event, ${}^{\forall}_{\mathscr{I}'}$Res.Ex is satisfied whilst ${}^{\forall}_{\mathscr{I}}$Res.Ex is not. This can happen only if *(i)* Φ evaluates to true and *(ii)* $\mathbf{O}\,\Psi$ and $\mathbf{F}\,\Psi$ evaluate to false. Under these circumstances in no case $\mathbf{O}(\Psi \wedge \alpha \circledast \nu_m) \vee \mathbf{F}(\Psi \wedge \alpha \circledast \nu_m)$ can evaluate to true by monotonicity of $\mathbf{O}$ and $\mathbf{F}$. For ${}^{\forall}_{\mathscr{I}'}$Res.Ex to hold true under universal closure there should exist an event for which no $[\delta \leftarrowtail \nu_m]$ exists such that $(\alpha = \nu_m)$ evaluates to true. This is however not possible because by Def. 1 every event is endowed with a value for all its attributes. Notice that it can be the case instead that, conversely, ${}^{\forall}_{\mathscr{I}}$Res.Ex is satisfied whilst ${}^{\forall}_{\mathscr{I}'}$Res.Ex is not, e.g., if *(i)* Φ evaluates to true and *(ii)* $\mathbf{O}\,\Psi$ evaluates to false, *(iii)* $\mathbf{F}\,\Psi$ evaluates to true, and for all $[\delta \leftarrowtail \nu_m]$ such that *(iv)* $\alpha = \nu_m$ evaluates to true, *(v)* $\alpha \circledast \nu_m$ evaluates to false. □

The same proof scheme can be applied on the other Declare templates, provided that the hypothesis of even negations on y applies. Theorem 3 intuitively entails that the set of events leading to satisfaction cannot increase if additional correlation formulae are added, which is what happens with check formulae concatenated to targets (Theorem 2) and the opposite effect of check-formulae concatenation to activations (Theorem 1). For example, $\forall \nu_1.\, \mathbf{G}((b \wedge \texttt{Score} = \nu_1) \rightarrow \mathbf{F}(c \wedge \texttt{Score} > \nu_1))$, stemming from the Response template again, is satisfied by the events of both τ_1 and τ_2 in Example 1, because after b the value of `Score` increases upon occurrence of c. $\forall \nu_1, \nu_2.\, \mathbf{G}((b \wedge \texttt{Score} = \nu_1 \wedge \texttt{Clerk} = \nu_2) \rightarrow \mathbf{F}(c \wedge \texttt{Score} > \nu_1 \wedge \texttt{Clerk} \neq \nu_2))$, is instead satisfied by events of τ_1 only: notice that indeed the `Clerk` attributes of $\varepsilon 4$ and $\varepsilon 5$ are both "Judith". If we apply a check expansion on the activation, events that satisfy the formula can extend again, as per Theorem 1. Indeed, $\forall \nu_1, \nu_2.\, \mathbf{G}((b \wedge \texttt{Score} < 50 \wedge \texttt{Score} = \nu_1 \wedge \texttt{Clerk} = \nu_2) \rightarrow \mathbf{F}(c \wedge \texttt{Score} > \nu_1 \wedge \texttt{Clerk} \neq \nu_2))$ is satisfied by events of τ_1 and τ_2.
We remark that for templates negating targets, such as NotResponse, the opposite holds: the more the correlation conditions, the higher the number of events supporting them. This rationale is at the basis of the following.

Corollary 3.1. *If* $y \in par_{\Rightarrow}(\Xi)$ *is under an odd number of negations, correlation expansion is anti-monotonic.*

The proof sketch can proceed analogously to Theorem 3, yet considering that the target lies under non-monotonic operator $\neg$.

3.3 MP-Declare constraints and models

MP-Declare pre-constraints are parametric temporal formulas, defined at an intensive level. Constraints are pre-constraints whose terminal symbols are assigned with actual elements,

at an extensive level. In particular, we observe in Def. 11 that the only terminal symbols subject to assignment are a, α, and δ, because ν-variables are bound to the universal closure. In the following definition, we shall indicate the unassigned symbols of an MP-Declare pre-constraint ${}^{\forall}_{\mathscr{I}}\Xi(x_1,\ldots,x_n)$ as $a\left(\mathscr{I}(x_1,\ldots,x_n)\right)$, $\alpha\left(\mathscr{I}(x_1,\ldots,x_n)\right)$, and $\delta\left(\mathscr{I}(x_1,\ldots,x_n)\right)$, respectively. In (8) we have, e.g., that $a\left(\mathscr{I}_1(x_1,x_2)\right)=\{a_1,a_2\}$, $\alpha\left(\mathscr{I}_1(x_1,x_2)\right)=\{\alpha_1,\alpha_2\}$, and $\delta\left(\mathscr{I}_1(x_1,x_2)\right)=\{\delta_1\}$; in (9) we have that $a\left(\mathscr{I}_2(x_1,x_2)\right)=\{a_1,a_2\}$, $\alpha\left(\mathscr{I}_2(x_1,x_2)\right)=\{\alpha_1\}$, and $\delta\left(\mathscr{I}_2(x_1,x_2)\right)=\{\delta_1\}$.

Definition 13 (MP-Declare constraint). *Let ${}^{\forall}_{\mathscr{I}}\Xi$ be an MP-*Declare *pre-constraint of template $\Xi(x_1,\ldots,x_n)$. Let A be a set of task symbols, $\widehat{\alpha}$ be the universe of attribute functions, and $\widehat{\Delta}$ be the universe of interpretation as per Def. 1. An* MP-Declare assignment *is a triple of mappings $\mathscr{A}=(\mathscr{A}_\Sigma,\mathscr{A}_{\widehat{\alpha}},\mathscr{A}_{\widehat{\Delta}})$ defined on $\mathscr{I}(par(\Xi))$ such that $\mathscr{A}_\Sigma : a\left(\mathscr{I}(par(\Xi))\right)\to\Sigma$, $\mathscr{A}_{\widehat{\alpha}} : \alpha\left(\mathscr{I}(par(\Xi))\right)\to\widehat{\alpha}$, and $\mathscr{A}_{\widehat{\Delta}}(\delta) : \delta\left(\mathscr{I}(par(\Xi))\right)\to\widehat{\Delta}$. An* MP-Declare constraint *$\mathcal{C}$ is an MP-*Declare *pre-constraint assigned by $\mathscr{A}$, henceforth denoted as $\mathcal{C}=\mathscr{A}\left({}^{\forall}_{\mathscr{I}}\Xi\right)$. Given an event log L as defined in Def. 1, an MP-*Declare *constraint $\mathcal{C}$ is* satisfied, *or* fulfilled, *in a trace τ by event $\varepsilon\in\tau$ if $\tau,\varepsilon\models\mathcal{C}$ according to Def. 11. It is otherwise* violated.

Consider, e.g., the pre-constraints ${}^{\forall}_{\mathscr{I}_1}\Xi$ and ${}^{\forall}_{\mathscr{I}_2}\Xi$, the log of Example 1, and the example MP-Declare assignments *(i)* $\mathscr{A}_{1_1}$ for ${}^{\forall}_{\mathscr{I}_1}\Xi$, and *(ii)* $\mathscr{A}_{2_1},\mathscr{A}_{2_2}$ for ${}^{\forall}_{\mathscr{I}_2}\Xi$. Constraints $\mathcal{C}_{1_1}=\mathscr{A}_{1_1}\left({}^{\forall}_{\mathscr{I}_1}\Xi\right)$, $\mathcal{C}_{2_1}=\mathscr{A}_{2_1}\left({}^{\forall}_{\mathscr{I}_2}\Xi\right)$, and $\mathcal{C}_{2_2}=\mathscr{A}_{2_2}\left({}^{\forall}_{\mathscr{I}_2}\Xi\right)$ are then:

$$\mathcal{C}_{1_1} \doteq \forall\nu_1.\,\mathbf{G}\big((b \wedge \texttt{Score} < 50 \wedge \texttt{Clerk} = \nu_1) \to \mathbf{F}(c \wedge \texttt{Clerk} \neq \nu_1)\big); \tag{10}$$

$$\mathcal{C}_{2_1} \doteq \mathbf{G}\big(a \to \mathbf{F}(c \wedge \texttt{Location} = \text{Office 2})\big); \tag{11}$$

$$\mathcal{C}_{2_2} \doteq \mathbf{G}\big(b \to \mathbf{F}(c \wedge \texttt{Clerk} = \text{Jane})\big). \tag{12}$$

In the remainder, we will extend w.l.o.g. the notion of activation and target to constraints.

Definition 14 (MP-Declare model). *An* MP-Declare model *is a tuple* $\mathrm{D}\mathcal{M}=\left(A,\widehat{\alpha},\widehat{\Delta},\widehat{\Xi},\{\mathscr{I}_i : i\in\mathbb{N}\},\{\mathscr{A}_{i_j} : j\in\mathbb{N}\},\left\{\mathcal{C}_{i_j} : \mathcal{C}_{i_j}\doteq\mathscr{A}_{i_j}\left({}^{\forall}_{\mathscr{I}_i}\Xi\right),\Xi\in\widehat{\Xi}\right\}\right)$ *where A is a set of task symbols, $\widehat{\alpha}$ is the universe of attribute functions, $\widehat{\Delta}$ is the universe of interpretation, $\widehat{\Xi}$ is the repertoire of* Declare *templates, $\mathscr{I}_i$ are MP-*Declare *pre-interpretations, $\mathscr{A}_{i_j}$ are assignments for $\mathscr{I}_i$, and $\mathcal{C}_{i_j}$ are constraints, each derived from the $\mathscr{A}_{i_j}$-assignment of a pre-constraint.*
*An MP-*Declare *model* $\mathrm{D}\mathcal{M}$ *is* satisfied *in a trace τ by event $\varepsilon\in\tau$ if and only if all its constraints are satisfied. It is otherwise* violated.

In the remainder, we shall indicate constraints succinctly by the template name and the assigned parameters, omitting the FO-LTL$_f$ formalisation and the universal closure of variables. Considering for example Example 1, both traces satisfy an MP-Declare model that comprises, e.g., the following constraints: Response$(a, c\wedge\texttt{Location}=\text{Office 2})$, ChainPrecedence$(d\wedge\texttt{Clerk}=\nu, c\wedge\texttt{Clerk}\neq\nu)$, Response$(b\wedge\texttt{Score}>90\wedge\texttt{Clerk}=\nu_1\wedge\texttt{Location}=\nu_2, d\wedge\texttt{Clerk}\neq\nu_1\wedge\texttt{Location}\neq\nu_2)$. Notice that an SP-Declare constraint, e.g., Response(a,c), is an MP-Declare constraint for which no check nor correlation expansion has been applied for the pre-interpretation of parameters. Furthermore, we remark that all propositions stated for (pre-interpreted) templates and pre-constraints naturally extend to constraints as well, the latter being at the extensive level as assignments of the former.

We finally remark that for MP-Declare a *temporal* condition can be specified through an interval indicating the minimum and the maximum temporal distance allowed between the

occurrence of the activation and the occurrence of the corresponding target. It plays a fundamental role in process modelling through constraints [35, 36], thus we consider it a first-class citizen in the categorisation of conditions in MP-Declare. However, it falls in the category of correlation conditions, as it is based on the comparison of values associated to both activation and target events. For instance, a Response constraint with a temporal condition can be $\forall \nu.\, \mathbf{G}((b \wedge \texttt{Time} = \nu) \implies \mathbf{F}(c \wedge \texttt{Time} \leqslant \nu + 1\ \mathrm{day}))$ requiring that after b occurs, c must eventually follow within one day. Notice that the operation intuitively written as $+1\ \mathrm{day}$ is an endomorphism from the domain of `Time` to itself. Because one operand is a constant, it is reducible to a function on single variables and, as such, not adding complexity to the general treatise.

4 Mining Metrics

Declarative process mining relies on different metrics. In this subsection, we describe the metrics that we use to discriminate those constraints that are fulfilled in the majority of cases from those that are rarely satisfied, namely, *support* and *confidence*. We consider two notions of support already defined in the literature, specifically the event-based support [14] and the trace-based support [37]. The former is meant to be used for all constraints wherein both activation and target are defined. For all the others, we use the second notion of support. Thereupon, we show that support and confidence preserve the properties of monotonicity and anti-monotonicity seen for check and correlation expansion of activation and target.

Definition 15 (Support). *Let L be an event log, C its set of cases, and $\models^{\varepsilon}_{\tau}(\psi)$ the set of* events *ε in a* trace *τ of L that satisfy an FO-LTL$_f$ formula ψ. Let $\models^{\varepsilon}_{L}(\psi)$ be the set of all the events ε in the traces of L that satisfy ψ. Let $\mathcal{C}$ be an MP-*Declare *constraint and $\Phi_\bullet$ its activation. We respectively define as* event-based support $\mathscr{S}^{\varepsilon}_{L}(\mathcal{C})$ *and* trace-based support $\mathscr{S}^{\tau}_{L}(\mathcal{C})$ *of $\mathcal{C}$ as follows:*

$$\mathscr{S}^{\varepsilon}_{L}(\mathcal{C}) = \frac{\sum_{i=1}^{|L|} \left|\models^{\varepsilon}_{\tau_i}(\Phi_\bullet \wedge \mathcal{C})\right|}{\left|\models^{\varepsilon}_{L}(\Phi_\bullet)\right|} \qquad (13) \qquad\qquad \mathscr{S}^{\tau}_{L}(\mathcal{C}) = \frac{\left|\models^{\tau}_{L}(\mathcal{C})\right|}{|C|} \qquad (14)$$

Intuitively, $\mathscr{S}^{\varepsilon}_{L}(\mathcal{C})$ sums up the number of events that activate and fulfil $\mathcal{C}$ for every trace, and divides it by the number of events that satisfy the activation of $\mathcal{C}$ along the whole log. Notice that $\mathcal{C}$ is an FO-LTL$_f$ formula as well as $\Phi_\bullet$, so it is $\Phi_\bullet \wedge \mathcal{C}$. $\mathscr{S}^{\tau}_{L}(\mathcal{C})$ counts instead the number of traces that fulfil the constraint, averaged over the whole amount of traces in the log – we recall that by Def. 2 each trace correspond to one case. Notice that because SP-Declare constraints are a sub-class of MP-Declare ones, the same definition applies to both. In Example 3, Response(a, b) is activated by the 6 a's occurring in σ_1 (1), σ_3 (2), σ_5 (1), and σ_6 (2). Out of those, all except the second occurrence of a in σ_6 also fulfil Response(a, b). Therefore $\mathscr{S}^{\varepsilon}_{L}(\text{Response}(a, b)) = 5/6 \approx 0.833$. For Existence1$(a)$ we consider the trace-based support because no activation is defined for it (cf. Table 2). In the four string-traces above, the constraint is fulfilled, whilst in the remaining two it is violated because no a occurs. Therefore $\mathscr{S}^{\tau}_{L}(\text{Existence1}(a)) = 4/6 \approx 0.667$.
The confidence metric scales the support by the fraction of traces in the log in which the activation condition is satisfied.

Definition 16 (Confidence). *Let L be an event log, C its set of cases, and $\models^{\tau}_{L}(\psi)$ the set of all the traces τ of L consisting only of events that fulfil ψ. Let $\mathcal{C}$ be an MP-*Declare *constraint*

and $\Phi_\bullet$ its activation. We respectively define as event-based confidence $\mathscr{K}_L^\varepsilon$ *and* trace-based confidence $\mathscr{K}_L^\tau$ *of $\mathcal{C}$ as follows:*

$$\mathscr{K}_L^\varepsilon(\mathcal{C}) = \mathscr{S}_L^\varepsilon \times \frac{|\models_L^\tau(\Phi_\bullet)|}{|C|} \quad (15) \qquad \mathscr{K}_L^\tau(\mathcal{C}) = \mathscr{S}_L^\tau \times \frac{|\models_L^\tau(\Phi_\bullet)|}{|C|} \quad (16)$$

The event-based confidence of Response(a, b) in the string-traces of Example 3, e.g., is $\mathscr{K}_L^\varepsilon(\text{Response}(a, b)) = 5/6 \times 4/6 \approx 0.556$. The definitions of support and confidence in Eq. 13-16 show that the discovery of a certain constraint requires the following values to be extracted from event logs: *(i)* the number of occurrences of the activation of the constraint, *(ii)* the number of fulfilments of the constraint, and *(iii)* the fraction of traces in the log where the activation holds. In the remainder, we will avoid the specification of trace-based or event-based for support and confidence when clear from the context.

Theorem 4 (Preservation of monotonicity in support and confidence)**.** *Let L be an event log and $\mathcal{C}$ an MP-*Declare *constraint. Support and confidence of $\mathcal{C}$ on L preserve (i) monotonicity w.r.t. check-expansion on non-negated target, (ii) monotonicity w.r.t. correlation-expansion with non-negated target, (iii) anti-monotonicity w.r.t. check-expansion on negated target, and (iv) anti-monotonicity w.r.t. correlation-expansion with negated target.*

Proof. Given two LTL-FO$_f$ formulae ψ and ψ', if $\psi' \models \psi$ then by definition all events that satisfy ψ' also satisfy ψ, therefore $\models_L^\varepsilon(\psi') \subseteq \models_L^\varepsilon(\psi)$. Consequently, traces consisting of events that all satisfy ψ' maintain the same characteristic for ψ, therefore $\models_L^\tau(\psi') \subseteq \models_L^\tau(\psi)$. Check-expansions on targets do not involve the activations, therefore they are idempotent for $\models_L^\varepsilon(\Phi_\bullet)$ and $\models_L^\tau(\Phi_\bullet)$. As shown in the proof of Theorem 3, it is not possible that correlation conditions violate the activation formula if the rest of the conjunction evaluates to true, whereas if the latter is false, then so is the whole conjunction. Therefore, also correlation-expansions are idempotent for $\models_L^\varepsilon(\Phi_\bullet)$ and $\models_L^\tau(\Phi_\bullet)$. The propositions here stated are thus direct consequences of *(i)* Theorem 2, *(ii)* Theorem 3, *(iii)* Corollary 2.1, and *(iv)* Corollary 3.1 by inspection of the definitions of support and confidence as per Defs. 15 and 16. □

Theorem 4 entails that adding attribute checks on targets and correlation comparisons implies that support and confidence cannot increase, provided that the target parameters do not lie under negation as in the case of Response. Otherwise the consequence is that those metrics cannot decrease, as in the case of NotResponse. Considering the log of Example 1, Response(b, c) is activated and fulfilled once per trace. Consequently, $\mathscr{S}_L^\varepsilon(\text{Response}(b, c)) = 2/2 = 1$ and $\mathscr{K}_L^\varepsilon(\text{Response}(b, c)) = 2/2 \times 2/2 = 1$. Applying a check-expansion on the target, a resulting constraint can be Response$(b \wedge \texttt{Clerk} = \nu, c \wedge \texttt{Clerk} \neq \nu)$. The number of events that lead to fulfilment decrease, and more specifically we have that $\mathscr{S}_L^\varepsilon(\text{Response}(b \wedge \texttt{Clerk} = \nu, c \wedge \texttt{Clerk} \neq \nu)) = 1/2 = 0.5$ and $\mathscr{K}_L^\varepsilon(\text{Response}(b \wedge \texttt{Clerk} = \nu, c \wedge \texttt{Clerk} \neq \nu)) = 1/2 \times 2/2 = 0.5$. We notice the same consequence with a check-expansion on the target, e.g., Response$(b, c \wedge \texttt{Clerk} \neq \text{Judith})$. Also in that case, support and confidence both decrease to 0.5. Support and confidence instead remain equal to 1 with Response$(b \wedge \texttt{Score} = \nu, c \wedge \texttt{Score} > \nu)$ and Response$(b, c \wedge \texttt{Score} > 50)$.

Corollary 4.1 (Preservation of monotonicity in trace-based support)**.** *Trace-based support of $\mathcal{C}$ on L preserve anti-monotonicity w.r.t. check-expansion on activation.*

Proof. As seen in the proof of Theorem 4, if, given two LTL-FO$_f$ formulae ψ and ψ', $\psi' \models \psi$, then $\models_L^\tau(\psi') \subseteq \models_L^\tau(\psi)$. This corollary thus follows from Theorem 1 by inspection of the definition of trace-based support as per Def. 15. □

Considering again the log of Example 1, $\mathscr{S}_L^\tau(\text{Response}(b \wedge \texttt{Clerk} = \nu, c \wedge \texttt{Clerk} \neq \nu)) = 1/2 = 0.5$ because all events of τ_1 satisfy it, whilst a violation occurs in τ_2. Let us now consider the following constraint resulting from the check-expansion on the activation. $\text{Response}(b \wedge \texttt{Score} < 50 \wedge \texttt{Clerk} = \nu, c \wedge \wedge \texttt{Clerk} \neq \nu)$. All events in both traces fulfil the constraint because the constraint is not activated in τ_2, therefore $\mathscr{S}_L^\tau(\text{Response}(b \wedge \texttt{Score} < 50 \wedge \texttt{Clerk} = \nu, c \wedge \wedge \texttt{Clerk} \neq \nu)) = 2/2 = 1.0$.

5 Concluding remarks and research outlook

So far we have described the syntax and semantics of a language to verify temporal constraints on logs of events bearing multiple contextual information. We have shown formal properties that such a language enjoys and proven that the support and confidence of its constraints is monotonic or anti-monotonic under certain conditions, therefore a partial order can be established thereupon. The next step is to take advantage of these investigations to implement a discovery algorithm for MP-Declare that run on linked-data by means of first-order queries. To this end, we will take inspiration a.o. from existing research on database querying through temporal logic, and in particular on the work of Abiteboul et al. [29] to reduce traces to (sequentially-)time-stamped tuples in a database, and by the investigation of Chomicki et al. [33] on the translation of LTL-FO queries to a temporal extension of SQL-92.

6 Summary

In this deliverable, we have described a declarative language for describing processes under multiple perspectives other than control-flow. To that extent, we have resorted on an extension of an existing language, namely Declare, to encompass conditions exerted on other perspectives than control-flow, such as time, resources, and data data would include physical and social systems information. We have defined for that language syntax and semantics on the basis of a first-order extension of linear temporal logics over finite traces. Furthermore, we have shown that the extensions to include multi-perspective conditions in constraints enjoy monotonicity properties. Finally, a research plan for future work in this project has been provided that builds upon the outcome of this deliverable D5.1 to progress towards upcoming D5.2.

References

[1] M. Dumas, M. L. Rosa, J. Mendling, and H. A. Reijers, *Fundamentals of Business Process Management*. Springer, 2013.

[2] H. M. W. Verbeek, J. C. A. M. Buijs, B. F. van Dongen, and W. M. P. van der Aalst, "XES, XESame, and ProM 6," in *Information Systems Evolution - CAiSE Forum 2010, Hammamet, Tunisia, June 7-9, 2010, Selected Extended Papers*, 2010, pp. 60–75.

[3] T. Baier, C. Di Ciccio, J. Mendling, and M. Weske, "Matching of events and activities - an approach using declarative modeling constraints," in *BPMDS/EMMSAD*, ser. Lecture Notes in Business Information Processing, K. Gaaloul, R. Schmidt, S. Nurcan, S. Guerreiro, and Q. Ma, Eds., vol. 214. Springer, June 2015, pp. 119–134.

[4] T. Baier, A. Rogge-Solti, J. Mendling, and M. Weske, "Matching of events and activities: an approach based on behavioral constraint satisfaction," in *SAC*, R. L. Wainwright, J. M. Corchado, A. Bechini, and J. Hong, Eds. ACM, 2015, pp. 1225–1230.

[5] T. Baier, C. Di Ciccio, J. Mendling, and M. Weske, "Matching events and activities by integrating behavioral aspects and label analysis," *Software & Systems Modeling*, pp. 1–26, 4 2017.

[6] S. Schönig, C. Cabanillas, S. Jablonski, and J. Mendling, "A framework for efficiently mining the organisational perspective of business processes," *Decision Support Systems*, vol. 89, pp. 87–97, 2016.

[7] A. Van Looy and A. Shafagatova, "Business process performance measurement: a structured literature review of indicators, measures and metrics," *SpringerPlus*, vol. 5, no. 1, p. 1797, Oct 2016.

[8] N. M. Zaki, A. Awad, and E. Ezat, "Extracting accurate performance indicators from execution logs using process models," in *12th IEEE/ACS International Conference of Computer Systems and Applications, AICCSA 2015, Marrakech, Morocco, November 17-20, 2015*. IEEE, 2015, pp. 1–8.

[9] E. Verbeek, J. Buijs, B. van Dongen, and W. van der Aalst, "XES, xESame, and ProM 6," in *Information Systems Evolution*, 2011, pp. 60–75.

[10] B. F. van Dongen and S. Shabani, "Relational XES: data management for process mining," in *CAiSE Forum 2015*, 2015, pp. 169–176.

[11] M. Pesic and W. Van der Aalst, "A declarative approach for flexible business processes management," in *BPM Workshops*. Springer, 2006, pp. 169–180.

[12] F. M. Maggi, J. C. Bose, and W. van der Aalst, "Efficient Discovery of Understandable Declarative Process Models from Event Logs," in *CAiSE*, 2012, pp. 270–285.

[13] F. Chesani, E. Lamma, P. Mello, M. Montali, F. Riguzzi, and S. Storari, "Exploiting Inductive Logic Programming Techniques for Declarative Process Mining," *ToPNoC*, vol. 5460, pp. 278–295, 2009.

[14] C. Di Ciccio and M. Mecella, "On the discovery of declarative control flows for artful processes," *ACM Trans. Management Inf. Syst.*, vol. 5, no. 4, pp. 24:1–24:37, 2015.

[15] E. A. Emerson, "Temporal and modal logic," in *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics*, J. van Leeuwen, Ed. Cambridge, MA, USA: MIT Press, 1990, ch. Volume B: Formal Models and Sematics, pp. 995–1072. [Online]. Available: http://dl.acm.org/citation.cfm?id=114891.114907

[16] T. Wilke, "Classifying discrete temporal properties," in *STACS*, ser. Lecture Notes in Computer Science, C. Meinel and S. Tison, Eds., vol. 1563. Springer, 1999, pp. 32–46.

[17] G. De Giacomo and M. Y. Vardi, "Linear temporal logic and linear dynamic logic on finite traces," in *IJCAI*, F. Rossi, Ed. IJCAI/AAAI, 2013, pp. 854–860. [Online]. Available: http://www.aaai.org/ocs/index.php/IJCAI/IJCAI13/paper/view/6997

[18] M. Montali, M. Pesic, W. van der Aalst, F. Chesani, P. Mello, and S. Storari, "Declarative Specification and Verification of Service Choreographies," *ACM Transactions on the Web*, vol. 4, no. 1, 2010.

[19] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang, "Symbolic model checking: 10^20 states and beyond," *Inf. Comput.*, vol. 98, no. 2, pp. 142–170, 1992.

[20] D. Gries, "Monotonicity in calculational proofs," in *Correct System Design, Recent Insight and Advances, (to Hans Langmaack on the occasion of his retirement from his professorship at the University of Kiel)*, ser. Lecture Notes in Computer Science, E. Olderog and B. Steffen, Eds., vol. 1710. Springer, 1999, pp. 79–85.

[21] C. Di Ciccio, F. M. Maggi, and J. Mendling, "Efficient discovery of target-branched declare constraints," *Inf. Syst.*, vol. 56, pp. 258–283, 2016.

[22] C. Di Ciccio, F. M. Maggi, M. Montali, and J. Mendling, "Resolving inconsistencies and redundancies in declarative process models," *Inf. Syst.*, vol. 64, pp. 425–446, 2017.

[23] O. Kupferman and M. Y. Vardi, "Vacuity Detection in Temporal Model Checking," *International Journal on Software Tools for Technology Transfer*, pp. 224–233, 2003.

[24] W. M. P. van der Aalst and M. Pesic, "DecSerFlow: Towards a truly declarative service flow language," in *WS-FM*, ser. Lecture Notes in Computer Science, M. Bravetti, M. Núñez, and G. Zavattaro, Eds., vol. 4184. Springer, 2006, pp. 1–23.

[25] A. Burattin, F. M. Maggi, and A. Sperduti, "Conformance checking based on multi-perspective declarative process models," *Expert Systems with Applications*, vol. 65, pp. 194–211, 2016.

[26] S. Abiteboul, R. Hull, and V. Vianu, *Foundations of Databases*. Addison-Wesley, 1995.

[27] M. Chechik, S. M. Easterbrook, and V. Petrovykh, "Model-checking over multi-valued logics," in *FME*, ser. Lecture Notes in Computer Science, J. N. Oliveira and P. Zave, Eds., vol. 2021. Springer, 2001, pp. 72–98.

[28] M. Chechik, B. Devereux, S. M. Easterbrook, and A. Gurfinkel, "Multi-valued symbolic model-checking," *ACM Trans. Softw. Eng. Methodol.*, vol. 12, no. 4, pp. 371–408, 2003.

[29] S. Abiteboul, L. Herr, and J. V. den Bussche, "Temporal versus first-order logic to query temporal databases," in *PODS*, R. Hull, Ed. ACM Press, 1996, pp. 49–57.

[30] S. Cerrito, M. C. Mayer, and S. Praud, "First order linear temporal logic over finite time structures," in *LPAR*, ser. Lecture Notes in Computer Science, H. Ganzinger, D. A. McAllester, and A. Voronkov, Eds., vol. 1705. Springer, 1999, pp. 62–76.

[31] V. Stolz, "Temporal assertions for sequential and concurrent programs," Ph.D. dissertation, RWTH Aachen University, Germany, 2007. [Online]. Available: http://darwin.bth.rwth-aachen.de/opus3/volltexte/2007/1977/

[32] ——, "Temporal assertions with parametrized propositions," *J. Log. Comput.*, vol. 20, no. 3, pp. 743–757, 2010.

[33] J. Chomicki, D. Toman, and M. H. Böhlen, "Querying atsql databases with temporal logic," *ACM Trans. Database Syst.*, vol. 26, no. 2, pp. 145–178, Jun. 2001.

[34] A. Deutsch, L. Sui, and V. Vianu, "Specification and verification of data-driven web services," in *PODS*, C. Beeri and A. Deutsch, Eds. ACM, 2004, pp. 71–82.

[35] A. Lanz, M. Reichert, and B. Weber, "Process time patterns: A formal foundation," *Inf. Syst.*, vol. 57, pp. 38–68, 2016.

[36] F. M. Maggi, "Discovering metric temporal business constraints from event logs," in *BIR*, 2014, pp. 261–275.

[37] F. M. Maggi, A. Mooij, and W. van der Aalst, "User-Guided Discovery of Declarative Process Models," in *CIDM*, 2011, pp. 192–199.