Fifty Shades of Green: How Informative is a Compliant Process Trace?

Burattin, Andrea; Guizzardi, Giancarlo; Maggi, Fabrizio M. ; Montali, Marco

Published in:
International Conference on Advanced Information Systems Engineering

Link to article, DOI:
10.1007/978-3-030-21290-2_38

Publication date:
2019

Document Version
Peer reviewed version

Fifty Shades of Green: How Informative is a Compliant Process Trace?

Andrea Burattin[1], Giancarlo Guizzardi[3], Fabrizio M. Maggi[2], and Marco Montali[3]

[1] Technical University of Denmark, Kgs. Lyngby, Denmark
[2] University of Tartu, Tartu, Estonia
[3] Free-University of Bozen-Bolzano, Bolzano, Italy

Abstract. The problem of understanding whether a process trace satisfies a prescriptive model is a fundamental conceptual modeling problem in the context of process-based information systems. In business process management, and in process mining in particular, this amounts to check whether an event log conforms to a prescriptive process model, i.e., whether the actual traces present in the log are allowed by all behaviors implicitly expressed by the model. The research community has developed a plethora of very sophisticated conformance checking techniques that are particularly effective in the detection of non-conforming traces, and in elaborating on where and how they deviate from the prescribed behaviors. However, they do not provide any insight to distinguish between conforming traces, and understand their differences. In this paper, we delve into this rather unexplored area, and present a new process mining quality measure, called *informativeness*, which can be used to compare conforming traces to understand which are more relevant (or informative) than others. We introduce a technique to compute such measure in a very general way, as it can be applied on process models expressed in any language (e.g., Petri nets, Declare, process trees, BPMN) as long as a conformance checking tool is available. We then show the versatility of our approach, showing how it can be meaningfully applied when the activities contained in the process are associated to costs/rewards, or linked to strategic goals.

Keywords: Conformance Checking · Business Value · Process Mining · Goals.

1 Introduction

The increasing availability of event data recorded by information systems, electronic devices, web services, and sensor networks provides detailed information about the actual processes in a wide range of systems and organizations. Process mining techniques [1] can use such event data to discover and enhance processes, check the conformance of actual with expected behaviours, and ultimately provide insights on how processes are executed in reality.

The typical starting point for process mining algorithms is an event log. Each event in a log refers to an activity (i.e., a well-defined step in a process) and is related to a particular case, in turn identifying a process instance. The events

belonging to a case are ordered and can be seen as one "run" of the process (often referred to as a trace of events). Event logs may store additional information about events such as the resource (i.e., person or device) responsible for the execution or triggering of the activity, the timestamp of the event, or additional data attributes recorded with the event. Typically, three types of process mining techniques can be distinguished [1]: *(i)* process discovery (learning a model from example traces in an event log), *(ii)* conformance checking (comparing the observed behavior in the event log with the expected behaviors expressed by a process model), and *(iii)* model enhancement (extending models based on additional information in the event logs, e.g., to highlight bottlenecks).

In particular, conformance checking has lately attracted a lot of attention both from the research community and the industry, being instrumental to understand the presence, extent, and nature of *deviations* separating the actual and expected courses of execution of the process [4]. In an organizational context, such deviations are key towards a better governance, risk management, and compliance, since they may reveal legal/normative issues, or opportunities to further improve and optimize processes. In this respect, virtually all approaches in the conformance checking spectrum focus on the detection of behaviors that do not comply with a prescriptive process model, and in turn on the fine-grained analysis of the resulting deviations. Interestingly, *no fine-grained insight is instead provided in the case of compliant behaviors.*

In this paper, we delve into this unexplored area, arguing that analyzing, classifying, and better understanding compliant behaviors is as important as in the case of non-compliant ones. To do so, we appeal to the fact that business processes intimately relate to the value chain of an organization, and hence activities in a process are executed because they ultimately *contribute to value.* This, in turn, provides a conceptual basis to measure compliant traces, which makes some traces more *informative* about the process than others.

We propose a novel notion of *informativeness* of a trace. Informativeness should not be confused with standard measures used to classify traces based on the value, or reward, they produce. Instead, it directly relates to conformance, in the following sense: a trace is informative if it conforms to the process model of interest, and, in addition, it contains behavior that: *(i)* is not necessary to achieve conformance, and *(ii)* such behavior impacts (positively or negatively) to the resulting value. These two aspects single out behaviors that could have been skipped according to the process, but that proved meaningful when ascribing value to the course of execution. Let us substantiate this with a very simple example. The payment phase of a shop selling process consists of a sequence of tasks where a clerk performs the payment using the data of the customer and, if the payment is accepted, puts the paid items in a bag, inserts the receipt in the bag, and optionally also includes a discount badge for future purchases. Consider two compliant executions, performed by John and Jane. John strictly follows the sequence of mandatory tasks. Jane instead decides to also put a discount badge in the customer bag. If we ascribe value to this process depending on the customer satisfaction, both executions positively contribute to it, as the customer gets the

items they wanted. However, the trace produced by Jane is more informative, as she exhibited a behavior that is not strictly required to conform to the process prescriptions, but that positively contributes to the customer satisfaction. Notice that the trace produced by Jane would have remained informative even if she performed some optional behavior negatively impacting on the customer satisfaction, such as inserting advertisement sheets in the customer bag.

Starting from this intuition, we formalize informativeness into a parameterized, model-agnostic *metric* that can be effectively used to measure informativeness, and classify traces on this basis. The metric is model-agnostic in the sense that it can be applied on process models expressed in any language (e.g., Petri nets, Declare, process trees, BPMN) as long as a conformance checking tool is available. A proof-of-concept implementation to compute this metric is available as a plug-in for the well-established ProM process mining framework.

To show the effectiveness and versatility of our approach, we ground it within two scenarios. In the first scenario, prescriptive processes emerge from the sophisticated requirement models introduced in [7]. Such models contain hard and soft goals, which can be decomposed into more specific goals and ultimately tasks, possibly marked as optional. In this setting, informativeness provides a tool to compare traces based on the presence of optional behaviors that achieve or prevent the achievement of soft goals. In the second scenario, we consider standard BPMN processes whose tasks are associated to costs/rewards.

The rest of the paper is structured as follow: Section 2 introduces the notion of informativeness, framing it in the context of conceptual modeling. It then provides the two scenarios used throughout the paper. Section 3 shows how the informativeness of a process trace can be actually computed, demonstrating that it reconstructs the intuitive understanding of informativeness as discussed in Section 2. Section 4 tackles related work and Section 5 concludes the paper and spells out directions for future work.

2 On the Notion of Informativeness

In this section, we introduce the notion of informativeness from the point of view of conceptual modeling of processes and we show, via realistic yet to-the-point examples, how such problems can impact typical scenarios.

2.1 Intended, Allowed, Compliant and Informative Traces

Modeling is the activity of representing the physical and the social world for the purposes of communication, understanding, problem solving, controlling and automation [10]. As discussed in depth in [5], according to this view, *models* are representations of *conceptualizations* of reality. Conceptualizations are the result of cognitive operations created by abstracting (filtering out) certain features of states of affairs of reality. These conceptualizations, in a sense, delimit the set of *abstractions* that can be carved out of reality, i.e., the abstractions that are deemed acceptable according to that conceptualization. So, for example,

if we have a proper conceptualization of genealogy, the concepts of ancestor, descendent, father, mother, offspring and their ties allow us to build individual abstractions representing particular states of affairs (e.g., John is the father of Paul and Mary is an ancestor of John), which are constructed by abstracting from a number of features of reality according to these concepts (e.g., from the weight and hair color of John, Paul and Mary). Moreover, this conceptualization proscribes a number of abstractions that represent states of affairs unacceptable according to that conceptualization (e.g., John being his own ancestor, John having two biological fathers, John's biological father changing with time).

If a model is a representation of a conceptualization, an adequate model of a conceptualization is one that accepts as valid instances (or *allowed model instances*) exactly those that represent the state of affairs deemed acceptable by that conceptualization [5]. We call these instances *intended model instances*. Models, however, are frequently *under-constrained*, thus, accepting as allowed, instances that are non-intended. Frequently, they are also *over-constrained*, thus, excluding as valid (i.e., as allowed) intended instances. In the case of process modeling, model instances are execution traces, i.e., possible executions of a process model.

In Figure 1, we contrast the set of intended instances (traces) of a process model with the set of allowed traces delimited by a given process specification. As we can observe, there are traces that are intended according to the underlying conceptualization of reality, but are excluded by the process model (the difference between intended and allowed traces). Moreover, we have traces that are allowed by the model, but do not correspond to intended traces (the difference between allowed and intended traces). Furthermore, as shown in Fig. 1, a subset of the intended traces (i.e., executions that are actually enacted in reality) can be recorded in event logs. The difference between the set of intended traces present in a log and the set of allowed ones delimits the set of *non-compliant traces*. In other words, a non-compliant trace is one that is intended by a given conceptualization (as evidenced by the event log), but is not prescribed by the process model that aims at capturing that conceptualization. In contrast, the intersection between the set of allowed traces and those recorded in the log is called the set of compliant traces.

Many authors in the literature of process mining have proposed metrics for characterizing and ranking non-compliant traces (according to different degrees of non-conformance). But what about the set of compliant traces? Traditionally, all compliant traces are thought of as standing in the same footing and, as such, are thought as having equal importance for the process analyst. As previously mentioned, in this paper, we defend the notion of *informativeness* of a trace, i.e., how informative that trace is for the process analysis. As we elaborate in the sequel, how informative a trace is depends on the relation between the activities constituting a process and the value-based elements outside the process, namely, the goals (and anti-goals) of suitable stakeholders. Specifically, the important aspect to consider is the value that is brought by the execution of activities that

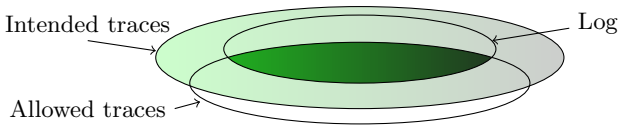

Fig. 1. Representation of the traces involved in the scenario presented in this paper. The color indicates the informativeness (light green: not informative; dark green: very informative) of intended traces. We aim at "giving colors" (i.e., assign informativeness) to compliant traces (i.e., traces in the log also allowed by the process model).

are not strictly required to make the trace compliant: these are the ones that make the execution interesting if they impact on value.

The main ingredients of process models are actions (intentional events). As discussed in [6], actions are manifestations of intentions inhering in the stakeholders. Goals (as expressed in frameworks such as i* [6, 7, 14]) are the propositional content of the stakeholders's intentions. In other words, every action is caused by a certain intention of a stakeholder, whose propositional content is formulated as a goal. Goals can be either mandatory or optional for specific stakeholders. Mandatory top goals (i.e., mandatory goals that are not components of other goals) are termed *hard goals*. Optional top goals are called *soft goals*.[4] Optional goals in general (i.e., "nice-to-have" goals) are also called *preferences*. Moreover, goals relate to each other via decomposition (or refinement) relations. These can be either OR-decompositions (representing that the satisfaction of any sub-goal is sufficient for satisfying the composed goal) or AND-decompositions (meaning the satisfaction of the composed goal requires the joint satisfaction of all mandatory sub-goals). Moreover, goals (both optional and mandatory) can relate to each other via contribution relations. These contributions can be either positive (e.g., the satisfaction of a goal A implies the satisfaction of another goal B - called a make contribution) or negative (e.g., the satisfaction of a goal B is incompatible with the satisfaction of a goal C - termed a break contribution). Analogously, the execution of actions can, while addressing a goal, exert positive or negative contributions to other goals.

Process models are designed to explicitly coordinate the execution of action types that together afford the satisfaction of the stakeholder's goals. So, all intended and compliant traces of a process model should *satisfy all the mandatory goals* that motivated the creation of that process. However, compliant traces can differ wrt. their informativeness, i.e., *the degree to which they positively or negatively contribute to the satisfaction of optional goals*. The stronger is the contribution that a trace makes wrt. these goals, the more value (positive or negative)[5] it adds to the stakeholder(s) at hand. In other words, the informa-

[4] We are fully aware of the different senses in which the term soft goal is employed in the literature [6], namely, alternatively as a synonym to non-functional requirement, fuzzy propositional content, or propositional contents without a generally accepted satisfaction criteria. Here, as in [7], we use the term in the specific sense just defined.

[5] As discussed in [12], even though people intuitively assume a positive connotation of the term value, value emerges from events that impact goals either positively

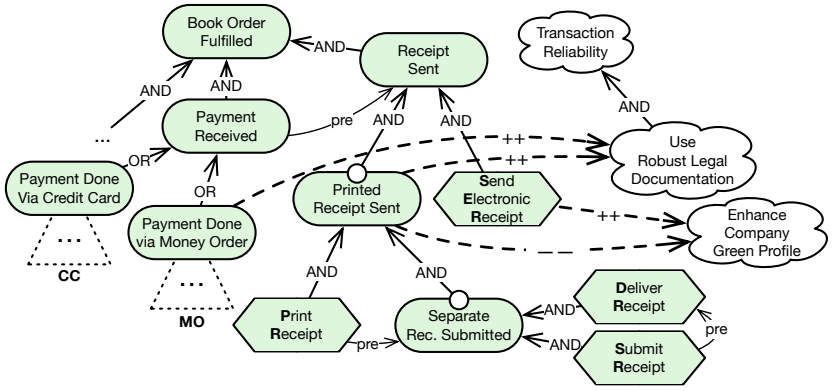

Fig. 2. Fragment of the wholesale book seller requirement model from [7].

tiveness of a trace is a measure of how much value (either negative or positive) a trace brings to the stakeholder(s) whose goals motivate the design of that process.

2.2 Running Examples

To better introduce the problem tackled in this paper, let us analyze some realistic examples. We start by introducing a language where the goals are explicitly reported in the formalism and then we move to another scenario where goals need to be manually elicited.

Example 1. We consider the setting of [7], where business processes implicitly emerge from the representation of requirements. In particular, [7] puts forward a sophisticated goal-oriented modeling framework. It uses, as a basis, the classical goal-oriented approach described before, where *(i)* both hard and soft goals are considered; *(ii)* goals are subject to AND/OR decomposition into sub-goals and ultimately tasks; *(iii)* hard goals achieve or prevent the achievement of soft goals. In addition, [7] adopts: *(i) optional goals*, to express goals that should not necessarily be achieved towards the achievement of their parent goals; *(ii) preference goals*, to capture the relative importance of certain soft goals; *(iii) temporal constraints* on the executability of tasks, in a way that is reminiscent of declarative, constraint-based processes. For self-containedness, we depict in Figure 2 an excerpt of the case study from [7], which tackles a wholesale book seller. Goals are represented as rounded rectangles, tasks as hexagons. Hard goals are colored, whereas soft goals are white. Optional goals are marked with a circle on top. Cloud-shaped soft goals denote preference goals. Beside AND/OR contribution arcs, we have dashed arcs labeled with "++" (resp., "−−") to indicate that when the source goal is achieved (or the source task is executed) then the target goal

or negatively. More specifically, the value ascribed to an event consists of *benefits* (positive value contributions) or *sacrifices* (negative value contributions).

is also achieved (resp., cannot be achieved); thin arcs labeled with "pre" indicate that the target element cannot be achieved/executed until the source one is achieved/executed. The **CC** and **MO** goals do not contain optional parts.

The presence of *optional goals that are sub-goals of hard goals* in the model, which in turn underlies the presence of optional tasks, is the basis to understand how informative different traces are. Informativeness depends on how behavior that is exhibited, but is not essential towards conformance, actually impacts (either negatively or positively) on the overall, resulting value.

Consider, for example, the soft goal *Transaction Reliability* as the main focus to understand the overall, produced value. Consider now the following four conforming scenarios:

Scenario Book-CC. The book order is paid via credit card, and then the corresponding receipt is sent by executing the Send Electronic Receipt task.

Scenario Book-CC-PR. Similar to the previous scenario, with the addition that also a printed receipt is sent, by executing the Print Receipt task.

Scenario Book-CC-2PR. Extension of the previous scenario, ensuring that a separate printed receipt is submitted. This is done by executing, after Print Receipt, the two tasks that Submit and Deliver the Receipt.

Scenario Book-MO-PR. An alternative scenario where payment is done using a money order, and then the corresponding receipt is sent both electronically and physically, by respectively performing the Print Receipt and Send Electronic Receipt tasks.

Scenario **Book-CC** is not particularly informative (when compared to other compliant traces), since it conforms to the requirements but does not include any non-mandatory behavior with the purpose of satisfying the *Transaction Reliability* soft goal. Scenario **Book-CC-PR** is more informative: the execution of Print Receipt could be removed without jeopardizing conformance, but its presence ensures the reliability of the overall transaction. Scenario **Book-CC-2PR** is as informative as scenario **Book-CC-PR**: the presence of the additional tasks for submitting and delivering the receipt is not essential to guarantee conformance, but at the same time does not interact with reliability of the transaction, which is already implicitly achieved by the fact that the receipt has been previously printed. Finally, scenario **Book-MO-PR** is not informative. In fact, it consists of an initial behavior that achieves the hard goal *Payment Done via Money Order*, and has also the effect of guaranteeing reliability of the overall transaction. This initial portion of the entire scenario is essential towards conformance, since its removal would jeopardize the achievement of the *Payment Received* goal, in turn preventing the fulfillment of the book order. The fact that this essential part satisfies the reliability of the transaction makes the execution of the Print Receipt task irrelevant, and hence not informative, as opposed to scenarios **Book-CC-PR** and **Book-CC-2PR**.

Interestingly, scenario **Book-MO-PR** would actually become informative if the scope of the analysis covers all soft goals, not just *Transaction Reliability*. In such a case, in fact, the optional execution of Print Receipt prevents the company to enhance its green profile, which would not happen if such a task

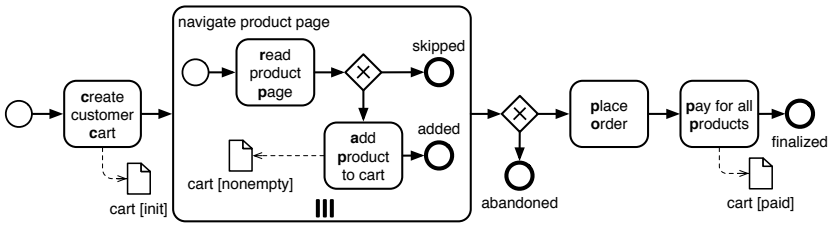

Fig. 3. Fragment of an e-commerce, order-to-delivery process.

was not present in the scenario. This is an example of informativeness arising from a negative impact to the overall produced value. ◁

For the next example, we consider a BPMN process model. In this case, the goals (i.e., the values) are not explicit so different interpretations are possible.

Example 2. Consider the fragment of an online order-to-delivery process shown in Figure 3. Each process instance relates to a *cart* case object, which is manipulated by the customer. When the process starts, the cart is *initialized*. Then, the customer may navigate through multiple product pages by browsing the e-commerce website. For each navigated page, after accessing the page itself, the customer may possibly decide to add the accessed product to her own cart (in this case, the cart enters into the *nonempty* state). When all pages of interest have been navigated, the customer may decide to quit, or to finalize the order.[6] This, in turn, amounts to placing the order and then paying for all the products in the cart, which has the effect of moving the cart into the *paid* state.

We consider now 4 scenarios, accounting for different instances of this process:

Cart-Empty. The customer visits the pages of some products, but does not add any product to the cart, and consequently abandons the process.

Cart-Abandon. The customer visits the pages of some products, and adds a product to her cart costing 20 euros. Then, the customer decides not to finalize the cart, and so simply abandons.

Cart-Pay. As the previous scenario, but now the customer finalizes the order, going through the placement task, and paying 20 euros.

Cart-PayMore. An extended version of the previous scenario, where the customer adds two products to the cart, one costing 20 euros and another costing 30 euros, finally paying a total amount of 50 euros.

All scenarios contain optional behaviors: the only mandatory task is Create Customer Cart, while all the others denote optional, possibly informative behavior.

We now analyze informativeness by considering, as value, the impact of each scenario on the evolution of the carts, considering that potentially informative traces should make the cart *nonempty*, and then possibly also *paid*. With this objective in mind, scenario **Cart-Empty** is not informative, since the cart simply stays in the *init* state. Scenario **Cart-Abandon** is instead informative: the customer adds an item to the cart making it *nonempty*, through the execution of

[6] For simplicity, we model this as a deferred choice, but in reality we should also ensure that the customer cannot proceed to the order finalization if the cart is empty.

the optional Add Product to Cart task. Scenario **Cart-Pay** is more informative than **Cart-Abandon**: the cart is further moved to the *paid* state through the execution of the optional process fragment consisting of the sequence of Place Order and Pay for All the Products tasks. Scenario **Cart-PayMore** is as informative as **Cart-Pay**: more items are added to the cart, but in terms of state evolution of the cart, this has the same overall effect than in **Cart-Pay**. In particular, the second execution of Add Product to Cart is not informative, as it was preceded by another instance of the same task that made the cart *nonempty*.

An alternative assignment of values (and hence informativeness) is by considering the potential/concrete monetary gain associated to the tasks involved in the process. In particular, assume that the Add Product to Cart task gets associated to the potential gain of the added products, and that the Pay for All Products task gets associated to the overall, paid amount. The other tasks do not come with any gain instead. With this objective in mind, scenario **Cart-Empty** is not informative, as it does not contain any optional behavior involving any gain. Scenario **Cart-Abandon** is instead informative: the customer decides to add a product to the cart for a potential gain of 20 euros. Scenario **Cart-Pay** is more informative than **Cart-Abandon**, because of the presence of a second optional task, in particular Pay for All Products, also coming with a concrete gain of 20 euros. Scenario **Cart-PayMore** is, in this interpretation of value, even more informative as **Cart-Pay**: it contains the execution of three optional tasks that come with positive gains: the two executions of Add Product to Cart, and the execution of Pay for All Products, incurring respectively in a gain of 20 (potential), 30 (potential), and 50 (concrete) euros.

Notice that various ways to aggregate such task-related costs into a single value may be singled out. Depending on such aggregation, different values for informativeness may be obtained, giving more weight to traces that end up with an actual gain (possibly for orders containing few items), or to traces with a cart containing very costly products that in the end are however not paid. ◁

The examples provided in this section can help to better understand the general concept and the aim of informativeness and, at the same time, they can serve as a guide to the actual realization of the metric.

3 Informativeness Metric

In this section, we present a possible realization of the informativeness metric. We formally define the metric and then investigate how it can be instantiated to the two examples previously described. Additionally, we report about a proof-of-concept implementation for the computation of the metric.

3.1 Realization and Formal Definition

Let us first define the basic elements that are needed to compute how informative a process trace is. Given a set of unique event identifiers U (e.g., $U \subseteq \mathbb{N}$), a set of

attribute names A_n and a set of attribute values[7] A_v, an *event* $e = (id, f)$ is a tuple made of a unique event identifier $id \in U$ and a key-value relation $f : A_n \to A_v$ mapping attribute names to the corresponding values. Common attributes contained in events are *name*, *timestamp*, and *originator*. With a projection operator π on events it is possible to extract specific values of specific attributes from the attribute key-value relation, i.e., $\pi_a((id, f)) = f(a)$. For example, consider event $e = (42, \{name = \text{'purchase'}, timestamp = \text{2019-06-03}, cost = 100\})$, then $\pi_{name}(e) = purchase$ and $\pi_{cost}(e) = 100$. If the key-value relation does not contain any mapping for the given attribute name, then the default value 1 is returned, i.e., $\pi_\perp(e) = 1$. Additionally, the event identifier can be extracted with $\pi_{id}(e) = 42$. For readability purposes, in the rest of the paper, we assume that no attribute in f is named *id*, thus π_{id} is always unambiguous.

The set of all events is denoted with E, the set of all possible sequences of events is called $T = E^*$, and a sequence of events $t \in T$ is called *trace*. Single events of $t = \langle e_1, e_2, \ldots, e_n \rangle$ are accessed by the corresponding index, e.g., $t(1) = e_1, t(2) = e_2$. The length of a trace is denoted as $|t| = n$. Given a trace $t = \langle e_1, e_2, \ldots, e_n \rangle$, the deletion of some events from it (without perturbing the order of the remaining ones) generates a so-called *sub-sequence*[8] of the original trace. Note that a sub-sequence is also a trace. The set of all possible sub-sequences of t, which are not empty and not equal to t itself, is denoted with $S(t)$ and contains $2^{|t|} - 2$ traces.[9]

Given a trace t and a sub-sequence $s \in S(t)$, we define the diff operator, which returns the sequence of events to be removed from t to generate s:

$$\text{diff}(t, s) = \langle e_i \mid e_i \in t \wedge e_i \notin s \rangle.$$

The actual implementation of the diff operator can use the unique event identifier (i.e., $\pi_{id}(e)$) to establish if an event belongs to the original trace or not.

In order to define the informativeness of a sub-sequence $s \in S(t)$, it is necessary to know which attributes bring value and how to aggregate them. Therefore, given an attribute name a, a value adapter m, and an aggregation operator $\odot$, we define:

$$\text{informativeness}(s, t, a, m, \odot) = m \bigodot_{e \in \text{diff}(t,s)} \pi_a(e). \tag{1}$$

The simplest implementation of informativeness just counts the number of removed events and is realized with $a = \perp$ (recall $\pi_\perp(e)$ always returns 1), $m = 1$ and $\odot = \Sigma$. In this case, we obtain the definition $\sum_{e \in \text{diff}(t,s)} 1 = |\text{diff}(t, s)|$. Another example of informativeness function is the inverse of the costs of the

[7] In this context, *value* denotes a symbolic/numeric constant.

[8] Sub-sequences should not be confused with sub-strings.

[9] Consider the set I of indexes of events in trace t: $I = \{1, 2, \ldots, |t|\}$. By taking only events from a subset of I, we can generate a possible sub-sequence of t. Therefore, the set of all possible sub-sets of I, also called power-set $\mathcal{P}(I)$, contains the indexes of all possible sub-sequences and $|\mathcal{P}(I)| = 2^{|I|} = 2^{|t|}$. From this value, we need to remove two special sub-sequences: the empty and the original ones. Therefore, we end up with $2^{|t|} - 2$ possible sub-sequences.

Algorithm 1: Trace informativeness

Input: $t \in \mathcal{T}$: a trace;
$M \in \mathcal{M}$: the reference model;
$C : \mathcal{M} \times \mathcal{T} \to \{true, false\}$: a conformance function which, given a model and a trace, returns '*true*' if the trace is compliant, '*false*' otherwise;
$a, m, \odot$: configuration of the informativeness function as in Eq. 1.
Output: The informativeness of the trace and the activities bringing it

```
▷ Check that the current trace is compliant with the model
1  if C(M, t) = false then
2  |  return error ▷ The initial trace must be compliant
3  end
   ▷ Initialize structures to keep maximal informativeness
4  smax ← 0
5  tmax ← t
   ▷ Iterate over all possible sub-sequences of t
6  foreach t' ∈ S(t) do
7  |  if C(M, t') = true then ▷ Continue if the sub-sequence is compliant
8  |  |  s ← informativeness(t', t, a, m, ⊙)
9  |  |  if s > smax then
10 |  |  |  smax ← s
11 |  |  |  tmax ← t'
12 |  |  end
13 |  end
14 end
15 return (smax, diff(t, tmax)) ▷ The difference between t and tmax contains
     the most informative activities. If no sub-sequence is compliant,
     then the informativeness is 0
```

tasks present in the original trace but not in the sub-sequence, i.e., $a = cost$, $m = -1$, $\odot = \Sigma$, thus obtaining $-\sum_{e \in \text{diff}_{(t,s)}} \pi_{cost}(e)$.

With these definitions in place, it is possible to obtain the informativeness of a trace t wrt. a model M as the maximal informativeness among all sub-sequences in $S(t)$ that are compliant with M. Alg. 1 reports a possible way of computing the metric. The algorithm takes as input the trace t, the model M and a conformance function C. Additional inputs are a, m and $\odot$, as previously mentioned in Eq. 1, to configure the informativeness function. The first check performed by the algorithm is to verify that the trace is indeed compliant with the model (lines 1-3). After that, all possible sub-sequences are iterated (line 6) and, for the compliant ones (line 7), the corresponding informativeness is computed (line 8). The best informative score is kept with the sub-sequence producing it (lines 9-12). Finally, the best informativeness, together with the tasks involved in its computation are returned to the user (line 15). If the trace does not contain any optional behavior, then no sub-sequence is compliant and therefore the maximal informativeness is never re-assigned, with 0 being returned.

The computational complexity of the algorithm is controlled by two main factors: the number of sub-sequences and the complexity of the conformance checking technique (the informativeness function has linear complexity on the number of events of the sub-sequence). Such complexity could represent a limitation for using this formalization in complex scenarios. However, in this paper, we prefer to focus on conceptual aspects, consciously leaving out all possible optimizations, to not sacrifice understandability.

3.2 Example of Measure Calculation

It is interesting to observe how the informativeness measure can be calculated on the previous examples (cf. Sec. 2.2). For Example 1, let us denote by **CC** and **MO** the traces that respectively achieve the two goals *Payment Done via Credit Card* and *Payment Done via Money Order* (recall that they contain all mandatory tasks). We use the following notation to represent task executions: $T(i_r, i_g)$, where T is a task (compactly denoted using the bold initials of its description, as shown in Figure 2), while i_r and i_g are two boolean (0/1) attributes respectively indicating whether the task impacts (in a positive or a negative way) on the achievement of soft goals *Transaction Reliability* and *Enhance Company Green Profile*. We can then compactly indicate, with a slight abuse of notation, $\mathbf{CC}(0,0)$ and $\mathbf{MO}(1,0)$. These two boolean attributes, if not natively present in the trace, can be computed by pre-processing the trace, e.g., by relying on the planning technique from [7]. To compute informativeness, we also need a conformance function that accepts the traces and the model, and that judges the trace compliant if and only if it achieves the top, hard goal *Book Order Fulfilled*. This, again, can be directly computed using the planning technique from [7].

We consider $a = i_r$, $m = 1$, and the aggregation operator $\odot = \Sigma$. Then the scenarios correspond to the following informative traces:

Scenario Book-CC: trace $\langle \mathbf{CC}(0,0), \mathsf{SER}(0,1) \rangle$ with informative value 0, since no sub-sequence is compliant.

Scenario Book-CC-PR: trace $\langle \mathbf{CC}(0,0), \mathsf{SER}(0,1), \mathsf{PR}(1,1) \rangle$ with informative value 1, since PR can be removed without threatening conformance, and it brings a value of 1 for i_r (since it achieves the corresponding soft goal).

Scenario Book-CC-2PR: trace $\langle \mathbf{CC}(0,0), \mathsf{SER}(0,1), \mathsf{PR}(1,0), \mathsf{SR}(0,0), \mathsf{DR}(0,0) \rangle$ with informative value 1, since the sub-sequence $\langle \mathsf{PR}(1,0), \mathsf{SR}(0,0), \mathsf{DR}(0,0) \rangle$ can be safely omitted, and only PR brings value (as in the previous scenario).

Scenario Book-MO-PR: trace $\langle \mathbf{MO}(1,0), \mathsf{SER}(0,1), \mathsf{PR}(0,1) \rangle$ with informative value 0. Notice that, in this case, PR has $i_r = 0$, since the corresponding soft goal has been already achieved via **MO**, which constitutes an essential, non-omittable behavior. Informativeness would become 2 if we consider $a = i_g$ instead, since both tasks SER and PR actually interact with the corresponding soft goal (the first by temporarily achieving it, the second by reverting the achievement).

For Example 2, let us adopt the following notation for representing task executions: $T(g_p, g_c)$, where T is a task (compactly represented using the bold

initials of the task descriptions, as shown in Figure 3), g_p is the potential gain associated with the execution of the activity, and g_c is the concrete gain of the task. If the activity does not bring any gain, the values are omitted. For example, scenario **Cart-Abandon** is described as $\langle \mathsf{CC}, \mathsf{RP}, \mathsf{RP}, \mathsf{AP}(20,0), \mathsf{RP} \rangle$.

As reported in the example, we can think on several value definitions. We consider here valuable executions with expensive items in the cart, or with paid items. To achieve that, we assume $m = 1$, $\odot = \Sigma$, and $a = g_p$ or $a = g_c$, respectively. Then we obtain:

Cart-Empty: $\langle \mathsf{CC}, \mathsf{RP}, \mathsf{RP}, \mathsf{RP} \rangle$ with informative value 0 in both cases (despite there are activities which are not needed).

Cart-Abandon: $\langle \mathsf{CC}, \mathsf{RP}, \mathsf{RP}, \mathsf{AP}(20,0), \mathsf{RP} \rangle$ with informative values 20 and 0 (an item is added to the cart, but no payment is done).

Cart-Pay: $\langle \mathsf{CC}, \mathsf{RP}, \mathsf{RP}, \mathsf{AP}(20,0), \mathsf{RP}, \mathsf{PO}, \mathsf{PP}(0,20) \rangle$ with informative values 20 and 20 (the added item is then paid).

Cart-PayMore: $\langle \mathsf{CC}, \mathsf{RP}, \mathsf{RP}, \mathsf{AP}(20,0), \mathsf{RP}, \mathsf{AP}(30,0), \mathsf{PO}, \mathsf{PP}(0,50) \rangle$ with informative values 50 and 50 (using the same line of reasoning of **Cart-Pay**).

We might derive new attributes combining values from others, e.g., $g_{tot} = g_p + 2g_c$, where we assign some value for having products in the cart but, when these are paid, the value significantly increases. Clearly, domain knowledge as well as specific goals are needed to define what the actual attribute is meant to capture.

A similar line of reasoning could be carried out when informativeness has to be calculated considering the impact of tasks to the cart states, assuming that the information about the state transitions of the cart is readily available in the trace (or that the trace has been pre-processed accordingly).

3.3 Implementation

A proof-of-concept implementation of the technique is available as a ProM plug-in (see `https://github.com/delas/informativeness`). The plug-in considers models represented as process trees and relies on existing conformance checking plug-ins. Nevertheless, as mentioned before, extending the technique to cope with other modeling languages is merely a trivial implementation exercise (as conformance checking techniques are already available in ProM). Our implementation uses, as parameters, $a = \bot$, $m = 1$ and $\odot = \Sigma$, grounding the informativeness function as in Eq. 1 to $\sum_{e \in \text{diff}(t,s)} 1 = |\text{diff}(t,s)|$.

4 Related Work

In [7], Liaskos et al. propose an approach for representing and reasoning about goal prioritization in Goal-Oriented Requirements Engineering (GORE). In particular, they employ a version of i* models with optional goals, preferences and preference priority markers (weights). In addition, they use an existing preference-based planner over that extended i* model to search for alternative plans that best satisfy a given set of mandatory and preferred requirements.

Their notion of a "preferred plan" addresses a similar issue as the notion of informativeness proposed here. However, their approach differs from ours in important ways. Given that their main focus is requirements engineering, their contribution addresses exclusively design-time (type-level) plans. Here, in contrast, given our emphasis on process mining, the focus is on the informative compliant (token-level) traces. As such, while their approach compare alternative plans defined over a model, we focus on comparing informative values of different variants of particular compliant traces by identifying the added value of optional parts of that trace. Furthermore, despite dealing with a similar notion, the author did not explicitly propose a precise value-based metric for measuring how informative process traces are.

Within process mining [1], many techniques have been proposed for model-to-log comparison, i.e., analyze the relationships between a log and a process model [4]. For example, to investigate the quality of a discovered model it is possible to compute three quality measures [3]: fitness [11, 2] (answering the question: "is the model able to replay the log?"), precision [9, 8] ("is the model underfitting the log?"), and generalization [13] ("is the model overfitting the log?"). These three measures are typically combined with a fourth one, called simplicity, which quantifies the "complexity" of the structure of the model (this measure is not particularly related to a specific log but measures specific properties of the model). As argued before though, conceptually speaking, all these measures are detecting the extent to which a model and a log "deviate". Therefore, traces that are compliant with the model cannot be distinguished between each other. Historically, process mining researchers always focused on quantifying the extent to which a trace is problematic (e.g., not fitting the model). Though, no effort has been put in establishing *how informative* a trace is.

5 Final Considerations and Future Work

We have presented a metric, namely informativeness, to identify compliant executions of business processes which are of particular interest. Such executions can be used to gain better understanding of the process and to trigger further investigations, such as improvement or redesign initiatives. Informativeness summarizes the performed behavior that goes beyond the necessary tasks required by the reference model and that impact on value. This non-trivially combines the employed definition of value and that of non-mandatory behavior. It is worth pointing out that in imperative languages non-mandatory behavior is typically observed in the presence of optional tasks or repetitions. Considering BPMN models, for example, optional tasks are realized with inclusive and exclusive gateways (i.e., OR and XOR), which are also involved in the construction of loops. To realize loops it is also possible to use task markers (i.e., loop and multiple instance markers). In the context of declarative languages, non-mandatory behavior can be caused by the presence of such control-flow structures (i.e., loops or optional tasks), but also by the fact that the model is under-constrained. Under-constrained models are fairly common in the declarative domain and, by using the informativeness

metric presented in this paper, it is possible to highlight undesired, compliant behaviors.

Many future works are conceivable, starting from a proper realization of the metric: the algorithm reported in this paper aimed solely to understandability, but for practical purposes it is inefficient. Specific implementations, tailored to concrete modeling languages, can leverage the corresponding semantic properties to drastically reduce the computational requirements.

Acknowledgements The work is supported by Innovation Fund Denmark project EcoKnow.org (7050-00034A). The authors would like to thank Marlon Dumas for providing the inspiration for Fig. 1.

References

1. van der Aalst, W.M.: Process Mining. Springer Berlin Heidelberg, 2nd edn. (2016)
2. Adriansyah, A., van Dongen, B., van der Aalst, W.M.: Conformance Checking Using Cost-Based Fitness Analysis. In: Proc. of EDOC. pp. 55–64. IEEE (2011)
3. Buijs, J., van Dongen, B., van der Aalst, W.M.: On the Role of Fitness, Precision, Generalization. In: Proc. of OTM. pp. 305–322. Springer (2012)
4. Carmona, J., van Dongen, B.F., Solti, A., Weidlich, M.: Conformance Checking - Relating Processes and Models. Springer (2018)
5. Guizzardi, G.: On ontology, ontologies, conceptualizations, modeling languages. In: Frontiers in Artificial Intelligence and Applications, Databases and Information Systems. IOS Press (2007)
6. Guizzardi, R.S.S., Franch, X., Guizzardi, G.: Applying a foundational ontology to analyze means-end links in the i* framework. In: Proc. of RCIS. pp. 1–11 (2012)
7. Liaskos, S., McIlraith, S.A., Sohrabi, S., Mylopoulos, J.: Representing and reasoning about preferences in requirements engineering. Requirements Engineering **16**(3), 227–249 (2011)
8. Munoz-Gama, J.: Conformance Checking and Diagnosis in Process Mining, LNBIP, vol. 270. Springer (2016)
9. Munoz-Gama, J., Carmona, J.: A fresh look at Precision in Process Conformance. In: Proc. of BPM. pp. 211–226. Springer (2010)
10. Mylopoulos, J.: Conceptual modelling and telos. Conceptual Modelling, Databases, and CASE pp. 49–68 (1992)
11. Rozinat, A., van der Aalst, W.M.: Conformance checking of processes based on monitoring real behavior. Information Systems **33**(1), 64–95 (2008)
12. Sales, T.P., Guarino, N., Guizzardi, G., Mylopoulos, J.: An ontological analysis of value propositions. In: Proc. of EDOC. pp. 184–193. IEEE Press (2017)
13. Vanden Broucke, S.K., De Weerdt, J., Vanthienen, J., Baesens, B.: Determining process model precision and generalization with weighted artificial negative events. IEEE Transactions on Knowledge and Data Engineering **26**(8), 1877–1889 (2014)
14. Yu, E., Giorgini, P., Maiden, N., Mylopoulos, J.: Social Modeling for Requirements Engineering. MIT Press (2011)