A Temporal Logic-Based Measurement Framework for Process Mining

Alessio Cecconi
WU Vienna
Vienna, Austria
alessio.cecconi@wu.ac.at

Giuseppe De Giacomo, Claudio Di Ciccio
Sapienza University of Rome
Rome, Italy
{name.surname}@uniroma1.it

Fabrizio Maria Maggi
Free University of Bolzano
Bolzano, Italy
maggi@inf.unibz.it

Jan Mendling
WU Vienna
Vienna, Austria
jan.mendling@wu.ac.at

***Abstract*—The assessment of behavioral rules with respect to a given dataset is key in several research areas, including declarative process mining, association rule mining, and specification mining. The assessment is required to check how well a set of discovered rules describes the input data, as well as to determine to what extent data complies with predefined rules. In declarative process mining, in particular, some measures have been taken from association rule mining and adapted to support the assessment of temporal rules on event logs. Among them, support and confidence are used most often, yet they are reportedly unable to provide a sufficiently rich feedback to users and often cause spurious rules to be discovered from logs. In addition, these measures are designed to work on a predefined set of rules, thus lacking generality and extensibility. In this paper, we address this research gap by developing a general measurement framework for temporal rules based on Linear-time Temporal Logic with Past on Finite Traces (LTLp$_f$). The framework is independent from the rule-specification language of choice and allows users to define new measures. We show that our framework can seamlessly adapt well-known measures of the association rule mining field to declarative process mining. Also, we test our software prototype implementing the framework on synthetic and real-world data, and investigate the properties characterizing those measures in the context of process analysis.**

***Index Terms*—Declarative Process Mining, Specification Mining, Association Rule Mining, Quality Measures, Temporal Rules**

I. Introduction

Measuring the degree to which process traces comply with behavioral rules is key in process analysis branches such as conformance checking [1], compliance assessment [2], and discovery of process constraints [3]. To date, several measures have been defined to this end. Among the most frequently used measures there are support and confidence. However, their definition has been customized to the specification languages in use and even for the specific mining algorithms under analysis. For instance, there is a significant difference in the definition of support used in [3] (percentage of traces fully compliant to a rule) and [4] (percentage of fulfilled triggers of the rule over the entire log), in a way that the support of rule "If a is executed, then b will be executed later" on a set of traces like $\{\langle \mathsf{a},\mathsf{b},\mathsf{c},\mathsf{d}\rangle, \langle \mathsf{a},\mathsf{b},\mathsf{c},\mathsf{a}\rangle, \langle \mathsf{a},\mathsf{c}\rangle\}$ is equal to 0.33 for [3] and 0.5 according to [4]. Furthermore, the definition of those measures are defined ad hoc for specific sets of rules, like DECLARE [5] templates. Such issues hinder the fair comparison and eventually the advancement of rule-based process mining.

A plethora of other measures are available in the context of association rule mining that are reportedly superior in comparison to support and confidence [6]. However, these measures do not take into consideration the temporal dimension, which is a first-class citizen dimension in process mining.

In this paper, we address this research problem by proposing a general measurement framework rooted in formal semantics, specifically in Linear-time Temporal Logic with Past on Finite Traces (LTLp$_f$), to express process rules in a reactive form [7] and abstract from specific rule-specification languages. More specifically, we show that a fine grained temporal logic interpretation of any formula in the form "if A then B" allows us to assess all the available association rule mining measures as-is for temporal rules.

Towards investigating the implications of using a large set of available measures on log analysis, we conduct an extensive set of simulation experiments. Through them, we observe that the measures respond differently to changes in the behavior evidenced by event logs, thus suggesting that different measures can be used to highlight different aspects of a process as per its recorded executions. Also, we test our implementation of the framework on both synthetic and real-world data.

The remainder of this paper is structured as follows. Section II discusses prior research on measures for declarative process mining and specification mining. Section III defines preliminaries upon which we define our framework. Section IV defines the measurement framework. Section V presents computational studies of our implementation and Section VI the results of its application to a series of simulation experiments. Finally, Section VII summarizes the contribution of this paper and points to opportunities for future research.

II. Related Work

Temporal logic and declarative specifications have been widely used to support process discovery and conformance checking. The assessment of rules with respect to traces is a key component of all these techniques.

In declarative process discovery, quality measures are used to prune candidate rules based on user-defined thresholds. This pruning is used for DECLARE discovery in [3], [4] and for DCRgraphs discovery in [8]. These techniques are mainly based on confidence and support measures. Nevertheless, the use of support and confidence is not sufficient to avoid a large

number of spurious results, thus threatening their statistical validity [9]. In addition, the definitions of these metrics are also different for different techniques. For instance, the support measure presented in [3] is different from the support of [4]. Also, both are defined for the sole DECLARE template set.

In the area of conformance checking, de Leoni et al. [1] use alignments and fitness to measure the degree of conformance of a DECLARE model with respect to an execution trace. Polyvyanyy et al. [10] rely on entropy to measure precision and recall of both procedural and declarative models. Finally, Burattin et al. [11] use specific measures like fulfillment ratio and violation ratio based on the evaluation of the number of activations of a rule that lead to a fulfillment and the number of activations that lead to a violation in a log. Quality measures are critical for all these techniques. However, the metrics provided in these works are bound to the specific techniques and the precise modeling languages used. Also, those techniques do not provide a general measurement framework that can be applied for any type of temporal-logic rule.

In software engineering, a number of works employ techniques for the discovery and quality assessment of temporal patterns. This stream of research is called specification mining. Yang et al. [12] discover 2-value temporal patterns using a trace measure that quantifies partial satisfactions of a rule. Yet, the technique lacks generality as it is limited only to alternation patterns (similar to ALTERNATERESPONSE and ALTERNATEPRECEDENCE in Table I) and the adopted computation heuristics are tailored to the software domain. Le et al. [6] emphasize the limits of using only support and confidence measures and investigate properties of other measures reviewed in [13]. Their results demonstrate that there are several measures outperforming support and confidence, and that the combination of different measures yields better results. However, they limit their study to 2-value temporal patterns (specifically, RESPONSE and PRECEDENCE in Table I). Furthermore, their computation of the probability for a temporal specification is based on a sliding window technique [14]: the traces are read in chunks of the size of a given window, then the probability of a rule is the percentage of windows in which it is valid. They test the effect of different window sizes, showing that their results depend not only on the input rules and the data, but also on this parameter selection. Lemieux et al. [15] extend specification mining to arbitrary Linear Temporal Logic (LTL) specifications (implicitly on finite traces) beyond 2-value templates. Yet, they resort only on support and confidence measures to prune uninteresting results, thus incurring in the already mentioned statistical limits [9].

In summary, we observe that measures are largely used as a tool to support research, but they are hardly made subject to evaluative research themselves. Next, we define preliminaries for a framework to assess various measures.

III. PRELIMINARIES

As the formal foundations of our framework, we consider rules specified in Linear Temporal Logic on Finite Traces (LTL_f) [16], as used in DECLARE [5], [17]. LTL_f has exactly the same syntax of LTL [18]. Its semantics is interpreted on finite traces, and hence takes into account that business processes are assumed to terminate sooner or later [19]. DECLARE focuses on some specific LTL_f formulas. Table I illustrates some of the most important rules for business process modeling.

Table I: Some DECLARE constraints expressed as RCons

Constraint	LTL_f expression [16]	RCon
PARTICIPATION(a)	$\Diamond \mathsf{a}$	$t_{\text{Start}} \boxdot\!\!\rightarrow \Diamond \mathsf{a}$
INIT(a)	a	$t_{\text{Start}} \boxdot\!\!\rightarrow \mathsf{a}$
END(a)	$\Box\Diamond \mathsf{a}$	$t_{\text{End}} \boxdot\!\!\rightarrow \mathsf{a}$
ATMOSTONE(a)	$\Box(\mathsf{a} \rightarrow \bigcirc(\neg\Diamond \mathsf{a}))$	$\mathsf{a} \boxdot\!\!\rightarrow \bigcirc(\neg\Diamond \mathsf{a})$
RESPONDEDEXISTENCE(a, b)	$\Diamond \mathsf{a} \rightarrow \Diamond \mathsf{b}$	$\mathsf{a} \boxdot\!\!\rightarrow (\Diamond \mathsf{b} \vee \Diamond\!\!\!-\, \mathsf{b})$
RESPONSE(a, b)	$\Box(\mathsf{a} \rightarrow \Diamond \mathsf{b})$	$\mathsf{a} \boxdot\!\!\rightarrow \Diamond \mathsf{b}$
ALTERNATERESPONSE(a, b)	$\Box(\mathsf{a} \rightarrow \Diamond \mathsf{b}) \wedge \Box(\mathsf{a} \rightarrow \bigcirc(\neg \mathsf{a}\ \mathbf{W}\ \mathsf{b}))$	$\mathsf{a} \boxdot\!\!\rightarrow \bigcirc(\neg \mathsf{a}\ \mathbf{U}\ \mathsf{b})$
CHAINRESPONSE(a, b)	$\Box(\mathsf{a} \rightarrow \Diamond \mathsf{b}) \wedge \Box(\mathsf{a} \rightarrow \bigcirc \mathsf{b})$	$\mathsf{a} \boxdot\!\!\rightarrow \bigcirc \mathsf{b}$
PRECEDENCE(a, b)	$\neg \mathsf{b}\ \mathbf{W}\ \mathsf{a}$	$\mathsf{b} \boxdot\!\!\rightarrow \Diamond\!\!\!-\, \mathsf{a}$
ALTERNATEPRECEDENCE(a, b)	$(\neg \mathsf{b}\ \mathbf{W}\ \mathsf{a}) \wedge \Box(\mathsf{b} \rightarrow \bigcirc(\neg \mathsf{b}\ \mathbf{W}\ \mathsf{a}))$	$\mathsf{b} \boxdot\!\!\rightarrow \ominus(\neg \mathsf{b}\ \mathbf{S}\ \mathsf{a})$
CHAINPRECEDENCE(a, b)	$(\neg \mathsf{b}\ \mathbf{W}\ \mathsf{a}) \wedge \Box(\bigcirc \mathsf{b} \rightarrow \mathsf{a})$	$\mathsf{b} \boxdot\!\!\rightarrow \ominus \mathsf{a}$

Specifically useful for our purposes is LTLp_f, which is an extension of LTL_f supporting the expression of properties of the past (hence the "p" suffix) [7]. Well-formed LTLp_f formulae are built from an alphabet $\Sigma \supseteq \{\mathsf{a}\}$ of propositional symbols and are closed under the boolean connectives, the unary temporal operators $\bigcirc$ (next) and $\ominus$ (previous), the binary temporal operators $\mathbf{U}$ (Until) and $\mathbf{S}$ (Since):

$$\varphi ::= \mathsf{a} | (\neg\varphi) | (\varphi_1 \wedge \varphi_2) | (\bigcirc\varphi) | (\varphi_1\ \mathbf{U}\ \varphi_2) | (\ominus\varphi) | (\varphi_1\ \mathbf{S}\ \varphi_2).$$

From these basic operators, it is possible to derive: classical boolean abbreviations *True*, *False*, $\vee$, $\rightarrow$; constant t_{End}, verified as $\neg \bigcirc \mathit{True}$, denoting the last instant of a trace; constant t_{Start}, verified as $\neg \ominus \mathit{True}$, denoting the first instant of a trace; $\Diamond\varphi$ as $\mathit{True}\ \mathbf{U}\ \varphi$ indicating that φ holds true eventually before t_{End}; $\varphi_1\ \mathbf{W}\ \varphi_2$ as $(\varphi_1\ \mathbf{U}\ \varphi_2) \vee \Box\varphi_1$, which relaxes $\mathbf{U}$ as φ_2 may never hold true; $\Diamond\!\!\!-\,\varphi$ as $\mathit{True}\ \mathbf{S}\ \varphi$ indicating that φ holds true eventually in the past after t_{Start}; $\Box\varphi$ as $\neg\Diamond\neg\varphi$ indicating that φ holds true from the current instant till t_{End}; $\boxminus\varphi$ as $\neg\Diamond\!\!\!-\,\neg\varphi$ indicating that φ holds true from t_{Start} to the current instant.

Given a finite trace t of length $n \in \mathbb{N}$, an LTLp_f formula φ is satisfied in a given instant i ($1 \leqslant i \leqslant n$) by induction of the following:
$t, i \models \mathit{True}$; $t, i \not\models \mathit{False}$;
$t, i \models \mathsf{a}$ iff $t(i)$ is assigned with a;
$t, i \models \neg\varphi$ iff $t, i \not\models \varphi$;
$t, i \models \varphi_1 \wedge \varphi_2$ iff $t, i \models \varphi_1$ and $t, i \models \varphi_2$;
$t, i \models \bigcirc\varphi$ iff $i < n$ and $t, i+1 \models \varphi$;
$t, i \models \ominus\varphi$ iff $i > 1$ and $t, i-1 \models \varphi$;
$t, i \models \varphi_1\ \mathbf{U}\ \varphi_2$ iff $t, j \models \varphi_2$ with $i \leqslant j \leqslant n$, and $t, k \models \varphi_1$ for all k s.t. $i \leqslant k < j$;
$t, i \models \varphi_1\ \mathbf{S}\ \varphi_2$ iff $t, j \models \varphi_2$ with $1 \leqslant j \leqslant i$, and $t, k \models \varphi_1$ for all k s.t. $j < k \leqslant i$.
A formula φ is satisfied by a trace t, written $t \models \varphi$ iff $t, 1 \models \varphi$. One of the central properties of LTLp_f, shared with LTL_f, is that one can compute a deterministic finite state automaton (DFS) A_φ such that for every trace t we have $t \models \varphi$ iff t is in the language recognized by A_φ.

IV. Temporal-extended Measurement Framework

In this section, we build on LTLp$_f$ to develop our measurement framework. Above, we discussed that relying only on confidence and support measures can produce spurious results [9] and that various measures beyond them have been proposed in association rule mining for time-unaware rules [13]. Table II presents a comprehensive list of those measures. Some of them, like confidence and support, are used for the temporal specification of processes, though interpreted in different, ad-hoc ways [3], [4], [7]. The framework we propose next is generic as it allows for the usage of any probabilistic measure (including those of Table II) on any rule for the temporal specification of processes. To this end, Section IV-A formalizes the reactive temporal specification of rules, Section IV-B discusses their probabilistic interpretation, and Section IV-C defines the overall framework.

A. Reactive Temporal Specification

Our first building block is the concept of Reactive Constraint (RCon), originally introduced in [7], which we extend here. A rule typically expresses that the occurrence of certain preconditions (activation) implies certain consequences (target). We codify such intuition in RCons based on LTLp$_f$.

Definition 4.1 (Reactive Constraint (RCon)): Given an alphabet $\Sigma \cup \{t_{\text{Start}}, t_{\text{End}}, \mathit{True}, \mathit{False}\}$, let φ_α and φ_τ be LTLp$_f$ formulae over Σ. A *Reactive Constraint (RCon)* Ψ is a pair $(\varphi_\alpha, \varphi_\tau)$ hereafter denoted as $\Psi \triangleq \varphi_\alpha \boxdot\!\!\rightarrow \varphi_\tau$.

An RCon is interpreted as follows: each time the activator is true, the target should be true at that point of the trace. For example, $\mathsf{a} \boxdot\!\!\rightarrow \Diamond \mathsf{b}$ is an RCon describing that every time a (the activator, φ_α) is *True*, then also $\Diamond \mathsf{b}$ (the target, φ_τ) must evaluate to *True*. Because at every event of the trace (i.e., any point in time) both the activator and target can be either *True* or *False*, the possible evaluation of an RCon can result in either of the following four combinations.

Definition 4.2 (RCon evaluation): Given an RCon $\Psi \triangleq \varphi_\alpha \boxdot\!\!\rightarrow \varphi_\tau$ and a trace t of length $n \in \mathbb{N}$, let t_i be the i^{th} event in the trace ($1 \leqslant i \leqslant n$). For each $t_i \in t$ the possible evaluations of Ψ are:

$\varphi_\alpha = \mathit{False}, \varphi_\tau = \mathit{False}$ if $\{t, i \not\models \varphi_\alpha \wedge t, i \not\models \varphi_\tau\}$;
$\varphi_\alpha = \mathit{False}, \varphi_\tau = \mathit{True}$ if $\{t, i \not\models \varphi_\alpha \wedge t, i \models \varphi_\tau\}$;
$\varphi_\alpha = \mathit{True}, \varphi_\tau = \mathit{False}$ if $\{t, i \models \varphi_\alpha \wedge t, i \not\models \varphi_\tau\}$;
$\varphi_\alpha = \mathit{True}, \varphi_\tau = \mathit{True}$ if $\{t, i \models \varphi_\alpha \wedge t, i \models \varphi_\tau\}$.

For example, the first two rows of Table III show the evaluation of RCon $(\Diamond \mathsf{b} \wedge \Diamond \mathsf{e}) \boxdot\!\!\rightarrow (\neg \mathsf{c} \vee \Diamond \mathsf{f})$ in each event of trace $\langle \mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{d}, \mathsf{f}, \mathsf{c}, \mathsf{e}, \mathsf{c}, \mathsf{h} \rangle$. The constraint states that, between the execution of tasks b and e, no occurrence of c is expected unless it is eventually followed by f. Notice that φ_α and φ_τ are evaluated separately at every event of a trace.

The RCon evaluation can be performed efficiently based on the automaton-based techniques defined in [7], adapting it for offline verification. The full description of this aspect goes beyond the scope of the paper, but we briefly outline the rationale here. Intuitively, we resort on [7, Theorem 4]: an RCon can be separated in pure-past, pure-present and pure-future components. The respective sub-formulas contain only past temporal operators, none, or only future ones. Therefore, by mirroring pure-past formulas and reversing their automata, a single replay of the sub-trace from the beginning to the activator event keeps track of the truth value of the pure-past formula till that point. As we know the suffix of the trace, we can apply the same principle to pure-future formulas too: a single replay from the end of the trace to the activator event keeps track of the truth value of the pure-future formula from that point *onwards*. In this way, we evaluate any formula at each event reading the trace only twice: once from t_{Start} to t_{End} (past components) and once from t_{End} to t_{Start} (future components). This implies that the computational cost depends linearly on the number of events in the event log and the number of rules to verify. Specifically, given an event log L with $|L|$ traces, assuming every trace $t \in L$ having length up to n events, and M rules, the cost to verify all rules on L is: $O(|L| \times n \times M)$.

B. Probabilistic interpretation

The evaluation of RCons indicates whether a rule holds true or false within a trace. In real life, traces often contain noise or partially deviate from desired process specifications. In such occasions wherein the trace contains also events that do not satisfy the rule, we are interested in understanding *to what degree* a rule is satisfied. As we have previously defined the notion of satisfaction for φ_α and φ_τ on single events (Def. 4.2), we can devise a probabilistic interpretation for RCons over traces.

Definition 4.3 (Probability of an LTLp$_f$ formula): Given an LTLp$_f$ formula φ and a trace t of length $|t| = n$, we define the probability of φ over $t = \langle t_1, \ldots, t_n \rangle$ as the proportion of events t_i ($i \in [1, n], n \in \mathbb{N}$) satisfying φ.

$$P(\varphi, t) = \frac{|\{i \in [1, n] : t, i \models \varphi\}|}{n}$$

Definition 4.4 (Probability of LTLp$_f$ formulae intersection): Given two LTLp$_f$ formulae φ_1 and φ_2 and a trace t of length n, we define the probability of the intersection of φ_1 and φ_2 over t as the proportion of events $t_i \in t$ satisfying both φ_1 and φ_2.

$$P(\varphi_1 \cap \varphi_2, t) = \frac{|\{i \in [1, n] : t, i \models \varphi_1, t, i \models \varphi_2\}|}{n}$$

The probability of all the RCon evaluations follows from the above definitions.

$$\begin{cases} P(\neg\varphi_\alpha \cap \neg\varphi_\tau, t) = \dfrac{|\{i \in [1, n] : t, i \not\models \varphi_\alpha, t, i \not\models \varphi_\tau\}|}{n} \\ P(\neg\varphi_\alpha \cap \varphi_\tau, t) = \dfrac{|\{i \in [1, n] : t, i \not\models \varphi_\alpha, t, i \models \varphi_\tau\}|}{n} \\ P(\varphi_\alpha \cap \neg\varphi_\tau, t) = \dfrac{|\{i \in [1, n] : t, i \models \varphi_\alpha, t, i \not\models \varphi_\tau\}|}{n} \\ P(\varphi_\alpha \cap \varphi_\tau, t) = \dfrac{|\{i \in [1, n] : t, i \models \varphi_\alpha, t, i \models \varphi_\tau\}|}{n} \end{cases}$$

For example, Table III shows the probabilities resulting form the evaluation of RCon $(\Diamond \mathsf{b} \wedge \Diamond \mathsf{e}) \boxdot\!\!\rightarrow (\neg \mathsf{c} \vee \Diamond \mathsf{f})$ on trace $\langle \mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{d}, \mathsf{f}, \mathsf{c}, \mathsf{e}, \mathsf{c}, \mathsf{h} \rangle$. Probabilities defined as above permit the application, in the context of temporal logic specifications, of measures defined for association rule mining [13], where an

Table II: Probabilistic measures from [13] on association rules in the form *"if A then B"*.

Measure	Formula	Range
Support	$P(AB)$	$[0, 1]$
Confidence/Precision	$P(B\|A)$	$[0, 1]$
Coverage	$P(A)$	$[0, 1]$
Prevalence	$P(B)$	$[0, 1]$
Recall	$P(A\|B)$	$[0, 1]$
Specificity	$P(\neg B\|\neg A)$	$[0, 1]$
Accuracy	$P(AB) + P(\neg A \neg B)$	$[0, 1]$
Lift/Interest	$\frac{P(AB)}{P(A)P(B)}$	$[0, +\infty)$
Leverage	$P(B\|A) - P(A)P(B)$	$[-1, 1]$
Added Value/ Change of Support	$P(B\|A) - P(B)$	$[-1, 1]$
Relative risk	$\frac{P(B\|A)}{P(B\|\neg A)}$	$[0, +\infty)$
Jaccard	$\frac{P(AB)}{P(A) + P(B) - P(AB)}$	$(-\infty, +\infty)$
Certainty factor	$\frac{P(B\|A) - P(B)}{1 - P(B)}$	$(-\infty, +\infty)$
Ø-Coefficient (Linear Correlation Coefficient)	$\frac{P(AB) - P(A)P(B)}{\sqrt{P(A)P(B)P(\neg A)P(\neg B)}}$	$(-\infty, +\infty)$

Measure	Formula	Range
Interestingness Weighting Dependency	$((\frac{P(AB)}{P(A)P(B)})^k - 1) \times P(AB)^m$	$[0, +\infty)$
Yule's Q	$\frac{P(AB)P(\neg A \neg B) - P(A\neg B)P(\neg AB)}{P(AB)P(\neg A \neg B) + P(A\neg B)P(\neg AB)}$	$(-\infty, +\infty)$
Yule's Y	$\frac{\sqrt{P(AB)P(\neg A \neg B)} - \sqrt{P(A\neg B)P(\neg AB)}}{\sqrt{P(AB)P(\neg A \neg B)} + \sqrt{P(A\neg B)P(\neg AB)}}$	$(-\infty, +\infty)$
Klosgen	$\sqrt{P(AB)} \times \max(P(B\|A) - P(B), P(A\|B) - P(A))$	$[-1, 1]$
Gini Index	$P(A) \times (P(B\|A)^2 + P(\neg B\|A)^2) +$ $P(\neg A) \times (P(B\|\neg A)^2 + P(\neg B\|\neg A)^2)$ $-P(B)^2 - P(\neg B)^2$	$[-2, 2]$
Collective Strength	$\frac{P(AB) + P(\neg B\|\neg A)}{P(A)P(B) + P(\neg A)P(\neg B)} \times \frac{1 - P(A)P(B) - P(\neg A)P(\neg B)}{1 - P(AB) - P(\neg B\|\neg A)}$	$(-\infty, +\infty)$
Laplace Correction	$\frac{N(AB) + 1}{N(A) + 2}$	$[0.5, 1]$
J-Measure	$P(AB) \log \frac{P(B\|A)}{P(B)} + P(A\neg B) \log \frac{P(\neg B\|A)}{P(\neg B)}$	$(-\infty, +\infty)$
Two-Way Support Variation	$P(AB) \log_2 \frac{P(AB)}{P(A)P(B)} + P(A\neg B) \log_2 \frac{P(A\neg B)}{P(A)P(\neg B)}$ $+ P(\neg AB) \log_2 \frac{P(\neg AB)}{P(\neg A)P(B)} + P(\neg A \neg B) \log_2 \frac{P(\neg A \neg B)}{P(\neg A)P(\neg B)}$	$(-\infty, +\infty)$
Zhang	$\frac{P(AB) - P(A)P(B)}{\max(P(AB)P(\neg B), P(B)P(A\neg B))}$	$(-\infty, +\infty)$

Measure	Formula	Range
Conviction	$\frac{P(A)P(\neg B)}{P(A\neg B)}$	$[0, +\infty)$
Piatetsky-Shapiro	$P(AB) - P(A)P(B)$	$[-1, 1]$
Cosine	$\frac{P(AB)}{\sqrt{P(A)P(B)}}$	$[0, +\infty)$
Loevinger	$1 - \frac{P(A)P(\neg B)}{P(A\neg B)}$	$(-\infty, 1]$
Information Gain	$\log \frac{P(AB)}{P(A)P(B)}$	$(-\infty, +\infty)$
Sebag-Schoenauer	$\frac{P(AB)}{P(A\neg B)}$	$[0, +\infty)$
Least Contradiction	$\frac{P(AB) - P(A\neg B)}{P(B)}$	$(-\infty, +\infty)$
Odd Multiplier	$\frac{P(AB)P(\neg B)}{P(B)P(A\neg B)}$	$[0, +\infty)$
Example and Counterexample Rate	$1 - \frac{P(A\neg B)}{P(AB)}$	$(-\infty, 1]$
Odds ratio	$\frac{P(AB)P(\neg A \neg B)}{P(A\neg B)P(\neg BA)}$	$[0, +\infty)$
One-Way Support	$P(B\|A) \log_2 \frac{P(AB)}{P(A)P(B)}$	$(-\infty, +\infty)$
Two-Way Support	$P(AB) \log_2 \frac{P(AB)}{P(A)P(B)}$	$(-\infty, +\infty)$

Table III: Evaluation (0 is *False* and 1 is *True*), probabilistic interpretation, and statistics computation of a sample of measures for RCon $(\Diamond\!\!\!-\mathsf{b} \wedge \Diamond\mathsf{e}) \boxdot\!\!\!\rightarrow (\neg\mathsf{c} \vee \Diamond\mathsf{f})$.

Trace $t_1 =$	$\langle$ a,	b,	c,	d,	f,	c,	e,	c,	h $\rangle$
φ_α: $(\Diamond\!\!\!-\mathsf{b} \wedge \Diamond\mathsf{e})$	0	1	1	1	1	1	1	0	0
φ_τ: $(\neg\mathsf{c} \vee \Diamond\mathsf{f})$	1	1	1	1	1	0	1	0	1

$P(\varphi_\alpha) = 6/9$	$P(\neg\varphi_\alpha\varphi_\tau) = 2/9$	$P(\neg\varphi_\alpha\neg\varphi_\tau) = 1/9$
$P(\varphi_\tau) = 7/9$	$P(\varphi_\alpha\neg\varphi_\tau) = 1/9$	$P(\varphi_\alpha\varphi_\tau) = 5/9$

Support: $P(\varphi_\alpha\varphi_\tau) = 0.56$ Confidence: $P(\varphi_\tau|\varphi_\alpha) = 0.83$

Specificity: $P(\neg\varphi_\tau|\neg\varphi_\alpha) = 0.33$ Lift: $P(\varphi_\alpha\varphi_\tau)/(P(\varphi_\alpha)P(\varphi_\tau)) = 1.07$

	Event log	Support	Confidence	Specificity	Lift
$t_1 =$	$\langle$a, b, c, d, f, c, e, c, h$\rangle$	0.56	0.83	0.33	1.07
$t_2 =$	$\langle$b, d, a, f, g, d, e, d$\rangle$	0.88	1.00	0.00	1.00
$t_3 =$	$\langle$a, c, d, b, c, e, f, c$\rangle$	0.38	1.00	0.20	1.14
$t_4 =$	$\langle$b, c, c, e, a$\rangle$	0.4	0.50	0.00	0.83
	Mean	0.55	0.83	0.13	1.01
	Standard deviation	0.23	0.24	0.16	0.13
	Variance	0.05	0.06	0.03	0.02

antecedent A and a consequent B are defined for every rule in the form *"if A then B"*. To that extent, it suffices to map φ_α to A and φ_τ to B, thus having $P(A)$ as $P(\varphi_\alpha)$, $P(B)$ as $P(\varphi_\tau)$, and $P(AB)$ as $P(\varphi_\alpha \cap \varphi_\tau)$. As a result, we can extend any measure defined for association rules to temporal rules as well, including those of Table II. For example, Table III shows few measures computed from the probabilities of RCon $(\Diamond\!\!\!-\mathsf{b} \wedge \Diamond\mathsf{e}) \boxdot\!\!\!\rightarrow (\neg\mathsf{c} \vee \Diamond\mathsf{f})$.

C. Measurement system

Given as input an event log L, a set of Reactive Constraints (RCons) R, and a set of probabilistic measures M, our framework returns the measurement of every measure in M for each constraint in R over log L. More precisely, the output can be reported at three different levels of detail:

Event level: distinct evaluation of φ_α and φ_τ for every constraint in R on each event of each trace in L;

Trace level: measurement of every measure in M for every trace in L for every constraint in R;

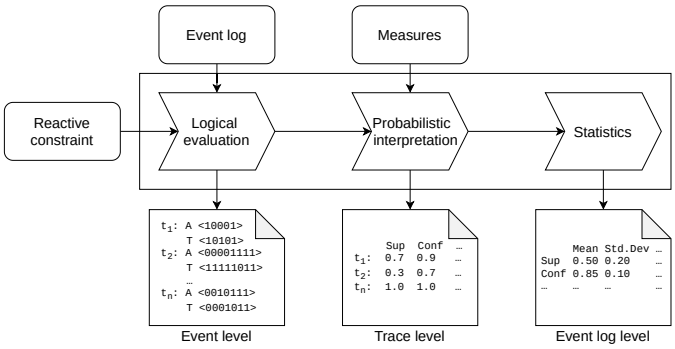

Figure 1: Measurement framework pipeline.

Log level: statistical distribution of every measure in M for every constraint in R.

For example, Table III shows the aggregation of trace-level measures for RCon $(\Diamond\!\!\!-\mathsf{b} \wedge \Diamond\mathsf{e}) \boxdot\!\!\!\rightarrow (\neg\mathsf{c} \vee \Diamond\mathsf{f})$ in a log that consists of 4 traces. Being able to perceive the overall status of a constraint is as important as the possibility to analyze its details in single traces. Therefore, we report the entire statistical distribution of a measure across the log to provide a complete information spectrum. Figure 1 depicts the pipeline of the framework from the input to the output. In the first stage, an RCon is evaluated in each trace of the log. Then, the evaluation result is used to compute probabilities and measures of the rule in each trace. In the final stage, the statistical distribution of each measure over the log is drawn.

The design of the RCons, in particular the choice of the activators, is crucial for the evaluation and the computation of the measures. Let us take as an example the constraint RESPONDEDEXISTENCE(a, b) from the repertoire of DECLARE (see Table I). The classical LTL_f formula underlying RESPONDEDEXISTENCE(a, b) for whole-trace evaluations is $\neg\Diamond\mathsf{a} \vee \Diamond\mathsf{b}$ [16]. However, the formulation of the rule as an RCon can lead to different interpretations:

- $\mathsf{a} \boxdot\!\!\!\rightarrow (\Diamond\mathsf{b} \vee \Diamond\!\!\!-\mathsf{b})$, i.e., when a occurs, b is expected to occur somewhere in the trace;
- $(\Diamond\mathsf{a} \vee \Diamond\!\!\!-\mathsf{a}) \boxdot\!\!\!\rightarrow (\Diamond\mathsf{b} \vee \Diamond\!\!\!-\mathsf{b})$, i.e., for every event of the trace

Table IV: Measurements of a constraint expressed with different formulations on trace $\langle \mathsf{d}, \mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{a} \rangle$.

RCon formulation	Evaluation	Support $P(\varphi_\alpha \varphi_\tau)$	Confidence $P(\varphi_\tau \mid \varphi_\alpha)$
$\mathsf{a} \boxdot\!\!\rightarrow (\Diamond \mathsf{b} \vee \Diamond\!\!\!-\, \mathsf{b})$	φ_α: $\langle 0,1,0,0,1 \rangle$ φ_τ: $\langle 1,1,1,1,1 \rangle$	$2/5 = 0.4$	$2/2 = 1$
$(\Diamond \mathsf{a} \vee \Diamond\!\!\!-\, \mathsf{a}) \boxdot\!\!\rightarrow (\Diamond \mathsf{b} \vee \Diamond\!\!\!-\, \mathsf{b})$	φ_α: $\langle 1,1,1,1,1 \rangle$ φ_τ: $\langle 1,1,1,1,1 \rangle$	$5/5 = 1$	$5/5 = 1$
$\mathit{True} \boxdot\!\!\rightarrow \neg(\Diamond \mathsf{a} \vee \Diamond\!\!\!-\, \mathsf{a}) \vee (\Diamond \mathsf{b} \vee \Diamond\!\!\!-\, \mathsf{b})$	φ_α: $\langle 1,1,1,1,1 \rangle$ φ_τ: $\langle 1,1,1,1,1 \rangle$	$5/5 = 1$	$5/5 = 1$
$t_{\text{Start}} \boxdot\!\!\rightarrow (\neg \Diamond \mathsf{a} \vee \Diamond \mathsf{b})$	φ_α: $\langle 1,0,0,0,0 \rangle$ φ_τ: $\langle 1,1,1,0,0 \rangle$	$1/5 = 0.2$	$1/1 = 1$

such that a occurs either in the past or in the future, also b should occur somewhere in the trace;

- $\mathit{True} \boxdot\!\!\rightarrow \neg(\Diamond \mathsf{a} \vee \Diamond\!\!\!-\, \mathsf{a}) \vee (\Diamond \mathsf{b} \vee \Diamond\!\!\!-\, \mathsf{b})$, i.e., at every event, if a occurs in the trace, also b is expected to occur;
- $t_{\text{Start}} \boxdot\!\!\rightarrow (\neg \Diamond \mathsf{a} \vee \Diamond \mathsf{b})$, i.e., at the beginning of the trace, if a occurs in the trace also b should occur.

All the formulations above are legitimate as they entail that the occurrence of a in the trace demands the occurrence of b. However, the difference in the way the activator is represented turns out to be crucial. The activator, indeed, encodes when the rule is of interest. For example: are we interested in each single occurrence of task a or only in its eventual occurrence in the trace? Do we want the rule to be satisfied in every point of the trace or just at the beginning of the trace? This choice has a clear impact on the measures. Table IV presents the evaluation of a trace with the different formulations seen above and their measurements for the confidence and support measures. While they are all perfectly compliant to the trace (confidence is equal to 1, i.e., each time the activator holds true, also the target holds true), the support varies considerably, i.e., the frequency of $\varphi_\alpha \cap \varphi_\tau$. Notice that this phenomenon comes with neither a good nor with a bad connotation, but stresses the idea that a full control over the formula implies a mindful decision about its design and subsequently on picking the right measures for it.

In summary, we have defined a measurement framework for declarative specifications defined as Reactive Constraints, capable of reporting customizable measures at both trace and event log levels.

V. Implementation and Performance Analysis

We have implemented our measurement framework as a proof-of-concept software prototype on top of an existing declarative process specification discovery tool [7]. The Java source-code can be found at github.com/Oneiroe/Janus. The core software architecture for the verification of RCons is shared with the discovery tool. That is why the new and the old software components are contained in the same repository despite the independence of the two modules (discovery and measurement). All the models used in the following experiments are discovered with [7].

In the remainder of this section, we report on the results of an experimental investigation on the computational performance of our implemented framework. In particular, we assess the performance of the implemented technique against an increase in the data size (i.e., the size of the event log) and the model size (i.e., the number of rules) with synthetic event logs. Finally, we test the performance on a set of real life logs.

Table V: The set of DECLARE rules used in the experiments.

Init(a)	Response(e, f)	ChainResponse(o, p)
End(b)	Precedence(g, h)	ChainPrecedence(q, r)
AtMostOne(c)	AlternatePrecedence(i, l)	RespondedExistence(s, t)
Participation(d)	AlternateResponse(m, n)	

We repeated every experiment 10 times to smooth random factors. The reported results average over the ones of the single repetitions. The machine used for the experiments was equipped with an Intel Core i5-7300U CPU at 2.60GHz, quad-core, 16Gb of RAM and an Ubuntu 18.04.4 LTS operating system.

To test the response of our implemented framework against the input data size, we set up a controlled experiment in which we first generated logs of varying sizes that are compliant with a fixed set of rules, resorting on the simulation engine of MINERful [20]. Thereupon, we computed the measures listed in Table II against all the rules of a larger test model (not fully compliant with the event log). For every run, we recorded the wall-clock time of our prototype.

The starting set of rules stems from the DECLARE repertoire of templates [5] and is provided in Table V. Notice that the set contains all the rule templates seen in Table I and is designed in a way that every constraint insists on different tasks. The test model consists of 649 constraints extracted by the discovery algorithm of Janus (setting the support and confidence threshold parameters to 0.05 and 0.8) from a synthetic event log of 834 963 events, 500 traces and tasks in $[\mathsf{a}, \mathsf{z}]$ that is compliant with the initial model.[1] Given the test model obtained as described above, we performed two tests of 65 iterations each, based on synthetic event logs that comply with the rules of Table V, by: (1) increasing the length of the traces (with a step of 100 events per iteration, keeping the number of traces per event log equal to 500); (2) increasing the number of traces in the event log (with a step of 50 new traces per iteration, keeping the trace lengths between 900 and 1000 events). Figure 2 illustrates the results of both experiments. We observe that the factor actually influencing the wall-clock time is the total amount of events rather than the trace length: indeed, Fig. 2 shows that the recorded timings of both experiments tend to lie on the same line. This experimental result confirms the linear relation between the total number of events in the log and the computational performance illustrated in Section IV-A.

Next, we investigate the response of the framework to an increase in the model size. To do so, we first generated an event log containing 1000 traces with a trace length between 100 and 500 events from the simulation of the rules in Table V. Thereupon, we used the discovery algorithm of Janus to automatically retrieve different test models with varying levels of compliance. To that extent, we made the confidence threshold range from 1.0 (full model compliance), down to 0.0 with a step

[1] Available at github.com/Oneiroe/Janus/blob/master/tests-SJ2T/model.zip

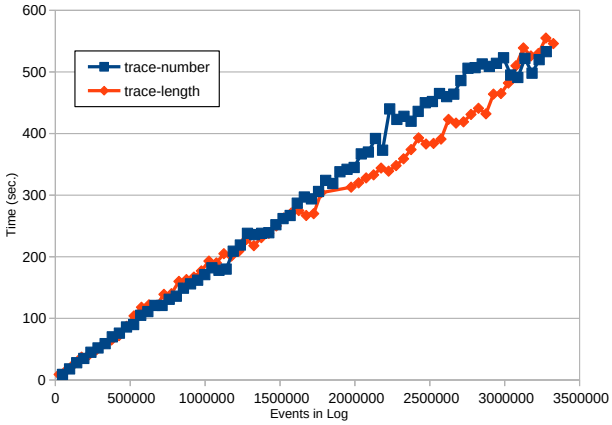

Figure 2: The computation time is linearly dependent on the total number of events in the event log.

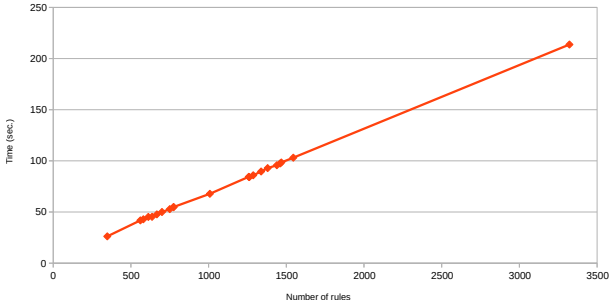

Figure 3: The computation time is linearly dependent on the total number of rules to check.

of 0.05. The rationale is, the lower the confidence threshold, the higher the number of constraints in the test model. Then, we calculated all the measures in Table II for every constraint of each test model. The time taken for the measurements are shown in Fig. 3. Notice that the computation time is linearly dependent on the number of rules to check, thus in line with the theoretical computational cost exposed in Section IV-A.

To test the performance also in real settings, we calculated the measurements on 11 openly available BPICs event logs[2] plus one event log stemming from a partner of a smart-city project in which the authors are involved (labelled as "Smart city" in Table VI). We included the latter event log due to its considerable size: as it can be noticed from the table, it is the one bearing the largest amount of events in this experiment.

For each log, we ran the discovery algorithm of Janus in order to extract a test model to check the event log against. We tuned the parameters of the discovery algorithm to obtain a set of rules that are highly compliant (confidence threshold of 0.8), even if not frequent (support threshold of 0.05). Table VI illustrates the results. For each log we report, along with the number of traces, tasks and events contained in the log and the number of rules in the test model, the total time from the launch to the termination of the software ("Total"), the time to evaluate the rules on every event ("Checks"), and the time to compute the measures aggregated by trace and log ("Measures"). We remark that the wall-clock time remains within acceptable ranges as the slowest run takes around 1.5 minutes to check about 500 constraints.

[2] https://data.4tu.nl/

Table VI: Performance records on real-life datasets.

Event log	Traces	Tasks	Events	Rules	Total [sec]	Check [msec]	Measures [msec]
BPIC12	13087	36	262200	519	94.7	20 589.40	70 414.90
BPIC13_cp	1487	7	6660	20	1.3	129.3	274.9
BPIC13_i	7554	13	65533	14	2.6	389.5	666.5
BPIC14_f	41353	9	369485	51	20.6	3871.70	13 477.10
BPIC15_1f	902	70	21656	3856	37.9	13 197.70	22 796.00
BPIC15_2f	681	82	24678	5889	56	19 199.60	35 130.60
BPIC15_3f	1369	62	43786	4098	72.3	24 449.00	45 503.70
BPIC15_4f	860	65	29403	4690	55	18 605.10	34 459.80
BPIC15_5f	975	74	30030	5164	62.9	21 975.00	39 039.20
RTFMP	150370	11	561470	49	62	12 666.30	43 145.40
SEPSIS	1050	16	15214	260	3.1	710.70	1590.70
Smart city	4347	20	692333	292	67	22 226.60	39 295.50

VI. Evaluation

In this section, we report on experiments that show interesting implications of having a vast availability of measures with customization options. We argue that having different measures yields a more precise characterization of the behavior at hand. Indeed, different measures respond differently to different stimuli in the data. To support this claim, we study the variation in the measures at the injection of specific noise types in the event log that directly or indirectly affect a constraint, thereby assessing their sensitivity or resilience to changes. In fact, while an observable reaction of measures could be desirable when, e.g., noise reveals a change in the process execution, resilience could be preferable, for example, when noise stems from a mere interference or technical issue affecting the information system that records the log. Ultimately, we observe how different measures sense different aspects of a constraint, thus confirming the need of using a multitude of measures and the importance of a proper selection thereof to conduct an in-depth behavioral analysis.

We conducted the experiment as follows. We took as reference model the set of rules in Table V and the synthetic event log that complies with it.[1] Notice that the rules are designed not to interfere with one another as each of them insists on different tasks. In this way, it is possible to observe the response of measures at varying noise levels targeting one constraint at a time, thus diminishing the effect of cross-interference. Thereupon, we injected noise in the event log, resorting on the technique described in [21], and calculated all the measures in Table II for the reference model. In particular, we made use of the following types of noise:

Events insertion: spurious events are included in the traces (mimicking, e.g., double records, alien events, etc.);

Events deletion: events are expunged from the log (mimicking, e.g., missing records, uncommitted transactions, etc.);

White noise: events are randomly inserted and deleted.

Addressing one rule at a time, we studied (1) the direct effect of noise on that constraint by altering the occurrences of its

activator and target via insertions and deletions, and (2) the indirect effect, by altering the occurrences of the other tasks in the log with white noise. We made the noise spread all over the log according to a controlled probability variable. For instance, setting the noise injection as the deletion of occurrences of task a with a probability of 20% results in the removal of 20% of the occurrences of task a from the log, picked at random.

More specifically, for every rule in the set of Table V, we ran a separate experiment for (i) event insertion noise affecting the activator or (ii) the target, (iii) event deletion noise affecting the activator or (iv) the target, (v) white noise affecting neither the activator nor the target. For each of the combinations above, we let the error-injection probability range from 0 to 100 % with a step of 10 %. Because of the random factor, we repeated each experiment 10 times and recorded the average results.

Figure 4 shows the results of such experiments on constraint RESPONSE(e, f). Every line corresponds to a measure. The RESPONSE constraint imposes that the target occurs eventually after each occurrence of the activator. The plots reveal through the starting point and the steepness of the slopes in the curves whether, and to what extent, the corresponding measures consider the frequency of the events satisfying the activator or the target in the traces. As it can be seen, the measures shows different trends for each stimulus.

At large, the RESPONSE constraint appears to be particularly sensitive to the deletions of the target and influenced by both the deletions and insertions of the activator, while mostly insensitive to spurious insertions of the target and white noise. More specifically, we can derive the following observations.

- The deletion of events satisfying the target leads to more violations of the rule (higher $P(\varphi_\alpha \neg\varphi_\tau)$), thus the negative effect is reflected in the drop of many measures based on $P(\varphi_\alpha\varphi_\tau)$ (e.g., confidence, lift, certainty factor). In contrast, the curves of specificity and accuracy grow because of their definition bound to $P(\neg\varphi_\alpha\neg\varphi_\tau)$.
- The deletion of events satisfying the activator, instead, does not bring more rule violations, but only less satisfactions (lower $P(\varphi_\alpha\varphi_\tau)$). Therefore, measures such as relative risk, lift, Zhang, or leverage, decrease. Measures focusing on the target (e.g., prevalence) are basically unaffected.
- The insertion of more events satisfying the target (higher $P(\varphi_\tau)$) does not influence the frequency with which the activator is satisfied. Most of the measures are stable, with a slightly decreasing trend for, e.g., lift or accuracy.
- The insertion of more events satisfying the activator is less characterizing, as the new elements may bring both new rule satisfactions (higher $P(\varphi_\alpha\varphi_\tau)$) and violations (higher $P(\varphi_\alpha\neg\varphi_\tau)$). That is why many measures show an increasing trend (e.g., cosine or Laplace), while others oscillate around the initial value (e.g., confidence). Even the most visible downward trends (see certainty factor, Zhang and Ylue's Y and Q) do not show a totally smooth trend.
- Lastly, the constraint is mostly stable against random alterations that affect neither the activator nor the target. The satisfactions and violations remain constant, whilst the only increase is in $P(\neg\varphi_\alpha\varphi_\tau)$ and $P(\neg\varphi_\alpha\neg\varphi_\tau)$. The slight fluctuations of the measures are due to the variation in the number of events in the traces.

Because of space limits, we cannot illustrate the results for the other rules of Table V. The interested reader can find them at github.com/Oneiroe/Janus/blob/master/tests-SJ2T/NOISE-INJECTION-PLOTS.zip. Overall, generalizing from this specific case, we observe the following: (1) The reaction of the measures to noise depends on the type of stimulus and on whether it affects the target or the activator of the rule. (2) Not all measures sense certain alterations. If a measure is mostly stable with and without an error, it means that it cannot sense that particular stimulus. (3) Some measures have similar trends, but different magnitude. This means that there are "classes" of measures that focus on the same aspects of a rule. (4) The steepness of the curves with which the measure evolves indicates how much the measure is resilient to the change in presence of noise. In particular, it shows the range of tolerance before the error becomes too large to recognize the specific constraint behavior. Notice that sometimes it is desirable to sense if the fundamental characteristics of a log are still visible despite the deviations (e.g., to implement discovery algorithms that are robust to noise).

VII. CONCLUSION

In this paper, we presented a comprehensive measurement framework for declarative specifications modeled as Reactive Constraints. Given an event log and a set of custom probabilistic measures, the framework accepts in input any RCon and returns as output the evaluation of the rule for each event of the log, the computed measures for all the traces, and their statistics over the entire event log. The framework goes beyond the current state of the art as it is not limited to a specific set of measures or rules. The experiments conducted reveal the possibility to characterize the behavior of a given constraint through the combination of different measures, which sense differently the behavior recorded in the log.

Future work. Different possibilities are now open upon the foundations of the measurement framework. It is possible to exploit the possibility to characterize a phenomenon by studying the evolution of different measures for, e.g., dynamic recognition of exceptions in process monitoring [22] or the identification of process drifts [23].

As different measures react differently to different stimuli for different types of rules, a method to select and combine the most appropriate measures depending on the context turns out to be key. To this extent, future research could resort to existing techniques like [24] or develop novel multi-measure heuristics. Also, the measures can be integrated for the assessment of multi-constraint specifications as a whole as in [1], [11].

Many specifications can be considered equivalently valid for static measurements, yet their evolution can show which of them is the most error-tolerant. In this sense, the framework can support the identification of resilient specifications.

While the analysis of multiple measures at once may be overwhelming for a human, machine learning techniques, dealing naturally with multidimensional data, could benefit

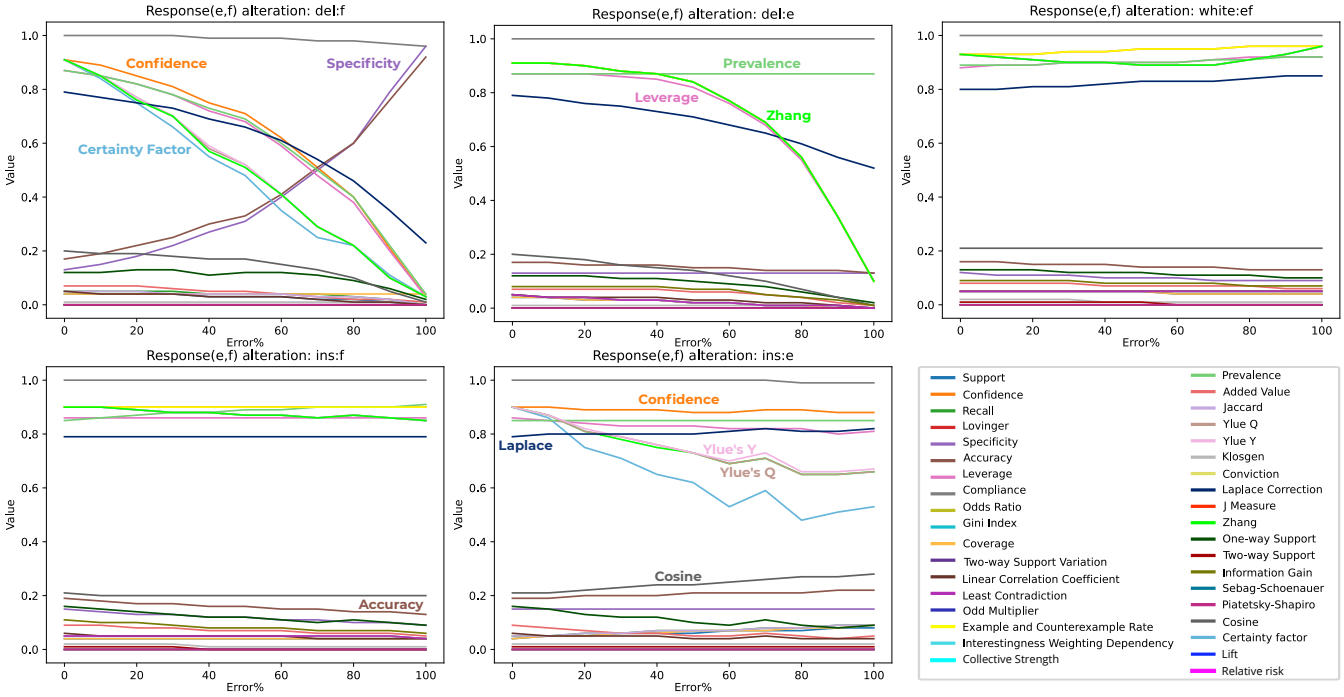

Figure 4: Effect of error injection on constraint RESPONSE(e, f) for all measures.

from the availability of the great amount of information returned by the proposed framework. Therefore, it seems to be also promisingly exploitable for feature selection tasks in sequence classification [25].

ACKNOWLEDGMENT

The work of C. Di Ciccio was partly supported by MIUR under grant "Dipartimenti di eccellenza 2018-2022" of the Department of Computer Science at Sapienza University of Rome.

REFERENCES

[1] M. de Leoni, F. M. Maggi, and W. M. P. van der Aalst, "An alignment-based framework to check the conformance of declarative process models and to preprocess event-log data," *Inf. Syst.*, vol. 47, pp. 258–277, 2015.
[2] L. T. Ly, F. M. Maggi, M. Montali, S. Rinderle-Ma, and W. M. P. van der Aalst, "Compliance monitoring in business processes: Functionalities, application, and tool-support," *Inf. Syst.*, vol. 54, pp. 209–234, 2015.
[3] F. M. Maggi, R. P. J. C. Bose, and W. M. P. van der Aalst, "Efficient discovery of understandable declarative process models from event logs," in *CAiSE*, 2012, pp. 270–285.
[4] 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.
[5] W. M. P. van der Aalst and M. Pesic, "DecSerFlow: Towards a truly declarative service flow language," in *WS-FM*, 2006, pp. 1–23.
[6] T. B. Le and D. Lo, "Beyond support and confidence: Exploring interestingness measures for rule-based specification mining," in *SANER*, 2015, pp. 331–340.
[7] A. Cecconi, C. Di Ciccio, G. De Giacomo, and J. Mendling, "Interestingness of traces in declarative process mining: The Janus LTLp_f approach," in *BPM*, 2018, pp. 121–138.
[8] S. Debois, T. T. Hildebrandt, P. H. Laursen, and K. R. Ulrik, "Declarative process mining for DCR graphs," in *SAC*, 2017, pp. 759–764.
[9] W. Hämäläinen and G. I. Webb, "A tutorial on statistically sound pattern discovery," *Data Min. Knowl. Discov.*, vol. 33, no. 2, pp. 325–377, 2019.
[10] A. Polyvyanyy, A. Solti, M. Weidlich, C. Di Ciccio, and J. Mendling, "Monotone precision and recall measures for comparing executions and specifications of dynamic systems," *ACM Trans. Softw. Eng. Methodol.*, vol. 29, no. 3, Jun. 2020.
[11] A. Burattin, F. M. Maggi, W. M. P. van der Aalst, and A. Sperduti, "Techniques for a posteriori analysis of declarative processes," in *EDOC*, 2012, pp. 41–50.
[12] J. Yang, D. Evans, D. Bhardwaj, T. Bhat, and M. Das, "Perracotta: mining temporal API rules from imperfect traces," in *ICSE*, 2006, pp. 282–291.
[13] L. Geng and H. J. Hamilton, "Interestingness measures for data mining: A survey," *ACM Comput. Surv.*, vol. 38, no. 3, p. 9, 2006.
[14] M. Gabel and Z. Su, "Online inference and enforcement of temporal properties," in *ICSE*, 2010, pp. 15–24.
[15] C. Lemieux, D. Park, and I. Beschastnikh, "General LTL specification mining (T)," in *ASE*, 2015, pp. 81–92.
[16] G. De Giacomo and M. Y. Vardi, "Linear temporal logic and linear dynamic logic on finite traces," in *IJCAI*, 2013, pp. 854–860.
[17] M. Pesic, D. Bosnacki, and W. M. P. van der Aalst, "Enacting declarative languages using LTL: avoiding errors and improving performance," in *SPIN*, ser. LNCS, vol. 6349. Springer, 2010, pp. 146–161.
[18] C. Baier, J.-P. Katoen, and K. Guldstrand Larsen, *Principles of Model Checking*. The MIT Press, 2008.
[19] M. Dumas, M. La Rosa, J. Mendling, and H. A. Reijers, *Fundamentals of Business Process Management, Second Edition*. Springer, 2018.
[20] C. Di Ciccio, M. L. Bernardi, M. Cimitile, and F. M. Maggi, "Generating event logs through the simulation of declare models," in *EOMAS*, 2015, pp. 20–36.
[21] C. Di Ciccio, M. Mecella, and J. Mendling, "The effect of noise on mined declarative constraints," in *SIMPDA*, 2013, pp. 1–24.
[22] F. M. Maggi, M. Montali, M. Westergaard, and W. M. P. van der Aalst, "Monitoring business constraints with linear temporal logic: An approach based on colored automata," in *BPM*, 2011, pp. 132–147.
[23] A. Yeshchenko, C. D. Ciccio, J. Mendling, and A. Polyvyanyy, "Comprehensive process drift detection with visual analytics," in *ER*, 2019, pp. 119–135.
[24] Z. Cao, Y. Tian, T. B. Le, and D. Lo, "Rule-based specification mining leveraging learning to rank," *Autom. Softw. Eng.*, vol. 25, no. 3, pp. 501–530, 2018.
[25] Z. Xing, J. Pei, and E. J. Keogh, "A brief survey on sequence classification," *SIGKDD Explorations*, vol. 12, no. 1, pp. 40–48, 2010.