Measurement of Rule-based LTLf Declarative Process Specifications

Alessio Cecconi  
WU Vienna, Austria  
alessio.ceccconi@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 (LTLf) 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 LTLf) 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, $\Psi_1$ and $\Psi_2$. $\Psi_1$ translates to “Whenever antibiotics are administered, they should be preceded by the registration of an antibiogram”. $\Psi_2$ 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 $\Psi_1$) given the satisfaction of the antecedent (activator, i.e., the detection of a viral infection in $\Psi_2$). Suppose the confidence of $\Psi_1$ and $\Psi_2$ 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., $\Psi_1 \land \Psi_2$) 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 $\Psi_1$ occurs 100 times in the log, whilst the activator of $\Psi_2$ 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 $\Psi_1$ and $\Psi_2$), namely a pair of LTLf formulae: activator and target. RCons cover the full spectrum of declarative process specification languages such as DECLARE [1] as any LTLf 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 LTLf-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: LTLf 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 of...
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 $\Sigma$, we name as event an assignment for the symbols in $\Sigma$ 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 $L = \{t_1^2, t_2^3, t_3^4, t_2^5\}$ of cardinality 17.

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 expressive power 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 $\circ$ (next) and $\ominus$ (yesterday), and the binary temporal operators $U$ (until) and $S$ (since) as follows:

$$\varphi ::= True|False|a | \neg \varphi | (\varphi_1 \land \varphi_2) | (\varphi_1 \land \varphi_2) | (\varphi_1 \land \varphi_2).$$

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

Semantics of LTL$_f$ is given in terms of finite traces, i.e., finite words over the alphabet $2^\Sigma$. We name the index of the element in the trace as instant. Intuitively, $\circ \varphi$ and $\neg \ominus \varphi$ indicate that $\varphi$ holds in the next and previous instant, respectively; $\varphi_1 \land \varphi_2$ states that $\varphi_2$ will eventually hold and, until then, $\varphi_1$ holds too; dually, $\varphi_1 \land \varphi_2$ signifies that $\varphi_2$ holds at some point in time, from that instant on, $\varphi_1$ 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 $\varphi$ is satisfied at a given instant $i$ ($1 \leq i \leq n$) by induction of the following:

$$(t, i) \models True; (t, i) \not\models False;$$

$$(t, i) \models a \iff a \text{ True in } t(i);$$

$$\varphi \models \neg \varphi \iff (t, i) \not\models \varphi;$$

$$(t, i) \models \varphi_1 \land \varphi_2 \iff (t, i) \models \varphi_1 \text{ and } (t, i) \models \varphi_2;$$

$$(t, i) \models \varphi \text{ iff } i < n \text{ and } (t, i+1) \models \varphi;$$

$$\varphi \models \varphi \text{ iff } i > 1 \text{ and } (t, i-1) \models \varphi;$$

$$\varphi_1 \models \varphi_2 \iff \varphi_1 \text{ for all } k \text{ s.t. } i \leq k < j;$$

$$(t, i) \models \varphi_1 \land \varphi_2 \iff \varphi_2 \text{ with } 1 \leq j \leq i, \text{ and } (t, k) \models \varphi_1 \text{ for all } k \text{ s.t. } j < k \leq i.$$

Without loss of generality, we consider here the non-strict semantics of $U$ and $S$ [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 $\langle a, b, c, d, c, e, c, b \rangle$, $\langle b, d, a, b, b, d, e, d, c \rangle,$ and $\langle b, c, a, e, a \rangle,$ respectively. We thereby indicate, e.g., that $(t_1, 1) \models a$, $(t_2, 4) \models b$, and $(t_4, 2) \models c$. Considering again the formula $(\circ \ominus \neg \ominus) U d$, we have that $(t_1, 1) \models$ it, whereas $(t_2, 6) \not\models$ does not.

From the above operators, the following can be derived:

- Classical boolean abbreviations $\vee, \neg$; $\varphi \models True \iff \varphi \models False$.
- $\varphi \models \neg \ominus \varphi$.
- $\varphi \models \circ \ominus \varphi$.
- $\varphi \models \ominus \circ \varphi$.
- $\varphi \models \ominus \varphi$.
- $\varphi \models \circ \varphi$.

The proof follows from Thm. 2.1 as the checking is feasible in $O(n^3 \times \|\varphi\|)$.

Corollary 2.1: Let $L$ be a finite trace of length $n \in \mathbb{N}$. Checking whether $(t, i) \models \varphi$ (with $1 \leq i \leq n$) satisfies an LTL$_f$ formula $\varphi$ is feasible in $O(n^3 \times \|\varphi\|)$.

Proof: It follows from the proof in [16] elaborated for future operators (\circ, \neg \circ, and U). Notice that the use of past modalities $\ominus$, $\varphi$ and $S$ 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]).

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 $\Sigma$, let $\varphi_{\circ}$ and $\varphi_{\ominus}$ be LTL$_f$ formulae over $\Sigma$. A Reactive Constraint (RCon) $\Psi$ is a pair
We can thus label every event in \( t \) 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 \( (\varphi_a) \) 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 LTLf FORMULAE

The interestiness measures for RCons are based on the probabilities of their activator \((\varphi_a)\) and target \((\varphi_t)\) LTLf formulae. Thus, in this section, we propose estimators for the probabilities of traces and logs satisfying LTLf 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 LTLf formula \( \varphi \) as the degree to which \( \varphi \) is satisfied in that trace, which we denote as \( P(\varphi(t)) \).

Throughout this work, we assume the existence of a labeling mechanism \( \Omega \) that, when given an event \( e \) in a trace \( t \) and a formula \( \varphi \), marks the event with 1 if the event satisfies \( \varphi \) 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 LTLf 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., \( t_2 = \langle e_{2,1}, \ldots, e_{2,9} \rangle = \langle b, d, a, b, b, d, e, c, d \rangle \) from Table I, and formula \( \varphi = d \land \Phi \). As only \((t_2, 2)\) and \((t_2, 6)\) satisfy \( \varphi \), \( \Omega(e_{2,1}, \varphi) \) and \( \Omega(e_{2,6}, \varphi) \) return 1 while \( \Omega(e_{2,i}, \varphi) = 0 \) for every \( i \in \{1, \ldots, 9\} \setminus \{2, 6\} \). Therefore, \( x_{\varphi, t_2} = \langle 0, 1, 0, 0, 1, 0, 0, 0, 0 \rangle \).

Moreover, we assume that \( P(\varphi(t)) \) (the probability of \( t \) to satisfy \( \varphi \) ), 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.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:

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

(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

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

(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 $\varphi_1$ formula (P of act.) and of the $\varphi_2$ formula (P of target) for each RCon $\varphi_0 \rightarrow \varphi_{\tau}$ in $S$ 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 $\Psi_1 = \varphi_{a_2} \rightarrow \varphi_{\tau_1} = \emptyset a$. As we have previously discussed in Sec. III, the evaluation of $\varphi_{a_2}$ and $\varphi_{\tau_1}$ on $t_4$ leads to the following sequences of labels, respectively: $(0, 1, 0, 1, 0, 0)$ and $(0, 0, 1, 1, 1, 1)$. Therefore, we conclude that $P(\varphi_{a_2}(t_4)) = \frac{2}{6}$ and $P(\varphi_{\tau_1}(t_4)) = \frac{5}{6}$.

In order to make measures of interest for formulae and specifications we must, in addition, obtain estimators for the intersection of two LTL$_f$ formulae $\varphi_1$ and $\varphi_2$ being satisfied (or violated) by a trace, e.g., $P(\varphi_1(t) \cap \varphi_2(t))$, for the conditional distribution of $\varphi_1$ to be satisfied (or violated) by trace $t$ conditional on $\varphi_2$ 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., $t_4 = \langle b, c, a, c, e, a \rangle$ from Table I and the pair of activation and target of $\Psi_1 = \emptyset a$, i.e., $\varphi_{a_1} = c$ and $\varphi_{\tau_1} = \emptyset a$, respectively. We have that $x_{\varphi_{a_1}, \varphi_{\tau_1}, t_4}$ is $(0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0)$. The resulting joint sequence $x_{\varphi_1, \varphi_2, t_4}$ 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:

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

(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 

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

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 $P(\varphi_{a_1}(t_4) \cap \varphi_{\tau_1}(t_4)) = \frac{1}{6}$ as only one element in $x_{\varphi_{a_1}, \varphi_{\tau_1}, t_4}$ 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 $X_{\varphi_1, t} | X_{\varphi_2, t} = x_2$ is a univariate Bernoulli that depends only on sequence $x_2$ (which results from applying $\Omega$ to $\varphi_2$ 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 

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

(5)

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

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

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

Take, e.g., $t_4 = \langle b, c, a, c, e, a \rangle$ from Table I as above, $\varphi_{a_1} = c$ and $\varphi_{\tau_1} = \emptyset a$, namely the activator and target of $\Psi_1$. We have that $P(\varphi_{\tau_1}(t_4) | \varphi_{a_1}(t_4)) = \frac{2}{6}$ as $\varphi_{a_1}$ is satisfied by two events, only one of which satisfies $\varphi_{\tau_1}$ 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 $\varphi$. Recall that an event log $L = \{t_1, \ldots, t_m\}$ is a bag of traces with trace $t_i$ occurring $j_i$ times in the log. Let us denote with $L = \{t_1, \ldots, t_m\}$ 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 $P(T = t)$. Let $T$ be the support of the probability distribution of $T$, i.e., $T = \{t \mid P(T = t) > 0\}$.

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., $L = T$. 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 $P(\varphi_1(t) | \varphi_2(t))$, 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

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

(7)

which provides a link between log-based evaluation of formulae and trace-based evaluation. Since we assume log completeness we may replace $T$ with $L$ 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

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

(8)

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

$$P(T = t_i) = \frac{j_i}{\sum_{j=1}^{k} j_i},$$

(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 $P(t = t_4) = \frac{12}{45}$. We saw in Sec. IV-A that $P(\varphi_{a_1}(t_4))$ is $\frac{2}{3}$ for $\varphi_{a_1} = c$. Therefore, the term for $i = 4$ in the summation of Eq. (8) is $\frac{12}{45} \cdot \frac{2}{3} \approx 0.09$ for $\varphi_{a_1}$. Extending the sum to all the traces in $L$, we have that the value of $P(\varphi_{a_1}(L))$ 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 $P(\varphi_1(L) \land \varphi_2(L))$ is

$$P(\varphi_1(L) \land \varphi_2(L)) = \sum_{t \in L} P(T = t) P(\varphi_1(t) \land \varphi_2(t)).$$

(10)

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

In Sec. IV-A, e.g., we showed that $P(\varphi_{a_1}(t_3) \land \varphi_{r_1}(t_4))$ is $\frac{1}{6}$ considering the example log, $\varphi_{a_1}$ as above, and $\varphi_{r_1} = \neg a$. Recalling that $P(t = t_4) = \frac{12}{45}$, we have that the term of the summation in Eq. (10) for $t = t_4$ is $\frac{12}{45} \cdot \frac{1}{6} \approx 0.04$. The value of $P(\varphi_{a_1}(L) \land \varphi_{r_1}(L))$ is computed by summing up the elements obtained for every $t \in L$, thus obtaining 0.22.

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

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

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

(11)

with $P(\varphi_1(L) \land \varphi_2(L))$ estimated as in Prop. 4.5, and $P(\varphi_2(L))$ 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 $P(\varphi_2(L)) > 0$ as before to avoid division by zero. For instance, we showed above that for the example log $L$ in Table I the estimations $P(\varphi_{a_1}(L) \land \varphi_{r_1}(L))$ and $P(\varphi_{a_2}(L))$ are 0.22 and 0.27, respectively. By applying the computation above, we have that $P(\varphi_{r_1}(L) \mid \varphi_{a_1}(L)) = 0.22 \approx 0.82$.

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 $S$ as follows: given a trace $t$ of length $n$, an instant $i$ with $1 \leq i \leq n$, and a specification $S = \{\Psi_1, \ldots, \Psi_s\}$, with $s \in \mathbb{N}$ and $\Psi_j = \varphi_{a_j} \circ \rightarrow \varphi_{r_j}$ for every $j$ s.t. $1 \leq j \leq s$, we say that $S$ is **activated by** $t$ **in** $i$, i.e., $(t, i) \models S$, iff there exists a $\Psi_j \in S$ s.t. $(t, i) \models \varphi_{a_j} \land \varphi_{r_j}$; $S$ is **satisfied by** $t$ **in** $i$, $(t, i) \models S$, iff $(t, i) \models S$ and there does not exist any $\Psi_j \in S$ s.t. $(t, i) \not\models \varphi_{a_j} \land \varphi_{r_j}$. $S$ is **violated by** $t$ **in** $i$, $(t, i) \not\models S$, iff there exists a $\Psi_j \in S$ s.t. $(t, i) \not\models \varphi_{a_j} \land \varphi_{r_j}$. $S$ is **unaffected by** $t$ **in** $i$, $(t, i) \not\models S$. 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 $S = \{\Psi_1, \ldots, \Psi_s\}$ as an RCon, $S = S_\alpha \circ \rightarrow S_\tau$, where $S_\alpha$ and $S_\tau$ are LTL$_f$ formulae expressed as follows:

$$S_\alpha = \bigvee_{j=1}^{s} \varphi_{a_j}; \quad S_\tau = \bigwedge_{j=1}^{s} (\neg \varphi_{a_j} \land \neg \varphi_{r_j}).$$

(12)

For example, $S$ from Table I is activated when either $\Psi_1$ or $\Psi_2$ are (i.e., $S_\alpha = c \circ \circ d$) and it is satisfied when all the activated constraints are satisfied, i.e., $S_\tau = (\neg c \circ \circ a) \land (\neg d \circ a)$. Hence, $S$ is violated in $(t_3, 1)$, e.g., because $\Psi_1$ is violated and satisfied in $(t_3, 2)$, because $\Psi_2$ is satisfied and $\Psi_1$ 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 \( \Omega_R \) be an RCon interpreter that takes an event and an RCon \( \Psi = \varphi_a \text{--} \varphi_r \) and returns a label \( \Omega_R(e, \Psi) \in \{0, 1, x\} \) corresponding to \( \Psi \) being violated, unaffected, and satisfied by \( e \), respectively. The labeling of an event given an RCon \( \Psi \) resorts to \( \Omega \) (explained in Sec. IV-A) as follows:

\[
\Omega_R(e, \Psi) = \begin{cases} 0, & \text{if } \Omega(e, \varphi_a) = 1 \text{ and } \Omega(e, \varphi_r) = 0, \\ 1, & \text{if } \Omega(e, \varphi_a) = 1 \text{ and } \Omega(e, \varphi_r) = 1, \\ x, & \text{otherwise}. 
\end{cases}
\]

Notice that \( x \) is a new outcome of the labeling function \( \Omega_R \) that solely applies to RCons but not to LTLf formulae (see the definition of \( \Omega \) in Sec. IV-A). The second column of Table I lists the labels assigned by \( \Omega_R \) to every event in the traces and all RCons in a sequence. For example, the evaluation of \( \Psi_1 = \varphi_a \text{--} \varphi_r \) \( \Rightarrow \sigma \text{--} \delta \) on \( t_4 = \langle e_{4,1}, \ldots, e_{4,6} \rangle \) is such that \( (t_4, 1) \models c \), thus \( \Omega_R(e_{4,1}, \Psi_1) = x \); also, we observe that \( (t_4, 2) \models c \) but \( (t_4, 2) \not\models \delta \); therefore \( \Omega_R(e_{4,2}, \Psi_1) = 0 \); finally, we have that \( (t_4, 4) \models c \) and \( (t_4, 4) \not\models \delta \), so \( \Omega_R(e_{4,4}, \Psi_1) = 1 \). Following this approach, we attain the following sequence of labels from \( t_4 \) via \( \Omega_R \) on \( \Psi_1 \): \( \langle x, 0, 0, 1, 1, x, x \rangle \).

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 \( P(\Psi(t)) = P(\varphi_r(t) \mid \varphi_a(t)) \) be the probability of an RCon \( \Psi \) to be satisfied by \( t \), and let \( P(\neg \Psi(t)) = P(\neg \varphi_r(t) \mid \varphi_a(t)) = 1 - P(\varphi_r(t) \mid \varphi_a(t)) \) be the complementary probability of \( P(\Psi(t)) \) being violated.

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

\[
P(\Psi(t)) = P(\varphi_r(t) \mid \varphi_a(t)) = \frac{p_{11}}{p_{01} + p_{11}},
\]

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

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

At this stage, we turn to estimate the probabilities of interest for a specification \( S \) (i.e., a set of RCons). Specifically, once a specification is interpreted similarly to a single RCon, once we obtained the interpretation for \( S_a \) and \( S_e \), we apply the labeling mechanism \( \Omega_R \) to get one of the three possible outcomes, which allows for the re-use of Prop. 5.1 to obtain estimators for \( P(S(t)) \) and \( P(\neg S(t)) \). 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 \( P(S(t)) \) is given by

\[
P(S(t)) = P(S(t) \mid S_a(t)) = \frac{p_{11}}{p_{01} + p_{11}},
\]

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

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

<table>
<thead>
<tr>
<th>Measure</th>
<th>Trace</th>
<th>Log</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support</td>
<td>( P(S_a \land S_e \mid t) )</td>
<td>( P(S_a \land S_e \mid L) )</td>
</tr>
<tr>
<td>Confidence</td>
<td>( P(S_a \mid S_e \mid t) )</td>
<td>( P(S_a \mid S_e \mid L) )</td>
</tr>
<tr>
<td>Recall</td>
<td>( P(S_a \mid S_e \mid t) )</td>
<td>( P(S_a \mid S_e \mid L) )</td>
</tr>
<tr>
<td>Specificity</td>
<td>( P(\neg S_a \mid \neg S_e \mid t) )</td>
<td>( P(\neg S_a \mid \neg S_e \mid L) )</td>
</tr>
<tr>
<td>Lift</td>
<td>( P(S_a \mid S_e \mid t) )</td>
<td>( P(S_a \mid S_e \mid L) )</td>
</tr>
</tbody>
</table>

**Log Estimators:** To derive log estimators we again assume that \( L \) is complete, i.e., \( L = T \) with \( T \) 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 \( P(S(L)) \) denote the probability of a log \( L \) to satisfy a specification \( S \). The MLE of \( P(S(L)) \) is given by \( P(S(L)) = \sum_{t \in L} P(T = t)P(S(t)) \), and the MLE of \( P(\neg S(L)) \) is given by \( 1 - P(S(L)) \).

The proof follows the same lines as that of Prop. 4.4.

Returning to the running example in Table I, \( P(S(L)) \) is \( 45 \) \( 45 \), which together with the probabilities of \( S \) yield a Support of \( 0.7 \) and Confidence of \( 0.79 \).
VI. EVALUATION

We implemented our technique in a proof-of-concept Java tool. 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\textsubscript{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. 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 variability: http://dx.doi.org/10.4121/uuid:915d2bfb-7e84-49ad-a286-dc35f063a460.

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 \( S_5 \), which suggests that the activator was satisfied in every event of the log. Also, Lift shows that in \( S_2 \) and \( S_1 \) the satisfaction of both activator and target is higher than their individual satisfaction degree.

VII. RELATED WORK

Different contributions in the literature aim at quantitative extensions of LTL/LTL\textsubscript{f} enriching the languages with quantitative operators. LTL\textsubscript{f}[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\textsubscript{f} with new operators, as we quantify the satisfaction of formulae based on standard LTL\textsubscript{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

---


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


