Measurement of Rule-based LTLf Declarative Process Specifications

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

Claudio Di Ciccio
Sapienza University of Rome, Italy
claudio.diciccio@uniroma1.it

Arik Senderovich
York University, Toronto, Canada
sariks@yorku.ca

***Abstract*—The classical checking of declarative Linear Temporal Logic on Finite Traces (LTL_f) specifications verifies whether conjunctions of sets of formulae are satisfied by collections of finite traces. The data on which the verification is conducted may be corrupted by a number of logging errors or execution deviations at the level of single elements within a trace. The ability to quantitatively assess the extent to which traces satisfy a process specification (and not only if they do so or not at all) is thus key, especially in process mining scenarios. Previous techniques proposed for this aim either require formulae to be extended with quantitative operators or cater to the coarse granularity of whole traces. In this paper, we propose a framework to devise probabilistic measures for declarative process specifications on traces at the level of events, inspired by association rule mining. Thereupon, we describe a technique that measures the degree of satisfaction of these specifications over bags of traces. To assess our approach, we conduct an evaluation with real-world data.**

***Index Terms*—Linear temporal logic, declarative process mining, specification mining, probabilistic modeling, statistical estimation**

I. 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].

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. Consider, for example, a declarative process specification $\mathcal{S}$ consisting of two rules, Ψ_1 and Ψ_2. Ψ_1 translates to "Whenever antibiotics are administered, they should be preceded by the registration of an antibiogram". Ψ_2 indicates that "If a viral infection is detected, an intravenous antiviral administration will follow". Confidence is the proportion of events in a log that satisfy the consequent (*target*, i.e., the preceding registration of an antibiogram in Ψ_1) given the satisfaction of the antecedent (*activator*, i.e., the detection of a viral infection 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: violating a single rule or violating them all would lead to the same result. Likewise, aggregating measures over the single rules (as in [7]) may be misleading: if the activator of Ψ_1 occurs 100 times in the log, whilst the activator of Ψ_2 occurs twice, there is one violation out of the overall 102 occurrences of the activators. Nevertheless, the average confidence amounts to $75\,\%$.

To overcome this issue, we adapt and extend the concept of Reactive Constraint (RCon), originally proposed in [8], and the measurement framework for single declarative rules expressed as RCons [7]. RCons are rules expressed in an if–then fashion (like Ψ_1 and Ψ_2), namely a pair of LTL_f formulae: activator and target. RCons cover the full spectrum of declarative process specification languages such as DECLARE [1] as any LTL_f formula can be translated into an RCon [8]. Equipped with this notion, we propose a measurement framework that takes inspiration from classical association rule mining [9] to assess whether, and in how far, process specifications consisting of LTL_f-based rules expressed in such an "if–then" fashion are satisfied by a trace, 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 non-trivial problem, we move from an ad-hoc counting approach to sound probabilistic theory based on maximum likelihood estimation. Finally, we conduct an evaluation on real-world data of our approach with its software prototype implementation.

In the remainder of this paper, Sec.s II and III formalize the background notions our work is based upon: LTL_f and its interpretation on event logs, and RCons. Sec. IV lays the foundations of our probabilistic theory, upon which the evaluation and measurement of declarative process specifications are based upon as described in Sec. V. We report on the evaluation

The work of C. Di Ciccio was supported by the MUR under the PRIN programme, grant B87G22000450001 (PINPOINT), the "Dipartimenti di eccellenza 2018-2022" grant, and by the DRONES and SPECTRA Sapienza research projects.

of our implemented prototype on real-world data in Sec. VI. Sec. VII analyzes the research in the literature that relates to our investigation. Finally, Sec. VIII concludes the paper with remarks on future work.

II. 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 $L = \{t_1^{j_1}, \ldots, t_m^{j_m}\}$ of cardinality $|L| = \sum_{i=1}^{m} j_i$.
For example, Table I presents a log $L = \{t_1^{17}, t_2^{6}, t_3^{5}, t_4^{12}, t_5^{5}\}$ defined over alphabet $\Sigma = \{\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{d}, \mathsf{e}\}$. Its cardinality is 45.

Linear Temporal Logic on Finite Traces (LTL$_f$) [10] 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) [11] and is at the basis of declarative process specification languages such as Declare [12]. In this paper, we endow LTL$_f$ with past modalities as in [13].

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 unary temporal operators $\bigcirc$ (*next*) and $\ominus$ (*yesterday*), and the binary temporal operators $\mathbf{U}$ (*until*) and $\mathbf{S}$ (*since*) as follows:

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

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

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 in 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 $(1 \leqslant i \leqslant n)$ by induction of the following:

$(t,i) \models \mathit{True}$; $(t,i) \not\models \mathit{False}$;
$(t,i) \models a$ iff a is *True* in $t(i)$;
$(t,i) \models \neg\varphi$ iff $(t,i) \not\models \varphi$;
$(t,i) \models \varphi_1 \wedge \varphi_2$ iff $(t,i) \models \varphi_1$ and $(t,i) \models \varphi_2$;
$(t,i) \models \bigcirc\varphi$ iff $i < n$ and $(t,i+1) \models \varphi$;
$(t,i) \models \ominus\varphi$ iff $i > 1$ and $(t,i-1) \models \varphi$;
$(t,i) \models \varphi_1\ \mathbf{U}\ \varphi_2$ iff $(t,j) \models \varphi_2$ with $i \leqslant j \leqslant n$, and $(t,k) \models \varphi_1$ for all k s.t. $i \leqslant k < j$;
$(t,i) \models \varphi_1\ \mathbf{S}\ \varphi_2$ iff $(t,j) \models \varphi_2$ with $1 \leqslant j \leqslant i$, and $(t,k) \models \varphi_1$ for all k s.t. $j < k \leqslant i$.

Without loss of generality, we consider here the non-strict semantics of $\mathbf{U}$ and $\mathbf{S}$ [14]. Also, notice that each event in Table I satisfies only one proposition (thus applying the "Declare assumption" [15]) for the sake of simplicity. In the following, we might directly refer to the sequence of events $\langle e_1, \ldots, e_n \rangle$ of a trace t of length n to indicate the sequence of assignments at instants $1, \ldots, n$. For example, t_1, t_2, and t_4 in Table I are written as $\langle \mathsf{a,b,c,d,b,c,e,c,b} \rangle$, $\langle \mathsf{b,d,a,b,b,d,e,d,c} \rangle$, and $\langle \mathsf{b,c,a,c,e,a} \rangle$, respectively. We thereby indicate, e.g., that $(t_1, 1) \models \mathsf{a}$, $(t_2, 4) \models \mathsf{b}$, and $(t_4, 2) \models \mathsf{c}$. Considering again the formula $(\bigcirc\neg\mathsf{e})\ \mathbf{U}\ \mathsf{d}$, we have that $(t_1, 1)$ satisfies it, whereas $(t_2, 6)$ does not.

From the above operators, the following can be derived:

- Classical boolean abbreviations $\vee, \rightarrow$;
- Constant $t_{\text{End}} \equiv \neg \bigcirc \mathit{True}$, the last instant of a trace;
- Constant $t_{\text{Start}} \equiv \neg \ominus \mathit{True}$, the first instant of a trace;
- $\Diamond\varphi \equiv \mathit{True}\ \mathbf{U}\ \varphi$, indicating that φ holds true in a following instant before t_{End} (*eventually*);
- $\varphi_1\ \mathbf{W}\ \varphi_2 \equiv (\varphi_1\ \mathbf{U}\ \varphi_2) \vee \Box\varphi_1$, which relaxes $\mathbf{U}$ as φ_2 may never hold true (*weak until*);
- $\diamondminus\varphi \equiv \mathit{True}\ \mathbf{S}\ \varphi$, indicating that φ holds true in a preceding instant after t_{Start} (*once*);
- $\Box\varphi \equiv \neg\Diamond\neg\varphi$, indicating that φ holds true from the current instant till t_{End} (*always*);
- $\boxminus\varphi \equiv \neg\diamondminus\neg\varphi$, indicating that φ holds true from t_{Start} to the current instant (*historically*).

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

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

Theorem 2.1 ([16]): Let t be a finite trace of length $n \in \mathbb{N}$. Checking whether $(t,i) \models \varphi$ (with $1 \leqslant i \leqslant n$) satisfies an LTL$_f$ formula φ is feasible in $O(n^2 \times \|\varphi\|)$.

Proof: It follows from the proof in [16] elaborated for future operators ($\bigcirc$, $\Diamond$, $\Box$, and $\mathbf{U}$). Notice that the use of past modalities $\ominus$, $\boxminus$, $\diamondminus$ and $\mathbf{S}$ do not alter the complexity. Indeed, they can be included in the parse tree of the constructive proof in [16] as the respective future counterparts and checked against the trace read in reverse (i.e., from end to start [8]). ■

Corollary 2.1: Let L be an event log as per Def. 2.1 consisting of m traces of length up to $n \in \mathbb{N}$ and cardinality $|L| \geqslant m$. 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 Thm. 2.1 as the checking is done for the $O(n)$ events of all m distinct traces in L. ■

III. Rules as Reactive Constraints (RCons)

Intuitively, Reactive Constraints (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

Table I: Measurements of RCons $\Psi_1 = \mathsf{c} \Box\!\!\rightarrow \diamondminus \mathsf{a}$, and $\Psi_2 = \mathsf{d} \Box\!\!\rightarrow \Diamond \mathsf{e}$, and of specification $\mathcal{S} = \{\Psi_1, \Psi_2\}$ on a log

Log	Evaluation	RCon / Specification	P of RCon / P of $\mathcal{S}$	P of act.	P of target	Support	Confidence	Recall	Specificity	Lift
	$\langle$x, x, 1, x, x, 1, x, 1, x$\rangle$	$\Psi_1 = \mathsf{c} \Box\!\!\rightarrow \diamondminus \mathsf{a}$	1.00	0.33	1.00	0.33	1.00	0.33	0.00	1.00
$t_1 = \langle \mathsf{a,b,c,d,b,c,e,c,b} \rangle$	$\langle$x, x, x, 1, x, x, x, x, x$\rangle$	$\Psi_2 = \mathsf{d} \Box\!\!\rightarrow \Diamond \mathsf{e}$	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$\rangle$	$\mathcal{S} = \{\Psi_1, \Psi_2\}$	1.00	0.44	0.89	0.44	1.00	0.50	0.20	1.13
	$\langle$x, x, x, x, x, x, x, x, 1$\rangle$	$\Psi_1 = \mathsf{c} \Box\!\!\rightarrow \diamondminus \mathsf{a}$	1.00	0.11	0.78	0.11	1.00	0.14	0.25	1.29
$t_2 = \langle \mathsf{b,d,a,b,b,d,e,d,c} \rangle$	$\langle$x, 1, x, x, x, 1, x, 0, x$\rangle$	$\Psi_2 = \mathsf{d} \Box\!\!\rightarrow \Diamond \mathsf{e}$	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$\rangle$	$\mathcal{S} = \{\Psi_1, \Psi_2\}$	0.75	0.44	0.78	0.33	0.75	0.43	0.20	0.96
	$\langle$0, x, x, x, 1, x, x, 1, x, 1$\rangle$	$\Psi_1 = \mathsf{c} \Box\!\!\rightarrow \diamondminus \mathsf{a}$	0.75	0.40	0.80	0.30	0.75	0.38	0.17	0.94
$t_3 = \langle \mathsf{c,d,a,b,c,e,b,c,b,c} \rangle$	$\langle$x, 1, x, x, x, x, x, x, x, x$\rangle$	$\Psi_2 = \mathsf{d} \Box\!\!\rightarrow \Diamond \mathsf{e}$	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$	$\mathcal{S} = \{\Psi_1, \Psi_2\}$	0.80	0.50	0.70	0.40	0.80	0.57	0.40	1.14
	$\langle$x, 0, x, 1, x, x$\rangle$	$\Psi_1 = \mathsf{c} \Box\!\!\rightarrow \diamondminus \mathsf{a}$	0.50	0.33	0.67	0.17	0.50	0.25	0.25	0.75
$t_4 = \langle \mathsf{b,c,a,c,e,a} \rangle$	$\langle$x, x, x, x, x, x$\rangle$	$\Psi_2 = \mathsf{d} \Box\!\!\rightarrow \Diamond \mathsf{e}$	NaN	0.00	0.83	0.00	NaN	0.00	0.17	NaN
	$\langle$x, 0, x, 1, x, x$\rangle$	$\mathcal{S} = \{\Psi_1, \Psi_2\}$	0.50	0.33	0.50	0.17	0.50	0.33	0.50	1.00
	$\langle$x, x, x$\rangle$	$\Psi_1 = \mathsf{c} \Box\!\!\rightarrow \diamondminus \mathsf{a}$	NaN	0.00	0.00	0.00	NaN	NaN	1.00	NaN
$t_5 = \langle \mathsf{b,b,b} \rangle$	$\langle$x, x, x$\rangle$	$\Psi_2 = \mathsf{d} \Box\!\!\rightarrow \Diamond \mathsf{e}$	NaN	0.00	0.00	0.00	NaN	NaN	1.00	NaN
	$\langle$x, x, x$\rangle$	$\mathcal{S} = \{\Psi_1, \Psi_2\}$	NaN	0.00	0.00	0.00	NaN	NaN	1.00	NaN
		$\Psi_1 = \mathsf{c} \Box\!\!\rightarrow \diamondminus \mathsf{a}$	0.73	0.89	0.73	0.73	0.82	1.00	1.00	1.13
$L = \{t_1^{17}, t_2^{6}, t_3^{5}, t_4^{12}, t_5^{5}\}$ $\lvert L\rvert = 45$		$\Psi_2 = \mathsf{d} \Box\!\!\rightarrow \Diamond \mathsf{e}$	0.58	0.62	0.80	0.58	0.93	0.72	0.41	1.16
		$\mathcal{S} = \{\Psi_1, \Psi_2\}$	0.70	0.89	0.70	0.70	0.79	1.00	1.00	1.13

NaN values denote a division by 0.

$(\varphi_\alpha, \varphi_\tau)$ hereafter denoted as $\Psi \triangleq \varphi_\alpha \Box\!\!\rightarrow \varphi_\tau$. We name φ_α as *activator* and φ_τ as *target*.

We define the semantics of an RCon $\Psi = \varphi_\alpha \Box\!\!\rightarrow \varphi_\tau$ as follows: given a trace t of length n and instant i with $1 \leqslant i \leqslant n$, 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. $1 \leqslant i \leqslant n$ and $t, i \models \varphi_\alpha$. For example, $\Psi_1 = \mathsf{c} \Box\!\!\rightarrow \diamondminus \mathsf{a}$ in Table I is satisfied any time the occurrence of c (the activator) is preceded eventually in the past by a (as $\diamondminus \mathsf{a}$ is the target), violated when the occurrence of c is not preceded eventually in the past by a, and unaffected by every event in which c does not occur. Notice that by declaring that the activator is c in Ψ_1, the user makes the "trigger" of the rule explicit. $\Psi_1 = \mathsf{c} \Box\!\!\rightarrow \diamondminus \mathsf{a}$ and $\Psi_2 = \mathsf{d} \Box\!\!\rightarrow \Diamond \mathsf{e}$ 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. The expressiveness of RCons fully covers that of LTL_f and, a fortiori, of DECLARE. The examination of the expressiveness of RCons goes beyond the scope of this paper, though, and is discussed more in detail in [7], [8].

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 $\mathcal{S} \triangleq \{\Psi_1, \ldots, \Psi_s\}$, with $s \in \mathbb{N}$.

For example, Table I presents a specification $\mathcal{S} = \{\Psi_1, \Psi_2\}$ composed by the Ψ_1 RCon above and $\Psi_2 = \mathsf{d} \Box\!\!\rightarrow \Diamond \mathsf{e}$.

Corollary 3.1: Let $\mathcal{S} = \{\Psi_1, \ldots, \Psi_s\}$ 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 $\mathcal{S}$ is feasible in $O(n^3 \times \|\varphi\| \times |L| \times s)$.

The proof follows from Corollary 2.1, as a pair of labels is sufficient for all RCons in the specification. Take, e.g., trace $t_4 = \langle \mathsf{b,c,a,c,e,a} \rangle$ from Table I and the aforementioned RCon $\Psi_1 = \mathsf{c} \Box\!\!\rightarrow \diamondminus \mathsf{a}$. The activator ($\mathsf{c}$) 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 ($\diamondminus \mathsf{a}$) 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 [17]. Next, we lay the foundations rooted in probabilistic theory to reach this goal.

IV. ESTIMATORS FOR LTL_f FORMULAE

The interestigness measures for RCons are based on the probabilities of their activator (φ_α) and target (φ_τ) LTL_f formulae. Thus, 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.

A. 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 $t = \langle e_1, \ldots, e_n \rangle$ 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 Ω 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., $\Omega(e, \varphi) \in \{0, 1\}$. This procedure can be achieved in polynomial time as shown in Corollary 2.1 through automata-based techniques for LTL_f formulae verification [7]. Therefore, every trace t can be associated with a binary sequence denoted by $x_{\varphi,t} = \langle \Omega(e_1, \varphi), \ldots, \Omega(e_n, \varphi) \rangle$. Take again, e.g., $t_2 = \langle e_{2,1}, \ldots, e_{2,9} \rangle = \langle \mathsf{b,d,a,b,b,d,e,d,c} \rangle$ from Table I, and formula $\varphi = \mathsf{d} \wedge \Diamond \mathsf{e}$. As only $(t_2, 2)$ and $(t_2, 6)$ satisfy φ, $\Omega(e_{2,1}, \varphi)$ and $\Omega(e_{2,6}, \varphi)$ return 1 while $\Omega(e_{2,i}, \varphi)$ is 0 for every $i \in \{1, \ldots, 9\} \backslash \{2, 6\}$. Therefore, $x_{\varphi,t_2} = \langle 0, 1, 0, 0, 0, 1, 0, 0, 0 \rangle$.

Moreover, 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. We intend to relax this assumption in future work.

Due to the event independence of evaluation assumption, the sequence $x_{\varphi,t}$ can be viewed as an independent and identically distributed (i.id.) draw from a Bernoulli random variable $X_{\varphi,t}$, which takes the value of 1 with probability $P(\varphi(t))$ and 0 otherwise:

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

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

Proposition 4.1: The maximum likelihood estimator (MLE) for $P(\varphi(t))$ is

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

The proof follows from the assumption that $X_{\varphi,t}$ is a univariate Bernoulli random variable, whose MLE is well-established in the literature (see, e.g., [18]).

Returning to our running example (Table I), 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 $\varphi_\alpha \,\Box\!\!\rightarrow\, \varphi_\tau$ in $\mathcal{S}$ and for every trace $t \in L$. Let us consider again trace $t_4 = \langle \mathsf{b}, \mathsf{c}, \mathsf{a}, \mathsf{c}, \mathsf{e}, \mathsf{a} \rangle$ and the RCon $\Psi_1 = \varphi_{\alpha_1} \,\Box\!\!\rightarrow\, \varphi_{\tau_1} = \mathsf{c} \,\Box\!\!\rightarrow\, \Diamond\mathsf{a}$. As we have previously discussed in Sec. III, 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 $P\left(\widehat{\varphi_{\alpha_1}(t_4)}\right)$ is $\frac{2}{6}$ and $P\left(\widehat{\varphi_{\tau_1}(t_4)}\right)$ 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 (or violated) by a trace, e.g., $P(\varphi_1(t) \cap \varphi_2(t))$, and for the conditional distribution of φ_1 to be satisfied (or violated) by trace t conditional on φ_2 being satisfied (or violated) 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 (e.g., quantifying $P(\neg\varphi_1(t) \cap \varphi_2(t))$ and $P(\neg\varphi_1(t) | \varphi_2(t))$). Formalizing the above, we wish to estimate the quantities of interest from a labeled sequence,

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

Take, e.g, $t_4 = \langle \mathsf{b}, \mathsf{c}, \mathsf{a}, \mathsf{c}, \mathsf{e}, \mathsf{a} \rangle$ from Table I and the pair of activation and target of $\Psi_1 = \mathsf{c} \,\Box\!\!\rightarrow\, \Diamond\mathsf{a}$, i.e., $\varphi_{\alpha_1} = \mathsf{c}$ and $\varphi_{\tau_1} = \Diamond\mathsf{a}$, 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.id. 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 [19]. The estimation of each p_{ij} is proposed in [20]. When estimating $P(\varphi_1(t) \cap \varphi_2(t))$ we are essentially interested in an estimator for p_{11}.

Proposition 4.2: The MLE for $P(\varphi_1(t) \cap \varphi_2(t))$ is

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

The proof follows from the use of estimators for p_{ij} (with $i, j \in \{0,1\}$) provided in [20]. From the example above, we conclude that $P(\varphi_{\alpha_1}\widehat{(t_4) \cap \varphi_{\tau_1}}(t_4))$ is $\frac{1}{6}$ as only one element in $x_{(\varphi_{\alpha_1},\varphi_{\tau_1}),t_4}$ is $(1,1)$.

Similarly, we can estimate the other combinations of satisfaction and violation of the two formulae (namely, $\hat{p}_{00}$, $\hat{p}_{01}$, and $\hat{p}_{10}$). 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 $X_{\varphi_1,t} \mid X_{\varphi_2,t} = x_2$ is a *univariate* Bernoulli that depends only on sequence x_2 (which results from applying Ω to φ_2 and t) and on the four parameters p_{ij} of the joint bivariate Bernoulli distribution (see the proof in [19]). This result leads to our estimators.

Proposition 4.3: The MLE for $P(\varphi_1(t) | \varphi_2(t))$ is

$$P(\varphi_1\widehat{(t) | \varphi_2}(t)) = \frac{\hat{p}_{11}}{\hat{p}_{01} + \hat{p}_{11}}, \quad (5)$$

with $\hat{p}_{11}$ as derived in Prop. 4.2 and $\hat{p}_{01}$ being

$$\hat{p}_{01} = \sum_{i=1}^{n} \frac{(1 - \Omega(e_i,\varphi_1))\Omega(e_i,\varphi_2)}{n}. \quad (6)$$

The outcome is due to the derivation of the conditional distribution for bivariate Bernoulli (see [19]) and MLE estimation in univariate Bernoulli (see [18]).[1]

Take, e.g., $t_4 = \langle \mathsf{b}, \mathsf{c}, \mathsf{a}, \mathsf{c}, \mathsf{e}, \mathsf{a} \rangle$ from Table I as above, $\varphi_{\alpha_1} = \mathsf{c}$ and $\varphi_{\tau_1} = \Diamond\mathsf{a}$, namely the activator and target of Ψ_1. We have that $P(\varphi_{\tau_1}\widehat{(t_4) | \varphi_{\alpha_1}}(t_4))$ is $\frac{1}{2}$ as φ_{α_1} is satisfied by two events, only one of which satisfies φ_{τ_1} too.

B. 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 $L = \{t_1^{j_1}, \ldots, t_m^{j_m}\}$ is a bag of traces with trace $t_i^{j_i}$ occurring j_i times in the log. Let us denote with $\bar{L} = \{t_1, \ldots, t_m\}$ 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 $P(T = t)$.[2] Let $\mathcal{T}$ be the support of the probability distribution of T, i.e., $\mathcal{T} = \{t \mid P(T = t) > 0\}$.

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 Bernoulli random variable, as its success probability, $P(X_{\varphi,T} = 1)$, changes for randomly

[1]When estimating $P(\varphi_1\widehat{(t) | \varphi_2}(t))$, 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.

[2]In practice, the trace can be generated via a random walk over, e.g., a finite-state automaton [21].

sampled traces. We shall use $X_{\varphi,T}$ together with our log completeness assumption to derive an estimator for $P(\varphi(L))$. From the law of total probability (LTP) we get that

$$P(X_{\varphi,T} = x) = \sum_{t \in \mathcal{T}} P(T = t) P(X_{\varphi,t} = x), \qquad (7)$$

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. (7), plug in $P(\varphi(L))$ for $P(X_{\varphi,T} = 1)$, and $P(\varphi(t))$ for $P(X_{\varphi,t} = 1)$, thus immediately arriving at an estimator for $P(\varphi(L))$.

Proposition 4.4: The MLE for $P(\varphi(L))$ is

$$\widehat{P(\varphi(L))} = \sum_{i=1}^{m} \widehat{P(T = t_i)} \widehat{P(\varphi(t_i))}, \qquad (8)$$

with $\widehat{P(\varphi(t_i))}$ estimated as in Prop. 4.1, and

$$\widehat{P(T = t_i)} = \frac{j_i}{\sum_{k=1}^{m} j_k}, \qquad (9)$$

with m being the number of unique traces in L.

The proof that Prop. 4.4 provides an MLE comes from LTP and the fact that T and $X_{\varphi,t}$ form a two-node Bayesian network where T is directly followed by $X_{\varphi,t}$, which allows for the use of MLE results for Bayesian networks (see [22]). For example, t_4 has a multiplicity of 12 considering the example log in Table I. The cardinality of the example log L is 45, so $\widehat{P(t = t_4)}$ is $\frac{12}{45}$. We saw in Sec. IV-A that $P\,\widehat{(\varphi_{\alpha_1}(t_4))}$ is $\frac{2}{6}$ for $\varphi_{\alpha_1} = \mathsf{c}$. Therefore, the term for $i = 4$ in the summation of Eq. (8) is $\frac{12}{45} \cdot \frac{2}{6} \approxeq 0.09$ for φ_{α_1}. Extending the sum to all the traces in $\bar{L}$, we have that the value of $P\widehat{(\varphi_{\alpha_1}(L))}$ is approximately 0.27.

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.

Proposition 4.5: The MLE of $P(\varphi_1(L) \cap \varphi_2(L))$ is

$$P(\varphi_1\widehat{(L) \cap} \varphi_2(L)) = \sum_{t \in \bar{L}} \widehat{P(T = t)} P(\varphi_1\widehat{(t) \cap} \varphi_2(t)). \qquad (10)$$

with $P(T = t)$ being estimated as in Prop. 4.4 and $P(\varphi_1(t) \cap \varphi_2(t))$ estimated using Prop. 4.2.

In Sec. IV-A, e.g., we showed that $P(\varphi_{\alpha_1}\widehat{(t_4) \cap} \varphi_{\tau_1}(t_4))$ is $\frac{1}{6}$ considering the example log, φ_{α_1} as above, and $\varphi_{\tau_1} = \Diamond\!\!\!-\,\mathsf{a}$. Recalling that $\widehat{P(t = t_4)}$ is $\frac{12}{45}$, we have that the term of the summation in Eq. (10) for $t = t_4$ is $\frac{12}{45} \cdot \frac{1}{6} \approxeq 0.04$. The value of $P(\varphi_{\alpha_1}\widehat{(L) \cap} \varphi_{\tau_1}(L))$ is computed by summing up the elements obtained for every $t \in \bar{L}$, thus obtaining 0.22.

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

Proposition 4.6: The MLE for $P(\varphi_1(L)|\varphi_2(L))$ is

$$P(\varphi_1\widehat{(L) \mid} \varphi_2(L)) = \frac{P(\varphi_1\widehat{(L) \cap} \varphi_2(L))}{P\widehat{(\varphi_2(L))}}, \qquad (11)$$

with $P(\varphi_1\widehat{(L) \cap} \varphi_2(L))$ estimated as in Prop. 4.5, and $P\widehat{(\varphi_2(L))}$ as in Prop. 4.4.

The proof follows from Kolmogorov's analysis (see the reasoning of the proof for Prop. 4.3). Notice that we assume $P\widehat{(\varphi_2(L))} > 0$ as before to avoid division by zero. For instance, we showed above that for the example log L in Table I the estimations $P(\varphi_{\alpha_1}\widehat{(L) \cap} \varphi_{\tau_1}(L))$ and $P\widehat{(\varphi_{\alpha_1}(L))}$ are 0.22 and 0.27, respectively. By applying the computation above, we have that $P(\varphi_{\tau_1}\widehat{(L) \mid} \varphi_{\alpha_1}(L))$ is $\frac{0.22}{0.27} \approxeq 0.82$.

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

Theorem 4.1: 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.

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.

V. 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 Sec. III), 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, continue proposing estimators of probabilities of traces and log satisfying such specifications, and finally describe the computation of interestingness measures.

A. Evaluating Specifications

Formally, we define the semantics of a specification $\mathcal{S}$ as follows: given a trace t of length n, an instant i with $1 \leqslant i \leqslant n$, and a specification $\mathcal{S} \triangleq \{\Psi_1, \ldots, \Psi_s\}$, with $s \in \mathbb{N}$ and $\Psi_j = \varphi_{\alpha_j} \mathrel{\Box\!\!\rightarrow} \varphi_{\tau_j}$ for every j s.t. $1 \leqslant j \leqslant s$, we say that $\mathcal{S}$ **is activated by** t **in** i, i.e., $(t, i) \models \mathcal{S}_\alpha$, iff there exists a $\Psi_j \in \mathcal{S}$ s.t. $(t, i) \models \varphi_{\alpha_j}$; $\mathcal{S}$ **is satisfied by** t **in** i, $(t, i) \models \mathcal{S}$, iff $(t, i) \models \mathcal{S}_\alpha$ and there does not exist any $\Psi_j \in \mathcal{S}$ s.t. $(t, i) \not\models \Psi_j$; $\mathcal{S}$ **is violated by** t **in** i, $(t, i) \not\models \mathcal{S}$, iff there exists a $\Psi_j \in \mathcal{S}$ s.t. $(t, i) \not\models \Psi_j$; $\mathcal{S}$ **is unaffected by** t **in** i iff $(t, i) \not\models \mathcal{S}_\alpha$. In other words, $\mathcal{S}$ 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 $\mathcal{S} = \{\Psi_1, \ldots, \Psi_s\}$ as an RCon, $\mathcal{S} = \mathcal{S}_\alpha \mathrel{\Box\!\!\rightarrow} \mathcal{S}_\tau$, where $\mathcal{S}_\alpha$ and $\mathcal{S}_\tau$ are LTL_f formulae expressed as follows:

$$\mathcal{S}_\alpha = \bigvee_{j=1}^{s} \varphi_{\alpha_j}; \qquad \mathcal{S}_\tau = \bigwedge_{j=1}^{s} \neg(\varphi_{\alpha_j} \wedge \neg\varphi_{\tau_j}). \qquad (12)$$

For example, $\mathcal{S}$ from Table I is activated when either Ψ_1 or Ψ_2 are (i.e., $\mathcal{S}_\alpha = \mathsf{c} \vee \mathsf{d}$) and it is satisfied when all the activated constraints are satisfied, i.e., $\mathcal{S}_\tau = (\neg\mathsf{c} \vee \Diamond\!\!\!-\,\mathsf{a}) \wedge (\neg\mathsf{d} \vee \Diamond\mathsf{e})$. Hence, $\mathcal{S}$ 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.

B. Estimators for Specifications

Trace Estimators: Let Ω_R be an RCon interpreter that takes an event and an RCon $\Psi = \varphi_\alpha \mathrel{\square\!\!\!\rightarrow} \varphi_\tau$ and returns a label $\Omega_R(e, \Psi) \in \{0, \mathsf{x}, 1\}$ corresponding to Ψ being violated, unaffected, and satisfied by e, respectively. The labeling of an event given an RCon Ψ resorts to Ω (explained in Sec. IV-A) as follows:

$$\Omega_R(e, \Psi) = \begin{cases} 0, & \text{if } \; \Omega(e, \varphi_\alpha) = 1 \;\text{ and }\; \Omega(e, \varphi_\tau) = 0, \\ 1, & \text{if } \; \Omega(e, \varphi_\alpha) = 1 \;\text{ and }\; \Omega(e, \varphi_\tau) = 1, \\ \mathsf{x}, & \text{otherwise.} \end{cases} \tag{13}$$

Notice that x is a new outcome of the labeling function Ω_R that solely applies to RCons but not to LTL_f formulae (see the definition of Ω in Sec. IV-A). The second column of Table I lists the labels assigned by Ω_R to every event in the traces and all RCons in a sequence. For example, the evaluation of $\Psi_1 = \varphi_{\alpha_1} \mathrel{\square\!\!\!\rightarrow} \varphi_{\tau_1} = \mathsf{c} \mathrel{\square\!\!\!\rightarrow} \Diamond \mathsf{a}$ on $t_4 = \langle e_{4,1}, \ldots, e_{4,6} \rangle = \langle \mathsf{b}, \mathsf{c}, \mathsf{a}, \mathsf{c}, \mathsf{e}, \mathsf{a} \rangle$ is such that $(t_4, 1) \not\models \mathsf{c}$, thus $\Omega_R(e_{4,1}, \Psi_1) = \mathsf{x}$; also, we observe that $(t_4, 2) \models \mathsf{c}$ but $(t_4, 2) \not\models \Diamond \mathsf{a}$, therefore $\Omega_R(e_{4,2}, \Psi_1) = 0$; finally, we have that $(t_4, 4) \models \mathsf{c}$ and $(t_4, 4) \models \Diamond \mathsf{a}$, so $\Omega_R(e_{4,4}, \Psi_1) = 1$. Following this approach, we attain the following sequence of labels from t_4 via Ω_R on Ψ_1: $\langle \mathsf{x}, 0, \mathsf{x}, 1, \mathsf{x}, \mathsf{x} \rangle$.

We are interested in estimating the probabilities of satisfaction and violation of a given RCon in a trace, which correspond to cases in which the RCon is activated. Formally, let $P(\Psi(t)) = P(\varphi_\tau(t) \mid \varphi_\alpha(t))$ be the probability of an RCon Ψ to be satisfied by t, and let $P(\neg\Psi(t)) = P(\neg\varphi_\tau(t) \mid \varphi_\alpha(t)) = 1 - P(\varphi_\tau(t) \mid \varphi_\alpha(t))$ be the complementary probability of $P(\Psi(t))$ being violated.

Proposition 5.1: The MLE of $P(\Psi(t))$ is given by

$$\widehat{P(\Psi(t))} = \widehat{P(\varphi_\tau(t) \mid \varphi_\alpha(t))} = \frac{\hat{p}_{11}}{\hat{p}_{01} + \hat{p}_{11}}, \tag{14}$$

with p_{11} and p_{01} estimated as in Prop. 4.2, and the MLE of $P(\neg\Psi(t))$ given by $\widehat{P(\neg\Psi(t))} = 1 - \widehat{P(\Psi(t))}$.

The proof follows from Prop.s 4.2 and 4.3. For example, $\widehat{P(\Psi_1(t_4))} = \frac{1}{2}$, as shown in Sec. IV-A.

At this stage, we turn to estimate the probabilities of interest for a specification $\mathcal{S}$ (i.e., a set of RCons). Specifically, since a specification is interpreted similarly to a single RCon, once we obtained the interpretation for $\mathcal{S}_\alpha$ and $\mathcal{S}_\tau$, we apply the labeling mechanism Ω_R to get one of the three possible outcomes, which allows for the re-use of Prop. 5.1 to obtain estimators for $P(\mathcal{S}(t))$ and $P(\neg\mathcal{S}(t))$. The following proposition provides our main results for estimating the probability of a trace to satisfy a specification of RCons.

Proposition 5.2: The MLE of $P(\mathcal{S}(t))$ is given by

$$\widehat{P(\mathcal{S}(t))} = \widehat{P(\mathcal{S}_\tau(t) \mid \mathcal{S}_\alpha(t))} = \frac{\hat{p}_{11}}{\hat{p}_{01} + \hat{p}_{11}}, \tag{15}$$

with p_{11} and p_{01} estimated as in Prop. 4.2, and the MLE of $P(\neg\mathcal{S}(t))$ given by $\widehat{P(\neg\mathcal{S}(t))} = 1 - \widehat{P(\mathcal{S}(t))}$.

Notice that we use a similar notation for single RCons, replacing $\Psi(t)$ with $\mathcal{S}(t)$ to denote satisfaction of a specification. For example, $\widehat{P(\mathcal{S}(t_4))} = \frac{1}{2}$, as we consider only 0 or 1 outcomes applying the computation described in Prop. 5.2.

Table II: Interestingness measures

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

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.

Proposition 5.3: Let $P(\mathcal{S}(L))$ denote the probability of a log L to satisfy a specification $\mathcal{S}$. The MLE of $P(\mathcal{S}(L))$ is given by $\widehat{P(\mathcal{S}(L))} = \sum_{t \in \bar{L}} \widehat{P(T = t)} \widehat{P(\mathcal{S}(t))}$, and the MLE of $P(\neg\mathcal{S}(L))$ is given by $1 - \widehat{P(\mathcal{S}(L))}$.

The proof follows the same lines as that of Prop. 4.4. Returning to the running example in Table I, $\widehat{P(\mathcal{S}(L))}$ is $\frac{(17 \cdot 1.0) + (6 \cdot 0.75) + (5 \cdot 0.80) + (12 \cdot 0.50)}{45} = 0.7$.

Similarly to Sec. IV, we conclude providing a result that states the computational complexity of our technique.

Theorem 5.1: The estimation of the interestingness measures over specifications of RCon given an event log is polynomial. The proof relies on a similar argument to the proof of Theorem 4.1: 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.

C. 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, as shown in Table II (additional interestingness measures can be found in [7]). These measures are based on the probabilities of the satisfaction of the activator and target conditions for a rule on traces or logs [9], which makes them suitable for measuring interestingness of RCons. Let us consider a few examples of such measures: Given a trace t, the Support measure, $P(\varphi_\alpha \cap \varphi_\tau, t)$, quantifies the satisfaction both of the activator and target; Confidence, $P(\varphi_\tau | \varphi_\alpha, t)$, considers the conditional occurrence of the target given the occurrence of the activator; finally, Specificity, $P(\neg\varphi_\tau | \neg\varphi_\alpha, t)$, measures the non-occurrence of the target given the non-occurrence of the activator.

Measuring the interestingness of specifications is now possible thanks to Prop.s 5.2 and 5.3. Specifically, the estimates that we derive for specifications enable us to compute a plethora of measures while considering the joint effects of multiple rules at once, beyond their strict boolean conjunction. Thereby, we advance the state of the art as measures were previously restricted to a single-rule scope [7].

In the running example (see Table I), we use Prop.s 5.2 and 5.3 to compute the measures that appear in the last five columns (Support, Confidence, etc.) for $\mathcal{S} = \{\Psi_1, \Psi_2\}$. Observing the result in the last line of Table I, e.g., we have that $\widehat{P(\mathcal{S}(L))}$ is 0.7, which together with the probabilities of activator and target of $\mathcal{S}$ yield a Support of 0.7 and Confidence of 0.79.

VI. Evaluation

We implemented our technique in a proof-of-concept Java tool.[3] The implementation natively supports a relevant set of rule templates based on [23], but we already showed in Sec. V that the technique is seamlessly applicable to any RCon. It supports the computation of 37 measures inspired by [9]. 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 [8], MINERful [24], and Perracotta [25]. The real-world dataset used for the experiments contains records of patient visits at a Dutch hospital.[4] The dataset exhibits high variability: 75% of the traces are unique, which makes it a good candidate to evaluate partially-satisfied specifications. Due to space limitations, we show the outcome for one dataset. The interested reader can find the scripts and input files to reproduce the tests alongside output reports, additional experimental data and the full collection of specification rules at https://oneiroe.github.io/DeclarativeSpecificationMeasurements-static/.

We remark that our tool 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.

a) Measuring Single Rules and Entire Specifications: First, we highlight the importance of checking an entire specification, as opposed to the analysis of single rules. The rationale is that while many specification miners use a threshold to retrieve only rules satisfied for a number of times above that value, the corresponding specification may present a satisfaction degree below the desired level. 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, at a step of 0.05. Finally, we compared the measures of the specifications to the input threshold. The results can be found in Fig. 1. Markedly, every miner returned specifications whose overall Confidence was lower than the initially set threshold. This issue may lead to sub-optimal results, similar to the multiple testing problem [26] in statistical inference. With our technique, it is possible to spot such behavior. Improving declarative process specification miners with the integration of our technique paves the path for an interesting future endeavor.

b) Differing Measures: We show how different specifications, though never violated in the log, may exhibit different characteristics through the usage of multiple measures. To retrieve multiple distinct specifications, we first mined a specification with the miner set to discard partly violated rules. Any miner would 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 III. The results show that, despite Confidence is 1 for every specification, other measures can still spot differences. For example, Specificity returns a division by zero for $\mathcal{S}_5$, which suggests that the activator was satisfied in every event of the log. Also, Lift shows that in $\mathcal{S}_2$ and $\mathcal{S}_1$ the satisfaction of both activator and target is higher than their individual satisfaction degree.

[3]Publicly available at https://github.com/Oneiroe/Janus.

[4]The hospital dataset is publicly available at the following address: http://dx.doi.org/10.4121/uuid:915d2bfb-7e84-49ad-a286-dc35f063a460.

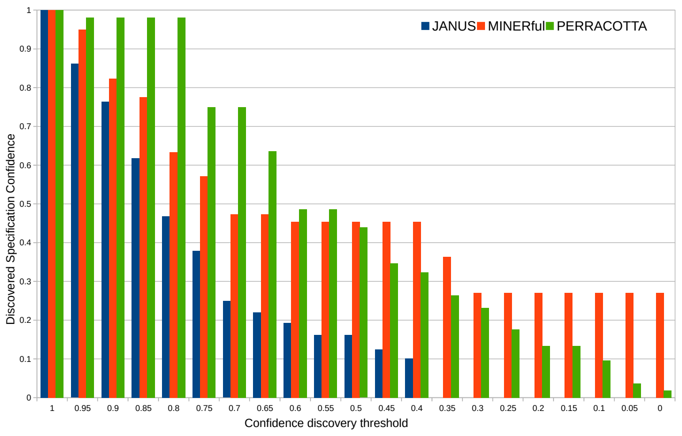

Figure 1: Confidence of the mined specifications with respect to the threshold used for their discovery

Table III: Measurements of sub-specifications

Measure	Original $\mathcal{S}$	$\mathcal{S}_1$	$\mathcal{S}_2$	$\mathcal{S}_3$	$\mathcal{S}_4$	$\mathcal{S}_5$
Confidence	1.000	1.000	1.000	1.000	1.000	1.000
Support	1.000	0.052	0.190	0.748	0.965	1.000
Recall	1.000	0.169	1.000	1.000	1.000	1.000
Specificity	NaN	0.728	1.000	1.000	1.000	NaN
Lift	1.000	3.221	5.250	1.338	1.037	1.000

VII. Related Work

Different contributions in the literature aim at quantitative extensions of LTL/LTL_f enriching the languages with quantitative operators. LTL[$\mathcal{F}$] [27] introduces quality operators quantifying over distinct satisfactions of a formula. Quantified-LTL [28] uses quantifiers over its propositional variables, also in probabilistic systems such as Markov chains. In [29], the quantification of satisfaction is based on associating costs to specification violations based on user ranking of tasks priority. Differently from all of them, 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 [30] 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 [31] 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 [32]. In [33] Fitness, Precision, and Generalization are devised for DECLARE models through alignments, while in [34] 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 [35] to understand the differences between compliant traces. We showed in Sec. VI how different measures can spot differences in compliant specifications, thus a deeper analysis in this direction is an interesting research outlook.

VIII. 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. We apply our prototype to real-world data, showing it is possible to evaluate existing specification miners beyond the boolean satisfaction of the intersection of all formulae.

Our result for LTL_f can be easily extended to Linear Dynamic Logic over Finite Traces [10], 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 [36] 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. Another interesting outlook would be the employment of specifications measures as features of the data for machine learning applications, e.g., trace clustering [37].

References

[1] M. Pesic, D. Bosnacki, and 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, and G. Perelli, "HyperLDLf: a logic for checking properties of finite traces process logs," in *IJCAI*, 2021, pp. 1859–1865.

[3] F. Bacchus and 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, and 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, and I. Beschastnikh, "General LTL specification mining (T)," in *ASE*, 2015, pp. 81–92.

[6] Z. Cao, Y. Tian, T. Le, and D. Lo, "Rule-based specification mining leveraging learning to rank," *Autom. Softw. Eng.*, vol. 25, no. 3, pp. 501–530, 2018.

[7] A. Cecconi, G. De Giacomo, C. Di Ciccio, F. Maggi, and J. Mendling, "Measuring the interestingness of temporal logic behavioral specifications in process mining," *Information Systems*, p. 101920, 2021.

[8] A. Cecconi, C. Di Ciccio, G. De Giacomo, and J. Mendling, "Interestingness of traces in declarative process mining: The Janus LTLp_f approach," in *BPM*, 2018, pp. 121–138.

[9] L. Geng and H. Hamilton, "Interestingness measures for data mining: A survey," *ACM Comput. Surv.*, vol. 38, no. 3, p. 9, 2006.

[10] G. De Giacomo and M. Vardi, "Linear temporal logic and linear dynamic logic on finite traces," in *IJCAI*, 2013, pp. 854–860.

[11] A. Pnueli, "The temporal logic of programs," in *FOCS*, 1977, pp. 46–57.

[12] C. Di Ciccio and M. Montali, "Declarative process specifications: Reasoning, discovery, monitoring," in *Process Mining Handbook*, W. M. P. van der Aalst and J. Carmona, Eds. Springer, 2022, pp. 108–152.

[13] O. Lichtenstein, A. Pnueli, and L. Zuck, "The glory of the past," in *Logic of Programs*, 1985, pp. 196–218.

[14] I. Hodkinson and M. Reynolds, "Separation - past, present, and future," in *We Will Show Them! (2)*, 2005, pp. 117–142.

[15] G. De Giacomo, R. De Masellis, and M. Montali, "Reasoning on LTL on finite traces: Insensitivity to infiniteness," in *AAAI*, 2014, pp. 1027–1033.

[16] V. Fionda and G. Greco, "The complexity of LTL on finite traces: Hard and easy fragments," in *AAAI*, 2016, pp. 971–977.

[17] G. De Giacomo, F. Maggi, A. Marrella, and S. Sardiña, "Computing trace alignment against declarative process models through planning," in *ICAPS*, 2016, pp. 367–375.

[18] P. Bickel and K. Doksum, *Mathematical statistics: basic ideas and selected topics, volumes I-II package*. CRC Press, 2015.

[19] B. Dai, S. Ding, and G. Wahba, "Multivariate bernoulli distribution," *Bernoulli*, vol. 19, no. 4, pp. 1465–1483, 2013.

[20] S. Ip and J. Xue, "A multivariate regression view of multi-label classification," University College London, Tech. Rep., 2015.

[21] C. Di Ciccio, M. L. Bernardi, M. Cimitile, and F. M. Maggi, "Generating event logs through the simulation of Declare models," in *EOMAS@CAiSE*, 2015, pp. 20–36.

[22] F. Jensen and T. Nielsen, *Bayesian networks and decision graphs*. Springer, 2007, vol. 2.

[23] M. Dwyer, G. Avrunin, and J. Corbett, "Patterns in property specifications for finite-state verification," in *ICSE*, 1999, pp. 411–420.

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

[25] J. Yang, D. Evans, D. Bhardwaj, T. Bhat, and M. Das, "Perracotta: mining temporal API rules from imperfect traces," in *ICSE*, 2006, pp. 282–291.

[26] W. Hämäläinen and G. Webb, "A tutorial on statistically sound pattern discovery," *Data Min. Knowl. Discov.*, vol. 33, no. 2, pp. 325–377, 2019.

[27] S. Almagor, U. Boker, and O. Kupferman, "Formally reasoning about quality," *J. ACM*, vol. 63, no. 3, pp. 24:1–24:56, 2016.

[28] J. Piribauer, C. Baier, N. Bertrand, and O. Sankur, "Quantified linear temporal logic over probabilistic systems with an application to vacuity checking," in *CONCUR*, 2021, pp. 7:1–7:18.

[29] M. Lahijanian, S. Almagor, D. Fried, L. Kavraki, and M. Vardi, "This time the robot settles for a cost: A quantitative approach to temporal logic planning with partial satisfaction," in *AAAI*, 2015, pp. 3664–3671.

[30] A. Legay, A. Lukina, L. Traonouez, J. Yang, S. Smolka, and R. Grosu, "Statistical model checking," in *Computing and Software Science - State of the Art and Perspectives*, 2019, pp. 478–504.

[31] F. Maggi, M. Montali, and R. Peñaloza, "Temporal logics over finite traces with uncertainty," in *AAAI*, 2020, pp. 10 218–10 225.

[32] J. C. Buijs, B. F. van Dongen, and W. M. van der Aalst, "Quality dimensions in process discovery: The importance of fitness, precision, generalization and simplicity," *Int. J. Cooperative Inf. Syst.*, vol. 23, no. 01, p. 1440001, 2014.

[33] M. De Leoni, F. Maggi, and W. van der Aalst, "An alignment-based framework to check the conformance of declarative process models and to preprocess event-log data," *Inf. Syst.*, vol. 47, pp. 258–277, 2015.

[34] A. Polyvyanyy, A. Solti, M. Weidlich, C. Di Ciccio, and J. Mendling, "Monotone precision and recall measures for comparing executions and specifications of dynamic systems," *ACM Trans. Softw. Eng. Methodol.*, vol. 29, no. 3, pp. 17:1–17:41, 2020.

[35] A. Burattin, G. Guizzardi, F. M. Maggi, and M. Montali, "Fifty shades of green: How informative is a compliant process trace?" in *CAiSE*, 2019, pp. 611–626.

[36] S. Schönig, C. Di Ciccio, F. M. Maggi, and J. Mendling, "Discovery of multi-perspective declarative process models," in *ICSOC*, 2016, pp. 87–103.

[37] J. De Weerdt, "Trace clustering," in *Encyclopedia of Big Data Technologies*, 2019.