Mining Process Task Post-Conditions

By Metta Santiputri

Mining Process Task Post-Conditions

Metta Santiputri(✉), Aditya K. Ghose, Hoa Khanh Dam, and Xiong Wen

Decision Systems Lab, School of Computing and Information Technology,
University of Wollongong, Wollongong, NSW 2522, Australia
{ms804,aditya,hoa,xw926}@uowmail.edu.au
http://www.uow.edu.au

Abstract. A large and growing body of work explores the use of semantic annotation of business process designs, but these annotations can be difficult and expensive to acquire. This paper presents a data-driven approach to mining these annotations (and specifically post-conditions) from event logs in process execution histories which describe both task execution events (typically contained in *process logs*) and state update events (which we record in *effect logs*). We present an empirical evaluation, which suggests that the approach provides generally reliable results.

Keywords: Semantic annotation · Data-driven · Process logs · Effect logs

1 Introduction

A large and growing body of work explores the use of semantic annotation of business process designs [4,5,10,16,29,32]. A large body of work also addresses the problem of semantic annotation of web services in a similar fashion [24–26,28]. Common to all of these approaches is the idea that semantic annotation of process tasks or services provides value in ways that the process or service model alone cannot. Our focus in this paper is on *post-conditions* of tasks in the context of process models (pre-conditions are also of interest and we believe that an extension of the machinery presented here can address these, but are outside the scope of the present work). Ideally process designs annotated with post-conditions help answer the following question for any part of a process design: *what changes will have occurred in the process context if the process were to execute upto this point?* Arguably, a sufficiently detailed process model (for instance one that decomposes tasks down to the level of individual read or write operations) will require no additional information to answer this question. However, process models are most valuable when described at higher levels of abstraction, in terms of concepts and activities that stakeholders are familiar with. Processes annotated with post-conditions thus serve a crucial modeling function, providing an effective summary of a substantial body of knowledge regarding the "lower-level" workings of a process. Annotation with post-conditions can also help solve a range of problems such as process compliance management [10], change management [20], enterprise process architectures [13] and the management of the business process life cycle [21].

P. Johannesson et al. (Eds.): ER 2015, LNCS 9381, pp. 514–527, 2015.
DOI: 10.1007/978-3-319-25264-3_38

The modeling and acquisition of these post-conditions poses a particularly difficult challenge. It is generally recognized that process modeling involves significant investment in time and effort, which would be multiplied manyfold if there were an additional obligation to specify semantic annotations. Analysts also tend to find semantic annotation difficult, particularly if the intent is to make these formal (as is required by all of the use cases referred to above). This paper seeks to address this challenge by offering a set of techniques that mine readily available data associated with process execution to generate largely accurate "first-cut" post-conditions for process tasks. Our approach leverages the generally understood notion of *event logging*. The events that occur in a process execution context can be viewed in general terms as being of two types: (1) events that describe the start or end of the execution of process tasks and (2) events that describe state changes in the objects impacted by a process. In many settings, the existing event logging machinery is capable of logging both kinds of events. In other settings, we need to instrument *object state monitors* (for either physical objects or computational objects, or both) to obtain events of the second kind. In line with the literature addressing these, we refer to a time-stamped record of events of the first kind as a *process log*. We shall refer to a time-stamped record of events of the second kind as an *effect log* (in view of the fact that the state transitions being recorded are in fact *effects* of the process in question). We leverage these two types of logs in juxtaposition, and the time-stamped sequences of task execution events and state-change events thus obtained, to generate the *sequence database* taken as input by a sequential rule miner (CMRules [6] in our instance, but others could be used instead). The key idea is to identify commonly occurring patterns of task execution events, followed by sequences of state change events (or *effects*). As we show, the approach is generally quite effective. We also define techniques which leverage a *state update operator* (that defines how a specification of a state of affairs is updated as a consequence of the execution of an action) and the actual history of process execution provided by the juxtaposed process and effect logs to determine whether the mined effects, if accumulated using the state update operator, would indeed generate the available execution histories. This forms a validation step for the mined results.

Our intent is to mine the *context-independent effects* (or *immediate effects*) of each task. These are *contextualized* via iterated applications of the state update operator to obtain the *context-dependent effects* of each task (in the context of a process model)—a complete collection of these for each task or event provides a semantically annotated process model. For instance, the immediate effect of turning a switch on is to complete a circuit. In the context of a light bulb circuit, the context-dependent effect of this task would be to turn the bulb on. In the context of a switching circuit for a chemical reactor, the context-dependent effect of that same task would be to bring the chemical reactor to an operational state. We engage the machinery we present below by using it in the following manner: given as input a process log, an effect log, a process model (or a set of process models in the event that the logs describe the execution of instances of multiple

process designs) and a state update operator, the machinery would generate the immediate effects of each task referred to in the process log. These effects could be used directly in annotating process models, or might be viewed as "first-cut" specifications, to be edited and refined by expert analysts.

We provide background on semantic annotation of processes and on process and effect logs in Sect. 2. In Sect. 3, we present our approach to mining task effects and validating these. In Sect. 4, we present results of an experimental evaluation exercise, before presenting concluding remarks.

2 Background

Semantic Annotation: We assume that each task or event in a process is associated with effects written as conjunctive normal form sentences in the underlying formal state description language, which might be propositional or first-order (we do not consider temporal logics in this work, but extensions are possible). We assume that each task or event has context-independent *immediate effects* that can be contextualized via iterated applications of a *state update operator* as in [10] and [16]. We permit the contextualized effects to be non-deterministic—at any given point in a process, the actual effects that might accrue would be one of a set of *effect scenarios*. We need to support this non-determinism for two reasons. First, in any process with XOR-branches, one might arrive at a given task via multiple paths, and the contextualized effects achieved must be contingent on the path taken. Since this analysis is done at design time, we need to admit the possibility of non-deterministic effects since the specific path taken can only be determined at run-time. Second, many state update operators generate non-deterministic outcomes, since inconsistencies (that commonly appear in state update) can be resolved in multiple different ways. Of the two well-known state update operators in the literature—the Possible Models Approach (PMA) and the Possible Worlds Approach (PWA)—our work leverages the PWA [11]. Specifically, we use the operator $\oplus$ defined below. In the following, we assume that all consistency checks implicitly include a background knowledge base ($\mathcal{KB}$) containing rules and axioms. Thus, the statement that $e'_i \cup e_j$ is consistent effectively entails the statement that $e'_i \cup e_j \cup \mathcal{KB}$ is consistent. We omit references to $\mathcal{KB}$ for ease of exposition.

For two effects e_i and e_j, and the knowledge base $\mathcal{KB}$, if $e_i \not\models \bot$ and $e_j \not\models \bot$, then the *pair-wise effect accumulation* (or *state update*) $e_i \oplus e_j$ is defined as:

$$e_i \oplus e_j = \{e'_i \cup e'_i \mid e'_i \subseteq e_i \wedge e'_i \cup e_j \cup \mathcal{KB} \not\models \bot \wedge \text{there does not exist } e''_i \text{ such that } e'_i \subset e''_i \subseteq e_i \wedge e''_i \cup e_j \cup \mathcal{KB} \not\models \bot\}$$

The outcome of the state update operation is not a unique effect specification, but a set of non-deterministic *effect scenarios*. To see why this might be the case, consider a task T with a single associated effect scenario given by $\{p, q\}$ which is followed by task T' whose immediate effect is to make r true. Given a background

knowledge base consisting of a single rule $r \rightarrow (\neg p \vee \neg q)$, the $\oplus$ operator would give us two distinct outcomes: $\{p, r\}$ and $\{q, r\}$.

To obtain a complete annotation of a process model, we repeatedly apply the $\oplus$ operator over pairs of contiguous tasks in a process model, with the first argument being an effect scenario associated with the prior task and the second argument being the immediate effect of the later task. Special techniques are provided for dealing with XOR and AND gateways in proposals such as [10,16] and [32], but these are not directly relevant for our current exposition and we omit details here.

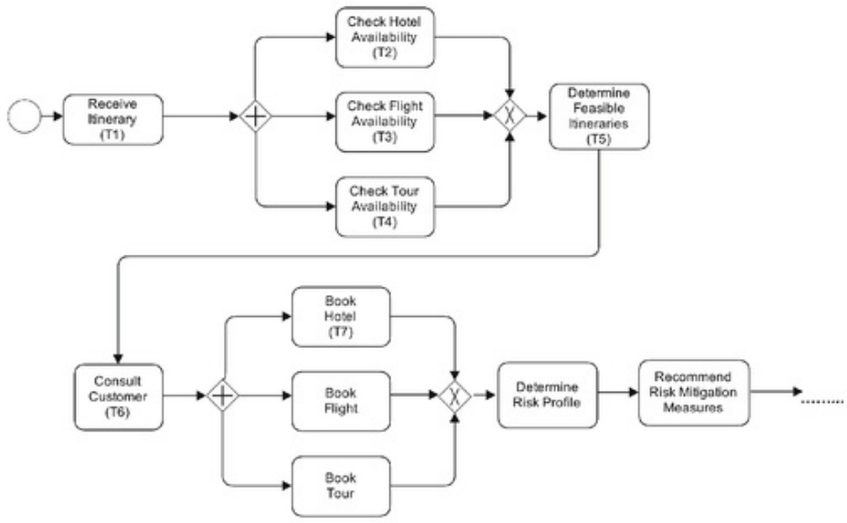

Fig. 1. A semantic effect-annotated BPMN process model for *Holiday Booking* process

Figure 1 illustrates a section of a semantically annotated BPMN process model for a *Holiday Booking* process followed by a travel agent. Table 1 outlines the semantic annotations of some of its tasks. The background knowledge base is given by a set of rules including the following representative examples. We use the standard convention of starting variables with upper-case letters.

(R1) $\forall Hotel, Dates, \exists Cust$ *hotel(Hotel, Dates)* $\rightarrow$ *hotel-pref(Cust, Hotel)*

(R2) $\forall Flight, Airline, ClassOfTravel, Dates, DepartTime, ArriveTime, \exists Cust$ *flight-available(Flight, Airline, ClassOfTravel, Dates, DepartTime, ArriveTime)* $\rightarrow$ *travel-dates(Cust, Dates)* $\wedge$ *airlines-preferences(Cust, Airline)* $\wedge$ *airline-classoftravel(Cust, ClassOfTravel)* $\wedge$ *departure-preferences(Cust, DepartTime)* $\wedge$ *arrival-prefs(Cust, ArriveTime)*

Process and Effect Logs: We assume that process execution data is available in the form a *process log* and an *effect log*. A process log (or event log) is a set of triples of the form $\langle CaseID, TimeStamp, TaskID \rangle$. The $TimeStamp$ value indicates the start time of *TaskID* while *CaseID* identifies the process instance that *TaskID* belongs to. We permit possibly many process instances (of the same process design or distinct process designs) to be executed at the same time.

Table 1. Annotation of holiday booking process in Fig. 1

Obtain customer requirements	*travel-dates(Cust, Dates)* ∧ *airline-preferences(Cust,Airline)* ∧ *airline-classoftravel(Cust, ClassOfTravel)* ∧ *departure-preferences(Cust,DepartTime)* ∧ *arrival-prefs(Cust, ArriveTime)* ∧ *meal-constraints(Cust, MealConstraints)* ∧ *freq-flyer(Cust, FreqFlyerDetails)* ∧ *hotel-pref(Cust, Hotel)* ∧ *room-prefs(Cust, RoomPrefs)* ∧ *tour-prefs(Cust, TourPrefs)*
Check Hotel Availability	*hotel-available(Hotel, Dates)*
Check flight availability	*flight-available(Flight, Airline, ClassOfTravel, DepartTimes, ArriveTimes)*
Check tour availability	*tour-available(Tour, DepartTimes, Depart-Location, Route, Stops)*
Determine feasibility itinerary	*feasible-itinerary(Flight, Hotel, Tour)*
Consult customer	*customer-confirmation(Cust, Itinerary)*

Table 2 shows a part of a process log describing the execution of multiple instances of the process model in Fig. 1.

An effect log consists of a set of tuples $\langle t_i, e_i \rangle$ where t_i is a timestamp and e_i is an effect assertion in the underlying state description language. The effect log records observed changes in the states of objects impacted by a process. It is important to note that only changes in state are recorded (and not state descriptions that persist). Effect logs can be obtained by instrumenting the process environment with object state monitors (both for physical objects as well as for computational/business objects). The underlying state description language might involve propositional state variables—the changes to describe would then be propositions becoming true or false, or more generally as disjunctions (in case state monitors have limited sensing capabilities). The underlying language might also admit non-Boolean state variables, in which case the effects logged would be the new value assignments to these variables. In many settings, it is convenient to represent effects in terms of first-order sentence schemas. In a travel booking process, the *"Obtain customer requirements"* task would lead to a ground instance of the sentence schema $airline\text{-}preference(Cust, Airline)$ becoming available. In this setting, the precise grounding of the *Cust* and *Airline* variables are not of particular interest. Indeed, recording the actual values of these variables in the effect log would lead to the effect mining procedure treating different groundings of the variables as distinct effects, when in fact we are only interested in recording the effect that a ground instance of that sentence schema

Table 2. A process log of the BPMN process model in Fig. 1

case ID	task ID	timestamp	case ID	task ID	timestamp
1	T_1	t_1	2	T_6	t_{103}
2	T_1	t_8	1	T_5	t_{125}
2	T_2	t_{25}	3	T_5	t_{145}
3	T_1	t_{27}	3	T_6	t_{156}
1	T_2	t_{28}	1	T_6	t_{172}
3	T_3	t_{32}	1	T_7	t_{174}
2	T_3	t_{33}	4	T_1	t_{377}
1	T_4	t_{36}	4	T_3	t_{379}
3	T_2	t_{71}	4	T_4	t_{438}
2	T_4	t_{80}	4	T_2	t_{440}
3	T_4	t_{88}	4	T_5	t_{461}
1	T_3	t_{95}	4	T_6	t_{471}
2	T_5	t_{98}			

Notes: T_1: *Receive Itinerary*, T_2: *Check Flight Availability*, T_3: *Check Hotel Availability*, T_4: *Check Tour Availability*, T_5: *Determine Feasible Itineraries*, T_6: *Consult Customer*, T_7: *Book Hotel*

becomes available. For effects of this sort, we only record a propositional effect of the form *airline-preference-known*. In other settings, we are interested in the precise instantiations of the variables in a sentence schema of the form $p(X, Y)$, in which case the full ground instance of $p(X, Y)$ is recorded in the effect log.

Table 3 illustrates a part of a effect log describing the effects of the execution of multiple instances of the process in Fig. 1.

3 Mining and Validating Effects

The effect mining technique we describe below takes as input: (1) a semantically annotated process model, (2) a state update operator $\oplus$, (3) a process log, and (4) an effect log (where both logs refer to the same history of process execution) and generates as output the immediate effect e_{T_i} for every task T_i referred to in the process log (these tasks may belong to one or more process designs). Our approach involves first mining a first-cut version of the immediate effects, and then filtering these by validating them against the process execution history as represented in the process log and effect log.

Effect mining procedure: Our approach to effect mining is predicated on the observation that the effects of executing a task occur soon after the execution of the task. Effects that manifest a long period after the execution of a task are typically not effects of that task alone, but of that task plus some others (e.g., one may think of the arrival of a traditional "snailmail" letter 3 days after

Table 3. An effect log of the BPMN process model in Fig. 1

timestamp	effects
t_2	*travel-dates-known*
t_3	*airline-preferences-known*
t_4	*airline-classoftravel-known*
t_4	*departure-preferences-known*
t_5	*arrival-prefs-known*
t_6	*meal-constraints-known*
t_6	*freq-flyer-known*
t_7	*hotel-pref-known*
t_7	*room-prefs-known*
t_7	*tour-prefs-known*
t_9	*travel-dates-known*
t_{10}	*airline-preferences-known*
t_{12}	*airline-classoftravel-known*
t_{13}	*departure-preferences-known*
t_{14}	*arrival-prefs-known*
t_{16}	*meal-constraints-known*
t_{16}	*freq-flyer-known*
t_{17}	*hotel-pref-known*
t_{45}	*room-prefs-known*
t_{22}	*tour-prefs-known*
t_{29}	*flight-available-known*

posting as an effect of the action of letter-posting, when it actually involves several other tasks executed by the postal service). The key pattern we leverage in mining effects is the *sequence* that involves the execution of a task and the manifestation of its effects, using a sequential pattern miner. We are interested in identifying all the effects that occur always (or most of the time) after each task is executed. Since the task executions are recorded in the process log and the effects occurrences are recorded in the effect log, we must first establish the correlations between the two logs to obtain a joined table that serves as the *sequence database* for a sequential rule miner. We use the CMRules algorithm [6] although a number of other candidates exist [3,7,8,15], and the framework is flexible enough to allow the use of any of these.

While our focus is on the sequential patterns that relate tasks to effects, we are not interested in the relative sequencing amongst effects. Indeed, it is undesirable for our purposes to have the sequential rule miner to view the sequences $\langle T, p, q \rangle$ and $\langle T, q, p \rangle$ as being distinct. We therefore enforce the rule that a contiguous sequence of effects in the sequence database must always be represented

in lexicographic order (this would require the second sequence above to be re-written as the first sequence).

We consider the problem of effect mining in two settings: (1) Settings characterized by the **unique task assumption** which stipulates that only one task may be performed at any point in time. This permits us to correlate all of the effects observed between the execution of a given task and the start of the next task with the first task. (2) Settings characterized by the **concurrent task assumption** which admits the possibility of multiple tasks executing concurrently (these could be tasks associated with distinct instances of the same process or associated with different processes). The second setting is more general, but the first setting simplifies the effect mining problem, and is worth considering if appropriate. We will apply the CMRules algorithm in both settings.

Both in settings with the *unique task assumption* and in settings with concurrent tasks, we create a joined table from the process log and the effect log. of the form:

$$\langle\langle T_1, \langle\langle e_{11}\rangle, \ldots, \langle e_{1n}\rangle\rangle\rangle, \ldots, \langle T_i, \langle\langle e_{i1}\rangle, \ldots, \langle e_{im}\rangle\rangle\rangle, \ldots \langle T_p, \langle\langle e_{p1}\rangle, \ldots, \langle e_{pk}\rangle\rangle\rangle$$

where each $\langle T_i, T_{i+1}\rangle$ pair represents contiguous tasks and each e_{ij} represents the j-th effect observed after the start of task T_i and before the start of task T_j. This table represents the sequence database provided as input to the sequential rule miner. A special provision is needed for the last task in case it does not have any subsequent task. Instead of using the last record in the process log as the end timestamp, we assume that we have prior information about the maximal time of process execution, ϵ, and use it as the end time of the last task in any case.

We then apply the CMRules algorithm, with the best results obtained when the values of *minSeqSup* and the *minSeqConf* are bounded from below by the number of distinct case-ID in which a specific task occurs (as with any association rule mining technique, *minSeqSup* and *minSeqConf* represent the support and confidence respectively—higher values of these can give us more reliable results but rule out potentially interesting rules and vice versa). In *unique task* settings with noise, the sequence of effects following the exection of each task and prior to the execution of the next task in the process instance should be largely identical if the process design is fixed—we apply CMRules mainly to mitigate the effects of noise. In *concurrent task* settings, these could vary significantly since the effects that follow a task might not be its effects but those of a distinct concurrent task. In these settings, the sequential rule miner is essential to identify the commonly occurring patterns of effects following a given task. In general terms, a sequential rule $X \rightarrow Y$ consists of two parts: the antecedent X and the consequent Y, which are both assumed to be sequences of transactions. The rule states that if the elements of X occur in a given sequence in the sequence database being mined, then the elements of Y will follow in the same sequence and in a manner that preserves the sequential relations between the elements of X and between the elements of Y. All sequential rules must also satisfy certain criteria regarding their accuracy (minimum confidence) and the proportion of the data that they actually represent (minimum support).

For example, consider case 1 in Table 2. The first task, task T_1 has timestamp t_1 and the next task in the case, T_2, has timestamp t_{28}; therefore, we group task T_1 with all effects in the effect log with the timestamp t_1 until t_{28}, which gives us the sequence (T_1)(*travel-dates-known*)(*airline-prefs-known*)(*airline-classoftravel-known*)(*departure-prefs-known*) (*arrival-prefs-known*) (*meal-constraints-known*) (*freq-flyer-known*) (*hotel-prefs-known*) (*room-prefs-known*) (*tour-prefs-known*) (*travel-dates-known*)(*airline-prefs-known*) (*airline-classoftravel-known*) (*departure-prefs-known*) (*arrival-prefs-known*) (*meal-constraints-known*) (*freq-flyer-known*)(*hotel-pref-known*)(*room-prefs-known*)(*tour-prefs-known*). Similarly, task T_2 is grouped with all effects with timestamp from t_{28} until t_{36}, and so on. Applying the same process to all the other cases, we obtain the sequences for task T_1, T_2, T_3, until T_7. Next, these sequences are grouped into sequence databases based on their task ID, for instance the sequence for task T_1 from case 1 goes into the same sequence database with the task T_1 sequence from case 2 (along with task T_1 sequences from other cases).

Although the CMRules algorithm is able to generate all sequential rules from the sequence databases, further post-processing is required. Since we are interested only in relations between a task and effects, only rules with a single task ID as antecedent are included in the results and all other rules are discarded.

Validation: We can use the state update operator and the available data to perform additional validation steps on the effects mined in the manner described above. In settings characterized by the *unique task assumption*, an element of the joined process log and effect log can be viewed as *semantic execution trace* of the form:

$$\langle\langle T_1, \langle\langle e_{11}\rangle, \ldots, \langle e_{1n}\rangle\rangle\rangle, \ldots, \langle T_i, \langle\langle e_{i1}\rangle, \ldots, \langle e_{im}\rangle\rangle\rangle, \ldots \langle T_p, \langle\langle e_{p1}\rangle, \ldots, \langle e_{pk}\rangle\rangle\rangle$$

for a process instance (case) with p tasks, with each T_i representing a task ID and each e_{ij} representing the j-th effect associated with task T_i. In other words, the sequence of effects associated with each task in the trace above represents the cumulative effects obtained at that point in the process instance. We shall refer to the sequence of tasks $\langle T_1, \ldots, T_p\rangle$ as the *signature* of the semantic execution trace above, and note that multiple semantic execution traces might be obtained for the same signature (due to the fact that we might find the process in one of many possible non-deterministic effect scenarios after the execution of a sequence of tasks). To validate the immediate effects mined using the procedure described in this section, we must establish for each semantic execution trace in the joined process log and effect log and for each task T_i in that trace that the following condition holds, where es_i is given by $e_{i1} \wedge e_{i2} \wedge \ldots \wedge e_{im}$ with m being the number of effect log entries associated with T_i: $es_i \models e$ for some $e \in e_{T_1} \oplus e_{T_2} \oplus \ldots \oplus e_{T_i}$ where each e_{T_i} denotes the mined immediate effects of task T_i. This represents a *soundness condition*, in the sense that we guarantee that every observed set of accumulated effects includes the accumulation of all mined effects of the tasks executed upto that point. For a sufficiently extensive collection of process and effect logs, we may also require that there must exist, for every $e \in e_{T_1} \oplus e_{T_2} \oplus \ldots \oplus e_{T_i}$, some entry in the joined process-effect log with an es_i associated with T_i such that $es_i \models e$. A completeness condition

would reverse the entailment relation (i.e., $e \models es_i$). If the mined effects failed all of these tests, a weaker condition of consistency ($es_i \wedge e \not\models \bot$) would suggest that the mined results were not entirely incorrect. In settings characterized by *concurrent tasks*, we cannot guarantee that the effects observed between the start of a task and the start of the next task in the same process instance are necessarily the effects of the former task (since concurrent tasks from other process instances might have led to these effects). In such settings, we validate by creating modified sequence databases, parameterized by a task sequence length parameter n for use with CMRules. For instance, when the task sequence length parameter is 2, for each contiguous pair of tasks $\langle T_i, T_j \rangle$, we take sequences of the form $\langle T_i, e_{i1}, \ldots, e_{in} \rangle$ and $\langle T_j, e_{j1}, \ldots, e_{jm} \rangle$ where the tasks belong to the same process instance and where the timestamps associated with each e_{ik} is earlier than the start of T_j and create an entry in this modified sequence database of the form $\langle T_i, T_j, e_{i1}, \ldots, e_{in}, e_{j1}, \ldots, e_{jm} \rangle$ by removing any effect e_{ik} associated with the earlier task T_i where $e_{ik} \wedge \mathcal{KB} \models \neg e_{jl}$ for some e_{jl} ($\mathcal{KB}$ is the background knowledge base). Since the effect log records only changes (and not persistent effects or non-changes), we perform the last step to ensure some form of state update is reflected in the combined sequence for $\langle T_i, T_j \rangle$. We use CMRules to obtain rules of the form $\langle T_i, T_j \rangle \rightarrow \langle e_1, \ldots, e_p \rangle$ with the support and confidence being set as earlier to refer only to those process instances where T_i and T_j appear contiguously. The following condition then provides a weak form of validation: $e_1 \wedge \ldots \wedge e_n \models e_{T_i} \oplus e_{T_j}$. The approach generalizes to task sequences of arbitrary length, but we omit details due to space constraints. A general validation strategy is to consider all task sequences of length $i = 1, \ldots, n$ where n is the length of the longest task sequence that conforms to the process design.

4 Evaluation

Evaluation with synthetic process models: Our aim is to establish that our approach generates reasonably reliable results. We ran the first set of experiments with a synthetic semantically annotated process model (i.e., a handcrafted one with $T_1, T_2, \ldots$ etc., for task names and $p, q, \ldots$ for effects). The model had 8 tasks, with an AND-split nested inside an XOR-split and with each task semantically annotated with 1 or 2 literals (in the 2 literal case, the effects were conjunctions of the 2 literals), and one rule in the $\mathcal{KB}$. We simulated a large number of possible execution traces of this model, and obtained synthetic process and effect logs. These logs involved the execution of multiple concurrent process instances. There were multiple effect scenarios associated with some of the tasks, owing to the fact that XOR gateway contributed to alternative flows that could have led to the same point (none of the effect scenarios were generated by alternative means of resolving inconsistency in the state update operator). We then investigated the effect of scaling up the complexity of the process model, by generating a second synthetic process model with 12 tasks with an XOR-split leading to two alternative flows, one of which included a nested AND-split and the other a nested XOR-split. The semantic annotations were 2 or 3 literals long

and involved a mix of conjunctions and disjunctions. The background $\mathcal{KB}$ had 4 rules. There were multiple effect scenarios associated with most of the tasks and these were generated both by alternative flows that could lead to a task (on account of XOR gateways) and by alternative resolutions of inconsistency by the state update operator.

Table 4 below describes the results of 4 experiments with each of these two process models. We used progressively larger numbers of overlapping instances of each process (i.e., T_i in instance 2 would start after the start of T_i in instance 1, but before the start of T_{i+1} in instance 1, and so on). We note that our problem would be no harder if the multiple concurrent process instances were of multiple distinct process models. We obtained progressively larger sizes of the sequence database. We recorded the precision (number of correct effect mined over the total number of effects mined) and recall (the number of correct effects mined over the total number of actual effects. Although not entirely monotonically improving, the results for process 2 confirm the intuition that better results are obtained with larger datasets. The results for process 2 also showed that the effects mined tended to be incorrect for the last task in a process instance (in those settings where precision and recall values were less than 1). This was due to the sequence of effects for the final task not being bounded by the start of the next task, but rather by the end of the log (artificially determined by length of the longest process).

Table 4. The recall and precision measures from the evaluation

	Process model 1				Process model 2			
Number of instances	5	10	100	500	5	10	100	500
Size of sequence DB	48	100	1082	5352	66	133	1297	6512
Recall	1.0	1.0	1.0	1.0	0.953	1.0	0.981	0.989
Precision	1.0	1.0	1.0	1.0	1.0	0.988	1.0	1.0

The synthetic process and effect logs used in these examples considered all possible flows. Real-life data might involve more imperfections (such as certain XOR flows never being executed, certain tasks never being executed and so on). We have also considered cases where noise is artificially added to the logs - we do not report these results here due to space constraints, but, as expected, precision and recall suffer as noise increases. We do not present an evaluation of the technique we propose for settings where we can make a *unique task assumption*, both because of space constraints and because the technique is simple, and, in our experience, almost always accurate. We do not present in detail an evaluation of the validation techniques from the previous section, but our experience suggests that it is effective in filtering out inaccurate effects.

User-mediated evaluation: To evaluate our approach in a more real-life setting, we took a real-life semantically annotated process model (Fig. 1) and obtained a set of process and effect logs from an expert process modeler. We

obtained a process log describing 10 execution instances (many of them with temporal overlaps) with a total of 110 entries, and an effect log with 154 entries.

We found that about 1 in 9 tasks, the effects mined were incorrect. The best explanation of this appears to be the fact that in the user-generated process log, there were other tasks that were exactly concurrent with the task for which the wrong effects were mined.

5 Related Work

A few examples of the benefits that can be exploited from semantically annotated business process models including compliance checking [4,10,14,17], management level strategic alignment of business processes [22], and exception handling [9]. Two groups of studies can be defined based on the purpose of adding semantics to business process model: (1) to specify the dynamic behaviour of the business process [32,34], and (2) to specify the meaning of the entities in the business process model itself [4,22,30]. In order to assist business analysts and process designers in the process of business process modelling, many studies have proposed frameworks to provide more user-friendly tools to add semantics into business process models [2,16,18].

In regards of assisting designers and analysts, many studies have emerged in harnessing historical data, specifically on software repositories, to discover useful information, with data such as programmer interaction history [23], software revision history [35,36], email history [1], visited web pages [27], and bug reports [19]. Particularly in business process modeling, there are extensive studies on process mining that exploit the historical data of process model executions, i.e. event logs. Process mining algorithms—such as alpha algorithm [31], heuristic miner [33], and fuzzy miner [12]—extract the structure of the process model.

Our research integrates these two approaches of (1) mining historical data to discover useful information and (2) adding semantics to business process models, in order to support the business process modeling. We use data sources similar to those used in process mining, which is the business process execution history in the form of event log or process log, but our research does not generate the structure of the process model, instead it discovers the semantics of the business processes.

6 Conclusions and Future Work

This paper offers an approach to mining business process task post-conditions (or effects) from process logs and logs of state changes in the process context. The empirical evaluation suggests that the results are generally reliable, pointing to prospects for further development of techniques that leverage these post-conditions in semantic analysis.

References

1. Bacchelli, A., Lanza, M., Humpa, V.: RTFM (Read The Factual Mails)–augmenting program comprehension with REmail. In: Proceedings of 15th IEEE European Conference on Software Maintenance and Reengineering (CSMR), pp. 15–24. IEEE (2011)
2. Born, M., Dörr, F., Weber, I.: User-friendly semantic annotation in business process modeling. In: Weske, M., Hacid, M.-S., Godart, C. (eds.) WISE Workshops 2007. LNCS, vol. 4832, pp. 260–271. Springer, Heidelberg (2007)
3. Chan, K.C., Au, W.H.: An effective algorithm for mining interesting quantitative association rules. In: Proceedings of the 1997 ACM Symposium on Applied Computing, pp. 88–90. ACM (1997)
4. Di Francescomarino, C., Ghidini, C., Rospocher, M., Serafini, L., Tonella, P.: Reasoning on semantically annotated processes. In: Bouguettaya, A., Krueger, I., Margaria, T. (eds.) ICSOC 2008. LNCS, vol. 5364, pp. 132–146. Springer, Heidelberg (2008)
5. Fensel, D., Facca, F., Simperl, E., Toma, I.: Web service modeling ontology. Semantic Web Services, pp. 107–129. Springer, Heidelberg (2011)
6. Fournier-Viger, P., Faghihi, U., Nkambou, R., Nguifo, E.M.: CMRules: mining sequential rules common to several sequences. Knowl. Based Syst. **25**(1), 63–76 (2012)
7. Fournier-Viger, P., Nkambou, R., Tseng, V.S.M.: RuleGrowth: mining sequential rules common to several sequences by pattern-growth. In: Proceedings of the 2011 ACM Symposium on Applied Computing, pp. 956–961. ACM (2011)
8. Garofalakis, M.N., Rastogi, R., Shim, K.: SPIRIT: sequential pattern mining with regular expression constraints. Proc. VLDB **99**, 7–10 (1999)
9. Ghidini, C., Rospocher, M., Serafini, L.: A formalisation of BPMN in description logics. FBK-irst, Technical Report TR, 06–004 (2008)
10. Ghose, Aditya K., Koliadis, George: Auditing business process compliance. In: Krämer, Bernd J., Lin, Kwei-Jay, Narasimhan, Priya (eds.) ICSOC 2007. LNCS, vol. 4749, pp. 169–180. Springer, Heidelberg (2007)
11. Ginsberg, M.L., Smith, D.E.: Reasoning about action I: a possible world approach. Artif. Intell. **35**(2), 165–195 (1988)
12. Günther, C.W., van der Aalst, W.M.P.: Fuzzy mining – adaptive process simplification based on multi-perspective metrics. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 328–343. Springer, Heidelberg (2007)
13. Hall, C., Harmon, P.: The 2005 Enterprise Architecture. Process Modeling and Simulation Tools Report, Technical report (2005)
14. Happel, H.J., Stojanovic, L.: Ontoprocess–a prototype for semantic business process verification using SWRL rules. In: Proceedings of the 3rd European Semantic Web Conference (ESWC), June 2006
15. Harms, S.K., Deogun, J.S.: Sequential association rule mining with time lags. J. Intell. Inf. Syst. **22**(1), 7–22 (2004)
16. Hinge, K., Ghose, A., Koliadis, G.: Process SEER: a tool for semantic effect annotation of business process models. In: Proceedings of the IEEE International Enterprise Distributed Object Computing Conference (EDOC 2009), pp. 54–63, September 2009
17. Hoffmann, J., Weber, I., Governatori, G.: On compliance checking for clausal constraints in annotated process models. Inf. Syst. Front. **14**(2), 155–177 (2012)
18. Hornung, T., Koschmider, A., Oberweis, A.: A recommender system for business process models. In: 17th Annual Workshop on Information Technologies and Systems (WITS) (2009)

19. Kim, M., Notkin, D.: Discovering and representing systematic code changes. In: Proceedings of the 31st International Conference on Software Engineering, pp. 309–319. IEEE Computer Society (2009)
20. Koliadis, G., Ghose, A.: Correlating business process and organizational models to manage change. In: Proceedings of the Australasian Conference on Information Systems, December 2006
21. Koliadis, G., Vranesevic, A., Bhuiyan, M., Krishna, A., Ghose, A.: A combined approach for supporting the business process model lifecycle. In: Proceedings of the 10th Pacific Asia Conference on Information Systems (PACIS 2006) (2006)
22. Koliadis, G., Ghose, A., Bhuiyan, M.: Correlating business process and organizational models to manage change. In: Proceedings of the Australasian Conference on Information Systems, pp. 1–10 (2006)
23. Lee, S., Kang, S., Kim, S., Staats, M.: The impact of view histories on edit recommendations. IEEE Trans. Softw. Eng. **41**(3), 314–330 (2015)
24. Martin, D., Paolucci, M., McIlraith, S.A., Burstein, M., McDermott, D., McGuinness, D.L., Parsia, B., Payne, T.R., Sabou, M., Solanki, M., Srinivasan, N., Sycara, K.: Bringing semantics to web services: the OWL-S approach. In: Cardoso, J., Sheth, A.P. (eds.) SWSWPC 2004. LNCS, vol. 3387, pp. 26–42. Springer, Heidelberg (2005)
25. Meyer, H.: On the semantics of service compositions. In: Marchiori, M., Pan, J.Z., Marie, C.S. (eds.) RR 2007. LNCS, vol. 4524, pp. 31–42. Springer, Heidelberg (2007)
26. Montali, M., Pesic, M., van der Aalst, W.M.P., Chesani, F., Mello, P., Storari, S.: Declarative specification and verification of service choreographiess. ACM Trans. Web **4**, 1–62 (2010)
27. Sawadsky, N., Murphy, G.C., Jiresal, R.: Reverb: recommending code-related web pages. In: Proceedings of the 2013 International Conference on Software Engineering, pp. 812–821. IEEE Press (2013)
28. Smith, F., Missikoff, M., Proietti, M.: Ontology-based querying of composite services. In: Ardagna, C.A., Damiani, E., Maciaszek, L.A., Missikoff, M., Parkin, M. (eds.) BSME 2010. LNCS, vol. 7350, pp. 159–180. Springer, Heidelberg (2012)
29. Smith, F., Proietti, M.: Rule-based behavioral reasoning on semantic business processes. In: ICAART, pp. 130–143. SciTePress (2013)
30. Thomas, O., Fellmann, M.: Semantic EPC: enhancing process modeling using ontology languages. In: SBPM, vol. 251 (2007)
31. van der Aalst, W., Weijters, T., Maruster, L.: Workflow mining: discovering process models from event logs. IEEE Trans. Knowl. Data Eng. **16**(9), 1128–1142 (2004)
32. Weber, I., Hoffmann, J., Mendling, J.: Semantic business process validation. In: Proceedings of the 3rd International Workshop on Semantic Business Process Management (SBPM08), CEUR-WS Proceedings, vol. 472 (2008)
33. Weijters, A., van Der Aalst, W.M., De Medeiros, A.A.: Process mining with the heuristics miner-algorithm. Technische Universiteit Eindhoven, Technical Report WP, vol. 166, pp. 1–34 (2006)
34. Wong, P.Y., Gibbons, J.: A relative timed semantics for BPMN. In: Proceedings of 7th International Workshop on the Foundations of Coordination Languages and Software Architectures, vol. 229 of ENTCS, July 2008
35. Ying, A.T., Murphy, G.C., Ng, R., Chu-Carroll, M.C.: Predicting source code changes by mining change history. IEEE Trans. Softw. Eng. **30**(9), 574–586 (2004)
36. Zimmermann, T., Zeller, A., Weissgerber, P., Diehl, S.: Mining version histories to guide software changes. IEEE Trans. Softw. Eng. **31**(6), 429–445 (2005)

Mining Process Task Post-Conditions

ORIGINALITY REPORT

12%

SIMILARITY INDEX

PRIMARY SOURCES

1	"Advances in Conceptual Modeling", Springer Nature America, Inc, 2014 Crossref	86 words — 2%
2	Kerry Hinge. "Process SEER: A Tool for Semantic Effect Annotation of Business Process Models", 2009 IEEE International Enterprise Distributed Object Computing Conference, 09/2009 Crossref	53 words — 1%
3	Dam, Hoa Khanh, and Aditya Ghose. "Mining version histories for change impact analysis in business process model repositories", Computers in Industry, 2015. Crossref	52 words — 1%
4	ilovestem.org Internet	47 words — 1%
5	"Business Process Management", Springer Nature America, Inc, 2015 Crossref	45 words — 1%
6	Lecture Notes in Computer Science, 2012. Crossref	31 words — 1%
7	"Goal-Aligned Categorization of Instance Variants in Knowledge-Intensive Processes", Lecture Notes in Computer Science, 2015. Crossref	27 words — < 1%
8	"Learning Relationships Between the Business Layer and the Application Layer in ArchiMate Models", Lecture Notes in Computer Science, 2015. Crossref	25 words — < 1%
9	"Advanced Information Systems Engineering", Springer Nature, 2017 Crossref	24 words — < 1%
10	www.docstoc.com Internet	19 words — < 1%
11	Yoo, EunHye, C. Rudra, M. Glasgow, and L. Mu. "Geospatial Estimation of Individual Exposure to Air Pollutants: Moving From Static Monitoring to Activity-Based Dynamic Exposure Assessment", Annals of the Association of	19 words — < 1%

	American Geographers, 2015. Crossref	
12	hal.archives-ouvertes.fr Internet	18 words — < 1%
13	Lecture Notes in Computer Science, 2008. Crossref	16 words — < 1%
14	"Conceptual Modeling", Springer Nature, 2016 Crossref	15 words — < 1%
15	Lecture Notes in Computer Science, 2011. Crossref	15 words — < 1%
16	Lecture Notes in Computer Science, 2013. Crossref	14 words — < 1%
17	Lecture Notes in Computer Science, 2009. Crossref	12 words — < 1%
18	dkm-static.fbk.eu Internet	11 words — < 1%
19	"Knowledge Engineering and Knowledge Management", Springer Nature, 2016 Crossref	10 words — < 1%
20	ceur-ws.org Internet	10 words — < 1%
21	udoo.uni-muenster.de Internet	9 words — < 1%
22	tel.archives-ouvertes.fr Internet	9 words — < 1%
23	Chuang, Yu-Cheng, PingYu Hsu, and Hung-Hao Chen. "Unifying Multi-level Business Process Discovered by Heuristic Miner Algorithm", Lecture Notes in Computer Science, 2013. Crossref	9 words — < 1%
24	epub.wu.ac.at Internet	9 words — < 1%
25	Alessandro Palù. "25 Years of Applications of Logic Programming in Italy", Lecture Notes in Computer Science, 2010 Crossref	8 words — < 1%
26	eprints-phd.biblio.unitn.it Internet	8 words — < 1%

27	"Enterprise, Business-Process and Information Systems Modeling", Springer Nature America, Inc, 2013 Crossref	8 words — < 1%
28	dbis.eprints.uni-ulm.de Internet	8 words — < 1%
29	"On the Move to Meaningful Internet Systems: OTM 2010", Springer Nature America, Inc, 2010 Crossref	8 words — < 1%
30	www.philippe-fournier-viger.com Internet	8 words — < 1%
31	Business Process Management Journal, Volume 16, Issue 5 (2010-09-25) Publications	8 words — < 1%
32	Liqiang Geng. "Discovering Structured Event Logs from Unstructured Audit Trails for Workflow Mining", Lecture Notes in Computer Science, 2009 Crossref	8 words — < 1%
33	Kurniawan, Tri A., Aditya K. Ghose, Hoa Khanh Dam, Lam-Son Le, and Tiancheng Zhang. "Design Maintenance in Process Eco-Systems", 2012 IEEE Ninth International Conference on Services Computing, 2012. Crossref	8 words — < 1%
34	Lecture Notes in Business Information Processing, 2016. Crossref	8 words — < 1%
35	Namiri, Kioumars. "Model-Driven Management of Internal Controls for Business Process Compliance", Universität Karlsruhe, 2008. Publications	8 words — < 1%
36	"Business Process Management Workshops", Springer Nature America, Inc, 2015 Crossref	7 words — < 1%
37	"On the Move to Meaningful Internet Systems. OTM 2017 Conferences", Springer Nature, 2017 Crossref	7 words — < 1%
38	Lecture Notes in Business Information Processing, 2013. Crossref	6 words — < 1%
39	Yingzhi Gou, Aditya Ghose, Hoa Khanh Dam. "Chapter 29 Leveraging Game-Tree Search for Robust Process Enactment", Springer Nature, 2017 Crossref	6 words — < 1%

EXCLUDE QUOTES	OFF	EXCLUDE MATCHES	OFF
EXCLUDE BIBLIOGRAPHY	ON		