

Highlights

Measuring Rule-based LTLf Process Specifications: A Probabilistic Data-driven Approach

Alessio Cecconi, Luca Barbaro, Claudio Di Ciccio, Arik Senderovich

- Interestingness measures for rule-based process specifications are introduced.
- A probabilistic estimation framework of single rules and entire specifications is proposed.
- Measures of specifications differ from the aggregation of the measures for their individual rules.
- Measures exhibit different sensitivity to process behavior changes over time.

Measuring Rule-based LTL_f Process Specifications: A Probabilistic Data-driven Approach

Alessio Cecconi^a, Luca Barbaro^b, Claudio Di Ciccio^{b,*} and Arik Senderovich^c

^aVienna University of Economics and Business, Welthandelsplatz 1, Vienna, 1020, Austria

^bSapienza University of Rome, Department of Computer Science, Viale Regina Elena 295, Rome, 00161, Italy

^cYork University, School of Information Technology (ITEC), 4700 Keele St, Toronto, M3J 1P3, Canada

ARTICLE INFO

Keywords:

Linear temporal logic
Declarative process mining
Specification mining
Probabilistic modeling
Statistical estimation

Abstract

Declarative process specifications define the behavior of processes by means of rules based on Linear Temporal Logic on Finite Traces (LTL_f). In a mining context, these specifications are inferred from, and checked on, multi-sets of runs recorded by information systems (namely, event logs). To this end, being able to gauge the degree to which process data comply with a specification is key. However, existing mining and verification techniques analyze the rules in isolation, thereby disregarding their interplay. In this paper, we introduce a framework to devise probabilistic measures for declarative process specifications. Thereupon, we propose a technique that measures the degree of satisfaction of specifications over event logs. To assess our approach, we conduct an evaluation with real-world data, evidencing its applicability for diverse process mining tasks, including discovery, checking, and drift detection.

1. Introduction

The declarative specification of a process allows users and designers to norm and control its behavior through rules. These rules consist of temporal logic formulae (such as LTL_f) that are verified against recorded runs of the process-aware systems in an event log, to check their compliance with the behavioral properties it must guarantee. This automated checking task shows wide adoption in multiple areas of computer science, including process mining [1, 2], planning [3, 4], and software engineering [5, 6]. Rule-based specifications allow for the definition of system and process behavior focusing on core constraints that must be satisfied, leaving other execution details to knowledge- and context-based decisions of the actors [7] with a so-called “open-ended” paradigm [8]. This aspect makes them particularly suitable to scenarios in which workflow flexibility is key [9, 10], as in the healthcare domain [11, 12] with the specification of clinical practice guidelines [13, 14].

Despite the increasing interest in this challenge, we observe that a fundamental problem remains unaddressed. Measuring the extent to which traces adhere to the admissible behavior in terms of *specifications*, or sets of rules, is still a problem that leaves ample margins for investigation.

Let us consider an example inspired by [15] on the clinical pathway for the treatment of unstable angina. A declarative process specification \mathcal{S} , e.g., consists of two rules, Ψ_1 and Ψ_2 . Rule Ψ_1 states that “Whenever a specific combination of medications is prescribed, it should be preceded by an electrocardiogram (ECG) and a cardiac enzyme test.” Rule

Ψ_2 indicates that “If a patient experiences severe chest pain, an immediate administration of sublingual nitroglycerin will follow.” Confidence is the ratio of events in a log that satisfy the consequent (i.e., the *target*, like the ECG and cardiac enzyme test in Ψ_1), given the satisfaction of the antecedent (i.e., the *activator*, like the occurrence of severe chest pain in Ψ_2).

Suppose the confidence of Ψ_1 and Ψ_2 are 100% and 50%, respectively. What is the confidence of $\mathcal{S} = \{\Psi_1, \Psi_2\}$? Considering the confidence of a single formula consisting of the conjunction of all rules (e.g., $\Psi_1 \wedge \Psi_2$) may be too coarse grained, since violating a single rule in one event or in multiple ones would lead to the same result of violating the whole specification for that trace. Similarly, aggregating measures over multiple rules (as in [16]) may be misleading. Let the activator of Ψ_1 , e.g., occur 100 times in the log, always leading to the satisfaction of the rule (Confidence 100%), and the activator of Ψ_2 occur twice, leading to the violation of the rule once (Confidence 50%). Consequently, there will be one violation out of 102 occurrences of the activators. Yet, the average confidence of the two rules will be 75%. A lack of a proper framework to gauge the degree to which specifications are satisfied by, or emerging from, recorded data, hinders their adoption for the discovery, checking and drift-detection of system and process behavior in rule-based settings. In turn, this issue slows down the evolution of dedicated techniques even where those tasks may turn out to be pivotal, like in the highly dynamic and flexible context of clinical pathways.

To overcome this issue, we propose a new approach adapting and extending the concept of Reactive Constraint (RCon), originally proposed in [17], and the measurement framework for single declarative rules expressed as RCons [16]. RCons are rules expressed in an if-then fashion (like Ψ_1 and Ψ_2), namely a pair of LTL_f formulae one of which is the activator (if) and the other is the target (then). RCons cover the full spectrum of declarative process specification

*Corresponding author

✉ alessio.cecconi.phd@gmail.com (A. Cecconi);

luca.barbaro@uniroma1.it (L. Barbaro); claudio.diciccio@uniroma1.it (C. Di Ciccio); sariks@yorku.ca (A. Senderovich)

ORCID(s): 0000-0001-5730-6332 (A. Cecconi); 0000-0002-2975-5330 (L. Barbaro); 0000-0001-5570-0475 (C. Di Ciccio); 0000-0003-4728-8024 (A. Senderovich)

languages such as DECLARE [1] as any LTL_f formula can be translated into an RCon [17]. Equipped with this notion, we propose a measurement framework that takes inspiration from classical association rule mining [18] to assess whether, and in how far, process specifications consisting of LTL_f -based rules expressed in an “if–then” fashion are satisfied by a trace. Our approach is rooted in probability theory and statistical inference. Specifically, in order to provide a non-binary interpretation for specification measurements, we model events of satisfaction and violation of formulae by traces (and logs of traces) using probability theory, and derive corresponding maximum-likelihood estimators for these probabilistic models. Moreover, we show that these estimators can be computed in polynomial time.

To the best of our knowledge, this work is the first to tackle and solve the problem of devising well-defined measures for *entire* declarative specifications consisting of multiple rules. To tackle this problem, we move from an ad-hoc counting approach to a sound probabilistic theory based on maximum likelihood estimation. Finally, we conduct an evaluation of our approach on real-world data with its software prototype implementation. This paper is an extended and revised version of our previous paper [19]. The advancements particularly the following aspects in particular:

- We enhanced the theoretical part of our contribution providing a more comprehensive analysis of the probabilistic approach along with detailed proofs for each proposition and theorem (Sections 5 and 6); in addition, we added an appendix for the basic concepts from probability theory used in the paper (Appendix A);
- We introduced a discussion in which we state the general properties and behaviors of the probability estimators (Section 6.4);
- We added a preliminary section about interestingness measures. It introduces these measures prior to their use thus clarifying the goals of the probabilistic approach and the value of the final outcome (Section 4);
- We extend the quantitative evaluation by using a larger collection of datasets (Section 7.1);
- We present a new and extensive use case where we measure the impact of drifts on declarative specifications. This part demonstrates how measured changes in the entire specification can signal shifts and anomalies in an ongoing process. (Section 7.3).

In the remainder of this paper, Sections 2 to 4 formalize the background notions our work is based upon: LTL_f and its interpretation on event logs, RCons, rule-based declarative process specifications, and interestingness measures. Section 5 lays the foundations of our probabilistic theory, upon which the evaluation and measurement of declarative process specifications are based upon as described in Section 6. We report on the evaluation of our implemented prototype on real-world data in Section 7. Section 8 analyzes the research in the

literature that relates to our investigation. Finally, Section 9 concludes the paper with remarks on future work.

2. Event logs and Linear Temporal Logic on Finite Traces (LTL_f)

In this paper, we are interested in the checking of specifications against collections of traces reporting on multiple executions of the process. As runs can recur, we formalize such structure as a multi-set of traces, namely an event log.

Definition 2.1 (Log). *Given a finite alphabet of propositional symbols Σ , we name as event an assignment for the symbols in Σ and as trace a finite sequence of events. An event log (or log for short) is a finite multi-set of traces $L = \{t_1^j, \dots, t_m^j\}$ of cardinality $|L| = \sum_{i=1}^m j_i$.*

For example, Table 1 presents a log $L = \{t_1^{17}, t_2^6, t_3^5, t_4^{12}, t_5^5\}$ defined over alphabet $\Sigma = \{a, b, c, d, e\}$. Its cardinality is 45.

Linear Temporal Logic on Finite Traces (LTL_f) [20] expresses propositions over linear discrete-time structures of finite length – namely, traces as per Def. 2.1. It shares its syntax with Linear Temporal Logic (LTL) [21] and is at the basis of declarative process specification languages such as DECLARE [7]. Here, we endow LTL_f with past modalities as in [22]. In the remainder of this section (and, as a compendium, in Appendix C), we outline the core concepts of LTL_f to which our approach resorts.

Definition 2.2 (Syntax of LTL_f). *Well-formed Linear Temporal Logic on Finite Traces (LTL_f) formulae are built from an alphabet $\Sigma \supseteq \{a\}$ of propositional symbols, auxiliary symbols ‘(’ and ‘)’, propositional constants True and False, the logical connectives \neg and \wedge , the unary temporal operators \bigcirc (next) and \ominus (yesterday), and the binary temporal operators \mathbf{U} (until) and \mathbf{S} (since) as follows:*

$$\begin{aligned} \varphi ::= & \text{True} | \text{False} | a | (\neg\varphi) | (\varphi_1 \wedge \varphi_2) | \\ & (\bigcirc\varphi) | (\varphi_1 \mathbf{U} \varphi_2) | (\ominus\varphi) | (\varphi_1 \mathbf{S} \varphi_2). \end{aligned}$$

We may omit parentheses when the operator precedence intuitively follows from the expression. Given $\{e, d\} \subseteq \Sigma$, e.g., the following is an LTL_f formula: $(\bigcirc\neg e) \mathbf{U} d$.

Semantics of LTL_f is given in terms of finite traces, i.e., finite words over the alphabet 2^Σ . We name the index of the element in the trace as *instant*. Intuitively, $\bigcirc\varphi$ and $\ominus\varphi$ indicate that φ holds true at the next and previous instant, respectively; $\varphi_1 \mathbf{U} \varphi_2$ states that φ_2 will eventually hold and, until then, φ_1 holds too; dually, $\varphi_1 \mathbf{S} \varphi_2$ signifies that φ_2 holds at some point and, from that instant on, φ_1 holds too. We formalize the above as follows.

Definition 2.3 (Semantics of LTL_f). *Given a finite trace t of length $n \in \mathbb{N}$, an LTL_f formula φ is satisfied at a given instant i ($1 \leq i \leq n$) by induction of the following:*

$$\begin{aligned} (t, i) & \models \text{True}; (t, i) \not\models \text{False}; \\ (t, i) & \models a \text{ iff } a \text{ is True in } t(i); \\ (t, i) & \models \neg\varphi \text{ iff } (t, i) \not\models \varphi; \end{aligned}$$

Table 1: Measurements of RCons $\Psi_1 = c \square \rightarrow \diamond a$, and $\Psi_2 = d \square \rightarrow \diamond e$, and of specification $S = \{\Psi_1, \Psi_2\}$ on a log

Log	Evaluation	RCon / Specification	P of RCon / P of S	P of act.	P of target	Support	Confidence	Recall	Specificity	Lift
$t_1 = \langle a, b, c, d, b, c, e, c, b \rangle$	$\langle x, x, 1, x, x, 1, x, 1, x \rangle$	$\Psi_1 = c \square \rightarrow \diamond a$	1.00	0.33	1.00	0.33	1.00	0.33	0.00	1.00
	$\langle x, x, x, 1, x, x, x, x, x \rangle$	$\Psi_2 = d \square \rightarrow \diamond e$	1.00	0.11	0.78	0.11	1.00	0.14	0.25	1.29
	$\langle x, x, 1, 1, x, 1, x, 1, x \rangle$	$S = \{\Psi_1, \Psi_2\}$	1.00	0.44	0.89	0.44	1.00	0.50	0.20	1.13
$t_2 = \langle b, d, a, b, b, d, e, d, c \rangle$	$\langle x, x, x, x, x, x, x, x, 1 \rangle$	$\Psi_1 = c \square \rightarrow \diamond a$	1.00	0.11	0.78	0.11	1.00	0.14	0.25	1.29
	$\langle x, 1, x, x, x, 1, x, 0, x \rangle$	$\Psi_2 = d \square \rightarrow \diamond e$	0.67	0.33	0.78	0.22	0.67	0.29	0.17	0.86
	$\langle x, 1, x, x, x, 1, x, 0, 1 \rangle$	$S = \{\Psi_1, \Psi_2\}$	0.75	0.44	0.78	0.33	0.75	0.43	0.20	0.96
$t_3 = \langle c, d, a, b, c, e, b, c, b, c \rangle$	$\langle 0, x, x, x, 1, x, x, 1, x, 1 \rangle$	$\Psi_1 = c \square \rightarrow \diamond a$	0.75	0.40	0.80	0.30	0.75	0.38	0.17	0.94
	$\langle x, 1, x, x, x, x, x, x, x, x \rangle$	$\Psi_2 = d \square \rightarrow \diamond e$	1.00	0.10	0.60	0.10	1.00	0.17	0.44	1.67
	$\langle 0, 1, x, x, 1, x, x, 1, x, 1 \rangle$	$S = \{\Psi_1, \Psi_2\}$	0.80	0.50	0.70	0.40	0.80	0.57	0.40	1.14
$t_4 = \langle b, c, a, c, e, a \rangle$	$\langle x, 0, x, 1, x, x \rangle$	$\Psi_1 = c \square \rightarrow \diamond a$	0.50	0.33	0.67	0.17	0.50	0.25	0.25	0.75
	$\langle x, x, x, x, x, x \rangle$	$\Psi_2 = d \square \rightarrow \diamond e$	NaN	0.00	0.83	0.00	NaN	0.00	0.17	NaN
	$\langle x, 0, x, 1, x, x \rangle$	$S = \{\Psi_1, \Psi_2\}$	0.50	0.33	0.50	0.17	0.50	0.33	0.50	1.00
$t_5 = \langle b, b, b \rangle$	$\langle x, x, x \rangle$	$\Psi_1 = c \square \rightarrow \diamond a$	NaN	0.00	0.00	0.00	NaN	NaN	1.00	NaN
	$\langle x, x, x \rangle$	$\Psi_2 = d \square \rightarrow \diamond e$	NaN	0.00	0.00	0.00	NaN	NaN	1.00	NaN
$L = \{t_1^{17}, t_2^6, t_3^5, t_4^{12}, t_5^5\}$ $ L = 45$		$\Psi_1 = c \square \rightarrow \diamond a$	0.80	0.27	0.75	0.22	0.80	0.29	0.27	1.07
		$\Psi_2 = d \square \rightarrow \diamond e$	0.85	0.10	0.69	0.08	0.85	0.12	0.33	1.24
		$S = \{\Psi_1, \Psi_2\}$	0.81	0.37	0.65	0.30	0.81	0.46	0.44	1.25

NaN values denote a division by 0.

- $(t, i) \models \varphi_1 \wedge \varphi_2$ 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 $i \leq j \leq n$, and
 $(t, k) \models \varphi_1$ for all k s.t. $i \leq k < j$;
 $(t, i) \models \varphi_1 \mathbf{S} \varphi_2$ iff $(t, j) \models \varphi_2$ with $1 \leq j \leq i$, and
 $(t, k) \models \varphi_1$ for all k s.t. $j < k \leq i$.

Without loss of generality, we consider here the non-strict semantics of \mathbf{U} and \mathbf{S} [23]. Also, notice that each event in Table 1 satisfies only one proposition (thus applying the ‘‘Declare assumption’’ [24]) for the sake of simplicity. In the following, we might directly refer to the sequence of events $\langle e_1, \dots, e_n \rangle$ of a trace t of length n to indicate the sequence of assignments at instants $1, \dots, n$. For example, t_1, t_2 , and t_4 in Table 1 are written as $\langle a, b, c, d, b, c, e, c, b \rangle$, $\langle b, d, a, b, b, d, e, d, c \rangle$, and $\langle b, c, a, c, e, a \rangle$, respectively. We thus indicate, e.g., that $(t_1, 1) \models a$, $(t_2, 4) \models b$, and $(t_4, 2) \models c$. Considering again the formula $(\bigcirc \neg e) \mathbf{U} d$, we have that $(t_1, 1)$ satisfies it (i.e., the formula is satisfied at the first instant of t_1), whereas $(t_2, 6)$ does not.

From the above operators, the following can be derived:

- Classical boolean abbreviations \vee, \rightarrow ;
- Constant $t_{\text{End}} \equiv \neg \bigcirc \text{True}$, the last instant of a trace;
- Constant $t_{\text{Start}} \equiv \neg \ominus \text{True}$, the first instant of a trace;
- $\diamond \varphi \equiv \text{True} \mathbf{U} \varphi$, indicating that φ holds true at an instant between the current one (included) and t_{End} (we name this operator *eventually*);
- $\diamond \varphi \equiv \text{True} \mathbf{S} \varphi$, indicating that φ holds true at an instant in the closed interval from t_{Start} to the current one (*once*);
- $\square \varphi \equiv \neg \diamond \neg \varphi$, indicating that φ holds true at every instant from the current one till t_{End} (*always*);
- $\boxplus \varphi \equiv \neg \diamond \neg \varphi$, indicating that φ holds true at every instant from t_{Start} to the current one (*historically*).

For example, $d \wedge \diamond e$ is satisfied in a trace when the propositional atom d holds true and e holds true at a later instant in the same trace. Considering the log in Table 1, we have that $(t_2, 6) \models d \wedge \diamond e$ whereas $(t_1, 1) \not\models d \wedge \diamond e$.

Notice that the semantics of the *past operators* $\ominus, \mathbf{S}, \diamond$, and \boxplus correspond to the semantics of the *future operators* $\bigcirc, \mathbf{U}, \diamond$, and \square , respectively, if we evaluate them on finite

traces read in reverse [17]. For example, the evaluation of $(\ominus \neg e) \mathbf{S} d$ on $\langle a, e, c, a, c, b \rangle$ is equivalent to the evaluation of $(\bigcirc \neg e) \mathbf{U} d$ on $\langle b, c, a, c, e, a \rangle$.

Let $\|\varphi\|$ denote the size of the LTL_f formula φ in terms of propositional symbols and connectives excluding parentheses. For example, $\|(\bigcirc \neg e) \mathbf{U} d\|$ is 5 and $\|d \wedge \diamond e\|$ is 4.

Theorem 2.1 [25]. *Let t be a finite trace of length $n \in \mathbb{N}$. Checking whether (t, i) (with $1 \leq i \leq n$) satisfies an LTL_f formula φ , $(t, i) \models \varphi$, is feasible in $O(n^2 \times \|\varphi\|)$.*

Proof. It follows from the proof in [25] elaborated for future operators ($\bigcirc, \diamond, \square$, and \mathbf{U}). Notice that the use of past modalities $\ominus, \boxplus, \diamond$ and \mathbf{S} do not alter the complexity. Indeed, they can be included in the parse tree of the constructive proof in [25] as the respective future counterparts and checked against the trace read in reverse (i.e., from end to start [17]). \square

Corollary 2.1. *Let L be an event log as per Def. 2.1 consisting of $m \in \mathbb{N}$ distinct traces of length up to $n \in \mathbb{N}$ and cardinality $|L| \geq m$. Labeling the events in L that satisfy an LTL_f formula φ is feasible in $O(n^3 \times \|\varphi\| \times m)$.*

Proof. The proof follows from Theorem 2.1: the checking is done for the $O(n)$ events of all m distinct traces in L . \square

3. Reactive Constraints (RCons) and process specifications

In this section, we illustrate how we adopt LTL_f to express rules specifying the behavior of a process in the form of Reactive Constraints (RCons).

3.1. Reactive Constraints (RCons)

RCons express LTL_f-based rules as antecedent-consequent pairs in an ‘‘if-then’’ fashion. Next, we formalize their definition.

Definition 3.1 (Reactive Constraint (RCon)). *Given an alphabet of propositional symbols Σ , let φ_α and φ_τ be LTL_f formulae over Σ . A Reactive Constraint (RCon) Ψ is a pair $(\varphi_\alpha, \varphi_\tau)$ hereafter denoted as $\Psi \triangleq \varphi_\alpha \square \rightarrow \varphi_\tau$. We name φ_α as activator and φ_τ as target.*

We define the semantics of an RCon $\Psi = \varphi_\alpha \square \rightarrow \varphi_\tau$ as follows: given a trace t of length n and an instant i with $1 \leq i \leq n$, we say that

Ψ is satisfied by t in i , i.e., $t, i \models \Psi$, iff $t, i \models \varphi_\alpha$ and $t, i \models \varphi_\tau$

Ψ is violated by t in i , $t, i \not\models \Psi$, iff $t, i \models \varphi_\alpha$ and $t, i \not\models \varphi_\tau$

Ψ is unaffected by t in i , iff $t, i \not\models \varphi_\alpha$.

We also say that Ψ is activated by t if there exists an instant i s.t. $1 \leq i \leq n$ and $t, i \models \varphi_\alpha$. For example, take $\Psi_1 = c \square \rightarrow \diamond a$ in Table 1. At every occurrence of c (the activator), Ψ_1 is either *satisfied* (if c is eventually preceded by a as $\diamond a$ is the target), or *violated*. Ψ_1 is instead *unaffected* by those events in which c does not occur. The activator of $\Psi_2 = d \square \rightarrow \diamond e$ in Table 1 is d and the target is $\diamond e$. Whenever d occurs, it is either satisfied (if eventually followed by e) or violated (otherwise). It is unaffected by events wherein d does not hold. Notice that by declaring that the activator of Ψ_1 is c , the user makes the “trigger” of the rule explicit. Ψ_1 and Ψ_2 are the RCon representation of what are known as PRECEDENCE(c, a) and RESPONSE(d, e) in the declarative process specification language DECLARE [1], respectively.

A Note on the Role of the Activators in RCons. We remark that any LTL_f formula ϕ can be expressed by means of an RCon. The expressiveness of RCons fully covers that of LTL_f and, a fortiori, of DECLARE. The examination of the expressiveness of RCons is discussed in detail in [17, 16]. On the other hand, the temporal conditions that an LTL_f formula such as $\square(\phi_1 \rightarrow \phi_2)$ exerts do not substantially differ from those of $\phi_1 \square \rightarrow \phi_2$. However, we adopt RCons to express rules in a mining context because the explicit definition of the activator makes it possible to distinguish an interesting satisfaction from a vacuous one [26]. The LTL_f formulae $\psi_1 = \square(c \rightarrow \diamond a)$ and $\psi_2 = \square((\neg c \text{ U } a) \vee \neg c)$ both express the PRECEDENCE(c, a) constraint. The rule dictates that, for every event, if c occurs, then it must be preceded by a . This explanation closely resembles ψ_1 . Therefore, if c does not occur, the constraint is also satisfied (regardless of whether a occurs or not: *ex falso sequitur quodlibet*), though vacuously. Notice that the latter condition is explicit in the second conjunct under \square in ψ_2 : “... $\vee \neg c$ ”. Indicating that a rule is satisfied though being unable to distinguish whether it is actually triggered or not adds limited information [27]. The adoption of RCons, though not solving the vacuity problem (activator and target could still be vacuously satisfiable formulae per se), lets the user explicitly define the conditions that make a satisfaction interesting. With $c \square \rightarrow \diamond a$, e.g., we state that the rule is unaffected (neither violated nor satisfied) by events in which c does not occur. Notice, however, that alternative expressions can be used to customize the interpretation of the rules. To adopt the classical binary interpretation of LTL_f formulae, PRECEDENCE(c, a) can be expressed, e.g., as $\Psi_3 = \text{True} \square \rightarrow ((\neg c \text{ U } a) \vee \neg c)$, or $\Psi_4 = t_{\text{Start}} \square \rightarrow \square(c \rightarrow \diamond a)$. In the first case, Ψ_3 indicates that every event activates the RCon (because *True* holds in every event). Therefore, the satisfaction of the target determines the satisfaction of the constraint. In the second case, the activator of Ψ_4 is t_{Start} . Thus, the first event acts as a representative for

the whole trace as either satisfying (if no events in it violates the constraint) or violating (if at least one event violates it). The constraint is unaffected by the trace in every other event. The definition of well-formed RCons and guidelines for their formulation in a mining context go beyond the scope of this paper, and suggests interesting theoretical investigation for future work.

3.2. Process specifications

Equipped with the above notion of RCons and the rationale behind their use in this context, we can define a process specification as follows.

Definition 3.2 (Rule-based LTL_f process specification). *A rule-based LTL_f process specification (henceforth, specification for short) is a finite non-empty set of RCons $S \triangleq \{\Psi_1, \dots, \Psi_s\}$, with $s \in \mathbb{N}$.*

For example, Table 1 presents a specification $S = \{\Psi_1, \Psi_2\}$ composed by the Ψ_1 RCon above and $\Psi_2 = d \square \rightarrow \diamond e$.

Corollary 3.1. *Let $S = \{\Psi_1, \dots, \Psi_s\}$ be a specification consisting of s RCons, the activator and target of which are of size up to $\|\varphi\|$. Labeling the events in L with the satisfaction of activator and target of every RCon in S is feasible in $O(n^3 \times \|\varphi\| \times |L| \times s)$.*

Proof. The proof follows from Corollary 2.1, as a pair of labels is sufficient for all RCons in the specification. \square

Take, e.g., trace $t_4 = \langle b, c, a, c, e, a \rangle$ from Table 1 and the aforementioned RCon $\Psi_1 = c \square \rightarrow \diamond a$. The activator (c) is satisfied in $(t_4, 2)$ and $(t_4, 4)$. We can thus label every event in t_4 thereby creating a new sequence as follows: $\langle 0, 1, 0, 1, 0, 0 \rangle$ where 1 and 0 indicate a satisfaction and a violation of the formula in the corresponding event, respectively. Similarly, we can create a sequence of labels denoting whether the target ($\diamond a$) is satisfied: $\langle 0, 0, 1, 1, 1, 1 \rangle$.

A trivial approach to classify traces as compliant with an RCons or not is to check whether no event violates it. Nevertheless, especially in checking contexts, understanding the extent to which a trace and a log satisfy a specification is key [28]. Next, we introduce the interestingness measures, i.e., the quantitative device we employ to quantify the extent of specification satisfaction. Subsequently, we lay the foundations rooted in probabilistic theory to reach this goal.

4. Interestingness Measures for RCons

The association rule mining field has a long history of measures development for association rules, also called interestingness measures [29, 18]. These rules have a standard “if- A -then- B ” form, where A and B are elements that may occur in a phenomenon, e.g., instructions in a set of database transactions, or events in a set of process traces. These measures are based on the (joint) probabilities of the occurrences (and co-occurrences) of A and B with the general goal of understanding whether there is a significant (directional) relation between the two elements or, alternatively, whether their

Table 2: A selection of interestingness measures

Measure	Definition
Support	$P(A \cap B)$
Confidence	$P(B A)$
Recall	$P(A B)$
Specificity	$P(\neg B \neg A)$
Lift	$\frac{P(A \cap B)}{P(A)P(B)}$

co-occurrence is uncorrelated. Historically, the development of these measures began in market basket analysis with the introduction of the A-Priori algorithm [30]. The problem that the algorithm addressed was to discover, given a set of transactions, association rules between sets of frequently co-occurring elements. Yet, the simple co-occurrence does not necessarily imply a correlation between the elements. Therefore, more refined measures that distinguish authentic associations from spurious ones were developed [18].

Let us consider a few examples of such measures, presented in Table 2: Given an “if- A -then- B ” rule, the Support measure, $P(A \cap B)$, quantifies the joint occurrences of the two elements; Confidence, $P(B|A)$, considers the occurrence of B only when we know that A occurred; Specificity, $P(\neg B|\neg A)$, measures the non-occurrence of B given the non-occurrence of A ; Recall, $P(A|B)$, measures the conditional occurrence of A given B ; finally, Lift, $\frac{P(A \cap B)}{P(A)P(B)}$, is the ratio of the probability of co-occurrence of A and B over the product of the individual probabilities of A and B to occur. Intuitively, Support is high when A and B occur frequently together; Confidence is high when B occurs every time A occurs, while Recall is high in the opposite situation, i.e., if A occurs every time B occurs; Specificity is high when B does not occur if A does not occur; Lift is high when the separate probability of A and B occurring is lower than the probability of their joint occurrence.

Clearly, these measures quantify different aspects of the rule and, depending on the needs of the user, one measure may turn out to be more useful than another. A plethora of other interestingness measures have been defined in the literature. We refer the reader to existing surveys for an extensive overview [18, 31, 32].

In the context of process mining, it has been already shown how to apply these interestingness measures to single RCon rules [16]. By definition, RCons are “if-then” rules too, which makes these measures suitable for interestingness measurement based on the probabilities of the satisfaction of the activator and target conditions in a trace or a log.

The main goal of this paper is to extend these results to specifications of RCons. To this end, we must assume a probabilistic model and derive sound statistical estimators for specifications satisfied by traces and, subsequently, logs.

5. Estimators for LTL_f formulae

The interestingness measures for RCons are based on the probabilities of their activator (φ_α) and target (φ_τ) LTL_f

formulae. In this section, we propose estimators for the probabilities of traces and logs satisfying LTL_f formulae and show that these estimators are computable in polynomial time.

5.1. Trace estimators

We start by defining probabilistic models for the evaluation of formulae over traces. One can consider the probability of an event in a given trace $t = \langle e_1, \dots, e_n \rangle$ to satisfy an LTL_f formula φ as the degree to which φ is satisfied in that trace, which we denote as $P(\varphi(t))$.

Throughout this work, we assume the existence of a labeling mechanism Λ that, when given an event e in a trace t and a formula φ , marks the event with 1 if the event satisfies φ or with 0 otherwise, i.e., $\Lambda(e, \varphi) \in \{0, 1\}$. This procedure can be achieved in polynomial time as shown in Corollary 2.1 through automata-based techniques for LTL_f formulae verification [16].

Therefore, every trace t can be associated with a binary sequence $x_{\varphi,t}$ as follows:

$$x_{\varphi,t} = \langle \Lambda(e_1, \varphi), \dots, \Lambda(e_n, \varphi) \rangle.$$

Take again, e.g., $t_2 = \langle e_{2,1}, \dots, e_{2,9} \rangle = \langle b, d, a, b, b, d, e, d, c \rangle$ from Table 1, and formula $\varphi = d \wedge \Diamond e$. As only $(t_2, 2)$ and $(t_2, 6)$ satisfy φ , $\Lambda(e_{2,1}, \varphi)$ and $\Lambda(e_{2,6}, \varphi)$ return 1 while $\Lambda(e_{2,i}, \varphi)$ is 0 for every $i \in \{1, \dots, 9\} \setminus \{2, 6\}$, i.e.,

$$x_{\varphi,t_2} = \langle 0, 1, 0, 0, 0, 1, 0, 0, 0 \rangle.$$

In what follows, we assume that $P(\varphi(t))$ (the probability of t to satisfy φ), is independent of the position of the event in the trace. This is an uninformative prior assumption, i.e., we assume that we are unaware of the values of other events and of the event location within the trace when evaluating a specific event. Thus, the sequence $x_{\varphi,t}$ can be viewed as an independent and identically distributed (i.i.d.) draw from a Bernoulli random variable $X_{\varphi,t}$, which takes the value of 1 with probability $P(\varphi(t))$ and 0 otherwise, i.e.,

$$X_{\varphi,t} = \begin{cases} 1, & \text{w.p. } P(\varphi(t)), \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

This leads us to our first estimator, namely that of $P(\varphi(t))$.¹

Proposition 5.1. *The maximum likelihood estimator (MLE)² for $P(\varphi(t))$ where $t = \langle e_1, \dots, e_n \rangle$ is*

$$\widehat{P(\varphi(t))} = \frac{1}{n} \sum_{i=1}^n \Lambda(e_i, \varphi). \quad (2)$$

Proof. Since $X_{\varphi,t}$ is a univariate Bernoulli random variable its MLE is well-established in the literature (see, e.g., [33]). Specifically, it equals to the ratio of ‘successful trials’ over the n trials. \square

¹We write $P(\varphi(t))$ for the probability of specific events such as the satisfaction of a formula, and $P(X = x)$ for the probability that a random variable X is set to a specific value x . See Appendix A for basic probability notation.

²MLE estimators exhibit important statistical properties such as unbiasedness and consistency (see Appendix B).

Returning to our running example (Table 1), the MLE estimator is used to compute the trace probabilities of the φ_α formula (P of act.) and of the φ_τ formula (P of target) for each RCon $\varphi_\alpha \square \rightarrow \varphi_\tau$ in \mathcal{S} and for every trace $t \in L$. Let us consider again trace $t_4 = \langle b, c, a, c, e, a \rangle$ and the RCon $\Psi_1 = \varphi_{\alpha_1} \square \rightarrow \varphi_{\tau_1} = c \square \rightarrow \diamond a$. As we have previously discussed in Section 3, the evaluation of φ_{α_1} and φ_{τ_1} on t_4 leads to the following sequences of labels, respectively: $\langle 0, 1, 0, 1, 0, 0 \rangle$ and $\langle 0, 0, 1, 1, 1, 1 \rangle$. Therefore, we conclude that $P(\widehat{\varphi_{\alpha_1}}(t_4))$ is $\frac{2}{6}$ and $P(\widehat{\varphi_{\tau_1}}(t_4))$ is $\frac{4}{6}$.

In order to obtain measures of interest for formulae and specifications we must, in addition, obtain estimators for the intersection of two LTL_f formulae φ_1 and φ_2 being satisfied by a trace, e.g., $P(\varphi_1(t) \cap \varphi_2(t))$, and for the conditional distribution of φ_1 to be satisfied by trace t conditional on φ_2 being satisfied by the trace, e.g., $P(\varphi_1(t)|\varphi_2(t))$.

The latter will be particularly useful to extend the estimators to entire process specifications. Notice that we provide results for the satisfaction of formulae, yet similar results can be derived for violations by quantifying, e.g., $P(\neg\varphi_1(t) \cap \varphi_2(t))$ and $P(\neg\varphi_1(t)|\varphi_2(t))$. Formalizing the above, we wish to estimate the quantities of interest from a labeled sequence,

$$x_{(\varphi_1, \varphi_2), t} = \langle (\Lambda(e_i, \varphi_1), \Lambda(e_i, \varphi_2)) \rangle_{i=1}^n,$$

for $t = \langle e_1, \dots, e_n \rangle$.

Take, for example, $t_4 = \langle b, c, a, c, e, a \rangle$ from Table 1 and the pair of activator and target of $\Psi_1 = c \square \rightarrow \diamond a$, i.e., $\varphi_{\alpha_1} = c$ and $\varphi_{\tau_1} = \diamond a$, respectively. We have that $x_{(\varphi_{\alpha_1}, \varphi_{\tau_1}), t_4}$ is $\langle (0, 0), (1, 0), (0, 1), (1, 1), (0, 1), (0, 1) \rangle$.

The resulting joint sequence $x_{(\varphi_1, \varphi_2), t}$ is again assumed to be an i.i.d. draw from a *bivariate* Bernoulli random variable $X_{(\varphi_1, \varphi_2), t}$. The bivariate Bernoulli corresponds to four parameters related to the four possible outcomes, namely:

$$X_{(\varphi_1, \varphi_2), t} = \begin{cases} (0, 0), & \text{w.p. } p_{00}, \\ (0, 1), & \text{w.p. } p_{01}, \\ (1, 0), & \text{w.p. } p_{10}, \\ (1, 1), & \text{w.p. } p_{11}, \end{cases} \quad (3)$$

such that $\sum_{i,j} p_{ij} = 1$. A more detailed definition of *bivariate* Bernoulli random variables is given in [34]. The MLE for each p_{ij} is proposed in [35]. When estimating $P(\varphi_1(t) \cap \varphi_2(t))$ we are essentially interested in an estimator for p_{11} , which is the probability that both formulae are satisfied by the trace.

Proposition 5.2. *The MLE for $P(\varphi_1(t) \cap \varphi_2(t))$ given a trace $t = \langle e_1, \dots, e_n \rangle$ is*

$$P(\widehat{\varphi_1(t) \cap \varphi_2(t)}) = \hat{p}_{11} = \frac{1}{n} \sum_{i=1}^n \Lambda(e_i, \varphi_1) \cdot \Lambda(e_i, \varphi_2). \quad (4)$$

Proof. The proof follows from the MLE estimators for Bivariate Bernoulli random variables with corresponding success probabilities, p_{ij} , with $i, j \in \{0, 1\}$ (see [35]). \square

From the example above, we conclude that

$$P(\widehat{\varphi_{\alpha_1}(t_4) \cap \varphi_{\tau_1}(t_4)}) = \frac{1}{6}, \quad (5)$$

as only one element in the sequence is (1, 1). Similarly to $P(\varphi_1(t) \cap \varphi_2(t))$, we can estimate the other combinations of satisfaction and violation of the two formulae (using \hat{p}_{00} , \hat{p}_{01} , and \hat{p}_{10}).

Remark 5.1. *One may be tempted to assume independence between the two sequences $\Lambda(e_i, \varphi_1)_{i=1}^n$ and $\Lambda(e_i, \varphi_2)_{i=1}^n$. However, this is seldomly the case. Consider two formulae: one that requires activity a to happen at least once in t , and the other that requires that activity a never occurs. Clearly, the evaluation of these two formulae would not be independent.*

Having modeled the joint probability of two LTL_f formulae satisfied by a trace, we can now define the probability of one formula being satisfied (or violated) by t conditioned on another formula being satisfied (or violated) by the same trace t . The conditional distribution of the random variable, $X_{\varphi_1, t} | X_{\varphi_2, t} = x_2$, is a *univariate* Bernoulli that depends on the sequence x_2 (which results from applying Λ to φ_2 and t) and on the four parameters p_{ij} of the joint bivariate Bernoulli distribution (see the proof in [34]). This result leads to the following estimator of the conditional probability.

Proposition 5.3. *The MLE for $P(\varphi_1(t)|\varphi_2(t))$ given a trace $t = \langle e_1, \dots, e_n \rangle$ is*

$$P(\widehat{\varphi_1(t)|\varphi_2(t)}) = \frac{\sum_{i=1}^n \Lambda(e_i, \varphi_1) \cdot \Lambda(e_i, \varphi_2)}{\sum_{i=1}^n \Lambda(e_i, \varphi_2)}. \quad (6)$$

Proof. The conditional distribution for bivariate Bernoulli is a univariate Bernoulli with success probability of $\frac{\hat{p}_{11}}{\hat{p}_{01} + \hat{p}_{11}}$ [34].

Therefore, the MLE estimation of $P(\widehat{\varphi_1(t)|\varphi_2(t)})$ is that of a univariate Bernoulli with that success probability [33].

This leads to the following estimator:

$$P(\widehat{\varphi_1(t)|\varphi_2(t)}) = \frac{\hat{p}_{11}}{\hat{p}_{01} + \hat{p}_{11}},$$

with \hat{p}_{11} as derived in Prop. 5.2 and \hat{p}_{01} being

$$\hat{p}_{01} = \frac{1}{n} \sum_{i=1}^n (1 - \Lambda(e_i, \varphi_1)) \cdot \Lambda(e_i, \varphi_2).$$

Therefore,

$$P(\widehat{\varphi_1(t)|\varphi_2(t)}) = \frac{\hat{p}_{11}}{\hat{p}_{01} + \hat{p}_{11}} = \frac{\sum_{i=1}^n \Lambda(e_i, \varphi_1) \cdot \Lambda(e_i, \varphi_2)}{\sum_{i=1}^n \Lambda(e_i, \varphi_2)}. \quad \square$$

Remark 5.2. *When estimating $P(\widehat{\varphi_1(t)|\varphi_2(t)})$, the denominator of the estimator may be equal to 0. In such a case, the conditional probability is ill-defined and the trace is ignored for log-level computations; the value is denoted as NaN.*

Take, e.g., $t_4 = \langle b, c, a, c, e, a \rangle$ from Table 1 as above, $\varphi_{\alpha_1} = c$ and $\varphi_{\tau_1} = \diamond a$, namely the activator and target of Ψ_1 .

We have that $P(\widehat{\varphi_{\tau_1}(t_4)} | \widehat{\varphi_{\alpha_1}(t_4)})$ is $\frac{1}{2}$ as φ_{α_1} is satisfied by two events, only one of which satisfies φ_{τ_1} too.

Similarly, we can estimate the remaining that correspond to the random variable, $X_{\varphi_1,t} | X_{\varphi_2,t} = x_2$, i.e., $P(\neg\varphi_1(t) | \neg\varphi_2(t))$, $P(\varphi_1(t) | \neg\varphi_2(t))$, and $P(\neg\varphi_1(t) | \varphi_2(t))$, since all of them are derived from the random variable $X_{\varphi_1,t} | X_{\varphi_2,t} = x_2$ and the fact that it is a univariate Bernoulli.

5.2. Log estimators

We lift our results from traces to logs by estimating $P(\varphi(L))$, i.e., the probability that the log L satisfies a formula φ . Recall that an event log $L = \{t_1^{j_1}, \dots, t_m^{j_m}\}$ is a bag of traces with trace $t_i^{j_i}$ occurring j_i times in the log. The size of the log $|L| = \sum_{i=1}^m j_i$ is the total number of traces.

Let us denote with $\bar{L} = \{t_1, \dots, t_m\}$ the set of unique traces in L . We assume that the traces in L are independently generated by a trace generator T , which is, in turn, associated with a discrete probability function $P(T = t)$.³ Let \mathcal{T} be the support of the probability distribution of T , i.e.,

$$\mathcal{T} = \{t \mid P(T = t) > 0\}. \quad (7)$$

First, we generalize our definitions from a given trace t to a *random* trace T . To this end, we assume log completeness: the log contains all possible traces that can be generated from T , i.e., $\bar{L} = \mathcal{T}$. We plan to lift this assumption in future work.

Next, we shall define $X_{\varphi,T}$ as a sequence of binary evaluations – similarly to the approach we adopted to define $X_{\varphi,t}$ in Eq. (1), yet over a random trace T . Note that $X_{\varphi,T}$ is essentially a doubly stochastic random variable, as its success probability, $P(X_{\varphi,T} = 1)$, changes for traces randomly sampled from the distribution of T (which we assume to be generalized Bernoulli, i.e., Bernoulli with multiple outcomes). We shall use $X_{\varphi,T}$ together with our log completeness assumption to derive an estimator for $P(\varphi(L))$.

Proposition 5.4. *The MLE for $P(\varphi(L))$ for a log L with a trace set $\bar{L} = \{t_i = \langle e_{i,1}, \dots, e_{i,n_i} \rangle\}_{i=1}^m$ is*

$$\widehat{P(\varphi(L))} = \frac{1}{|L|} \sum_{i=1}^m \frac{j_i}{n_i} \sum_{k=1}^{n_i} \Lambda(e_{i,k}, \varphi). \quad (8)$$

with m being the number of unique traces in L and $|L|$ denoting the number of traces in the log.

Proof. From the law of total probability (see Appendix A) we get that

$$\begin{aligned} P(X_{\varphi,T} = x) &= \sum_{t \in \mathcal{T}} P(T = t) P(X_{\varphi,T} = x \mid T = t) = \\ &= \sum_{t \in \mathcal{T}} P(T = t) P(X_{\varphi,t} = x), \end{aligned} \quad (9)$$

³In practice, the trace can be generated via a random walk over, e.g., a finite-state automaton [36].

which provides a link between log-based evaluation of formulae and trace-based evaluation. Since we assume log completeness, we may replace \mathcal{T} with \bar{L} in Eq. (9), and plug in $P(\varphi(L))$ for $P(X_{\varphi,T} = 1)$ and $P(\varphi(t))$ for $P(X_{\varphi,t} = 1)$, thus yielding the following:

$$\widehat{P(\varphi(L))} = \sum_{i=1}^m P(\widehat{T = t_i}) \widehat{P(\varphi(t_i))}. \quad (10)$$

The term $\widehat{P(\varphi(t_i))}$ is estimated using Prop. 5.1, and T is assumed to be a generalized Bernoulli random variable which has a known MLE estimator as follows (it is a special case of the Multinomial random variable),

$$P(\widehat{T = t_i}) = \frac{j_i}{\sum_{k=1}^m j_k} = \frac{j_i}{|L|}. \quad (11)$$

The proof for the latter can be found in [37].

Finally, to show that the proposed estimator for $\widehat{P(\varphi(L))}$ is indeed an MLE we use a well-known result in statistics is that any function of multiple MLE estimators yields an MLE estimator [38, Ch. 7], which completes our proof. \square

For example, t_4 has a multiplicity of 12 considering the example log in Table 1. The cardinality of the example log L is 45, so $\widehat{P(t = t_4)}$ is $\frac{12}{45}$. We saw in Section 5.1 that

$\widehat{P(\varphi_{\alpha_1}(t_4))}$ is $\frac{2}{6}$ for $\varphi_{\alpha_1} = c$. Therefore, the term for $i = 4$ in the summation of Eq. (8) is $\frac{12}{45} \cdot \frac{2}{6} \approx 0.09$ for φ_{α_1} . Extending the sum to all the traces in \bar{L} , we have that the value of $\widehat{P(\varphi_{\alpha_1}(L))}$ is approximately 0.274.

To lift the estimators of intersection and conditional probabilities from traces to logs, we can again apply the law of total probability and derive the following.

Theorem 5.1. *The MLE of $P(\varphi_1(L) \cap \varphi_2(L))$ for a log L with a trace set $\bar{L} = \{t_i = \langle e_{i,1}, \dots, e_{i,n_i} \rangle\}_{i=1}^m$ is*

$$\widehat{P(\varphi_1(\bar{L}) \cap \varphi_2(\bar{L}))} = \frac{1}{|L|} \sum_{i=1}^m \frac{j_i}{n_i} \sum_{k=1}^{n_i} \Lambda(e_{i,k}, \varphi_1) \Lambda(e_{i,k}, \varphi_2). \quad (12)$$

Proof. From the law of total probability, we get that

$$\begin{aligned} P(\varphi_1(L) \cap \varphi_2(L)) &= \sum_{t_i \in \bar{L}} P(T = t_i) P(\varphi_1(T_i) \cap \varphi_2(T_i) \mid T_i = t_i) = \\ &= \sum_{t_i \in \bar{L}} P(T = t_i) P(\varphi_1(t_i) \cap \varphi_2(t_i)), \end{aligned}$$

which leads to the following estimate:

$$\begin{aligned} \widehat{P(\varphi_1(L) \cap \varphi_2(L))} &= \sum_{t_i \in \bar{L}} \widehat{P(T = t_i)} \widehat{P(\varphi_1(t_i) \cap \varphi_2(t_i))} = \\ &= \frac{1}{|L|} \sum_{i=1}^m \frac{j_i}{n_i} \sum_{k=1}^{n_i} \Lambda(e_{i,k}, \varphi_1) \Lambda(e_{i,k}, \varphi_2). \end{aligned} \quad (13)$$

To obtain the result in Eq. (13) we estimate $P(T = t_i)$ as in Prop. 5.4 (Multinomial MLE) and $P(\varphi_1(t) \cap \varphi_2(t))$ as in Prop. 5.2 (bivariate Bernoulli). Since both are MLEs, we get that $\widehat{P(\varphi_1(L) \cap \varphi_2(L))}$ is an MLE [38, Ch. 7]. \square

In Section 5.1, e.g., we showed that $P(\widehat{\varphi_{\alpha_1}(t_4) \cap \varphi_{\tau_1}(t_4)})$ is $\frac{1}{6}$ considering the example log, φ_{α_1} as above, and $\varphi_{\tau_1} = \diamond a$. Recalling that $P(t = t_4)$ is $\frac{12}{45}$, we have that the term of the summation in Eq. (12) for $t = t_4$ is $\frac{12}{45} \cdot \frac{1}{6} \cong 0.04$. The value of $P(\widehat{\varphi_{\alpha_1}(L) \cap \varphi_{\tau_1}(L)})$ is computed by summing up the elements obtained for every $t \in \bar{L}$, thus obtaining 0.219.

Lastly, we show an estimator for the conditional distribution $P(\varphi_1(L) \mid \varphi_2(L))$ – similarly, one can derive estimators for the other conditional probabilities as for traces.

Theorem 5.2. *The MLE for $P(\varphi_1(L) \mid \varphi_2(L))$ for a log L with a trace set $\bar{L} = \{t_i = \langle e_{i,1}, \dots, e_{i,n_i} \rangle\}_{i=1}^m$ is*

$$P(\widehat{\varphi_1(L) \mid \varphi_2(L)}) = \frac{\sum_{i=1}^m \frac{1}{n_i} \sum_{k=1}^{n_i} \Lambda(e_{i,k}, \varphi_1) \cdot \Lambda(e_{i,k}, \varphi_2)}{\sum_{i=1}^m \frac{1}{n_i} \sum_{k=1}^{n_i} \Lambda(e_{i,k}, \varphi_2)}. \quad (14)$$

Proof. By definition of conditional probability we get that

$$P(\widehat{\varphi_1(L) \mid \varphi_2(L)}) = \frac{P(\widehat{\varphi_1(L) \cap \varphi_2(L)})}{P(\widehat{\varphi_2(L)})},$$

with $P(\widehat{\varphi_1(L) \cap \varphi_2(L)})$ estimated as in Theorem 5.1, and $P(\widehat{\varphi_2(L)})$ as in Prop. 5.4. Plugging the two expressions gets us to the estimator in Eq. (14). We once more use the result in [38, Chapter 7] to prove that the estimator is an MLE. \square

For instance, we showed above that for the example log L in Table 1 the estimations $P(\widehat{\varphi_{\alpha_1}(L) \cap \varphi_{\tau_1}(L)})$ and $P(\widehat{\varphi_{\alpha_1}(L)})$ are 0.219 and 0.274, respectively. By applying the computation above, we have that $P(\widehat{\varphi_{\tau_1}(L) \mid \varphi_{\alpha_1}(L)})$ is $\frac{0.219}{0.274} \cong 0.80$.

We conclude this section by highlighting that the computation of the estimators described thus far is tractable.

Theorem 5.3. *The estimators for LTL_f formulae being satisfied by a trace or a log, and the intersection and conditional probabilities thereof are computable in polynomial time.*

Proof. The proof relies on Theorem 2.1 and Corollary 2.1: once we have checked and labeled the traces, the computation of estimators requires only queries over the resulting labels, which can be performed in $O(|L| \times n)$ considering n as the length of the longest trace in the log. \square

6. Evaluation and measurement of specifications

In the previous section, we estimated the probabilities of any LTL_f formula. Yet, as the evaluation of an RCon differs from that of an LTL_f formula (see Section 3), the

evaluation of a specification consisting of RCons should take into account the interplay of activators and targets. The key point is in how to evaluate an *intersection* of RCons (i.e., a specification) on events. The rationale is that once we have the evaluation of the specification on every event, we can estimate the probabilities and consequently its measures like for any other RCon. In the remainder of this section, we start formalizing the semantics of RCon intersections in Section 6.1, continue proposing estimators of probabilities of traces and log satisfying such specifications (Section 6.2.2), describe the computation of interestingness measures (Section 6.3), and finally highlight advantages and practical implications of our approach (Section 6.4).

6.1. Evaluating specifications

Formally, we define the semantics of a specification S as follows: given a trace t of length n , an instant i with $1 \leq i \leq n$, and a specification $S \triangleq \{\Psi_1, \dots, \Psi_s\}$, with $s \in \mathbb{N}$ and $\Psi_j = \varphi_{\alpha_j} \square \rightarrow \varphi_{\tau_j}$ for every j s.t. $1 \leq j \leq s$, we say that

- S is activated by t in i , i.e., $(t, i) \vDash S_\alpha$, iff there exists a $\Psi_j \in S$ s.t. $(t, i) \vDash \varphi_{\alpha_j}$;
- S is satisfied by t in i , $(t, i) \vDash S$, iff $(t, i) \vDash S_\alpha$ and there does not exist any $\Psi_j \in S$ s.t. $(t, i) \not\vDash \Psi_j$;
- S is violated by t in i , $(t, i) \not\vDash S$, iff there exists a $\Psi_j \in S$ s.t. $(t, i) \not\vDash \Psi_j$;
- S is unaffected by t in i iff $(t, i) \not\vDash S_\alpha$.

In other words, S is activated if at least one of its RCons is activated, satisfied if all and only its activated RCons are satisfied, violated if at least one activated RCon is violated, and unaffected if it is not activated.

In light of the above, we can express a specification $S = \{\Psi_1, \dots, \Psi_s\}$ as an RCon, $S = S_\alpha \square \rightarrow S_\tau$, where S_α and S_τ are LTL_f formulae expressed as follows:

$$S_\alpha = \bigvee_{j=1}^s \varphi_{\alpha_j}; \quad S_\tau = \bigwedge_{j=1}^s \neg (\varphi_{\alpha_j} \wedge \neg \varphi_{\tau_j}). \quad (15)$$

For example, S from Table 1 is activated when either Ψ_1 or Ψ_2 are (i.e., $S_\alpha = c \vee d$) and it is satisfied when all the activated constraints are satisfied, i.e., $S_\tau = (\neg c \vee \diamond a) \wedge (\neg d \vee \diamond e)$. Hence, S is violated in $(t_3, 1)$, e.g. because Ψ_1 is violated and satisfied in $(t_3, 2)$, because Ψ_2 is satisfied and Ψ_1 is unaffected.

The possibility to reduce a specification to a single RCon is key to defining the corresponding estimators and thereby computing the probability a trace and a log satisfy it.

6.2. Estimators for Specifications

In this part, we define what trace and log estimators for single RCons, as well as for RCon specifications.

6.2.1. Trace Estimators

We first start by estimating the interestingness degree of an RCon. Let Λ_R be an RCon interpreter that takes an event and an RCon $\Psi = \varphi_\alpha \square \rightarrow \varphi_\tau$ and returns a label $\Lambda_R(e, \Psi) \in \{0, \times, 1\}$ corresponding to Ψ being violated, unaffected, and satisfied by e , respectively. The labeling of an

event given an RCon Ψ resorts to Λ (explained in Section 5.1) as follows:

$$\Lambda_R(e, \Psi) = \begin{cases} 0, & \text{if } \Lambda(e, \varphi_\alpha) = 1 \text{ and } \Lambda(e, \varphi_\tau) = 0, \\ 1, & \text{if } \Lambda(e, \varphi_\alpha) = 1 \text{ and } \Lambda(e, \varphi_\tau) = 1, \\ \times, & \text{otherwise.} \end{cases} \quad (16)$$

Notice that \times is a new outcome of the labeling function Λ_R that solely applies to RCons but not to LTL_f formulae (see the definition of Λ in Section 5.1). The second column of Table 1 lists the labels assigned by Λ_R to every event in the traces and all RCons in a sequence. For example, the evaluation of $\Psi_1 = \varphi_{\alpha_1} \square \rightarrow \varphi_{\tau_1} = c \square \rightarrow \diamond a$ on $t_4 = \langle e_{4,1}, \dots, e_{4,6} \rangle = \langle b, c, a, c, e, a \rangle$ is such that $(t_4, 1) \not\models c$, thus $\Lambda_R(e_{4,1}, \Psi_1) = \times$; also, we observe that $(t_4, 2) \models c$ but $(t_4, 2) \not\models \diamond a$, therefore $\Lambda_R(e_{4,2}, \Psi_1) = 0$; finally, we have that $(t_4, 4) \models c$ and $(t_4, 4) \models \diamond a$, so $\Lambda_R(e_{4,4}, \Psi_1) = 1$. Following this approach, we attain the following sequence of labels from t_4 via Λ_R on Ψ_1 : $\langle \times, 0, \times, 1, \times, \times \rangle$.

We aim at estimating the probabilities of *interesting* satisfaction and violation of a given RCon in a trace, which correspond to cases in which the RCon is activated by it (see Section 3.1). Therefore, the corresponding random variable that describes an ‘interesting’ RCon (i.e., an RCon that is activated by a trace and satisfied in it), is

$$X_{\varphi_\tau, t} \mid X_{\varphi_\alpha, t} = 1.$$

As before, the latter is a univariate Bernoulli random variable with success probability of

$$P(\Psi(t)) = P(\varphi_\tau(t) \mid \varphi_\alpha(t)).$$

In other words, $P(\Psi(t))$ is the probability that an RCon Ψ is interestingly satisfied by some trace t . We are now ready to state our first estimation result.

Proposition 6.1. *The MLE of $P(\Psi(t))$ for a given trace $t = \langle e_1, \dots, e_n \rangle$ is*

$$\widehat{P(\Psi(t))} = \frac{\sum_{i=1}^n \Lambda(e_i, \varphi_\tau(t)) \cdot \Lambda(e_i, \varphi_\alpha(t))}{\sum_{i=1}^n \Lambda(e_i, \varphi_\alpha(t))}. \quad (17)$$

Proof. We need to estimate the probability that an RCon is interestingly satisfied by t ,

$$P(\Psi(t)) = P(\varphi_\tau(t) \mid \varphi_\alpha(t)).$$

To this end, we can employ Prop. 5.3 to compute the corresponding MLE:

$$P(\widehat{\varphi_\tau(t) \mid \varphi_\alpha(t)}) = \frac{\sum_{i=1}^n \Lambda(e_i, \varphi_\tau(t)) \cdot \Lambda(e_i, \varphi_\alpha(t))}{\sum_{i=1}^n \Lambda(e_i, \varphi_\alpha(t))}. \quad \square$$

For example, $P(\widehat{\Psi_1(t_4)}) = \frac{1}{2}$, as shown in Section 5.1. Moreover, note that for t_4 , Ψ_2 is never activated, which leads

to $P(\widehat{\Psi_2(t_4)}) = \text{NaN}$. To obtain an MLE for $P(\widehat{\neg\Psi(t)})$ we can write

$$P(\widehat{\neg\Psi(t)}) = 1 - \widehat{P(\Psi(t))}, \quad (18)$$

due to the result that a function of an MLE is an MLE [38].

At this stage, we turn to estimate the probabilities of interest for a specification S (i.e., a set of RCons). Specifically, since a specification is interpreted similarly to a single RCon, once we obtained the interpretation for S_α and S_τ , we apply the labeling mechanism Λ_R again to obtain one of the three possible outcomes:

$$\Lambda_R(e, S) = \begin{cases} 0, & \text{if } \Lambda(e, S_\alpha) = 1 \text{ and } \Lambda(e, S_\tau) = 0, \\ 1, & \text{if } \Lambda(e, S_\alpha) = 1 \text{ and } \Lambda(e, S_\tau) = 1, \\ \times, & \text{otherwise.} \end{cases} \quad (19)$$

Note that φ_α and φ_τ are replaced by S_α and S_τ , respectively, but the event-based labeling mechanism Λ_R remains unchanged. This allows for the re-use of Prop. 6.1 to obtain estimators for interesting satisfaction and violation of specifications denoted by $P(S(t))$ and $P(\neg S(t))$, respectively.

For estimating interesting satisfaction recall that $P(S(t)) = P(S_\tau(t) \mid S_\alpha(t))$. Thus, we get the following result.

Proposition 6.2. *The MLE of $P(S(t))$ for a given trace $t = \langle e_1, \dots, e_n \rangle$ is*

$$\widehat{P(S(t))} = P(\widehat{S_\tau(t) \mid S_\alpha(t)}) = \frac{\sum_{i=1}^n \Lambda(e_i, S_\tau) \cdot \Lambda(e_i, S_\alpha)}{\sum_{i=1}^n \Lambda(e_i, S_\alpha)}. \quad (20)$$

Proof. Since S is an RCon, the MLE for the probability of interestingly satisfying it is given in Prop. 6.1. We plug-in the activator and target of the RCon, S_α and S_τ instead of φ_α and φ_τ , respectively, to complete the proof. \square

Notice that we use a similar notation for single RCons, replacing $\Psi(t)$ with $S(t)$ to denote satisfaction of a specification. For example, $P(\widehat{S(t_4)}) = \frac{1}{2}$, as we consider only 0 or 1 outcomes applying the computation described in Prop. 6.2.

To obtain an MLE for $P(\widehat{\neg S(t)})$ we can write

$$P(\widehat{\neg S(t)}) = 1 - \widehat{P(S(t))}, \quad (21)$$

due to the result that a function of an MLE is an MLE [38].

6.2.2. Log Estimators

To derive log estimators we again assume that L is complete, i.e., $\bar{L} = \mathcal{T}$ with \mathcal{T} being the support of the probability distribution of T . In what follows, we provide a result for a specification of RCons (including the special case that the specification is a single RCon). Let $P(S(L))$ denote the probability of a log L to interestingly satisfy a specification S .

Theorem 6.1. *The MLE of $P(S(L))$ for a log L with a trace set $\bar{L} = \{t_i = \langle e_{i,1}, \dots, e_{i,n_i} \rangle\}_{i=1}^m$ is given by*

Table 3: Interestingness measures from Table 2 defined for specifications

Measure	Trace	Log
Support	$P(S_\alpha \cap S_\tau, t)$	$P(S_\alpha \cap S_\tau, L)$
Confidence	$P(S_\tau S_\alpha, t)$	$P(S_\tau S_\alpha, L)$
Recall	$P(S_\alpha S_\tau, t)$	$P(S_\alpha S_\tau, L)$
Specificity	$P(\neg S_\tau \neg S_\alpha, t)$	$P(\neg S_\tau \neg S_\alpha, L)$
Lift	$P(S_\alpha \cap S_\tau, t)$	$P(S_\alpha \cap S_\tau, L)$
	$\frac{P(S_\alpha, t) P(S_\tau, t)}{P(S_\alpha, L) P(S_\tau, L)}$	$\frac{P(S_\alpha, L) P(S_\tau, L)}{P(S_\alpha, L) P(S_\tau, L)}$

$$\begin{aligned} \widehat{P(S(L))} &= P(\widehat{S_\tau(L)} | \widehat{S_\alpha(L)}) = \\ &= \frac{\sum_{i=1}^m \frac{j_i}{n_i} \sum_{k=1}^{n_i} \Lambda(e_{i,k}, S_\tau) \cdot \Lambda(e_{i,k}, S_\alpha)}{\sum_{i=1}^m \frac{j_i}{n_i} \sum_{k=1}^{n_i} \Lambda(e_{i,k}, S_\alpha)}. \end{aligned} \quad (22)$$

Proof. The proof follows from Theorem 5.2. Specifically, we replace φ_α and φ_τ in Eq. (14) with S_α and S_τ , respectively. \square

Returning to the running example in Table 1, we have that $\widehat{P(S(L))}$ is $\frac{(17 \cdot 0.44) + (6 \cdot 0.33) + (5 \cdot 0.40) + (12 \cdot 0.17) + (5 \cdot 0.0)}{(17 \cdot 0.44) + (6 \cdot 0.44) + (5 \cdot 0.50) + (12 \cdot 0.33) + (5 \cdot 0.0)} \approx 0.81$. To obtain an MLE for $\widehat{P(\neg S(L))}$ we can write

$$\widehat{P(\neg S(L))} = 1 - \widehat{P(S(L))}, \quad (23)$$

due to the result that a function of an MLE is an MLE [38]. Notice that to obtain an estimator for a single RCon with activator φ_α and target φ_τ , we replace S_α and S_τ with φ_α and φ_τ , respectively.

Similarly to Section 5, we conclude by providing a result that states the computational complexity of our technique.

Theorem 6.2. *The estimation of the interestingness measures over specifications of RCon given an event log is polynomial.*

Proof. The proof relies on a similar argument to the proof of Theorem 5.3: each of the estimators is a query over the labeled sequences that can be computed in $O(|L| \times n)$, considering n as the length of the longest trace in L . \square

6.3. Computing Measures of Interestingness for Specifications

Having defined the estimators, we can now quantify the interestingness of rule-based LTL_f process specifications relying on association rule mining measures. Specifically, the estimates that we derive for specifications (i.e., Prop. 6.2 and Theorem 6.1) allow us to compute a plethora of interestingness measures while considering the joint effects of multiple rules at once. Thereby, we advance the state of the art as measures were previously restricted to a single-rule scope [16].

Table 3 shows the measures presented in Table 2 computed for a specification S . In the following, we describe what these measures intuitively mean given a specification and an event log. As previously described, the event log serves

as a means to estimate probabilities, which in turn are the building block of the interestingness measures. We define these measures on two levels. At a trace-level, they gauge interestingness of a specification based on the number of events that satisfy specific conditions. At a log-level, they take into account events *and* traces, which we shall collectively name as “fraction of the log” (or log fraction) for simplicity. Without loss of generality, we will describe the measures at a log-level. As it can be seen in Table 3, the same concepts seamlessly apply by restricting the notion of “log fraction” to that of “trace fraction” as the part of events in a single trace for the trace-level, due to the regularity of the definitions.

To determine the interestingness of a specification, Support considers the fraction of the log satisfying both its activator and its target. Confidence assesses in how far the target is satisfied within the log fraction that satisfies the activator. Dually, Recall reports on the events and traces that satisfy the activator among those that belong to the log fraction satisfying the target. Specificity quantifies the violation of the target within the fraction of the log that violates the activator. Finally, Lift scales Support by the product of the estimated probabilities of activator and target.

In the running example (see Table 1), we use Prop. 6.2 and Theorem 6.1 to compute the measures that appear in the last five columns (Support, Confidence, etc.) for $S = \{\Psi_1, \Psi_2\}$. Observing the result in the last line of Table 1, e.g., we have that $\widehat{P(S(L))}$ is 0.81, which together with the probabilities of activator and target of S yield a Support of 0.3 and Confidence of 0.81.

The measures used here are only an explanatory subset of the available ones. Thanks to the estimators presented throughout Section 6, it is possible to seamlessly compute all the ones presented in [16] and define new ones, based on the probabilities of activators and targets.

6.4. Discussion

Having laid down the theoretical foundations, we turn to discussing the proposed approach from a practical perspective. Intuitively, the probability estimators of an entire specification can be viewed as a relaxed metric of satisfaction, i.e., the extent to which the specification is satisfied in the given trace or log. It is a relaxed notion of satisfaction because it does not employ a strict boolean evaluation for the entire trace (as commonly done when evaluating LTL_f formulae). The logical evaluation is performed at the level of single events, while for traces and logs we use probability theory leading to relaxed statistical estimators. These estimators provide details about the severity of the observed violations in traces and logs.

Furthermore, log estimators are not biased by the length of the traces, but only by their frequency. Consider the following example: given a log of two traces, one composed of 100 events and the other of 10 events, suppose that every event in the first trace satisfies the specification, whereas every event of the second one violates it. Thus, 100 out of 110 events, namely about the 91% of them, satisfy the specification. However, only 50% of the traces satisfy

Table 4: Example of a specification violated by high probability constraints.

Log	Evaluation	RCon / Specification	P of RCon / P of S
$t_1 = \langle a, b, c, d, e, f \rangle$	$\langle 0, 1, 1, 1, 1, 1 \rangle$	$\Psi_1 = True \square \rightarrow \neg a$	0.83
	$\langle 1, 0, 1, 1, 1, 1 \rangle$	$\Psi_2 = True \square \rightarrow \neg b$	0.83
	$\langle 1, 1, 0, 1, 1, 1 \rangle$	$\Psi_3 = True \square \rightarrow \neg c$	0.83
	$\langle 1, 1, 1, 0, 1, 1 \rangle$	$\Psi_4 = True \square \rightarrow \neg d$	0.83
	$\langle 1, 1, 1, 1, 0, 1 \rangle$	$\Psi_5 = True \square \rightarrow \neg e$	0.83
	$\langle 1, 1, 1, 1, 1, 0 \rangle$	$\Psi_6 = True \square \rightarrow \neg f$	0.83
	$\langle 0, 0, 0, 0, 0, 0 \rangle$	$S = \{\Psi_1, \dots, \Psi_6\}$	0.00
$t_2 = \langle a, y, y, y, e, f \rangle$	$\langle 0, 1, 1, 1, 1, 1 \rangle$	$\Psi_1 = True \square \rightarrow \neg a$	0.83
	$\langle 1, 1, 1, 1, 1, 1 \rangle$	$\Psi_2 = True \square \rightarrow \neg b$	1.00
	$\langle 1, 1, 1, 1, 1, 1 \rangle$	$\Psi_3 = True \square \rightarrow \neg c$	1.00
	$\langle 1, 1, 1, 1, 1, 1 \rangle$	$\Psi_4 = True \square \rightarrow \neg d$	1.00
	$\langle 1, 1, 1, 1, 0, 1 \rangle$	$\Psi_5 = True \square \rightarrow \neg e$	0.83
	$\langle 1, 1, 1, 1, 1, 0 \rangle$	$\Psi_6 = True \square \rightarrow \neg f$	0.83
	$\langle 0, 1, 1, 1, 0, 0 \rangle$	$S = \{\Psi_1, \dots, \Psi_6\}$	0.50
$L = \{t_1^5, t_2^{10}\}$ $ L = 15$		$\Psi_1 = True \square \rightarrow \neg a$	0.83
		$\Psi_2 = True \square \rightarrow \neg b$	0.94
		$\Psi_3 = True \square \rightarrow \neg c$	0.94
		$\Psi_4 = True \square \rightarrow \neg d$	0.94
		$\Psi_5 = True \square \rightarrow \neg e$	0.83
		$\Psi_6 = True \square \rightarrow \neg f$	0.83
		$S = \{\Psi_1, \dots, \Psi_6\}$	0.33

it. A trace represents the execution of a process instance. Indeed, regardless of the number of steps required, the main information is the overall outcome from that instance, e.g., to which extent a specification holds true in that trace. This aspect is reflected in our log estimators: Given their definition in Section 6.2.2, it can be intuitively observed that the information of the trace used for the log aggregation is a fraction of events in the trace (i.e., those that satisfy a formula) and not their total number. Notice that the frequency of the trace in the log plays a major role, as it signals the probability of running the process in the way the trace reports.

The combination of rules in a specification is also a pivotal point of the contribution of our approach. Recall that a specification is not simply a container of independent rules, but a unique system composed of their simultaneous interactions. The classical approach to reason about specifications is to build a unique LTL_f formula formed by the logical conjunction of all the rules [39]. The disadvantage of such an approach is that the specification may be considered as violated by an entire trace for the violation of a rule by a single event. Since the specification is a unique system, it is agreeable that a violated rule implies the violation of the entire RCons specification, yet the finer granular level of the analysis confines this violation to a single event, and not to the entire trace. Therefore, an estimator for specifications should score low values in traces only if multiple events violate the rule. Extending the focus on event logs, the estimator should assign low values if numerous events in considerably many traces violate it.

Table 4 exemplifies the above reasoning. The specification $S = \{\Psi_1, \dots, \Psi_6\}$ is composed only of constraints formulated as $True \square \rightarrow \neg z$, i.e., every event should be different from a given event z (with $z \in \{a, \dots, f\}$). In the first trace, t_1 , every single constraint is individually violated only by one event of the trace and satisfied by the others. Accordingly, their estimators assign high values (1.00). Nonetheless, the specification is violated in every single event by at least one

Table 5: Details of the real-world event logs used for the evaluation

Event log	Traces	Tasks	Events
BPIC12 [41]	13 087	36	262 200
BPIC13_cp [42]	1487	7	6660
BPIC14_f [43]	41 353	9	369 485
BPIC15_1f [44]	902	70	21 656
RTFMP [45]	150 370	11	561 470
Sepsis [14]	1050	16	15 214
Help-Desk [46]	4580	14	21 348

constraint, thus the estimator's score for S is 0.00. Instead, every event of the second trace, t_2 , satisfies Ψ_2 , Ψ_3 and Ψ_4 (scoring 1.00), while Ψ_1 , Ψ_5 and Ψ_6 are violated by one event therein each (scoring 0.83). In this case, then, the estimator assigns 0.50 to the specification. Since the numerosity of t_1 in the event log is 5 and the numerosity of t_2 is twice as much (10), the specification on the entire log is assigned 0.33 by the estimator.

This is a desirable behavior: A specification is evaluated for the combination of its constraints, not its single parts. The first trace shows clearly that, despite the single constraints being mostly satisfied, together they do not properly capture the execution of the trace. Thus, the average of the measures would be a descriptive statistic for the single rules but a misleading indicator for the specification. Exploring extensions of weighted evaluations of specifications where specific constraints may be more relevant than others and treated unequally is an interesting future outlook.

7. Evaluation

The goal of this section is to demonstrate how gauging the interestingness of an entire process specification with multiple measures leads to novel results that are not achievable via the analysis of typical, single-rule based measures. To this end, we conduct three experiments. In Section 7.1, we show how the measure of a specification differs from the assemblage of the measures of single rules; in Section 7.2, we provide experimental evidence of the fact that distinct measures show different aspects of a specification, even in the case of full compliance with a Confidence level of 1.0; in Section 7.3, we propose a use-case application of our approach, analyzing its support to process drift detection. Finally, Section 7.4 concludes the section with a discussion of our findings.

We implemented our technique in a proof-of-concept Java tool, publicly available at <https://github.com/Oneiroe/Janus>. The implementation natively supports a relevant set of rule templates based on [40], but we already showed in Section 6 that the technique is seamlessly applicable to any RCon. Furthermore, inspired by [18], it supports the computation of 37 interestingness measures.

In our experiments, we used a set of openly available, real-world event logs. The details and references to the datasets are reported in Table 5. Note that our tool scales linearly with

the size of the event log, the size of the specification, and the number of measures to compute. For example, for the Sepsis dataset the measurement took around 35 s with the heaviest setup (i.e., to compute 37 measures for a specification containing 3202 rules and a log with 1050 traces) on an Intel Core i5-7300U CPU at 2.60 GHz, quad-core, 16 GB of RAM, running Ubuntu 18.04.6.

7.1. Measuring single rules and entire specifications

In the following, we assess the usefulness of specifications measurement on real-life data, applying our technique to the results of various pattern-based LTL_f specification miners, i.e., Janus [17], MINERful [47], and Perracotta [48]. The goal is to highlight the importance of checking an entire specification, as opposed to the analysis of individual rules. The rationale is that while many specification miners use a threshold to retrieve only rules satisfied above that value, the corresponding specification may present a satisfaction degree below the desired level, following Morin's principle that the whole may be less than the sum of its parts [49].

The interested reader can find the scripts and input files to reproduce the tests alongside output reports and the full collection of specification rules at <https://oneiroe.github.io/DeclarativeSpecificationMeasurements-Journal-static/>. We conducted the experiment as follows. We discovered a specification from the log with each miner. Then, we used our tool to compute the interestingness measures on the log. Here we focus on Confidence, as all miners implement a custom calculation for it. We repeated the discovery step with increasing Confidence thresholds, from 0 to 1, with step size of 0.05. Finally, we compared the measures of the specifications to the input threshold.

The results can be found in Figs. 1 to 7. As highlighted in Figs. 1, 4 and 5, every miner returned specifications whose overall Confidence is lower than the initially set threshold, even for thresholds greater than 0.7. This means that although every rule in a specification has a Confidence value greater or equal to the threshold, the overall specification performs worse than the user-defined minimum accepted level. Other datasets (see Figs. 2, 3, 6 and 7), exhibit a similar behavior for lower thresholds.

This issue may lead to sub-optimal results, akin to the multiple testing problem [50] in statistical inference: assuming the independence of each hypothesis, thus not considering their inter-dependency, leads to erroneous outcomes. With our technique, this kind of issue becomes evident. Improving declarative process specification miners with the integration of our technique paves the path for interesting future endeavors. However, we remark that this analysis focuses on the quantitative matching of the specification to the logs, regardless of the semantics of the discovered specifications. Next, we broaden the perspective of interestingness from the sole Confidence to a larger set of measures.

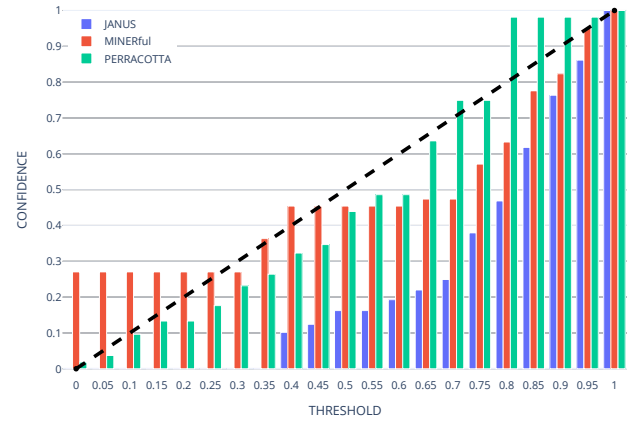


Figure 1: Confidence of the mined specifications with respect to the threshold used for their discovery with the Sepsis dataset [14].

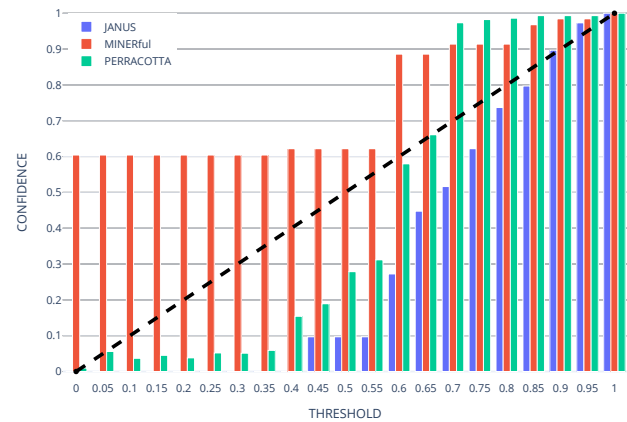


Figure 2: Confidence of the mined specifications with respect to the threshold used for their discovery with the BPIC12 dataset [41].

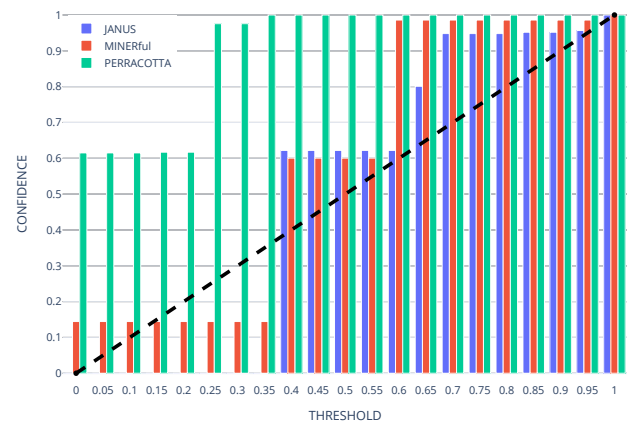


Figure 3: Confidence of the mined specifications with respect to the threshold used for their discovery with the BPIC13 dataset [42].

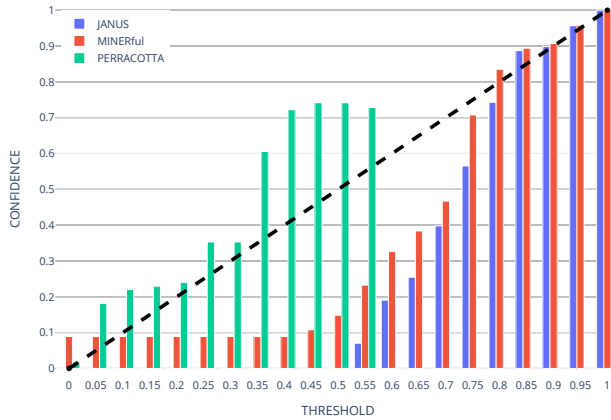


Figure 4: Confidence of the mined specifications with respect to the threshold used for their discovery with the BPIC14_f dataset [43]. Note: Perracotta did not return any rules with thresholds above 0.55 with this dataset.

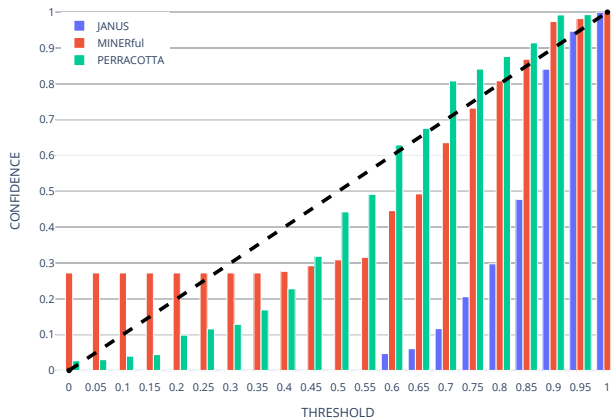


Figure 5: Confidence of the mined specifications with respect to the threshold used for their discovery with the BPIC15_1f dataset [44].

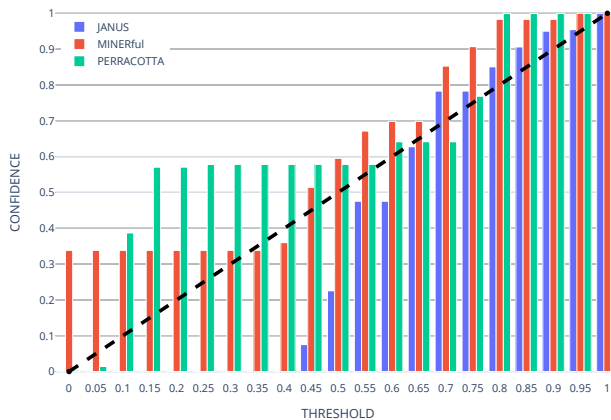


Figure 6: Confidence of the mined specifications with respect to the threshold used for their discovery with the RTFMP dataset [45].

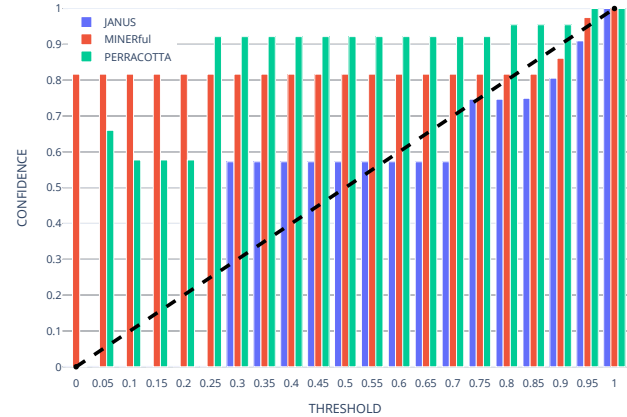


Figure 7: Confidence of the mined specifications with respect to the threshold used for their discovery for the Help-Desk dataset [46].

Table 6: Measurements of sub-specifications

Measure	Original S	S_1	S_2	S_3	S_4	S_5
Confidence	1.000	1.000	1.000	1.000	1.000	1.000
Support	1.000	0.003	0.011	0.074	0.522	0.683
Recall	1.000	0.011	1.000	1.000	1.000	1.000
Specificity	NaN	0.741	1.000	1.000	1.000	1.000
Lift	1.000	3.831	89.691	13.512	1.916	1.465

7.2. Differing measures

In this part, we show how different specifications, though never violated in the log, may exhibit different characteristics through the inspection of multiple measures.

In this experiment, we make use of the Sepsis event log [14]. The dataset refers to a real-world process in the healthcare domain and contains records of patient visits with sepsis symptoms at a Dutch hospital. The dataset exhibits high variability: 75% of the traces are unique, which makes it a good candidate to evaluate partial specifications. To retrieve multiple distinct specifications, we first mined a specification with the miner set to discard partly violated rules. Any miner would be fit for the task, thus without loss of generality we employed Janus with a Confidence threshold set to 1 for all rules. The resulting specification consisted of 238 rules, which we partitioned randomly into 5 subsets of RCons. Finally, we computed interestingness measures for each of the sub-specifications. An excerpt of the results is reported in Table 6.

The results show that despite a Confidence level of 1 for every specification (i.e., there are no violated rules), other measures can still detect differences. For example, Support shows that S_1, S_2 , and S_3 are applied to a very small portion of events of the log, while S_4 and S_5 are activated by more than half of it. Specificity and Recall signal that for every specification the respective targets and activators always occur together, except for S_1 : its target occurs without the activator the vast majority of the time. The division by zero for the original S Specificity suggests

that the specification was activated in every event of the log. Also, Lift shows that, in all the models, the joint probability of activator and target is higher than their individual one, yet with different strengths, e.g., S_2 towering any other one. The input specifications and output measurements are available for reproducibility purposes at <https://oneiroe.github.io/DeclarativeSpecificationMeasurements-Journal-static/>.

To conclude, this experiment demonstrates the advantages of the availability of a full set of measures, going beyond the mere satisfaction of specifications. Indeed, the choice of the most appropriate measure will depend on the specific application of use, as we will exemplify next.

7.3. Impact of process drift on specification measures

Throughout this section, we analyze the impact that drifts [51] in process specifications may have on the various interestingness measures. Our objective is twofold. On the one hand, we aim to undertake a preliminary assessment of the capability of measures to reflect modifications in the process behavior. On the other hand, we want to observe the difference in sensitivity to changes exhibited by the measures originally suggested in the literature for association rule mining [32, 31] and adopted in the context of declarative process mining in [19]. Specifically, we perform the analysis for entire specifications and not only for individual rules.

We used the Visual Drift Detection (VDD) tool [52] as a benchmark to detect the points in which the process is subject to variations over time. VDD divides event logs into sub-logs of consecutive traces, measures the Confidence of all discoverable DECLARE constraints in each sub-log, and analyzes the time sequences generated by the Confidence measurements while grouping together constraints that exhibit similar trends over time. It automatically detects change points within groups based on the oscillation of values in the sequence. Then, it automatically classifies those change points as either process drifts or evidence of erratic behavior and outliers [53].

7.3.1. Experimental setting

Taking the results illustrated in [53, 54] as a reference, we conducted our experiments with the following two real-world event logs: (i) the aforementioned Sepsis log [14], which reports on the tasks executed from the registration to the discharge of patients affected by sepsis, and (ii) the Help-Desk log [46], recording the activities carried out within a management process of the Help-desk of an Italian software company. The scripts we use for our tests can be found alongside the analysis results at the following address: <https://github.com/l2brb/Measurement-change-point-evaluation>.

To carry out our experiments, we went through the following steps. We first mined a process specification consisting of rules with high Confidence (rarely violated whenever triggered) from the event logs taken as a whole. To this end, we used Janus⁴ [17] with the Confidence threshold set to 0.95. Then, we sliced the event log into consecutive sub-logs with

⁴<https://github.com/Oneiroe/Janus/>

a tumbling window approach, namely extracting sequences of 50 consecutive, non-overlapping trace sets. We resorted to the dedicated pre-processing feature of MINERful⁵ [47] to this extent. We fed the sub-logs into VDD for the detection of change points, setting the parameters as follows: (i) 50 for the window size, (ii) 50 for slide size, and (iii) 300 for the cut threshold. Finally, we made our tool compute the value of the measures of the initial specification on every sub-log in a sequence. Thereupon, we analyzed the trend of the specification measurements of the specification in correspondence with the change points detected by VDD, to observe if, and in how far, they exhibit variations. In the following, we discuss the results of our analysis.

7.3.2. Experimental results

Figures 8 and 9 illustrate the trends of Confidence (on the y-axis) for the mined specifications on the sub-logs of the Help-Desk and the Sepsis datasets, respectively. The points on the x-axis represent the timestamp of the first event in every sub-log. We add colored bars in correspondence with the change points detected by VDD. As Yeshchenko et al. explain in [53], two drifts occur in the course of the process executions recorded in the Help-Desk log. They are located in the first and the last quarter of the diagram.

In Fig. 8, we highlight the corresponding points with a dark green bar. The lighter bars refer to the erratic behavior of the process in correspondence with other changing points of lower impact. We observe that all changing points correspond to upward or downward peaks in the poly-line plotting the Confidence values. However, they are not the sole steep slopes that can be noticed in the diagram. The reason is, VDD considers the trend of Confidence values for *all* discoverable constraints. In our case, instead, we focus on a selection of constraints, namely those that were discovered by Janus from the whole log setting a minimum threshold of 95 % for Confidence. Therefore, variations in the measure can be more visible considering the derived specification, while the oscillation may be mitigated by taking into account other constraints that are violated more often. Also, notice that VDD takes every constraint in isolation, while our holistic approach aims at measuring the interestingness of a specification as a whole.

Similar results are visible in Fig. 9. The bars in the diagram highlight the change points identified by VDD. As discussed in [54], none of the detected change points corresponds to actual process drifts. Instead, these oscillations are evidence of the erratic behavior characterizing the event log, most likely due to the reportedly flexible nature of the healthcare process involved [14]; moreover, the latest oscillation is an outlier that occurs towards the end of the event log. We observe that this characteristic is also evidenced by the fact that seasonal oscillations characterize the entire diagram, though their amplitude is limited compared to the Help-Desk case.

Our approach allows one to gauge the interestingness of a specification for additional measures beyond Confidence.

⁵<https://github.com/cdc08x/MINERful/>

Measurement of Rule-based LTLf Declarative Process Specifications

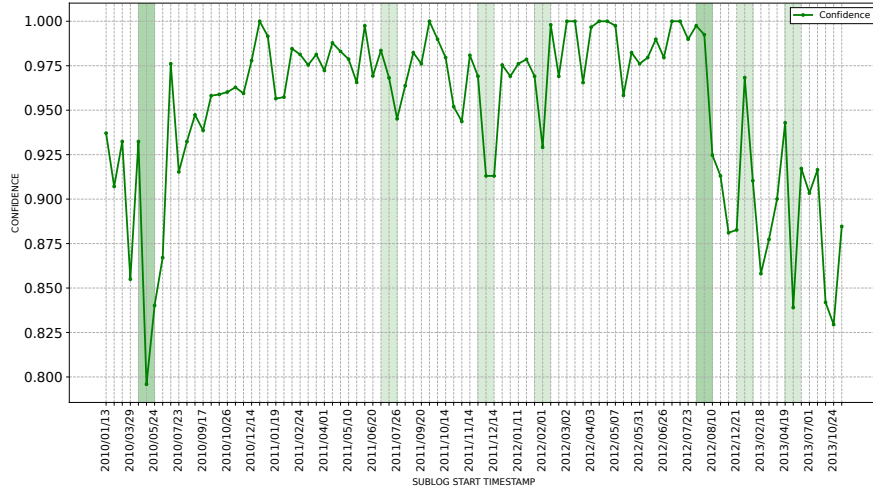


Figure 8: Help-Desk sub-log confidence values. The plot is visually edited with the addition of vertical bands in the areas where drifts (darker) and erratic behavior (lighter) are detected by VDD as in [53].

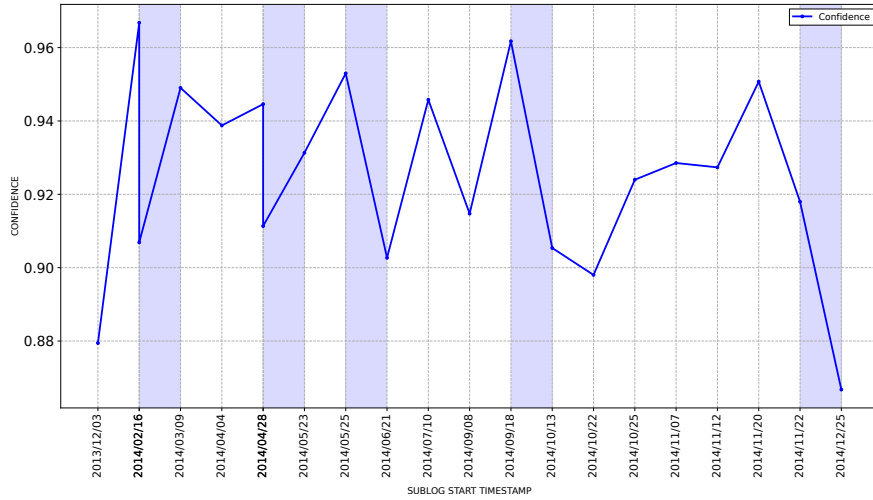


Figure 9: Sepsis sub-log confidence values. The plot is visually edited with the addition of vertical bands in the areas where erratic behavior is detected by VDD as in [54].

Specifically, Table 7 lists a comprehensive set of the measures that we used in our experiments. For every measure, we report the formula to compute it and its range.

The Sebag-Schoenauer measure of a specification S given an event log L , e.g., is computed as

$$1 - \frac{P(S_\alpha, L)P(\neg S_\tau, L)}{P(S_\alpha \cap \neg S_\tau, L)},$$

and its values range from 0 (included) to $+\infty$. Least Contradiction is defined as

$$\frac{P(S_\alpha \cap S_\tau, L) - P(S_\alpha \cap \neg S_\tau, L)}{P(S_\tau, L)},$$

on the range $(-\infty, +\infty)$. Equipped with this set of measures, we conducted a comparative analysis to evaluate their ability to signal change points.

To allow for a comparison between measures defined on different ranges (e.g., $[0, +\infty)$ for the Sebag-Schoenauer measure, and $(-\infty, +\infty)$ for Least Contradiction), we normalize

the values applying the method used in [19], which projects all values on a $[0, 1]$ interval (i.e., the range of Confidence, among others).

Tables 8(a) and 8(b) show the result of our analysis on both the Sepsis and Help-Desk logs, respectively. We report only measures with non-null values. For every measure, we indicate the mean value, the standard deviation, and the Coefficient of Variation (CV). CV, expressed as the ratio of the standard deviation to the mean, shows the extent of variability in relation to the average value along the sequence of collected values. We sort the measures in descending order by CV, standard deviation and mean. Measures that exhibit relatively more ample oscillations (hence, most sensitive to changes) are thus at the top. Both with the Sepsis and Help-Desk logs, the Sebag-Schoenauer measure ranks first, followed by a group of measures including Support and Confidence with very close values. From the Least Contradiction on, all measures follow with a CV that is

Table 7: Quality measures employed in the sensitivity experiment. For the sake of readability, we omit parameter L (the event log) from the formulae.

Measure	Formula	Range
Support	$P(S_a \cap S_r)$	[0, 1]
Confidence/ Precision	$P(S_r S_a)$	[0, 1]
	Recall	$P(S_a S_r)$
Accuracy	$P(S_a \cap S_r) + P(\neg S_a \cap \neg S_r)$	[0, 1]
Lift/ Interest	$\frac{P(S_a \cap S_r)}{P(S_a)P(S_r)}$	[0, +∞)
Leverage	$P(S_r S_a) - P(S_a)P(S_r)$	[-1, 1]
Added Value	$P(S_r S_a) - P(S_r)$	[-1, 1]
Jaccard	$\frac{P(S_a \cap S_r)}{P(S_a) + P(S_r) - P(S_a \cap S_r)}$	[0, 1]
	Certainty factor	$\frac{P(S_a S_r) - P(S_r)}{1 - P(S_r)}$
Klogsen	$\frac{\sqrt{P(S_a \cap S_r)}}{\max(P(S_r S_a) - P(S_r), P(S_a S_r) - P(S_a))}$	[-1, 1]
Conviction	$\frac{P(S_a)P(\neg S_r)}{P(S_a \cap \neg S_r)}$	[0, +∞)
J-Measure	$P(S_a \cap S_r) \log \frac{P(S_r S_a)}{P(S_r)} + P(S_a \cap \neg S_r) \log \frac{P(\neg S_r S_a)}{P(\neg S_r)}$	(-∞, +∞)
One-Way Support	$P(S_r S_a) \log_2 \frac{P(S_r \cap S_a)}{P(S_r)P(S_a)}$	(-∞, +∞)
Two-Way Support	$P(S_a \cap S_r) \log_2 \frac{P(S_a \cap S_r)}{P(S_a)P(S_r)}$	(-∞, +∞)
Piatetsky-Shapiro	$P(S_a \cap S_r) - P(S_a)P(S_r)$	[-1, 1]
Cosine	$\frac{P(S_a \cap S_r)}{\sqrt{P(S_a)P(S_r)}}$	[0, +∞)
	Loevinger	$1 - \frac{P(S_a \cap \neg S_r)}{P(S_a \cap S_r)}$
Information Gain	$\log \frac{P(S_a \cap S_r)}{P(S_a)P(S_r)}$	(-∞, +∞)
Sebag-Schoenauer	$\frac{P(S_a \cap S_r)}{P(S_a \cap \neg S_r)}$	[0, +∞)
Least Contradiction	$\frac{P(S_a \cap S_r) - P(S_a \cap \neg S_r)}{P(S_r)}$	(-∞, +∞)
Odd Multiplier	$\frac{P(S_a \cap S_r)P(\neg S_r)}{P(S_r)P(S_a \cap \neg S_r)}$	[0, +∞)
Example and Counterexample Rate	$1 - \frac{P(S_a \cap \neg S_r)}{P(S_a \cap S_r)}$	(-∞, 1]
Zhang	$\frac{P(S_a \cap S_r) - P(S_a)P(S_r)}{\max(P(S_a \cap S_r)P(\neg S_r), P(S_r)P(S_a \cap \neg S_r))}$	(-∞, +∞)

lower by orders of magnitude (from 10^{-4} down to 10^{-13} and below). The Recall measure closes the list with a steady mean of 1.0, as both CV and standard deviation equate to 0. Notice that, besides minimal variations in the ranking, the aforementioned groups are composed by the same measures both in Tables 8(a) and 8(b), thus regardless of the event log under examination.

Figures 10 and 11 plot the trends of those measures to visually compare their variations with the sub-logs extracted from the Sepsis and Help-Desk datasets, respectively. In both cases, we observe that the amplest oscillations are exhibited by the Sebag-Schoenauer measure, as expected in light of the previous discussion. Similar trends characterize the polylines associated with measures representing the group at the top of Tables 8(a) and 8(b), such as Confidence and Cosine. Least Contradiction lies in the middle of the diagrams and, though subject to variations, shows a lower sensitivity to changes. Lift is a representative of a large group of measures showing only imperceptible fluctuations. Finally, as anticipated above, Recall remains steadily equal to 1.0.

The trends exhibited in Figs. 8 to 11 pertain to full specifications mined from the original event logs. Our

approach, however, can gauge the interestingness of any sets of constraints, including sub-specifications (as illustrated in the experiment on synthetic data in Section 7.1). Here, we observe the variations that Confidence undergoes for some sub-specifications in particular. To have a better understanding of the rules that have led to the drifts, we form these sub-specifications by joining the constraints in the clusters that exhibit the most erratic trends in the Help-Desk log [46] and the Sepsis log [46] according to [53] and [54], respectively. The authors of [53, 54] indicate that those clusters are the ones that mainly signal change points in the process. Here, we leverage the capability of our approach to assess the behavior of those constraints taken together as sub-specifications, rather than individually as in [53, 54]. Thereafter, we also create specifications that join those sub-specifications. Finally, we measure Confidence for every sub-specification and specification on the respective sub-logs, using the same tumbling window approach as in the experiments above.

Tables 9 and 10 report on the minimum, maximum, and average values of Confidence together with the standard deviation and coefficient of variation for the clusters and the specifications derived therefrom on the Help-Desk and Sepsis event logs, respectively. We can observe that the erraticity of the clusters' behavior is confirmed by the CV, which is orders of magnitude higher than the values reported in Tables 8(a) and 8(b). Figures 12 and 13 illustrate the trend of Confidence over the subsequent sub-logs from the Help-Desk and Sepsis datasets, respectively. Both figures consider the individual clusters alone (Figs. 12(a) to 12(c), 13(a) and 13(b)) and the joined specifications (Figs. 12(d) and 13(c)). The plots in Figs. 12(a) to 12(c), 13(a) and 13(b), pertaining to the individual sub-specifications, closely resemble the trends illustrated in [53, 54]. However, we can also observe particular phenomena due to the definition of Confidence itself (see Table 7): (i) The Confidence of joined specifications tend to take the non-zero values from the sub-specifications, on one hand; (ii) On the other hand, the height of the peaks stemming from some sub-specifications is reduced whenever in other sub-specifications a corresponding lower Confidence is reported.

The former effect is particularly visible comparing Figs. 13(a) and 13(b) with Fig. 13(c): the plateaus lying at 0 occur in Fig. 13(c) only in correspondence of 0 values for the Confidence of *both* the clusters the specification is composed of. Such a plateau, e.g., occurs between the marks "2014/03/09" and "2014/12/25" in Fig. 13(b) (cluster 12), but not in Fig. 13(a) (cluster 8). Then, the plateau does not occur in the Confidence plot for the whole specification in Fig. 13(c). We see plateaus of Confidence equal to 0 in Fig. 13(c) right before the "2014/03/09" and "2014/11/07", instead, as the Confidence of neither of the sub-specifications goes beyond the minimum.

The conjunction of the two aforementioned effects is instead noticeable from Figs. 12(c) and 12(d). Confidence for cluster 4 (Fig. 12(c)) neither drops to 0 nor raises to 1 with any sub-log, as opposed to clusters 9 and 11 (Figs. 12(a) and 12(b);

Table 8: Variability of specification measures in the context of drift detection.

(a) Sepsis [14]				(b) Help-Desk [46]			
Measure	Mean	Std. Dev.	CV	Measure	Mean	Std. Dev.	CV
Sebag-Schoenauer	0.861 505 846 320	0.002 053 050 180	0.002 383 094 890	Sebag-Schoenauer	0.900 819 237 282	0.006 297 508 246	0.006 990 867 852
Compliance	0.924 989 480 532	0.000 696 413 129	0.000 752 887 620	Accuracy	0.950 622 074 373	0.002 179 473 562	0.002 292 681 414
Jaccard	0.924 989 494 280	0.000 696 413 103	0.000 752 887 582	Support	0.950 622 074 373	0.002 179 473 562	0.002 292 681 414
Accuracy	0.924 989 494 150	0.000 696 412 945	0.000 752 887 410	Compliance	0.950 622 071 166	0.002 179 473 472	0.002 292 681 328
Support	0.924 989 494 150	0.000 696 412 945	0.000 752 887 410	Confidence	0.950 622 076 105	0.002 179 473 334	0.002 292 681 170
Confidence	0.924 989 487 357	0.000 696 412 870	0.000 752 887 335	Jaccard	0.950 622 081 188	0.002 179 473 205	0.002 292 681 022
Cosine	0.962 422 081 608	0.000 175 943 255	0.000 182 812 987	Cosine	0.975 239 803 262	0.000 551 582 239	0.000 565 586 266
Example and Counterexample Rate	0.969 838 405 557	0.000 159 563 237	0.000 164 525 591	Example and Counterexample Rate	0.979 413 924 601	0.000 524 304 351	0.000 535 324 584
Least Contradiction	0.727 378 805 369	$8.975 428 878 985 \times 10^{-5}$	$1.233 941 491 385 \times 10^{-4}$	Least Contradiction	0.734 560 445 157	0.000 294 921 192	0.000 401 493 430
Certainty factor	0.499 999 875 531	$3.627 784 544 022 \times 10^{-13}$	$7.255 570 894 236 \times 10^{-13}$	Certainty factor	0.500 000 047 986	$3.021 126 604 882 \times 10^{-13}$	$6.042 252 629 875 \times 10^{-13}$
Zhang	0.499 999 929 596	$1.143 870 607 449 \times 10^{-13}$	$2.287 741 537 031 \times 10^{-13}$	Zhang	0.500 000 026 811	$8.434 163 170 167 \times 10^{-14}$	$1.686 832 543 582 \times 10^{-13}$
Leverage	0.499 999 971 617	$1.833 876 167 754 \times 10^{-14}$	$3.667 752 543 712 \times 10^{-14}$	Leverage	0.500 000 009 070	$4.133 232 125 709 \times 10^{-15}$	$8.266 464 101 459 \times 10^{-15}$
Added Value	0.499 999 985 808	$4.584 691 433 807 \times 10^{-15}$	$9.169 383 127 870 \times 10^{-15}$	One-way Support	0.500 000 008 712	$1.037 287 060 395 \times 10^{-15}$	$2.074 574 084 643 \times 10^{-15}$
Piatetsky-Shapiro	0.499 999 985 808	$4.584 691 433 807 \times 10^{-15}$	$9.169 383 127 870 \times 10^{-15}$	Two-way Support	0.500 000 008 712	$1.037 287 000 451 \times 10^{-15}$	$2.074 573 964 754 \times 10^{-15}$
Klosgen	0.499 999 986 751	$3.992 108 862 561 \times 10^{-15}$	$7.984 217 936 682 \times 10^{-15}$	Added Value	0.500 000 004 535	$1.033 308 212 580 \times 10^{-15}$	$2.066 616 406 415 \times 10^{-15}$
Two-way Support	0.499 999 994 921	$2.812 996 585 923 \times 10^{-15}$	$5.625 993 228 996 \times 10^{-15}$	Piatetsky-Shapiro	0.500 000 004 535	$1.033 308 212 580 \times 10^{-15}$	$2.066 616 406 415 \times 10^{-15}$
One-way Support	0.499 999 994 921	$2.812 996 585 923 \times 10^{-15}$	$5.625 993 228 996 \times 10^{-15}$	Klosgen	0.500 000 004 274	$9.205 376 653 863 \times 10^{-16}$	$1.841 075 315 036 \times 10^{-15}$
Lovinger	0.666 666 658 768	$2.571 487 448 916 \times 10^{-15}$	$3.857 231 219 076 \times 10^{-15}$	J Measure	0.500 000 007 383	$6.863 433 069 274 \times 10^{-16}$	$1.372 686 593 585 \times 10^{-15}$
Information Gain	0.499 999 995 743	$1.801 732 986 214 \times 10^{-15}$	$3.603 466 003 112 \times 10^{-15}$	Information Gain	0.500 000 006 803	$6.266 684 578 018 \times 10^{-16}$	$1.253 336 898 551 \times 10^{-15}$
J Measure	0.499 999 996 042	$1.750 513 976 295 \times 10^{-15}$	$3.501 027 980 303 \times 10^{-15}$	Lovinger	0.666 666 669 648	$8.286 719 455 800 \times 10^{-16}$	$1.243 007 912 811 \times 10^{-15}$
Lift	0.500 000 002 619	$3.690 476 483 444 \times 10^{-16}$	$7.380 952 928 225 \times 10^{-16}$	Lift	0.500 000 007 826	$1.111 562 317 104 \times 10^{-16}$	$2.223 124 599 412 \times 10^{-16}$
Conviction	0.500 000 006 667	$2.583 333 354 669 \times 10^{-16}$	$5.166 666 640 448 \times 10^{-16}$	Conviction	0.500 000 003 095	$8.849 684 779 794 \times 10^{-17}$	$1.769 936 945 002 \times 10^{-16}$
Odd Multiplier	0.500 000 005 000	$5.176 899 690 513 \times 10^{-32}$	$1.035 379 927 749 \times 10^{-31}$	Odd Multiplier	0.500 000 005 000	$1.995 913 133 692 \times 10^{-31}$	$3.991 826 227 465 \times 10^{-31}$
Recall	1.000 000 000 000	0.000 000 000 000	0.000 000 000 000	Recall	1.000 000 000 000	0.000 000 000 000	0.000 000 000 000

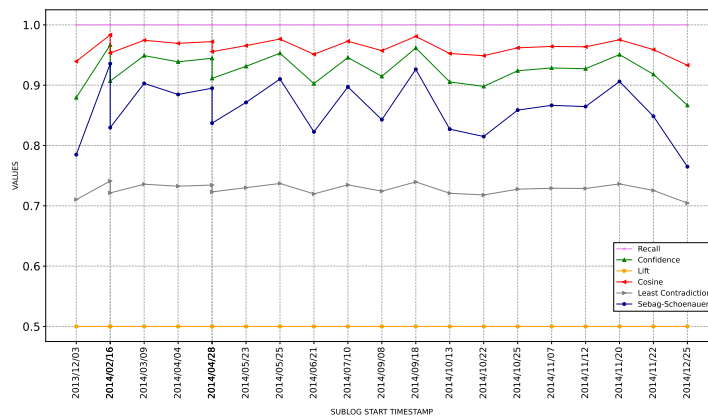


Figure 10: Specification measure oscillations on the Sepsis event log

Table 9: Aggregate confidence in the Help-Desk sub-logs [46] for the constraint clusters exhibiting the most erratic behavior [53]

Cluster	Mean	Std. Dev	CV	Max	Min
Cluster 4	0.6336	0.0083	0.0131	0.8912	0.5129
Cluster 9	0.3164	0.2110	0.6670	1.0000	0.0000
Cluster 11	0.0543	0.0520	0.9560	1.0000	0.0000
Joint specification	0.6346	0.0082	0.0128	0.8918	0.5129

see also the minimum and maximum values reported in Table 8(b)). Then, the plot in Fig. 12(d) tends to almost completely overlap with that of Fig. 12(c). We remark that this effect differs from what we would have obtained by

Table 10: Aggregate confidence in the Sepsis sub-logs [14] for the constraint clusters exhibiting the most erratic behavior [54]

Cluster	Mean	Std. Dev	CV	Max	Min
Cluster 8	0.4977	0.2404	0.4831	1.0000	0.0000
Cluster 12	0.3025	0.1775	0.5866	1.0000	0.0000
Joint specification	0.5518	0.1481	0.2683	1.0000	0.0000

merely averaging the Confidence values of the clusters' sub-specifications to assign the Confidence values of the joined specification.

Measurement of Rule-based LTLf Declarative Process Specifications

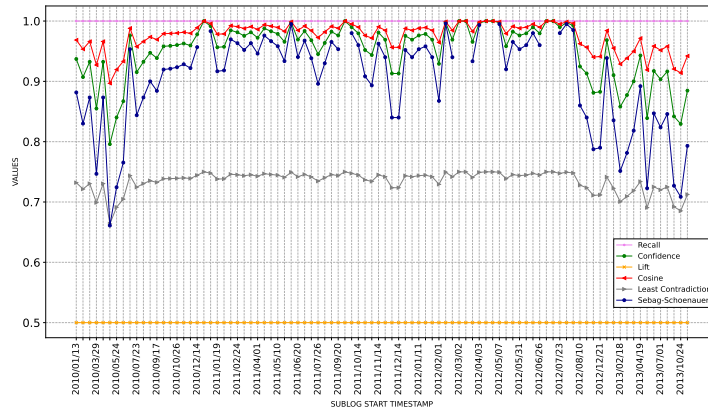
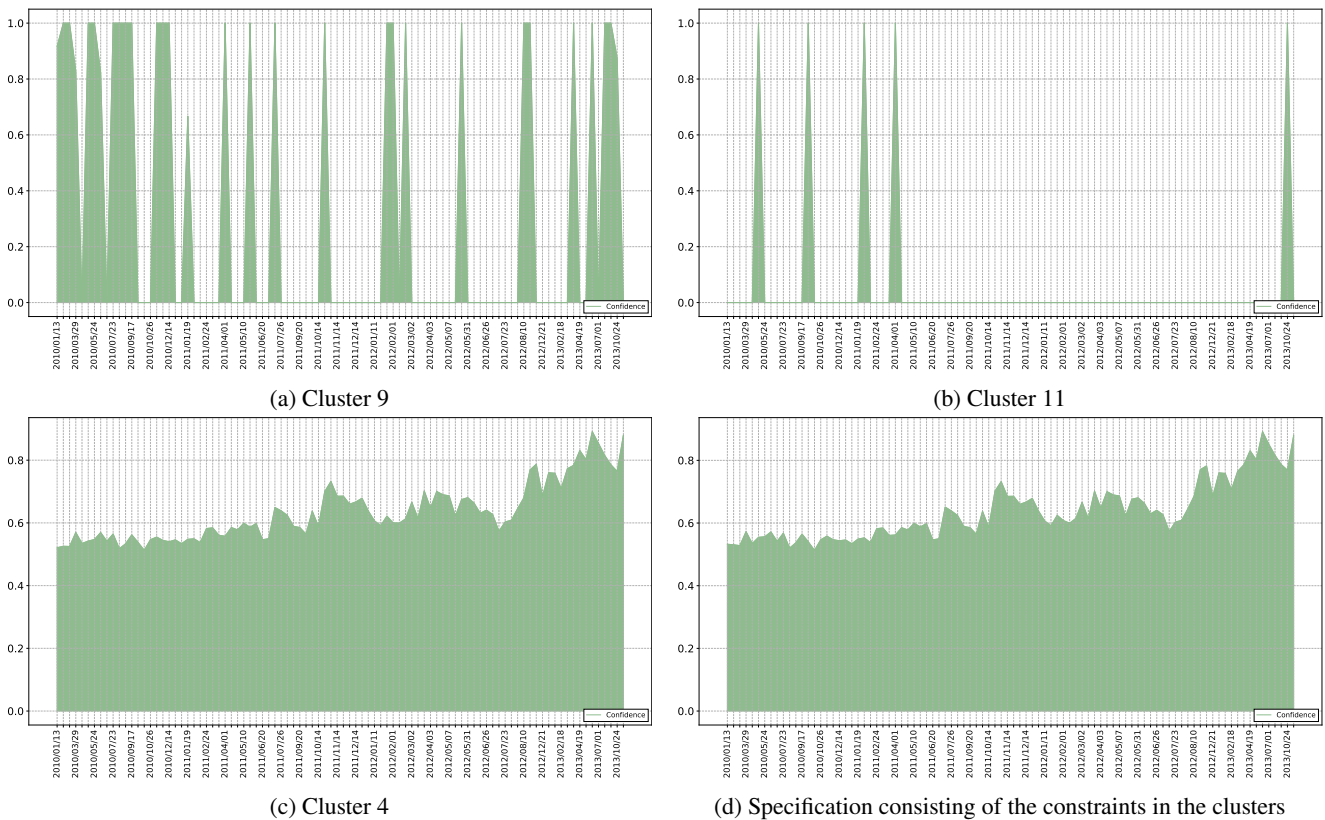


Figure 11: Specification measure oscillations on the Help-Desk event log



(a) Cluster 9

(b) Cluster 11

(c) Cluster 4

(d) Specification consisting of the constraints in the clusters

Figure 12: Confidence of the constraint clusters with the most erratic behavior with the Help-Desk log, and of the specification stemming from their union

7.4. Discussion

Thus far, we have investigated various aspects to which the application of our measurement framework contributes with novel insights. We provide their summary below.

First of all, we observed in Section 7.1 that the overall compliance of an event log to a whole specification tends to be lower than its compliance to every rule taken individually. This aspect is of considerable relevance in the context of process and specification mining, as current works in the

literature still tend to resort to an analysis centered around individual rules, thus neglecting their effect on the expected behavior in combination [48, 55, 56, 47, 57, 58].

Also, the opportunity to compute a large array of measures, including but not limited to those that derive from association rule mining, sheds light on interesting facts from different perspectives. The empirical piece of evidence presented in Section 7.2 confirms that a single measure is not

Measurement of Rule-based LTL Declarative Process Specifications

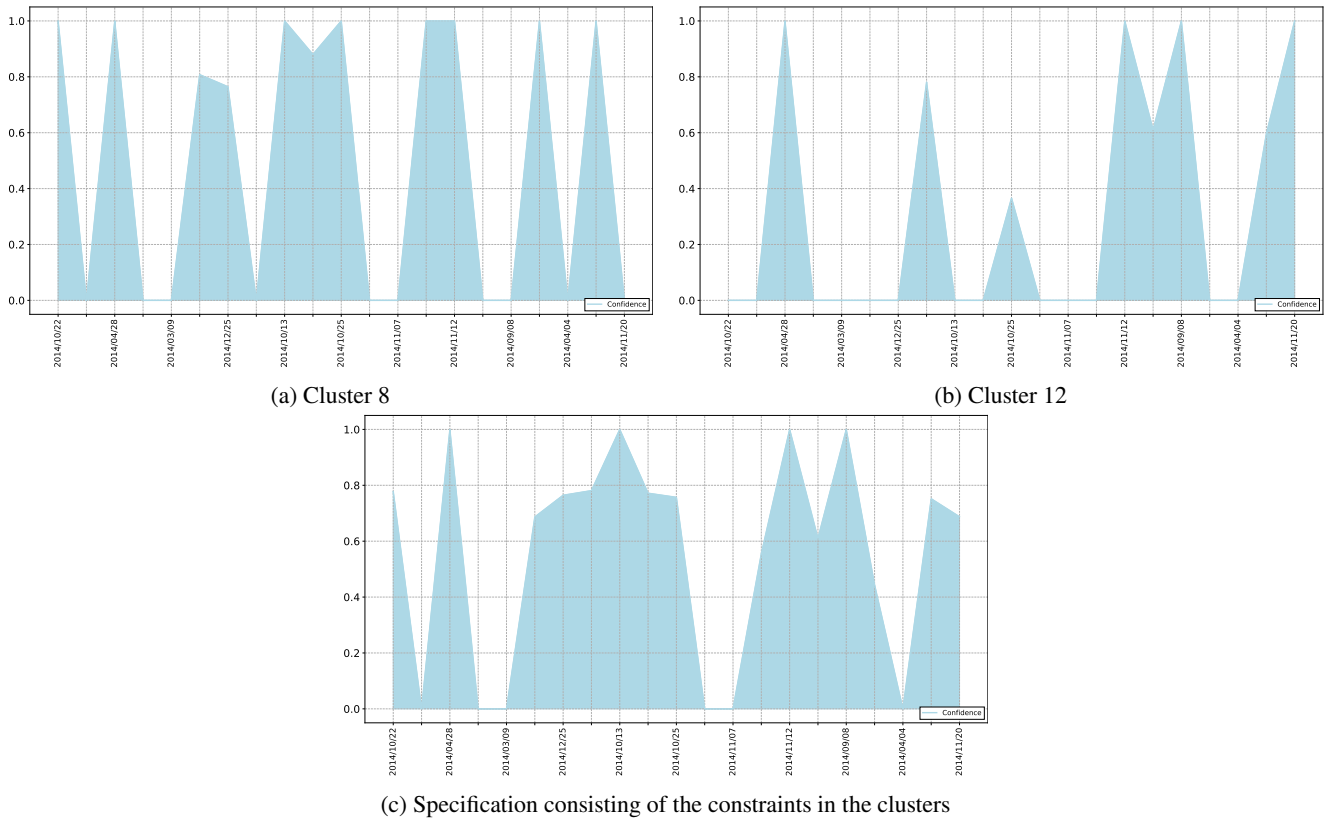


Figure 13: Confidence of the constraint clusters with the most erratic behavior with the Sepsis log, and of the specification stemming from their union

sufficient to provide a full account of the degree of interestingness of a mined process specification. The use case illustrated in Section 7.3 shows a possible application of our approach to assessing the variability of the degree of compliance of process executions with the overall specification over time. Our tool is not meant to be a drift identification technique. Other approaches, such as the aforementioned VDD [54], are specifically designed to achieve this goal. However, our method can be used in practice to observe the effect that drifts and other change points in the process may have on an entire specification [59]. The most suitable measure to consider for this kind of analysis appears to be the Least Contradiction. Also, the oscillations of measures in the sequence highlight that process executions are subject to variations over time, while the mining of a whole event log cannot represent these changes in behavior (either temporary or not). Furthermore, we have observed that the mere aggregation of measures stemming from sub-specifications does not properly account for the measurement of a conjoint specification.

In the next section, we provide an overview of the related literature and position our contribution against the existing background.

8. Related work

Different contributions in the literature aim at quantitative extensions of LTL/LTL_f enriching the languages with quantitative operators. [60, 61, 62] all proposed the addition of quantitative operators into the logic. LTL[F] [60] introduces quality operators quantifying over distinct satisfactions of a formula. Quantified-LTL [62] uses quantifiers over its propositional variables, also in probabilistic systems such as Markov chains.

In [61], the quantification of satisfaction, applied in the context of planning, is based on associating costs to specification violations based on user ranking of tasks priority. Differently from these methods, we do not extend the syntax and semantics of LTL_f with new operators, as we quantify the satisfaction of formulae based on standard LTL_f.

As for the interplay of temporal logic and probabilities, statistical model checking techniques [63] retrieve the probability for a formula to be satisfied in a probabilistic environment as Markov chains. Their goal is to predict the likelihood of a formula for any possible execution (a probabilistic relaxation of traditional model checking), while we study only already executed traces. The method proposed in [64] is close to our investigation, as it resorts to the association of a probability threshold to each rule. The threshold is used to perform relaxed conformance checking: each rule should hold in at least a portion of the log that

is greater than that value. However, only single rules are analyzed and the trace satisfaction is not quantified but considered as boolean, whereas we can assess the partial satisfaction of specifications, also on single traces.

In process mining, the compliance of process models to the data is usually gauged with four scores: Fitness, Precision, Generalization, and Simplicity [65]. In [66] Fitness, Precision, and Generalization are devised for DECLARE models through alignments, while in [67] Fitness and Precision are computed for any regular language through entropy. Our framework focuses on a different set of measures, inspired by association rule mining. The comparison and integration of the four measures above paves the path for future research endeavors. Notably, the novel measure of informativeness is proposed in [68] to understand the differences between compliant traces. We showed in Section 7 how separate measures can spot differences in compliant specifications, thus a deeper analysis in this direction is an interesting research outlook.

9. Conclusion

In this paper, we presented a tractable approach to quantify the satisfaction degree of rule-based LTL_f specifications on bags of traces. Our approach is grounded in probabilistic models with which we have derived maximum-likelihood estimators for LTL_f formulae, RCons, and process specifications. We apply our prototype to real-world data, showing its broad range of employment. We provide experimental evidence that the Confidence of a mined specification is often below the minimum levels set for the discovery of its rules. Also, we observe that a single measure does not give a full picture of the level of interestingness of a specification for an event log. Furthermore, we describe the effect of drifts and anomalies on the specification measurements, with insights into the measures that are more suitable for the signaling of behavioral changes over time.

The advantage of analyzing processes at the higher level of a specification is complementary to the details provided by its single rules. The specification measurement gives a holistic view of the status of the process, which could not be achieved by the sum of its single parts. This information can guide subsequent detailed analysis, e.g., highlighting the overall mismatch between the specification and the data, whenever it diverges significantly enough to call for an in-depth analysis.

Looking onward, our result for LTL_f can be easily extended to Linear Dynamic Logic over Finite Traces [20], which has the expressive power of Monadic Second Order Logic, but with the same computational cost of LTL_f . Moreover, we aim to explore the enrichment of log labeling with additional contextual data (such as patient diagnoses) akin to [69] and construct estimators that take this information into account. Specifically, a possible extension would be to model the event of satisfying a formula conditional on context via logistic regression. Also, a relevant direction for practical implementations is the assessment of measures based on the most common specification and process mining

tasks. To this end, the definition of desirable propositions for the interestingness measures and the properties they should guarantee is crucial, similarly to what has been done in [70] for conformance measures in process mining. Another interesting outlook is the employment of specifications measures as data features for machine learning applications, e.g., trace clustering [71]. Similarly, as hinted in the evaluation, our measurement framework can be used to support process mining applications, such as drift detection (with statistically significant identification of change points and pinpointing of the sub-specifications exhibiting the most erratic behavior), or post-processing filtering for declarative process discovery techniques. A highly stimulating research endeavor can be spurred by the application of our technique on the field, in the context of highly flexible, dynamic system and process execution scenarios such as healthcare with the checking and monitoring of rules defined in clinical pathways [72], with extensions aimed at weighting differently constraints according to their compulsory or best-practice nature [73].

Acknowledgments

L. Barbaro and C. Di Ciccio were partly supported by the Italian Ministry of University and Research (MUR) under PRIN grant B87G22000450001 (PINPOINT). L. Barbaro received funding from the Latium Region under the PO FSE+ grant B83C22004050009 (“Predictive process monitoring for production planning”). C. Di Ciccio was also supported by project SERICS (PE00000014) under the NRRP MUR program funded by the EU-NGEU. The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

- [1] M. Pesic, D. Bosnacki, W. van der Aalst, Enacting declarative languages using LTL: avoiding errors and improving performance, in: SPIN, 2010, pp. 146–161.
- [2] G. De Giacomo, P. Felli, M. Montali, G. Perelli, HyperLDL: a logic for checking properties of finite traces process logs, in: IJCAI, 2021, pp. 1859–1865.
- [3] F. Bacchus, F. Kabanza, Planning for temporally extended goals, in: AAAI/IAAI, Vol. 2, 1996, pp. 1215–1222.
- [4] A. Camacho, E. Triantafillou, C. Muise, J. Baier, S. McIlraith, Non-deterministic planning with temporally extended goals: LTL over finite and infinite traces, in: AAAI, 2017, pp. 3716–3724.
- [5] C. Lemieux, D. Park, I. Beschastnikh, General LTL specification mining (T), in: ASE, 2015, pp. 81–92.
- [6] Z. Cao, Y. Tian, T. Le, D. Lo, Rule-based specification mining leveraging learning to rank, *Autom. Softw. Eng.* 25 (2018) 501–530.
- [7] C. Di Ciccio, M. Montali, Declarative process specifications: Reasoning, discovery, monitoring, in: W. M. P. van der Aalst, J. Carmona (Eds.), *Process Mining Handbook*, Springer, 2022, pp. 108–152.
- [8] F. M. Maggi, A. J. Mooij, W. M. P. van der Aalst, User-guided discovery of declarative process models, in: *Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining, CIDM 2011*, part of the IEEE Symposium Series on Computational Intelligence 2011, April 11–15, 2011, Paris, France, IEEE, 2011, pp. 192–199. doi:10.1109/CIDM.2011.5949297.

- [9] C. Di Ciccio, A. Marrella, A. Russo, Knowledge-intensive Processes: Characteristics, requirements and analysis of contemporary approaches, *J. Data Semantics* 4 (2015) 29–57.
- [10] W. M. P. van der Aalst, M. Pesic, H. Schonenberg, Declarative workflows: Balancing between flexibility and support, *Computer Science - R&D* 23 (2009) 99–113.
- [11] M. Rovani, F. M. Maggi, M. de Leoni, W. M. P. van der Aalst, Declarative process mining in healthcare, *Expert Syst. Appl.* 42 (2015) 9236–9251.
- [12] J. Munoz-Gama, N. Martin, C. Fernández-Llatas, O. A. Johnson, M. Sepúlveda, E. Helm, V. Galvez-Yanjari, E. Rojas, A. Martínez-Millana, D. Aloini, I. A. Amantea, R. Andrews, M. Arias, I. Beerepoot, E. Benevento, A. Burattin, D. Capurro, J. Carmona, M. Comuzzi, B. Dalmas, R. de la Fuente, C. D. Francescomarino, C. D. Ciccio, R. Gatta, C. Ghidini, F. Gonzalez-Lopez, G. Ibáñez-Sánchez, H. B. Klasky, A. P. Kurniati, X. Lu, F. Mannhardt, R. Mans, M. Marcos, R. M. de Carvalho, M. Pegoraro, S. K. Poon, L. Pufahl, H. A. Reijers, S. Remy, S. Rinderle-Ma, L. Sacchi, F. Seoane, M. Song, A. Stefanini, E. Sulis, A. H. M. ter Hofstede, P. J. Toussaint, V. Traver, Z. Valero-Ramon, I. van de Weerd, W. M. P. van der Aalst, R. J. B. Vanwersch, M. Weske, M. T. Wynn, F. Zerbo, Process mining for healthcare: Characteristics and challenges, *J. Biomed. Informatics* 127 (2022) 103994.
- [13] A. Guzzo, A. Rullo, E. Vocaturo, Process mining applications in the healthcare domain: A comprehensive review, *WIREs Data Mining Knowl. Discov.* 12 (2022).
- [14] F. Mannhardt, Sepsis cases - event log, 2016. doi:<https://doi.org/10.4121/uuid:915d2bfb-7e84-49ad-a286-dc35f063a460>.
- [15] Z. Huang, W. Dong, L. Ji, L. Yin, H. Duan, On local anomaly detection and analysis for clinical pathways, *Artif. Intell. Medicine* 65 (2015) 167–177.
- [16] A. Ceconi, G. De Giacomo, C. Di Ciccio, F. Maggi, J. Mendling, Measuring the interestingness of temporal logic behavioral specifications in process mining, *Information Systems* (2021) 101920.
- [17] A. Ceconi, C. Di Ciccio, G. De Giacomo, J. Mendling, Interestingness of traces in declarative process mining: The Janus LTLp_f approach, in: *BPM*, 2018, pp. 121–138. doi:[10.1007/978-3-319-98648-7_8](https://doi.org/10.1007/978-3-319-98648-7_8).
- [18] L. Geng, H. Hamilton, Interestingness measures for data mining: A survey, *ACM Comput. Surv.* 38 (2006) 9.
- [19] A. Ceconi, C. Di Ciccio, A. Senderovich, Measurement of rule-based ltlf declarative process specifications, in: *ICPM*, IEEE, 2022, pp. 96–103. doi:[10.1109/ICPM57379.2022.9980690](https://doi.org/10.1109/ICPM57379.2022.9980690).
- [20] G. De Giacomo, M. Vardi, Linear temporal logic and linear dynamic logic on finite traces, in: *IJCAI*, 2013, pp. 854–860.
- [21] A. Pnueli, The temporal logic of programs, in: *FOCS*, 1977, pp. 46–57. doi:[10.1109/SFCS.1977.32](https://doi.org/10.1109/SFCS.1977.32).
- [22] O. Lichtenstein, A. Pnueli, L. Zuck, The glory of the past, in: *Logic of Programs*, 1985, pp. 196–218.
- [23] I. Hodkinson, M. Reynolds, Separation - past, present, and future, in: *We Will Show Them!* (2), 2005, pp. 117–142.
- [24] G. De Giacomo, R. De Masellis, M. Montali, Reasoning on LTL on finite traces: Insensitivity to infiniteness, in: *AAAI*, 2014, pp. 1027–1033.
- [25] V. Fionda, G. Greco, The complexity of LTL on finite traces: Hard and easy fragments, in: *AAAI*, 2016, pp. 971–977.
- [26] O. Kupferman, M. Vardi, Vacuity detection in temporal model checking, *Int. J. Softw. Tools Technol. Transf.* 4 (2003) 224–233.
- [27] F. M. Maggi, A. J. Mooij, W. M. P. van der Aalst, User-guided discovery of declarative process models, in: *CIDM*, 2011, pp. 192–199. doi:[10.1109/CIDM.2011.5949297](https://doi.org/10.1109/CIDM.2011.5949297).
- [28] G. De Giacomo, F. Maggi, A. Marrella, S. Sardiña, Computing trace alignment against declarative process models through planning, in: *ICAPS*, 2016, pp. 367–375.
- [29] J. Adamo, Data mining for association rules and sequential patterns - sequential and parallel algorithms, Springer New York, 2001. doi:[10.1007/978-1-4613-0085-4](https://doi.org/10.1007/978-1-4613-0085-4).
- [30] R. Agrawal, R. Srikant, Fast algorithms for mining association rules in large databases, in: J. B. Bocca, M. Jarke, C. Zaniolo (Eds.), *Vldb'94*, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile, Morgan Kaufmann, 1994, pp. 487–499. URL: <http://www.vldb.org/conf/1994/P487.PDF>.
- [31] P. Lenca, P. Meyer, B. Vaillant, S. Lallich, On selecting interestingness measures for association rules: User oriented description and multiple criteria decision aid, *Eur. J. Oper. Res.* 184 (2008) 610–626.
- [32] P.-N. Tan, V. Kumar, J. Srivastava, Selecting the right objective measure for association analysis, *Information Systems* 29 (2004) 293 – 313. Knowledge Discovery and Data Mining (KDD 2002).
- [33] P. Bickel, K. Doksum, Mathematical statistics: basic ideas and selected topics, volumes I-II package, CRC Press, 2015.
- [34] B. Dai, S. Ding, G. Wahba, Multivariate bernoulli distribution, *Bernoulli* 19 (2013) 1465–1483.
- [35] S. Ip, J. Xue, A multivariate regression view of multi-label classification, Technical Report, University College London, 2015.
- [36] C. Di Ciccio, M. L. Bernardi, M. Cimitile, F. M. Maggi, Generating event logs through the simulation of Declare models, in: *EO-MAS@CAISE*, 2015, pp. 20–36. doi:[10.1007/978-3-319-24626-0_2](https://doi.org/10.1007/978-3-319-24626-0_2).
- [37] C. R. Rao, Maximum likelihood estimation for the multinomial distribution, *Sankhyā: The Indian Journal of Statistics* (1933-1960) 18 (1957) 139–148.
- [38] G. Casella, R. L. Berger, Statistical inference, Cengage Learning, 2021.
- [39] J. Li, L. Zhang, G. Pu, M. Y. Vardi, J. He, Ltlf satisfiability checking, in: *ECAI*, volume 263 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, 2014, pp. 513–518. doi:[10.3233/978-1-61499-419-0-513](https://doi.org/10.3233/978-1-61499-419-0-513).
- [40] M. Dwyer, G. Avrunin, J. Corbett, Patterns in property specifications for finite-state verification, in: *ICSE*, 1999, pp. 411–420.
- [41] B. van Dongen, BPI Challenge 2012, 4TU.ResearchData, 2012. doi:[10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f](https://doi.org/10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f).
- [42] W. Steeman, BPI Challenge 2013, 2014. doi:[10.4121/uuid:a7ce5c55-03a7-4583-b855-98b86e1a2b07](https://doi.org/10.4121/uuid:a7ce5c55-03a7-4583-b855-98b86e1a2b07).
- [43] B. van Dongen, BPI Challenge 2014, 4TU.ResearchData, 2014. doi:[10.4121/uuid:c3e5d162-0cfd-4bb0-bd82-af5268819c35](https://doi.org/10.4121/uuid:c3e5d162-0cfd-4bb0-bd82-af5268819c35).
- [44] B. van Dongen, BPI Challenge 2015, 4TU.ResearchData, 2015. doi:[10.4121/uuid:31a308ef-c844-48da-948c-305d167a0ec1](https://doi.org/10.4121/uuid:31a308ef-c844-48da-948c-305d167a0ec1).
- [45] M. de Leoni, F. Mannhardt, Road traffic fine management process, 2015. doi:[10.4121/uuid:270fd440-1057-4fb9-89a9-b699b47990f5](https://doi.org/10.4121/uuid:270fd440-1057-4fb9-89a9-b699b47990f5).
- [46] M. Polato, Dataset belonging to the help desk log of an italian company, 2017. doi:<https://doi.org/10.4121/uuid:0c60edf1-6f83-4e75-9367-4c63b3e9d5bb>.
- [47] C. Di Ciccio, M. Mecella, On the discovery of declarative control flows for artful processes, *ACM Trans. Manag. Inf. Syst.* 5 (2015) 24:1–24:37.
- [48] J. Yang, D. Evans, D. Bhardwaj, T. Bhat, M. Das, Perracotta: mining temporal API rules from imperfect traces, in: *ICSE*, 2006, pp. 282–291.
- [49] E. Morin, La méthode: la nature de la nature, Média Diffusion, 2013.
- [50] W. Hämmäläinen, G. Webb, A tutorial on statistically sound pattern discovery, *Data Min. Knowl. Discov.* 33 (2019) 325–377.
- [51] A. Maaradji, M. Dumas, M. L. Rosa, A. Ostovar, Detecting sudden and gradual drifts in business processes from execution traces, *IEEE Trans. Knowl. Data Eng.* 29 (2017) 2140–2154.
- [52] A. Yeshchenko, J. Mendling, C. Di Ciccio, A. Polyvyanyy, VDD: A visual drift detection system for process mining, in: *ICPM Doctoral Consortium / Tools*, 2020, pp. 31–34. URL: <https://ceur-ws.org/Vol-2703/paperTD4.pdf>.
- [53] A. Yeshchenko, C. Di Ciccio, J. Mendling, A. Polyvyanyy, Comprehensive process drift detection with visual analytics, in: *ER*, 2019, pp. 119–135. doi:[10.1007/978-3-030-33223-5_11](https://doi.org/10.1007/978-3-030-33223-5_11).
- [54] A. Yeshchenko, C. Di Ciccio, J. Mendling, A. Polyvyanyy, Visual drift detection for event sequence data of business processes, *IEEE Trans. Vis. Comput. Graph.* 28 (2022) 3050–3068.
- [55] C. Lemieux, D. Park, I. Beschastnikh, General LTL specification mining (T), in: *ASE*, 2015, pp. 81–92.
- [56] T. B. Le, D. Lo, Beyond support and confidence: Exploring interestingness measures for rule-based specification mining, in: *SANER*, 2015, pp. 331–340. doi:[10.1109/SANER.2015.7081843](https://doi.org/10.1109/SANER.2015.7081843).

- [57] A. Alman, C. Di Ciccio, F. M. Maggi, M. Montali, H. van der Aa, Rum: Declarative process mining, distilled, in: BPM, 2021, pp. 23–29. doi:10.1007/978-3-030-85469-0_3.
- [58] C. O. Back, T. Slaats, T. T. Hillebrandt, M. Marquard, DisCoveR: accurate and efficient discovery of declarative process models, *Int. J. Softw. Tools Technol. Transf.* 24 (2022) 563–587.
- [59] N. Schützenmeier, C. Corea, P. Delfmann, S. Jablonski, Efficient computation of behavioral changes in declarative process models, in: BPMDS/EMMSAD@CAiSE, volume 479 of *Lecture Notes in Business Information Processing*, Springer, 2023, pp. 136–151. doi:10.1007/978-3-031-34241-7_10.
- [60] S. Almagor, U. Boker, O. Kupferman, Formally reasoning about quality, *J. ACM* 63 (2016) 24:1–24:56.
- [61] M. Lahijanian, S. Almagor, D. Fried, L. Kavraki, M. Vardi, This time the robot settles for a cost: A quantitative approach to temporal logic planning with partial satisfaction, in: AAI, 2015, pp. 3664–3671.
- [62] J. Piribauer, C. Baier, N. Bertrand, O. Sankur, Quantified linear temporal logic over probabilistic systems with an application to vacuity checking, in: CONCUR, 2021, pp. 7:1–7:18. doi:10.4230/LIPIcs.CONCUR.2021.7.
- [63] A. Legay, A. Lukina, L. Traonouez, J. Yang, S. Smolka, R. Grosu, Statistical model checking, in: *Computing and Software Science - State of the Art and Perspectives*, 2019, pp. 478–504. doi:10.1007/978-3-319-91908-9_23.
- [64] F. Maggi, M. Montali, R. Peñaloza, Temporal logics over finite traces with uncertainty, in: AAI, 2020, pp. 10218–10225.
- [65] J. C. Buijs, B. F. van Dongen, W. M. van der Aalst, Quality dimensions in process discovery: The importance of fitness, precision, generalization and simplicity, *Int. J. Cooperative Inf. Syst.* 23 (2014) 1440001.
- [66] M. De Leoni, F. Maggi, W. van der Aalst, An alignment-based framework to check the conformance of declarative process models and to preprocess event-log data, *Inf. Syst.* 47 (2015) 258–277.
- [67] A. Polyvyanyy, A. Solti, M. Weidlich, C. Di Ciccio, J. Mendling, Monotone precision and recall measures for comparing executions and specifications of dynamic systems, *ACM Trans. Softw. Eng. Methodol.* 29 (2020) 17:1–17:41.
- [68] A. Burattin, G. Guizzardi, F. M. Maggi, M. Montali, Fifty shades of green: How informative is a compliant process trace?, in: CAiSE, 2019, pp. 611–626.
- [69] S. Schöning, C. Di Ciccio, F. M. Maggi, J. Mendling, Discovery of multi-perspective declarative process models, in: ICSOC, 2016, pp. 87–103. doi:10.1007/978-3-319-46295-0_6.
- [70] A. F. Syring, N. Tax, W. M. P. van der Aalst, Evaluating conformance measures in process mining using conformance propositions, *Trans. Petri Nets Other Model. Concurr.* 14 (2019) 192–221.
- [71] J. De Weerd, Trace clustering, in: *Encyclopedia of Big Data Technologies*, 2019. doi:10.1007/978-3-319-63962-8_91-1.
- [72] J. D. Weerd, F. Caron, J. Vanthienen, B. Baesens, Getting a grasp on clinical pathway data: An approach based on process mining, in: *Emerging Trends in Knowledge Discovery and Data Mining - PAKDD 2012 International Workshops: DMHM, GeoDoc, 3Clust, and DSDM*, Kuala Lumpur, Malaysia, May 29 - June 1, 2012, Revised Selected Papers, volume 7769 of *Lecture Notes in Computer Science*, Springer, 2012, pp. 22–35. doi:10.1007/978-3-642-36778-6_3.
- [73] I. A. Amantea, L. Robaldo, E. Sulis, G. Governatori, G. Boella, Business process modelling in healthcare and compliance management: A logical framework, *FLAP 9* (2022) 1131–1154.
- [74] W. Feller, *An introduction to probability theory and its applications*, Wiley, 1957.

A. Basic Probability Notation and Results

The theory of probability is a fundamental mathematical concept that underpins a wide range of fields, including statistics, economics, and physics [74]. The key concepts in probability theory include the sample space Ω , events

A, B, C, \dots , probability functions $P(\cdot)$, and conditional probabilities $P(A | B)$. The sample space Ω represents the set of all possible outcomes of an experiment. For example, when flipping a coin, the sample space is $\{H, T\}$, where H represents heads and T represents tails.

Events A, B, C, \dots in probability theory are subsets of the sample space ($A \cup B \cup C \dots \subseteq \Omega$) representing specific outcomes of interest. For example, A could represent the event of getting heads when flipping a coin. Probability functions $P(\cdot)$ assign probabilities to events: $P(A)$, e.g., represents the probability of event A occurring, and it is a number between 0 and 1, inclusive. $P(A \cap B)$ represents the probability of both A and B occurring, also known as the intersection of A and B . $P(A \cup B)$ represents the probability of either A or B occurring, also known as the union of A and B . Notice that the notion of event in probability slightly differs from that of event in the context of process and specification mining (see Def. 2.1). $P(A)$ represents the probability that a record in the trace satisfies A . In this setting, then, A is a statement that can hold true, or not, in the elements that a trace consists of.

A.1. Conditional Probability

Conditional probability is the probability of an event A given that another event B has occurred. We write it as $P(A | B)$ and define it using Kolmogorov’s analysis:

$$P(A | B) = \frac{P(A \cap B)}{P(B)}. \quad (24)$$

This formula allows us to update our beliefs about the occurrence of an event based on new information. The following properties of conditional probability hold. Firstly, $P(A | B) \geq 0$ for all events A and B . This means that the conditional probability of A given B is always non-negative.

Secondly, $P(\Omega | B) = 1$ for any event B . This means that the probability of the entire sample space given that B has occurred is equal to 1. Thirdly, if A and B are mutually exclusive events (i.e., $A \cap B = \emptyset$), then $P(A | B) = 0$. This is because if A and B cannot occur simultaneously, then the occurrence of B rules out the possibility of A occurring.

The law of total probability (LTP) can be used to compute conditional probabilities. If A_1, A_2, \dots, A_n are mutually exclusive events that partition the sample space, then

$$P(B) = \sum_{i=1}^n P(B | A_i)P(A_i). \quad (25)$$

This formula can be rearranged to compute conditional probabilities as follows (for any $1 \leq j \leq n$):

$$P(A_j | B) = \frac{P(B | A_j)P(A_j)}{\sum_{i=1}^n P(B | A_i)P(A_i)}. \quad (26)$$

A.2. Discrete Random Variables

A discrete random variable is a random variable that takes on a countable number of values, such as the number of heads obtained in a series of coin flips, or the number of defects in

a batch of products. We denote a discrete random variable as X and its possible values as x_1, x_2, \dots, x_n . The probability distribution of X specifies the probabilities $P(X = x_k)$ for each possible value x_k with $1 \leq k \leq n$. For random variables, the conditional probability is defined as

$$P(X = x|Y = y) = \frac{P(X = x \cap Y = y)}{P(Y = y)} \quad (27)$$

where X and Y are discrete random variables, and x and y are possible values that X and Y can take, respectively. $P(X = x \cap Y = y)$ is the probability that $X = x$ and $Y = y$ occur simultaneously, and $P(Y = y)$ is the probability that $Y = y$ occurs.

The law of total probability for discrete random variables can be written as follows. Let X be a discrete random variable and let Y_1, Y_2, \dots, Y_m be a partition of the sample space Ω , i.e., $\Omega = Y_1 \cup Y_2 \cup \dots \cup Y_m$ and $Y_i \cap Y_j = \emptyset$ for $1 \leq i \leq m$, $1 \leq j \leq m$, and $i \neq j$. Then, for any event A , the law of total probability states:

$$P(A) = \sum_{i=1}^m P(Y_i)P(A|Y_i) \quad (28)$$

where the sum is taken over all possible values of i .

B. Maximum Likelihood Estimation

Maximum Likelihood Estimation (MLE) is a popular statistical method for estimating the parameters of a statistical model. MLE is widely used in many fields, including econometrics, biostatistics, and engineering, due to its desirable properties (see [38, Chapter 7]). One of the key strengths of MLE is its asymptotic efficiency, which means that as the sample size increases, the MLE estimates converge to the true parameter values at the fastest possible rate among all consistent estimators. This means that MLE produces the most precise estimates possible, given the available data.

Furthermore, MLE is a consistent estimator, meaning that as the sample size increases, the MLE estimates converge to the true parameter values. As a result, the accuracy of the MLE estimates increases with more data becoming available. Under certain conditions, MLE is also an unbiased estimator, meaning that the expected value of the estimates is equal to the true parameter value. This requires that the model is correctly specified and the sample size is sufficiently large.

Finally, MLE has solid theoretical foundations based on sound statistical theory and has been extensively studied in the literature, providing a large body of knowledge for understanding its properties and effective use. Additionally, the computation of MLE is widely supported by several existing software packages encoded in R and Python, among others.

C. Linear Temporal Logic on Finite Traces

Linear Temporal Logic on Finite Traces (LTL_f) [20] expresses propositions over linear discrete-time structures of finite length – namely, traces. LTL_f has the same syntax

of Linear Temporal Logic (LTL) [21], but is interpreted on finite traces. In this paper, in particular, we consider the LTL dialect including past modalities [22].

LTL_f formulae are built from an alphabet $\Sigma \supseteq \{a\}$ of propositional symbols, auxiliary symbols ‘(’ and ‘)’, propositional constants *True* and *False*, the logical connectives \neg (*negation*, unary) and \wedge (*conjunction*, binary), the unary temporal operators \bigcirc (*next*) and \ominus (*yesterday*), and the binary temporal operators \mathbf{U} (*until*) and \mathbf{S} (*since*). Typically, their syntax is enriched with additional binary logical connectives, namely \vee (*disjunction*) and \rightarrow (*implication*), and unary temporal operators \diamond (*eventually*), \diamondleftarrow (*once*), \square (*always*), and \squareleftarrow (*historically*). Finally, we introduce two additional temporal constants, t_{Start} and t_{End} , intuitively denoting the initial and final event of a trace.

Although they do not add expressive power, they contribute to more succinct formulations. The following grammar provides the syntax rules to build a well-formed LTL_f formula:

$$\begin{aligned} \varphi ::= & \text{True} \mid \text{False} \mid t_{\text{Start}} \mid t_{\text{End}} \mid a \mid \\ & (\neg\varphi) \mid (\varphi_1 \wedge \varphi_2) \mid (\varphi_1 \vee \varphi_2) \mid (\varphi_1 \rightarrow \varphi_2) \mid \\ & (\bigcirc\varphi) \mid (\varphi_1 \mathbf{U} \varphi_2) \mid (\diamond\varphi) \mid (\square\varphi) \mid \\ & (\ominus\varphi) \mid (\varphi_1 \mathbf{S} \varphi_2) \mid (\diamondleftarrow\varphi) \mid (\squareleftarrow\varphi) \end{aligned}$$

We may omit parentheses when the operator precedence intuitively follows from the expression. Given $\{a, b\} \subseteq \Sigma$, e.g., the following is a well-formed LTL_f formula: $(\bigcirc\neg a) \mathbf{U} b$.

Semantics of LTL_f is given in terms of finite traces, i.e., finite words over the alphabet 2^Σ . We name the index of the element in the trace as *instant*. Intuitively, *True* and *False* denote truth and falsity, $\neg\varphi$ means that φ does not hold true, $\varphi_1 \wedge \varphi_2$ signifies that both φ_1 and φ_2 hold true, $\varphi_1 \vee \varphi_2$ indicates that φ_1 or φ_2 (or both) hold true, and $\varphi_1 \rightarrow \varphi_2$ states that if φ_1 holds true then φ_2 must be verified (whereas if φ_1 does *not* hold true, no condition is exerted on φ_2). Formulae consisting of propositional symbols, constants and logical connectives are verified in a specific point in time (an *instant*). Temporal operators and constants require an evaluation of the formula on a trace of subsequent instants. $\bigcirc\varphi$ and $\ominus\varphi$ indicate that φ holds true at the next and previous instant, respectively; $\varphi_1 \mathbf{U} \varphi_2$ states that φ_2 will eventually hold and, until then, φ_1 holds too; dually, $\varphi_1 \mathbf{S} \varphi_2$ signifies that φ_2 holds at some point and, from that instant on, φ_1 holds too. $\diamond\varphi$ and $\diamondleftarrow\varphi$ mean that φ holds true eventually in the future, or at some instant in the past. Finally, $\square\varphi$ and $\squareleftarrow\varphi$ express the truthness of φ in every instant from the current one on, and in every instant from the current one back in the trace, respectively. We formalize the above as follows.

Given a finite trace t of length $n \in \mathbb{N}$, we write $(t, i) \models \varphi$ to denote that an LTL_f formula φ is satisfied at a given instant $i \in \mathbb{N}$, with $1 \leq i \leq n$, by induction of the following:

$$\begin{aligned} (t, i) \models & \text{True}; (t, i) \not\models \text{False}; \\ (t, i) \models & a \text{ iff } a \text{ is True in } t \text{ at instant } i; \end{aligned}$$

- $(t, i) \models \neg\varphi$ iff $(t, i) \not\models \varphi$;
 $(t, i) \models \varphi_1 \wedge \varphi_2$ iff $(t, i) \models \varphi_1$ and $(t, i) \models \varphi_2$;
 $(t, i) \models \varphi_1 \vee \varphi_2$ iff $(t, i) \models \varphi_1$ or $(t, i) \models \varphi_2$;
 $(t, i) \models \varphi_1 \rightarrow \varphi_2$ iff $(t, i) \not\models \varphi_1$ or $(t, i) \models \varphi_2$;
 $(t, i) \models \bigcirc\varphi$ 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 there exists a $j \in \mathbb{N}$ with $i \leq j \leq n$
 s.t. $(t, j) \models \varphi_2$ and $(t, k) \models \varphi_1$ for every $k \in \mathbb{N}$ s.t.
 $i \leq k < j$;
 $(t, i) \models \varphi_1 \mathbf{S} \varphi_2$ iff there exists a $j \in \mathbb{N}$ with $1 \leq j \leq i$
 s.t. $(t, j) \models \varphi_2$ and $(t, k) \models \varphi_1$ for every $k \in \mathbb{N}$ s.t.
 $j < k \leq i$.
 $(t, i) \models \diamond\varphi$ iff there exists a $j \in \mathbb{N}$ with $i \leq j \leq n$ s.t.
 $(t, j) \models \varphi$;
 $(t, i) \models \blacklozenge\varphi$ iff there exists a $j \in \mathbb{N}$ with $1 \leq j \leq i$ s.t.
 $(t, j) \models \varphi$;
 $(t, i) \models \square\varphi$ iff $(t, k) \models \varphi$ for every $k \in \mathbb{N}$ s.t. $i \leq k \leq n$;
 $(t, i) \models \boxminus\varphi$ iff $(t, k) \models \varphi$ for every $k \in \mathbb{N}$ s.t. $1 \leq k \leq i$;
 $(t, i) \models t_{\text{Start}}$ iff $(t, 1) \models \varphi$;
 $(t, i) \models t_{\text{End}}$ iff $(t, n) \models \varphi$.

For example, $(t, i) \models a \wedge \diamond b$ (i.e., $a \wedge \diamond b$ is satisfied in a trace t of length n at instant i) when the propositional atom a holds true in t at $i \leq n$ and b holds true at a later instant j with $i \leq j \leq n$ in the same trace t .

From the above operators, we observe the following logical equivalences:

- $\varphi_1 \vee \varphi_2 \equiv \neg(\neg\varphi_1 \wedge \neg\varphi_2)$;
- $\varphi_1 \rightarrow \varphi_2 \equiv \neg\varphi_1 \vee \varphi_2$;
- $t_{\text{End}} \equiv \neg(\bigcirc\text{True})$;
- $t_{\text{Start}} \equiv \neg(\ominus\text{True})$;
- $\diamond\varphi \equiv \text{True} \mathbf{U} \varphi$;
- $\blacklozenge\varphi \equiv \text{True} \mathbf{S} \varphi$;
- $\square\varphi \equiv \neg\blacklozenge\neg\varphi$;
- $\boxminus\varphi \equiv \neg\diamond\neg\varphi$.

This document is a pre-print copy of the manuscript
(Cecconi et al. 2024)
published by Elsevier.

The final version of the paper is identified by DOI: [10.1016/j.is.2023.102312](https://doi.org/10.1016/j.is.2023.102312)

References

Cecconi, Alessio, Luca Barbaro, Claudio Di Ciccio, and Arik Senderovich (2024). “Measuring rule-based LTLf process specifications: A probabilistic data-driven approach”. In: *Information Systems* 120, p. 102312. ISSN: 0306-4379. DOI: [10.1016/j.is.2023.102312](https://doi.org/10.1016/j.is.2023.102312).

BibTeX

```
@Article{
  author      = {Cecconi, Alessio and Barbaro, Luca and Di Ciccio, Claudio
                and Senderovich, Arik},
  journal     = {Information Systems},
  title       = {Measuring rule-based {LTLf} process specifications: A
                probabilistic data-driven approach},
  year        = {2024},
  issn        = {0306-4379},
  pages       = {102312},
  volume      = {120},
  doi         = {10.1016/j.is.2023.102312},
  keywords    = {Linear temporal logic; Declarative process mining;
                Specification mining; Probabilistic modeling; Statistical
                estimation},
  publisher   = {Elsevier}
}
```