Annotating and Mining for Effects of Processes

By Suman Roy

Annotating and Mining for Effects of Processes

Suman Roy[1(✉)], Metta Santiputri[2], and Aditya Ghose[2]

[1] Infosys Ltd., # 44 Electronics City, Hosur Road, Bangalore 560 100, India
Sunam_Roy@infosys.com

[2] School of Computer Science and Software Engineering, University of Wollongong, Wollongong, NSW 2500, Australia
ms804@uowmail.edu.au, aditya@uow.edu.au

Abstract. We provide a novel explicit annotation of a process model by way of accumulating effects of individual tasks specified by analysts using belief bases and computing the accumulated effect up to the point of execution of the process model in an automated manner. This technique permits the analyst to specify immediate effect annotations in a practitioner-accessible simple propositional logic formulas and generates a sequence of tasks along with cumulative effects, called effect logs. Further we propose and solve an effect mining problem, that is, given an effect log discover the process model with effect annotations of individual tasks which is close to the original annotated process model.

Keywords: Business process modeling · Semantic annotation · Effects · Belief bases · Annotated processes · Effect logs · Effect mining

1 Introduction

In this work we describe a technique for an explicit semantic annotation of process models. We require practitioners to provide a description of the immediate effects of each task. An effect of a task becomes true when the latter gets executed. These effects are propagated across different nodes (tasks and gateways) in the process model and then accumulated in a context-sensitive manner automatically, such that the cumulative effect annotation associated with any task in a BPMN process model[1] would describe the effects achieved by the process as if the process were executed up to that point [3]. The cumulative effect description of a task might be non-deterministic, this non-determinism may be caused by the execution of parallel gateways resulting in interleaving of paths in the process model, and belief update leading to multiple alternative means of resolving inconsistencies generated by the "undoing" of effects. Our effect annotation borrows ideas from Semantic Business Process Validation (SBPV) which

S. Roy—This work was done when the author visited University of Wollongong during July–Dec'14 to work on Infosys-CRC funded project on data-driven process discovery.

[1] Process models captured using industry standard notation BPMN.

I. Comyn-Wattiau et al. (Eds.): ER 2016, LNCS 9974, pp. 302–310, 2016.
DOI: 10.1007/978-3-319-46397-1_24

combines concepts from the workflow [6,8,13] and AI action and changes [4,15]. While we adopt the token semantics from workflow literature [14,15] for determining execution traces of processes we use belief operator from [5,7] for updating the effect and their accumulation on enabling of nodes. By this way we are able to compute an effect trace of the process which is a sequence of pairs consisting of task and the cumulative effect at this node.

Next we consider an inverse problem: given an effect log, *i.e.*, a set of effect traces, how one can determine the original effects at the tasks? We call this effect mining problem, *aka* process mining problem [11,12]. Although process designers take utmost care in designing process models there is no guarantee that they indeed reflect the correct models under consideration. The goal of process discovery is to derive some sort of model that describe the process as accurately as possible. Similarly, the goal of effect discovery is to generate a semantically annotated process which is close to the original semantic description of the process model. As one solution approach for this effect mining problem we propose a modification of process discovery algorithm, *viz.*, α-mining problem to find out a process model along with the effects associated with each task.

Related Work. There is a rich body of work on semantic annotation for web services. In one of them Weber *et al.* proposed an approach, Semantic Business Process Validation (SBPV) [15], in which axiomatic task descriptions are annotated and propagated across process models. The SBPV approach requires the user to completely specify pre-conditions and post-conditions that are context sensitive. In another approach Hinge *et al.* [3] proposed a technique for obtaining semantic effect descriptions of BPMN process models, without requiring the analyst to express excessive formal specification. Using an approach similar to SBPV in [4] Hoffman *et al.* proposed a framework of annotating processes for capturing the semantics of task execution in which compliance is checked against a set of constraints imposed on process states. Motivated by SBPV [4,15] and the annotation method suggested in [3] we lift ideas from AI and use belief bases to annotate the tasks with effects, propagate these effects across the nodes automatically in a context sensitive manner, and compute the accumulated effects of each task up to the execution point.

There is not much work on effect mining we could find. In [10] Santiputri *et al.* present a data-driven approach for mining semantically annotated business processes. The authors assume event logs in execution histories of business processes describing both task execution events (found in process logs) and state update events (recorded in effect logs) at disposal and mine for immediate effect annotations for each task in the process model to be edited and refined by analysts. In our work we only consider effect logs which is a collection of pairs consisting of tasks and accumulated effects, and discover the process model with effect annotations for each task. While in the previous work the authors use a variant of sequence mining algorithms for effect mining we use a modified process mining algorithm for effect discovery.

The paper is organized as follows. We introduce a semantic annotation of process models in Sect. 2 along with an example of effect annotation. The problem of effect mining is defined in Sect. 3. Finally we conclude in Sect. 4.

2 A Semantic Annotation of Processes

We introduce an annotation framework for business processes where tasks are annotated with effects. An effect of a task is some fact which materializes when the task is executed. It is captured by analysts providing a description of the immediate effects of each process task, i.e., a context independent specification of the functionality of each task. For the sake of easier readability we first introduce a simple process graph without annotation. We use a formalism of a business process which bears close resemblance with those described in [1,4,15] and that of workflows [8].

A *BPM process* is a graph (also called a process model graph) $\mathbf{P} = (\mathcal{N}, \mathcal{F})$ where $\mathcal{N}$ is a finite set of nodes which is partitioned into the set of tasks $\mathcal{T}$, the set of gateways $\mathcal{G}$, and the set of events $\mathcal{E}$, *i.e.*, $\mathcal{N} \widehat{=} \mathcal{T} \uplus \mathcal{G} \uplus \mathcal{E}$; $\mathcal{G}$ can be further partitioned into disjoint sets of decision merges, $\mathcal{G}_M$ ($\mathcal{G}_M^{and}$ (synchronizer) and $\mathcal{G}_M^{xor}$ (merge)) and decision splits, $\mathcal{G}_S$ ($\mathcal{G}_S^{and}$ (fork) and $\mathcal{G}_S^{xor}$ (choice)); a set $\mathcal{E}$ of events which is a disjoint union of two sets of events $\mathcal{E}_s$ and $\mathcal{E}_f$, where $\mathcal{E}_s$ is the set of start events with no incoming edges, $\mathcal{E}_f$ is the set of end events with no outgoing edges; and $\mathcal{F} \subseteq (\mathcal{N} \backslash \mathcal{E} \times \mathcal{N} \backslash \mathcal{E}) \bigcup (\mathcal{E}_s \times \mathcal{N} \backslash \mathcal{E}) \bigcup (\mathcal{N} \backslash \mathcal{E} \times \mathcal{E}_f)$ corresponds to sequence flows connecting tasks with tasks, tasks with gateways, gateways with tasks, start nodes with tasks and tasks with end nodes.

Let $in(n)$ ($out(n)$) be the set of incoming (outgoing) edges to (out of) node $n \in \mathcal{N}$. We impose the following conditions: $\forall n_s \in \mathcal{E}_s, |in(n_s)| = 0$, and $|out(n_s)| = 1$; $\forall n_f \in \mathcal{E}_f, |in(n_f)| = 1$, and $|out(n_f)| = 0$; for every $n \in \mathcal{T}, |in(n)| = |out(n)| = 1$; for every $n \in \mathcal{G}_S, |in(n)| = 1$ and $|out(n)| > 1$; for every $n \in \mathcal{G}_M, |in(n)| > 1$ and $|out(n)| = 1$; any outgoing edge $out(n)$ from a fork node n $\mathcal{G}_S^{and}$ will have a task node t appearing immediately after n: $(n, t) \in \mathcal{F}$, and every node is on a path from some start node to some end node. If these conditions hold then we say the business process to be *well-formed*. We shall consider only well-formed business processes henceforth.

Let us now specify the semantics of control elements of a business process. Given a process $\mathbf{P} = (\mathcal{N}, \mathcal{F})$, a *state* of $\mathbf{P}$ is a marking $\mu : \mathcal{F} \rightarrow \mathbb{N}$, also called a token mapping. The number of tokes may change during the execution of the process, when the transitions are enabled. A state μ' is reached from state μ via node n, written as $\mu \xrightarrow{n} \mu'$, when n can be a task, AND-split, AND-join or XOR-split or XOR-join, for details see [9,15].

The initial state is given by a marking μ_0 where $\mu_0(e_s) = 1$, for all $e_s \in \mathcal{E}_s$, and $\mu_0(e) = 0$ for all other edges e. A node n is said to be *activated* in a state μ if there exists state μ' such that $\mu \xrightarrow{n} \mu'$. A state μ' is reachable from a state μ, denoted as $\mu \xrightarrow{*} \mu'$ if there exists a (possibly finite) path, $\rho : n_s, n_1, \ldots, n_f \in (\mathcal{N})$ and a finite sequence of markings $\mu_1, \ldots \mu_k$ such that μ is activated in n_s and $\mu \xrightarrow{n_s} \mu_1 \xrightarrow{n_1} \cdots \xrightarrow{n_l} \mu_k$ and $\mu' = \mu_k$.

For a path ρ we denote projection of ρ on set of tasks $\mathcal{T}$ as $\rho_{|_\mathcal{T}}$. A *complete trace* (also called *trace*) of a business process $\mathbf{P}$ is a sequence of tasks $\tau = t_1, \ldots t_l$, $(t_i \in \mathcal{T},\ 1 \leq i \leq l)$ such there is a path $\rho = n_s, n_1, \ldots, n_f$ in $\mathcal{N}$ in $\mathbf{P}$ and $\tau = \rho_{|_\mathcal{T}}$. The set of all traces of a process $\mathbf{P}$ is denoted as $\mathbb{T}_\mathbf{P}$. For the remainder of the paper we assume our process to be sound [2,15].

For annotating processes with effects we shall use logical propositions and assume the existence of a countable set $\mathcal{P}$ of propositions. The set of all literals over $\mathcal{P}$ is denoted as $\mathcal{L}_\mathcal{P}$. A belief base $\mathcal{B}$ is a conjunction of literals in $\mathcal{P}$ which is logically consistent. It can be written using set-theoretic notation. For example, if a belief base $B = a \wedge \neg b \wedge c$, where $a, b, c \in \mathcal{P}$ then $B \equiv \{a, \neg b, c\}$. A theory $\mathbf{T}$ over $\mathcal{P}$ can be taken to be any propositional theory. A knowledge base $\mathcal{K}$ is a pair $(\mathcal{P}, \mathbf{T})$, where $\mathbf{T}$ can be assumed to consist of rules and facts. Wlog we can assume $\mathbf{T}$ to be the conjunction of those rules and facts.

We define an annotated process model/graph as $\mathbb{G}_\mathbf{P} = (\mathcal{N}, \mathcal{F}, \mathcal{K}, \mathcal{A})$[2], where $\mathbf{P} = (\mathcal{N}, \mathcal{F})$ is the underlying process model as before, $\mathcal{K} = (\mathcal{P}, \mathbf{T})$ is the underlying knowledge base annotation, $\mathcal{A}$ is a partial function mapping $n \in \mathcal{T}$ to $\text{eff}(n) \subseteq \mathcal{B}(\mathcal{P})$, and mapping $e \in \text{out}(n)$ for $n \in \mathcal{G}^{xor}_S$ to $(\text{con}(e), \text{pos}(e))$ where $\text{con}(e) \in \mathcal{L}_\mathcal{P}$ and $\text{pos}(e) \in \{1, \ldots, |\text{out}(n)|\}$[3]. The following technical conditions need to be imposed: there does not exist an e such that $\mathbf{T} \wedge \text{con}(e)$ is unsatisfiable; there do not exist n, e, and e' so that $e, e' \in \text{out}(n)$ (e and e' being distinct), $\mathcal{A}(e)$ and $\mathcal{A}(e')$ are defined, and $\text{pos}(e) = \text{pos}(e')$. Moreover, the cardinality of the set of reachable states immediately preceding a choice gateway must be bounded from below by the number of outgoing edges from the gateway.

In an execution of process we shall assume the effects of the activities of the process will be dynamically changed with the corresponding knowledge environment. Let us assume the current available information be represented by an a-priori available knowledge base $\mathcal{K}$; and an accumulated effect B_0 that we assume to be true (*i.e.* to persist) until the next task has been executed. Suppose that a task t in a process is executed in an instance of a process whose effect can be captured by the belief base $B = \text{eff}(t)$. What would be our knowledge after t is executed? Borrowing concepts from artificial intelligence [5,7], we revise our knowledge using belief update to capture the changing scenario (we treat belief revision and belief update as same). This can be achived using an update operators Δ details of which can be found in [9].

Let us now formally define semantics of an annotated process graph, a similar semantic annotation for business processes is provided in [4,15]. A *state* s of $\mathbb{G}$ is a pair $\iota = (\mu, \mathcal{E}_a)$ where μ is a token mapping as defined before, and $\mathcal{E}_a : 2^\mathcal{T} \mapsto 2^{(\mathcal{P})}$ is a cumulative effect accumulation function. Assume that the current set of tasks for which the effects are accumulated till date is $\mathcal{T}_c$ and its accumulated effect is $\mathcal{E}_a(\mathcal{T}_c)$. The updated set of tasks will be $\mathcal{T}'_c$ when a new task is executed. The initial state is $\iota_0 = (\mu_0, \mathcal{E}_{a0})$ where $\mathcal{T}_c = \emptyset$. By default, we assume $\mathcal{E}_{a0}(\emptyset) = \emptyset$. The effects across the tasks/gateways are going to be

[2] we shall drop the subscript $\mathbf{P}$ when it is clear from the context.

[3] For the sake of rigor $\mathcal{P}$ can be partitioned into two sets $\mathcal{P} = \mathcal{P}_t \uplus \mathcal{P}_x$, where tasks are annotated with symbols from $\mathcal{P}_t$, and conditions on choices come from $\mathcal{P}_x$.

accumulated in a recursive manner; the accumulated effect will be denoted as $\mathcal{E}'_a(\mathcal{T}'_c)$. Let ι and ι' be two states. A state ι' is reached from state ι via node n, written as $\iota \xrightarrow{n} \iota'$, if and only if $n \in (T \cup g)$ and $(n_i, n) \in \mathcal{F}$. We consider different cases of n being an task, a split gateway, a join gateway etc. in [9]. As before for a given a business process $\mathbf{P}$ we consider a trace $\mathbf{P}$ to be a sequence of tasks $\tau = t_1, \ldots t_l$, where $t_i \in \mathcal{T}$, $1 \leq i \leq l$. Similarly an *effect trace* is defined to be a sequence $\varepsilon : (t_1, \mathcal{E}_{a1}), (t_2, \mathcal{E}_{a2}), \ldots, (t_m, \mathcal{E}_{am})$, where $\tau = t_1, \ldots t_m$ is called the underlying *trace* of ε. Given an annotated process model we denote $\mathcal{E}_{\mathbb{G}}$ to be the set of its effect traces.

Figure 1 depicts an example of a business process for illustrating effect annotation. This process diagram is drawn using BPM notation. We use this process graph as the running example throughout this paper. This process contains a start and an end node, and various tasks, such as "order", "reject order", "fulfill order" etc. It also has a number of routing constructs such as an XOR-split after the task order and an AND-split after fulfill order. Only one of the branches after the XOR-split is executed depending on the condition (approved or fulfilled) which is true.

Table 1. Effect annotations for process model in Fig. 1

Task	Effects
Order (A)	$ordered \wedge received$
Reject order (C)	$rejected$
Fulfill order (B)	$fulfilled$
Send invoice (D)	$invoiceSent \wedge paymentExpected$
Ship order (E)	$shipped$
Receive payment (F)	$paymentReceived$
Accept payment (G)	$paymentAccepted \wedge \neg paymentExpected \wedge paid$
Close order	$closed$

In Fig. 1 the semantic annotation of each task is given in Table 1. The knowledge base is given by $\mathcal{K} = \{\mathcal{P}, \mathbf{T}\}$, where

$$\mathbf{P} = \{ordered, received, rejected, fulfilled, invoiceSent, paymentExpected, shipped, paymentReceived, paymentAccepted, closed, approved, cancelled; \text{and}$$
$$\mathbf{T} = \{closed \rightarrow \neg(ordered \wedge received \wedge fulfilled);$$
$$(cancelled \wedge rejected) \rightarrow \neg(ordered \wedge received)\}.$$

Assuming an initial state to be $\iota_0 = (\mu_0, \mathcal{E}_{a0})$ where $\mathcal{T}_c = \emptyset$ and $\mathcal{E}_{a0}(\emptyset) = \emptyset$ we can compute the cumulative effect on the execution of each task/gateway. The

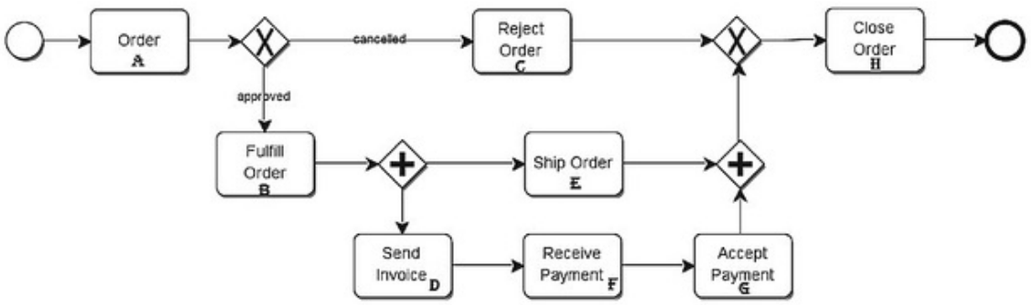

Fig. 1. An example of a process using BPM notation

Table 2. Accumulated effect annotations for process model in Fig. 1

Task	Accumulated effects
Order	$\{ordered, received\}$
Reject order	$\{ordered, received, cancelled, rejected\}$
Fulfill order	$\{ordered, received, approved, fulfilled\}$
Send invoice	$\{ordered, received, approved, fulfilled, shipped, invoiceSent, paymentExpected\}$
	$\{ordered, received, approved, fulfilled, invoiceSent, paymentExpected\}$
Ship order	$\{ordered, received, approved, fulfilled, shipped\}$
	$\{ordered, received, approved, fulfilled, invoiceSent, paymentReceived, paymentAccepted, paid, shipped\}$
Receive payment	$\{ordered, received, approved, fulfilled, shipped, invoiceSent, paymentExpected, paymentReceived\}$
	$\{ordered, received, approved, fulfilled, invoiceSent, paymentExpected, paymentReceived\}$
Accept payment	$\{ordered, received, approved, fulfilled, shipped, invoiceSent, paymentReceived, paymentAccepted, paid\}$
	$\{ordered, received, approved, fulfilled, invoiceSent, paymentReceived, paymentAccepted, paid\}$
Close order	$\{cancelled, rejected, closed, \neg fulfilled\}$
	$\{ordered, received, approved, fulfilled, invoiceSent, paymentReceived, paymentAccepted, paid, shipped, closed\}$

effect accumulation corresponding to the execution of each node is given in the Table 2. The accumulated effect is computed by finding the maximal consistent set computed out of the union of individual effect of the node and the current accumulated effect [9].

3 Effect Log Mining

The log entries that we consider contain event of one type (*viz.*, completion of the event), hence we drop event from the log entries in subsequent discussions and consider only tasks (similar convention is followed in ProM-framework). Let $\mathcal{T}$ to be the set of tasks/activities. We denote trace as $\sigma \in T^*$, where $\sigma = t_0, t_1, \ldots t_{n-1}$, such that $t_i \in \mathcal{T}, 0 \leq i \leq n-1$. A process log is a define as a set of traces, denoted as $W \in \mathcal{P}(\mathcal{T}^*)$. An effect log $\Theta \subseteq \mathcal{P}(\mathcal{T} \times \mathcal{B}(P))$ is defined as a set of effect traces.

The problem of process mining takes process log as input and produces process models as output. That is given a process log $W \in \mathcal{P}(\mathcal{T}^*$ over a set $\mathcal{T}$ of activities, find process model $\mathbf{P} = (\mathcal{N}, \mathcal{F})$ such that $\mathbb{T}_{\mathbf{P}} = W$. In this work we propose a variant of process mining problem which we call effect log mining

(also called effect discovery). It says, given an effect log $\Theta \subseteq \mathcal{P}(\mathcal{T} \times \mathcal{B}(P))$ find annotated process model $\mathbb{G} = (\mathcal{N}, \mathcal{F}, \mathcal{K}, \mathcal{A})$ such that $\mathcal{E}_{\mathbb{G}} = \Theta$.

We made some simplifications of our effect mining problem. We assume our knowledge base $\mathcal{K}$ is a pair $(\mathcal{P}, \mathbf{T})$, and $\mathbf{T}$ is empty, that is, it contains no facts or no rules. Further we make the unique name assumption for tasks, *i.e.*, a task appears only once in the process model, which is just a matter of renaming. Next we propose an algorithm for effect discovery using log abstractions [12] based on log-based ordering relations. Our log-based ordering relation is defined on annotated task nodes (pairs of tasks with accumulated effects). We also define an edge relation between annotated nodes based on the above ordering relation as part of our discovery algorithm. Using this algorithm it will be possible to discover the process and find effect for each individual task in the discovered process. In this work we choose to use a modified α-algorithm to discover our process model [11] (in spite of its inability to address loops, non-free-choice structures, and limited ability to address variability and closed path). For the details of the algorithm the reader is referred to [9].

We employ our process discovery technique on our example where we assume the knowledge base to be $\mathcal{K} = \{\mathcal{P}, \emptyset\}$, where $\mathcal{P}$ is defined before. We consider the effect log ε of the process model in Table 3. The workflow $\mathcal{W}_{\mathbf{P}}$ discovered thus is shown in Fig. 2. Using our technique we can only recover gross annotation of

Table 3. Effect traces and its underlying trace of the process model

Effect trace ε	The underlying trace
((start, ∅), (A,{*ordered, received*}), (A,{*ordered, received, cancelled*}), (C,{*ordered, received, cancelled, rejected*}), (H,{*ordered, received, cancelled, rejected, closed*}), (end,{*ordered, received, cancelled, rejected, closed*}))	(start,A,C,H,end)
((start, ∅), (A, {*ordered, received*}), (A, {*ordered, received, approved*}), (B, {*ordered, received, approved, fulfilled*}), (E, {*ordered, received, approved, fulfilled, shipped*}), (D, {*ordered, received, approved, fulfilled, shipped, invoiceSent, paymentExpected*}), (F, {*ordered, received, approved, fulfilled, shipped, invoiceSent, paymentExpected, paymentReceived*}), (G, {*ordered, received, approved, fulfilled, shipped, invoiceSent, paymentReceived, paymentAccepted, paid*}), (H, {*ordered, received, approved, fulfilled, shipped, invoiceSent, paymentReceived, paymentAccepted, paid, closed*}), (end, {*ordered, received, approved, fulfilled, shipped, invoiceSent, paymentReceived, paymentAccepted, paid, closed*}))	(start,A,B,E,D,F,G,H,end)
((start, ∅), (A, {*ordered, received*}), (A, {*ordered, received, approved*}), (B, {*ordered, received, approved, fulfilled*}), (D, {*ordered, received, approved, fulfilled, invoiceSent, paymentExpected*}), (F, {*ordered, received, approved, fulfilled, invoiceSent, paymentExpected, paymentReceived*}), (G, {*ordered, received, approved, fulfilled, invoiceSent, paymentReceived, paymentAccepted, paid*}), (E, {*ordered, received, approved, fulfilled, invoiceSent, paymentReceived, paymentAccepted, paid, shipped*}), (H, {*ordered, received, approved, fulfilled, invoiceSent, paymentReceived, paymentAccepted, paid, shipped, closed*}), (end, {*ordered, received, approved, fulfilled, invoiceSent, paymentReceived, paymentAccepted, paid, shipped, closed*}))	(start,A,B,D,F,G,E,H,end)

individual task instead of exact effect, however the satisfiability of the former implies that of the latter.

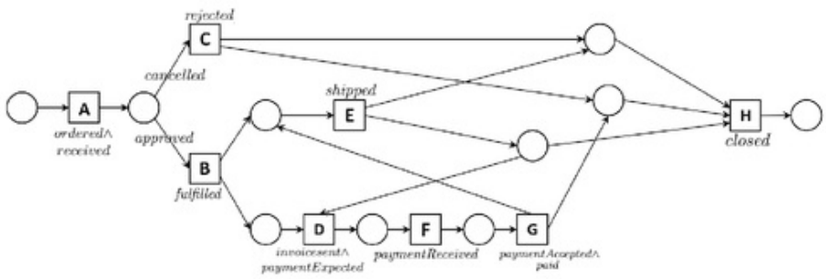

Fig. 2. Discovered workflow $\mathcal{W}_{\mathbf{P}}$

4 Conclusion

Our semantic annotation of process models can help bridge between the high-level process and the actual IT infrastructure as support functions like discovery, composition, mediation can be implemented with concrete services using Semantic Web formalisms. Given an implementation of services and observed snapshots, we should be able to discover the process with its effect annotations using our algorithm. In future we plan to integrate our effect mining technique with data-driven approach to mining effect annotations (and specifically post conditions) from event logs arising in process execution histories [10].

References

1. Dijkman, R.M., Dumas, M., Ouyang, C.: Semantics and analysis of business process models in BPMN. Inf. Softw. Technol. **50**(12), 1281–1294 (2008)
2. Fahland, D., Favre, C., Jobstmann, B., Koehler, J., Lohmann, N., Völzer, H., Wolf, K.: Analysis on demand: instantaneous soundness checking of industrial business process models. Data Knowl. Eng. **70**(5), 448–466 (2011)
3. Hinge, K., Ghose, A.K., Koliadis, G., Process, S.: A tool for semantic effect annotation of business process models. In: Proceedings of the 13th IEEE International Enterprise Distributed Object Computing Conference, EDOC 2009, pp. 54–63 (2009)
4. Hoffmann, J., Weber, I., Governatori, G.: On compliance checking for clausal constraints in annotated process models. Inf. Syst. Front. **14**(2), 155–177 (2012)
5. Katsuno, H., Mendelzon, A.O.: Propositional knowledge base revision and minimal change. Artif. Intell. **52**(3), 263–294 (1991)
6. Kiepuszewski, B., ter Hofstede, A., van der Aalst, W.: Fundamentals of control flow in workflows. Acta Informatica **39**, 143–209 (2003)
7. Liberatore, P.: The complexity of belief update. Artif. Intell. **119**(1–2), 141–190 (2000)

8. Liu, R., Kumar, A.: An analysis and taxonomy of unstructured workflows. In: Aalst, W.M.P., Benatallah, B., Casati, F., Curbera, F. (eds.) BPM 2005. LNCS, vol. 3649, pp. 268–284. Springer, Heidelberg (2005). doi:10.1007/11538394_18
9. Roy, S., Santriputri, M., Ghose, A.: Annotating and mining for effects of processes. Technical report, University of Wollongong, Australia (2015, Available on request)
10. Santiputri, M., Ghose, A.K., Dam, H.K., Wen, X.: Mining process task post-conditions. In: Johannesson, P., Lee, M.L., Liddle, S.W., Opdahl, A.L., López, Ó.P. (eds.) ER 2015. LNCS, vol. 9381, pp. 514–527. Springer, Heidelberg (2015). doi:10.1007/978-3-319-25264-3_38
11. van der Aalst, W.M.P., Weijters, T., Maruster, L.: Workflow mining: discovering process models from event logs. IEEE Trans. Knowl. Data Eng. **16**(9), 1128–1142 (2004)
12. Dongen, B.F., Aalst, W.M.P.: Multi-phase process mining: building instance graphs. In: Atzeni, P., Chu, W., Lu, H., Zhou, S., Ling, T.-W. (eds.) ER 2004. LNCS, vol. 3288, pp. 362–376. Springer, Heidelberg (2004). doi:10.1007/978-3-540-30464-7_29
13. Vanhatalo, J., Völzer, H., Koehler, J.: The refined process structure tree. In: Dumas, M., Reichert, M., Shan, M.-C. (eds.) BPM 2008. LNCS, vol. 5240, pp. 100–115. Springer, Heidelberg (2008). doi:10.1007/978-3-540-85758-7_10
14. Vanhatalo, J., Völzer, H., Leymann, F.: Faster and more focused control-flow analysis for business process models through SESE decomposition. In: Krämer, B.J., Lin, K.-J., Narasimhan, P. (eds.) ICSOC 2007. LNCS, vol. 4749, pp. 43–55. Springer, Heidelberg (2007). doi:10.1007/978-3-540-74974-5_4
15. Weber, I., Hoffmann, J., Mendling, J.: Beyond soundness: on the verification of semantic business process models. Distrib. Parallel Databases **27**(3), 271–343 (2010)

Annotating and Mining for Effects of Processes

ORIGINALITY REPORT

28%

SIMILARITY INDEX

PRIMARY SOURCES

1	"Transactions on Petri Nets and Other Models of Concurrency XI", Springer Nature, 2016 Crossref	220 words — 6%
2	"FM 2014: Formal Methods", Springer Nature America, Inc, 2014 Crossref	112 words — 3%
3	sbpm2008.fzi.de Internet	89 words — 2%
4	Suman Roy, Sidharth Bihary, Jose Alfonso Corso Laos. "A CSP-theoretic Framework of Checking Conformance of Business Processes", 2012 19th Asia-Pacific Software Engineering Conference, 2012 Crossref	70 words — 2%
5	Jörg Hoffmann. "On compliance checking for clausal constraints in annotated process models", Information Systems Frontiers, 05/23/2009 Crossref	53 words — 1%
6	www.imweber.de Internet	52 words — 1%
7	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	50 words — 1%
8	Lecture Notes in Computer Science, 2013. Crossref	40 words — 1%
9	Kerry Hinge, Aditya Ghose, Andrew Miller. "A Framework for Detecting Interactions Between Co-Incident Clinical Processes", International Journal of E-Health and Medical Communications, 2010 Crossref	27 words — 1%
10	Adriano Augusto, Raffaele Conforti, Marlon Dumas, Marcello La Rosa, Giorgio Bruno. "Chapter 25 Automated Discovery of Structured Process Models: Discover Structured vs. Discover and Structure", Springer Nature, 2016 Crossref	26 words — 1%

11	Sidharth Bihary, Jagadish Koneti, Suman Roy. "Process conformance using CSP", Proceedings of the 5th India Software Engineering Conference on - ISEC '12, 2012 Crossref	21 words — 1%
12	www.processmining.org Internet	19 words — 1%
13	Alberto Bastias, Sidharth Bihary, Suman Roy. "An Automated Analysis of Errors for BPM Processes Modeled Using an In-house Infosys Tool", 2011 18th Asia-Pacific Software Engineering Conference, 2011 Crossref	18 words — 1%
14	ceur-ws.org Internet	17 words — < 1%
15	"Business Process Management Workshops", Springer Nature America, Inc, 2011 Crossref	16 words — < 1%
16	Lecture Notes in Computer Science, 2011. Crossref	14 words — < 1%
17	W. M. P. Aalst. "Process mining: a two-step approach to balance between underfitting and overfitting", Software & Systems Modeling, 11/25/2008 Crossref	12 words — < 1%
18	Maya Lincoln. "Content-Based Validation of Business Process Modifications", Lecture Notes in Computer Science, 2011 Crossref	12 words — < 1%
19	ro.uow.edu.au Internet	12 words — < 1%
20	dspace.lib.cranfield.ac.uk Internet	10 words — < 1%
21	Lecture Notes in Computer Science, 2016. Crossref	10 words — < 1%
22	www.rug.nl Internet	10 words — < 1%
23	"Advanced Information Systems Engineering", Springer Nature, 2017 Crossref	9 words — < 1%
24	tel.archives-ouvertes.fr Internet	9 words — < 1%

25	"On the Move to Meaningful Internet Systems. OTM 2018 Conferences", Springer Nature, 2018 Crossref	9 words — < 1%
26	"Conceptual Modeling", Springer Nature, 2017 Crossref	9 words — < 1%
27	ie.technion.ac.il Internet	8 words — < 1%
28	Ingo M. Weber. "Verification of Annotated Process Models", Lecture Notes in Business Information Processing, 2009 Crossref	8 words — < 1%
29	Web Services Foundations, 2014. Crossref	8 words — < 1%
30	Hoa Khanh Dam, Aditya Ghose. "Mining version histories for change impact analysis in business process model repositories", Computers in Industry, 2015 Crossref	7 words — < 1%
31	"Advanced Information Systems Engineering", Springer Nature America, Inc, 2015 Crossref	6 words — < 1%
32	"On the Move to Meaningful Internet Systems. OTM 2017 Conferences", Springer Nature, 2017 Crossref	6 words — < 1%
33	Lecture Notes in Computer Science, 2010. Crossref	6 words — < 1%
34	"Business Process Management", Springer Nature America, Inc, 2014 Crossref	6 words — < 1%
35	Lecture Notes in Business Information Processing, 2013. Crossref	6 words — < 1%

EXCLUDE QUOTES OFF
EXCLUDE BIBLIOGRAPHY ON
EXCLUDE MATCHES OFF